Matematički fakultet, Univerzitet u Beogradu

Magistarska teza

Dizajn, implementacija i verifikacija protokola za plaćanje putem Interneta

Autor:
Ognjen Marić

Mentor:
dr Filip Marić

Članovi komisije:
dr Dušan Tošić,
Matematički fakultet, Beograd
dr Zoran Đurić
Elektrotehnički fakultet, Banja Luka
dr Filip Marić
Matematički fakultet, Beograd

oktobar 2010.

Master's Thesis: Design, Implementation and Verification of an Internet Payment System
Author: Ognjen Marić (1981 -)

Keywords: cryptographic protocols, software verification, formal methods, payment systems, e-commerce, internet, security kriptografski
Pages: 93
Thesis Committee: Filip Marić (supervisor), Dušan Tošić, Zoran Đurić
Date: October 22, 2010
Language: Serbian

Abstract

In this thesis we describe a new system called IPS (Internet Payment System). We present an existing general payment system classification, and provide a closer look at several existing systems. Based on their advantages and disadvantages, we define goals that the new system needs to fulfill, as an attempt to reconcile the security requirements with usability requirements. As is generally true for distributed systems, Internet payment systems also secure their communication through cryptographic protocols. Hence, we provide a detailed description of the proposed protocol, which constitutes the core component of the system. We also provide an overview of a pilot implementation of the system.

Security requirements are of vital importance for payment systems. Since their cryptographic protocols are usually directly exposed to attackers, it is important to provide a formal proof that these protocols satisfy their security requirements, that is, the protocols need to be verified. We provide an overview of modern verification methods, a then provide a proof of correctness of the proposed protocol. The proof is first performed using the AVISPA package, and then using Isabelle/HOL, ensuring automated proof checking. The Isabelle/HOL proof uses a model developed specifically for the IPS protocol, which provides certain advantages in comparison to some earlier models.

Naslov teze: Dizajn, implementacija i verifikacija sistema za plaćanje putem Interneta
Autor: Ognjen Marić (1981 -)

Ključne riječi: kriptografski protokoli, verifikacija programa, formalne metode, platni sistemi, Internet, elektronska trgovina, sigurnost
Broj strana: 93
Komisija: Filip Marić (mentor), Dušan Tošić, Zoran Đurić
Datum: 22. oktobar 2010.
Jezik: Srpski

Sažetak

U ovoj tezi opisujemo novi sistem za plaćanje putem Interneta, nazvan IPS (Internet Payment System). Prikazujemo jednu opštu klasifikaciju sistema za plaćanje, a potom dajemo pregled nekoliko postojećih sistema. Uvidom u njihove prednosti i mane, definišemo ciljeve koje novi sistem treba da ispuni, kao pokušaj da se pomire zahtjevi sigurnosti sa zahtjevima lakoće i efikasnosti korišćenja. Kao i većina distribuiranih sistema, i sistemi za plaćanje putem Interneta sigurnost svoje komunikacije obezbjeđuju kriptografskim protokolima. Stoga u opisu predloženog sistema pružamo i detaljan pregled predloženog protokola, koji je ujedno i centralna komponenta sistema. Potom dajemo i pregled izvršene probne implementacije sistema.

Sigurnosni zahtjevi su od centralne važnosti za platne sisteme. Kako su njihovi kriptografski protokoli direktno izloženi spoljnim napadačima, neophodno je pružiti formalan dokaz ispunjenja sigurnosnih zahtjeva od strane ovih protokola, odnosno protokole je potrebno verifikovati. Dajemo pregled modernih metoda verifikacije, a zatim pružamo i dokaz korektnosti predloženog protokola. Dokaz je proveden pomoću programskog paketa AVISPA, a zatim i u sistemu Isabelle/HOL, čime je obezbijeđena provjera dokaza od strane računara. Dokaz u sistemu Isabelle/HOL je proveden koristeći model posebno razvijen za IPS protokol, koji pruža određene prednosti u odnosu na neke prethodne modele.

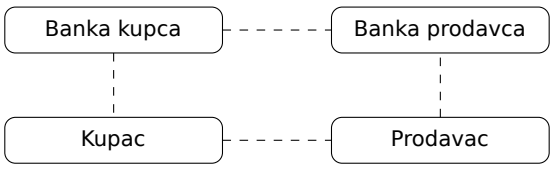
Sadržaj

Glava 1

Uvod

Internet je postao jedno od glavnih sredstava za trgovinu i pružanje finansijskih usluga. Elektronska trgovina donosi brojne pogodnosti. Kupovina je moguća bez obzira na lokaciju kupca ili prodavca, a ukoliko se kupuju elektronska dobra poput knjiga u elektronskom obliku, zvučnog ili video sadržaja, kupac trenutno dobija plaćenu robu, bez čekanja. Međutim, elektronska trgovina sa sobom nosi i određene opasnosti. Internet je nesiguran i nepouzdan prenosnik podataka, i jedna od glavnih prepreka za brži rast ove vrste trgovine jeste strah od prevare i gubitka povjerljivih informacija.

U tipičnom scenariju trgovine putem Interneta, kupac kupuje robu od prodavca koristeći kreditnu karticu. Za obavljanje kupovine potrebno je da kupac i prodavac imaju otvorene račune u bankama. Nakon uspješno završene kupovine, novac je potrebno prebaciti sa računa kupca na račun prodavca. Ovaj osnovni scenario je predstavljen na na slici 1.1. Veze prikazane na slici su logičke veze među učesnicima. Fizički, veze se ostvaruju putem komunikacionih kanala, čija topologija može biti različita od one prikazane na slici.



Slika 1.1: Tipičan scenario elektronske trgovine

Radi jednostavnosti, pretpostavićemo da kupac i prodavac imaju račune u istoj banci. Stoga posmatramo samo tri učesnika: kupca (K), prodavca (P) i banku (B). Kao što smo već napomenuli, učesnici su povezani komunikacionim kanalima. Za izvršenje kupovine, neophodan je prenos određenih podataka preko tih kanala. Ukoliko se podaci prenose preko kanala bez ikakve zaštite, postoji mogućnost prisluškivanja. Bilo koja strana koja ima mogućnost praćenja mrežnog saobraćaja na nekom od kanala bi lako mogla doći u posjed osjetljivih informacija, poput, na primjer, brojeva kreditnih kartica. Pred ovakvim *pasivnim napadačima* opasnost se efikasno može ukloniti korišćenjem nekih od brojnih kriptografskih algoritama i tehnika, razvijenih u proteklim decenijama. Međutim, u prisustvu tzv. *aktivnih napadača*, koji imaju mogućnost ne samo praćenja mrežnog saobraćaja, već i promjene postojećih poruka ili kreiranja novih, postoje i druge vrste opasnosti. Napa-

dač bi se tada mogao lažno predstaviti kao neki od učesnika protokola. Na primjer, mogao bi uvjeriti kupca da je on (napadač) u stvari prodavac. Osim toga, napadač bi mogao i da promijeni sadržaj poruka, na primjer da bi promijenio adresu na koju se plaćena roba šalje. Ovakav scenario je sasvim realan, pogotovo u slučaju bežične komunikacije.

Stoga kriptografski algoritmi, sami po sebi, nisu dovoljni. Neophodno je da se K, P i B dogovore o skupu pravila za razmjenu poruka i upotrebu kriptografije, tj. o *kriptografskom protokolu*, tačnije *protokolu za plaćanje*, koji bi im garantovao bezbjednost. Mnogi protokoli ove vrste su već predloženi u prošlosti. Svi oni pokušavaju da riješe prethodno navedene probleme, sa manje ili više uspjeha. Pored navednih, postoje i drugi problemi; ovdje navodimo tri veoma bitna.

1. P mora biti siguran da je K ovlašćen da koristi kreditnu karticu za plaćanje; u suprotnom, P će biti odgovoran za tzv. ,,card not present" povrate novca.
2. Ukoliko P ima pristup broju kreditne kartice kupca K, K mu mora vjerovati da neće zloupotrijebiti taj broj; štaviše, K se mora pouzdati u sigurnost P-ovog sistema, jer bi ,,provaljivanje" u P-ov sistem od strane trećeg lica napadaču između ostalog razotkrilo i broj kreditne kartice kupca K. Stoga, krajnje je poželjno da P nema pristup ovom broju.
3. B ne bi trebalo da ima pristup sadržaju narudžbe. Ovaj sadržaj bi mogao da bude poslovna tajna za P, kao i pitanje privatnosti za K.

Da bi neki kriptografski protokol bio šire prihvaćen i korišćen, neophodno je dokazati njegovu ispravnost. Ovakav dokaz je posebno bitan za protokole za plaćanje, gdje bilo kakav propust u protokolu može imati ozbiljne posljedice po njegove učesnike. U svrhu dokazivanja ispravnosti, neophodno je prvo formalizovati kako sam protokol, tako i ciljeve koje protokol treba da ostvari. Danas postoje mnoge tehnike za ovu vrstu formalizacije, odnosno mnogi načini za formalno modeliranje protokola. Po izvršenoj formalizaciji, možemo pristupiti i samom dokazu korektnosti. Ovo pak, nije nimalo trivijalan posao; dokazi su obično veoma kompleksni i moguće su greške. Stoga nije rijedak slučaj da se u ,,dokazivo sigurnim" sistemima za plaćanje nakon nekog vremena otkriju slabosti, koje su obično vezane upravo za neki nedostatak u dokazu ili u korišćenom modelu protokola ([PW92], [PSW95], [Kai95]). Iz ovih razloga, poželjno je da formalizacija omogući mašinsku provjeru dokaza (tj. provjeru dokaza od strane računara i specijalizovanog softvera).

Druga bitna klasa problema sa kojima se sistemi za plaćanje suočavaju nije vezana za sigurnost. Ni najsigurniji protokol ne znači mnogo ukoliko ga neće koristiti značajan broj korisnika. Platni sistemi stoga moraju biti dovoljno jednostavni da bi ih prosječni korisnici (kupci) mogli upotrebljavati. Osim toga, za kupce je poželjno da im platni sistem omogući kupovinu sa bilo koje lokacije. Sa stanovišta prodavaca, sistem mora opravdati ulaganje u njegovu implementaciju.

Glavni doprinos ovog rada je novi sistem za plaćanje, nazvan IPS (Internet Payment System). Izvlačeći pouke iz postojećih sistema za plaćanje, IPS pokušava da nađe rješenje za obe navedene klase problema. Drugi bitan doprinos je široko provedena verifikacija sigurnosti ovog sistema, koja zbog kompleksnosti i specifičnosti samog protokola postavlja nove izazove.

Dijelovi rada su već objavljeni u [DMG07], koji je nastao u saradnji sa Zoranom Đurićem i Draganom Gaševićem. Ovo se prvenstveno odnosi na sadržaj glave 3, kao i na dio glave 5, tačnije na verifikaciju pomoću paketa AVISPA.

1.1 Kratak pregled rada

Kratak pregled osnovnih pojmova neophodnih za razumijevanje ovog rada dat je u glavi 2. Glavu započinjemo pregledom kriptografskih alata (poglavlja 2.1 i 2.2). Drugi dio glave je posvećen kriptografskim protokolima. Uvod u kriptografske protokole dajemo u poglavlju 2.3, a zahtjeve koje ovi protokoli treba da ispune u poglavlju 2.4. Glavu završavamo primjerima napada na kriptografske protokole (poglavlje 2.5).

Ostatak rada se, kako i naslov sugeriše, sastoji od tri glavne cjeline, od kojih je svakoj posvećena po jedna glava. U glavi 3 obrađen je dizajn sistema IPS i njegovog protokola za plaćanje. Glava započinje pregledom osnovnih osobina elektronskih sistema plaćanja, te njihovom klasifikacijom. Zatim slijedi detaljniji pregled nekoliko poznatih, već postojećih sistema za plaćanje i njihovih osobina. Rezultat ovog pregleda je skup ciljeva koje IPS treba da ispuni. Potom dajemo opis predloženog sistema, uz detaljno objašnjenje njegovih dijelova, a sa posebnim akcentom na IPS protokol za plaćanje. Konačno, precizno definišemo sigurnosne zahtjeve koje protokol treba da ispuni.

U sljedećoj glavi, glavi 4, dat je kratak opis provedene implementacije sistema. Poglavlje 4.1 sadrži pregled implementacije u cjelini, njene arhitekture i korišćenih tehnologija. Poglavlje 4.3 opisuje tzv. shemu označavanja poruka, dok poglavlja 4.4 i 4.5 daju nešto detaljniju sliku o implementaciji pojedinih komponenti sistema.

Posljednja cjelina, verifikacija sigurnosti protokola, je obrađena u glavi 5. Glava započinje pregledom postojećih tehnika verifikacije sigurnosnih protokola (poglavlje 5.1), sa posebnim osvrtom na alat nazvan *AVISPA* (poglavlje 5.2), te na tehniku nazvanu *induktivni pristup* (poglavlje 5.4). Zatim počinje opis provedene verifikacije IPS protokola. Verifikacija je provedena dvojako. Prvo, protokol je modeliran upravo pomoću alata AVISPA, koji omogućava automatsku provjeru osobina protokola, ali i postavlja određena ograničenja na okruženje u kojem se protokol ispituje. Nakon rezultata verifikacije ovim paketom, završni dio glave opisuje verifikaciju protokola provedenu bez ovih ograničenja, koristeći dokazivač teorema nazvan Isabelle. Naš pristup ovoj verifikaciji različit je od induktivnog pristupa, a razlike su opisane u odjeljku 5.5.4.

Rad se završava glavom 6 koja, kao što je to uobičajeno, sumira rezultate ove teze i sugeriše neke pravce za dalji rad.

Zahvalnost

Kao valjda i sve druge teze na ovom svijetu, ni ova ne bi bila ista bez pomoći drugih ljudi. Najveću zahvalnost za samu tezu dugujem naravno mentoru, dr Filipu Mariću, bez kojeg

bi ona sigurno izgledala drugačije (to jest - gore), a sasvim sigurno bi i sam rad išao mnogo teže. Isabelle je dama, ali zna da se ponaša i na načine koje nije uputno spominjati u jednoj magistarskoj tezi; mentorova zasluga je što sam se na kraju ipak susreo sa damom.

Za moj prvi kontakt sa kriptografskim protokolima je zaslužan dr Zoran Đurić. Sam IPS protokol je nastao upravo na osnovu njegove ideje, a i dobar dio ove teze je proistekao iz gotovo jednogodišnjeg zajedničkog rada sa njim na tom polju. Taj period je za mene bio izuzetno poučan i interesantan, a Zoranova neiscrpna energija i nevjerovatna radna etika su bili ključni za prevazilaženje nekih mukotrpnih trenutaka. Stoga mi je zaista drago da je pristao da bude član komisije za ocjenu moje teze, a povrh svega mi je čast da takvu osobu mogu da nazovem i svojim prijateljem.

Moja radna etika, s druge strane, nažalost nije uvijek bila na tako visokom nivou, stoga ovu tezu i pišem krajem 2010. godine, pet godina nakon svog diplomskog rada. S druge strane, u tom periodu sam stekao i neke nove dobre prijatelje, zadržao neke stare, a neke nažalost i izgubio. Hvala im svima za sve vesele zajedničke trenutke, i što nisu zaboravili na mene čak i ponekad kada bih ja zaboravio na njih.

Želio bih da se zahvalim i dr Dušanu Tošiću što je prihvatio članstvo u komisiji za moju tezu u ,,kriznom" trenutku, kao i na ekspeditivnosti oko cijelog postupka.

Zahvalnost na ekspeditivnosti dugujem i ostalim članovima komisije, a i svom osoblju na Matematičkom fakultetu koje mi je redovno izlazilo u susret po raznoraznim pitanjima, kada sam bio pritisnut pred neposredno nadolazeći rokovima. Posljednjih 7 ili 8 mjeseci su za mene protekli u užasavajućem tempu; hvala im što su mi taj period olakšali u mjeri u kojoj su mogli.

Nekim drugim ljudima sam pak zahvalan na strpljenju, pogotovo u ovom posljednjem periodu. Prvenstveno hvala mojoj djevojci, koja je podnosila sve moje divne osobine poput sjajne moći koncentracije (kada zaboravim na neke druge stvari na koje ne bih trebao) ili sklonosti ka neobičnom radnom vremenu (nerijetko cjelodnevnom). Milijana, hvala ti na svemu.

Konačno, hvala mojoj porodici, roditeljima Slavku i Biljani, te bratu Nenadu. Najbolji ste.

Glava 2

Osnove

U ovoj glavi prezentujemo osnovne pojmove kriptografije i kriptografskih protokola. Čitaoci upoznati sa problematikom mogu da preskoče ovu glavu, uz napomenu da ćemo u radu kao definiciju jake, odnosno slabe autentifikacije koristiti definicije injektivne, odnosno neinjektivne saglasnosti iz [Low96a].

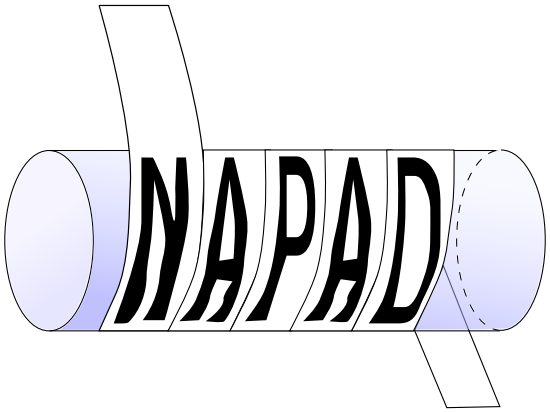
Poglavlja 2.1 i 2.2 na veoma neformalan način uvode osnovne kriptografske pojmove. Za strožiji tretman materijala preporučujemo neku od brojnih odličnih knjiga iz oblasti kriptografije, kao što je [MOV01]. Formalne definicije su namjerno preskočene iz dva razloga. Prvi razlog je što bi u ovako ograničenom prostoru teško bilo objasniti značenje ovakvih definicija. Drugi razlog je što se ova teza ipak ne bavi kriptografijom, već kriptografskim protokolima. Ostatak glave bi trebalo da objasni smisao ovakve tvrdnje, uvodeći kriptografske protokole, definišući ciljeve ovakvih protokola, te dajući primjere napada na njih.

2.1 Kriptografski algoritmi

Osnovna uloga kriptografije je da omogući skrivanje informacija. Kriptografija se, kao nauka i vještina, razvija već nekoliko hiljada godina. Jedan od prvih primjera je šifra zvana Skitale, koja se navodno koristila u staroj Sparti, oko 400. godine prije nove ere. Skitale je u osnovi obična šipka, tačnije par šipki istog prečnika. Jedna šipka je kod pošiljaoca poruke, druga kod primaoca. Oko šipke se više puta obmota dugačka traka papira, na kojoj se zatim ispiše poruka. Poruka se ispisuje s lijeva na desno, u više redova.

Na primjer, pretpostavimo da želimo da svojim saveznicima pošaljemo poruku ,,napad u zoru" (koju ćemo spojiti u ,,napaduzoru"), a da se pri tom obezbijedimo u slučaju da protivnik presretne poruku. Prvo obmotamo papir oko šipke, a zatim zapišemo poruku. Ako je prečnik šipke toliki da se na svakoj njenoj strani može napisati po jedno slovo, onda kao na slici 2.1 na prednjoj strani možemo napisati ,,napad"; na zadnjoj bismo napisali ,,uzoru". Ako sada odmotamo ovu traku papira, dobijamo tekst ,,nuazpoardu".

Početna poruka, u ovom slučaju ,,napad u zoru" (ili preciznije, ,,napaduzoru") je tzv. *otvoreni tekst*. Poruka dobijena rednim čitanjem slova na papiru, u ovom slučaju ,,nuazpoardu"



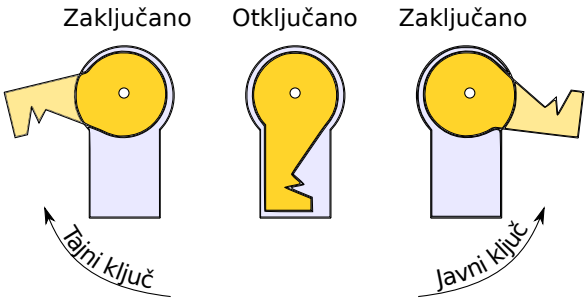
Slika 2.1: Skitale

je tzv. *šifrat*. Da bi se iz šifrata dobio otvoreni tekst, potrebno je znati ispravan prečnik šipke; kažemo da je ovaj prečnik *ključ*. Čak i kada bi protivnik došao u posjed šifrata, bilo bi mu potrebno da poznaje ovaj ključ.

Skitale naravno nije posebno dobar način za šifrovanje poruka. Ako znamo da je korišćen baš Skitale, i ako posjedujemo šifrat, uz malo truda bismo prostom metodom pokušaja i pogrešaka mogli odrediti i otvoreni tekst i ključ; mogli bismo *razbiti* ovaj sistem. Vremenom, smišljane su sve bolje i bolje metode šifrovanja, koje nije bilo jednostavno razbiti. Čuvena mašina Enigma koje su njemačke snage koristile u Drugom svjetskom ratu je jedan primjer takve metode. Enigma je za šifrovanje upotrebljavala komplikovan mehanizam zupčanika.

Sve ove metode šifrovanja su se mogle predstaviti određenim matematičkim transformacijama. Veoma važan korak u postavljanju matematičkih osnova za ovakve metode je napravio Claude Shannon u svom čuvenom radu o teoriji informacija 1949. godine. Pronalaskom digitalnih računara, mogućnosti izvršavanja matematičkih transformacija su znatno unaprijeđene, a time su znatno unaprijeđene i metode šifrovanja.

Ipak, mada neuopredivo bolje nego što je to bio Skitale, sve ove metode su imale jednu zajedničku karakteristiku sa njim: i pošiljalac i primalac su morali da posjeduju isti ključ prije nego što bi mogli početi komunikaciju. Revolucija počinje 1976, kada Whitfield Diffie i Martin Hellman objavljuju svoj rad u kojem uvode mehanizam koji će kasnije postati poznat pod nazivom *asimetrično šifrovanje* (ili *asimetrična kriptografija*). Razliku u odnosu na klasične, odnosno *simetrične* metode šifrovanja možemo ilustrovati analogijom. Zamislimo da Alisa šalje Bobanu poruku za koju želi da ostane tajna. Alisa može da stavi poruku u kutiju sa bravicom, da je zaključa i pošalje Bobanu. Da bi otključao kutiju, Boban mora da posjeduje istovjetan ključ. Štaviše, niko drugi ne smije da posjeduje takav ključ. Ovo je mehanizam simetričnog šifrovanja. Diffie i Hellman su uspjeli da naprave poseban mehanizam za bravu, za koji su potrebne dvije vrste ključa: jedna koja se može okrenuti samo u smjeru kazaljke na satu, i druga koja funkcioniše u obrnutom smjeru (slika 2.2). Boban tada mirno može mirno da ostavi i kopije kutijice i kopije jednog od ovih ključeva ispred vrata; ukoliko neko stavi poruku u jednu od ovih kutijica i zaključa ju ostavljenim ključem, samo će je Boban moći otključati. Ključ čije je kopije ostavio dostupnim je njegov *javni* ključ; ključ koji je ostao kod njega je njegov *privatni* ili *tajni* ključ.



Slika 2.2: Ilustracija asimetričnog šifrovanja

Zbog toga se ponekad asimetrična kriptografija naziva i *kriptografija javnim ključevima* (eng. public-key cryptography). Za odgovarajući par privatnih i javnih ključeva kažemo da su međusobno *inverzni*; za simetrične ključeve potom kažemo da su inverzni sami sebi.

Na temelju ovih otkrića, nastali su mnogi algoritmi za simetrično i asimetrično šifrovanje, od kojih su neki prihvaćeni i kao međunarodni standardi. Rad na njihovom unapređenju i razvoju traje i dan danas. Neki od najpoznatijih modernih kriptografskih algoritama su tako:

- *D-H* (Diffie-Hellman) algoritam za razmjenu simetričnih ključeva na osnovu javnih;
- *RSA* (Rivest-Shamir-Adelman) i *ElGamal* za asimetrično šifrovanje;
- *AES* (Advanced Encryption Standard), *DES* (Digital Encryption Standard) i *Triple-DES* za simetrično šifrovanje, uz napomenu da su DES i Triple-DES nešto stariji algoritmi čija se upotreba ne preporučuje za nove sisteme. Drugi poznati algoritmi za simetrično šifrovanje uključuju algoritme *IDEA* (International Data Encryption Algorithm), *RC4* (Rivest Cipher 4), *RC2* (Rivest Cipher 2), *Blowfish* i *Camelia*.

Iz ovog kratkog pregleda, moglo bi se zaključiti da je sve što nam je potrebno za ostvarivanje savršeno sigurne komunikacije upravo savršeno siguran algoritam za šifrovanje, koji se ne može razbiti. Nažalost, istina je malo drugačija. Algoritmi šifrovanja su neophodan element sigurnosti, ali sami po sebi ne mogu da garantuju sigurnu komunikaciju. Ili, da opet iskoristimo analogiju: nije vam dovoljan savršen lanac da biste sačuvali svoj bicikl. Ako ne zavežete lanac oko bicikla na pravi način, lopov ga i dalje može ukrasti. Isto tako, u distribuiranim sistemima, neophodno je kriptografske komponente posložiti na pravi način.

Osim samih algoritama za šifrovanje, postoje i mnogi drugi bitni kriptografski mehanizmi. U našem radu koristićemo tri takva mehanizma: digitalne potpise, kriptografske heš funkcije i kriptografski sigurne generatore pseudoslučajnih brojeva.

Digitalni potpisi su mehanizmi kojima se utvrđuje autor neke poruke. U osnovi, digitalne potpise možemo posmatrati kao asimetrično šifrovanje privatnim ključem. Zaista, u našoj analogiji, ako Boban stavi neki poruku u kutijicu i zaključa je svojim privatnim ključem, osoba koja otključa kutijicu javnim ključem može biti sigurna da je baš Boban taj koji je ubacio poruku.

Jedan problem sa ovakvom vrstom potpisa je da je potpis dugačak isto onoliko koliko i sama poruka koja se potpisuje. Tako bi se za slanje potpisane poruke udvostručila količina podataka neophodnih za prenos. Jedno rješenje leži u *kriptografskim heš funkcijama* (mi ćemo obično preskakati pridjev ,,kriptografski" i koristiti samo naziv heš funkcije). U osnovi, ove funkcije pokušavaju da naprave **jedinstvenu** smanjenu sliku svake poruke, zvanu *sažetak poruke* (eng. message digest). Veličina ove slike je fiksirana za heš funkciju, i za dvije najpopularnije danas korišćene funkcije ona iznosi 128, odnosno 160 bita. Kako nam je domen funkcije beskonačan, a kodomen konačan, iz Dirihleovog principa slijedi da ove funkcije ne mogu zaista da naprave jedinstvenu smanjenu sliku. Zato se zadovoljavamo slabijim uslovom: za proizvoljnu poruku M_1, i zadanu heš funkciju $h(\cdot)$, zahtijevamo da je praktično nemoguće pronaći drugu poruku M_2, takvu da bude $h(M_1) = h(M_2)$. Osim toga, ove heš funkcije zadovoljavaju i dodatni uslov: ako nam je poznato samo $h(M)$, praktično je nemoguće izračunati samo M, mada je računanje u obrnutom smjeru (računanje $h(M)$ iz M) jednostavno; za takve funkcije kažemo da su *jednosmjerne* (eng. one-way). Značenje termina ,,praktično nemoguće" se može i strogo definisati, ali mi to ovdje nećemo uraditi; u osnovi, svodi se na to da za gotovo nijedan ulaz ne možemo napraviti efikasan algoritam koji daje ispravan izlaz. Heš funkcije imaju i neke druge korisne osobine i mogu se iskoristiti u razne svrhe, ali ovo je jedina svrha koja nas interesuje u ovom radu: poruku M ćemo pretvoriti u njen sažetak, $h(M)$, njega ćemo digitalno potpisati tajnim ključem, te ovaj potpis koristiti da utvrdimo autora poruke. Osim toga, zbog jednosmjernosti heš funkcija, samo pomoću potpisa neke poruke neće biti moguće utvrditi njen sadržaj.

Najpoznatiji postojeći kriptografski heš algoritmi su *MD5* (Message Digest 5) i *SHA-1* (Secure Hash Algorithm 1), a među najpoznatijim algoritmima za digitalne potpise je *DSA* (Digital Signature Algorithm), kao i algoritam zasnovan na RSA algoritmu.

Posljednji kriptografski mehanizam koji ćemo koristiti su *kriptografski sigurni generatori pseudoslučajnih brojeva*. U mnogim primjenama, biće nam potrebni *slučajni brojevi*, koje napadač ne može da pogodi. Računarski sistemi obično nemaju izvore stvarne slučajnosti (npr. slučajnosti potekle iz kvantnih procesa). S druge strane, kombinacije određenih sistemskih parametara mogu napadaču biti teške za saznati i predvidjeti, tj. mogu predstavljati izvor *tajne entropije*. Međutim, čak i ovakvi izvori su obično ograničeni. Zato se najčešće moramo zadovoljiti *pseudoslučajnim brojevima*, tj. brojevima koji su generisani determinističkim algoritmom, počevši od nekog zaista slučajnog broja (ili broja poteklog iz izvora tajne entropije). Međutim, za kriptografske primjene ovakvih brojeva moramo da postavimo i uslov da, čak i ako napadač sazna veći broj uzastopnih pseudoslučajnih brojeva koje smo koristili, ne može da pogodi naredni. Ovaj dodatni zahtjev pravi razliku između uobičajenih generatora pseudoslučajnih brojeva, i onih koji su *kriptografski sigurni*, a samim tim i između pseudoslučajnih brojeva i kriptografski sigurnih pseudoslučajnih brojeva. Ipak, mada je ova razlika veoma bitna, mi ćemo u daljem tekstu često biti neprecizni i govoriti samo o slučajnim brojevima, pri čemu se podrazumijeva da su ovi brojevi u stvari pseudoslučajni, te da su kriptografski sigurni. Većina ovih slučajnih brojeva se nikad ne upotrebljava više od jedan put za istu namjenu; za takve brojeve ćemo upotrebljavati i engleski termin *nonce*. Jedan poznat kriptografski siguran generator je *Blum-Blum-Shub* generator.

Generatori slučajnih brojeva imaju i još jednu bitnu primjenu: ključevi u računarskim sistemima šifrovanja su uvijek brojevi (ili se barem mogu tako posmatrati), pa se ovi generatori koriste i za generisanje novih ključeva.

2.2 Infrastruktura javnih ključeva

Asimetrična kriptografija rješava jedan veliki problem: obezbjeđuje sigurnu komunikaciju između udaljenih učesnika, bez potrebe da prethodno razmijene tajne ključeve za šifrovanje. Dakle, ako Alisa (A) želi da pošalje poruku Bobanu (B), ne mora više da prethodno usaglasi tajni ključ sa njim. Dovoljno je da ima Bobanov javni ključ. Sada, međutim, Alisa ima novi problem: kako može biti sigurna da je ključ koji posjeduje zaista Bobanov?

Infrastruktura javnih ključeva (eng. public key infrastructure, skraćeno PKI) služi da učesnicima u mreži obezbijedi utvrđivanje vlasništva nad javnim ključevima. Najčešći način da se ovo obezbijedi je takozvana *hijerarhija povjerenja* (eng. hierarchy of trust). Za funkcionisanje ovog mehanizma, potrebno je da postoji neka strana kojoj svi vjeruju. U našem primjeru, neka je to neka Toma (T). B sada može da ode kod T i preda mu svoj javni ključ. T ovaj ključ stavlja u kovertu i u nju ubacuje pisamce sa imenom B, pečati kovertu, a zatim na nju stavlja svoj potpis. Ukoliko A sada dobije jednu ovakvu kovertu, može da bude sigurna da je ključ u koverti zaista javni ključ koji pripada B, budući da je kovertu pečatirao i potpisao T, a sadrži pisamce sa imenom B. Ovakva ovjerena koverta koja sadrži javni ključ i ime njegovog vlasnika se naziva *sertifikat*. U ovom slučaju, T se naziva *sertifikacionim tijelom* (eng. certificate authority).

Ako je T previše zauzet i ne može da stigne da ispotpisuje i ispečatira sve zahtjeve koji mu pristignu, a pri tom vjeruje Stanoju (S), može dio posla da prebaci na njega. Da bi i pečati koji pripadaju S bili vjerodostojni, T na više listova svojim pečatom i potpisom ovjerava potpis i pečat koji pripadaju S. Na ovaj način, i S postaje sertifikaciono tijelo. Sada B, umjesto da se obraća T, može da ode kod S. S će, kao i T, ubaciti ključ u kovertu, zajedno sa pisamcem na kojem stoji ime B. Međutim, S ne može sad samo da zatvori i pečatira kovertu. Da bi mu A vjerovala, prvo mora da ubaci jedan od listova koje je T ovjerio za njega, i tek tada zatvara kovertu.

Kada primi i otvori kovertu, A vidi pečat koji pripada S, ime B na pisamcu, te ključ, ali još uvijek ne može biti sigurna da je to zaista ključ koji pripada B. Da bi se uvjerila u to, prvo mora da nađe list koji je T ovjerio za S, a zatim da uporedi pečat i potpis na koverti koji je udario, odnosno potpisao S sa onim koji je T ovjerio za njega. Jasno, ovakav scenario se sada može nastaviti i dalje, gdje S potpisuje pečat i potpis koji pripadaju R, R potpisuje pečat i potpis za Q, Q za P i tako dalje.

2.3 Kriptografski protokoli

Kao što smo već pomenuli na kraju poglavlja 2.1, komponente distribuiranog sistema koje žele da osiguraju svoju komunikaciju moraju na ispravan način koristiti kriptografske al-

goritme. Kao prvi korak, moraju se dogovoriti oko nekog skupa pravila kojim će odrediti:

1. tačan način na koji treba koristiti kriptografske algoritme
2. na koji način treba da razmjenjuju poruke.

Ovakav skup pravila se naziva *kriptografski protokol*.

Jedan uobičajen način za specifikaciju kriptografskih protokola je takozvana *Alice-Bob notacija*, u kojoj se prvo navode pošiljalac i primalac poruke, a zatim i sam sadržaj poruke. Pregled notacije daje tabela 2.1.

Notacija	**Opis**
(m_1, m_2)	Poruka dobijena spajanjem poruka m_1 i m_2
$\{m\}_X$	Šifrovanje poruke m ključem X
$PubK\_X$	Javni ključ učesnika X
$sign_X(m)$	Poruka m, potpisana od strane učesnika X
$signOnly_X(m)$	Potpis poruke m od strane učesnika X (ne uključuje samu poruku m)
$X \rightarrow Y : m$	Poruka m, poslana od strane X za Y

Tabela 2.1: Notacija za opis protokola

Daćemo sada i opis jednog poznatog kriptografskog protokola koristeći ovu notaciju. U pitanju je Needham-Schoeder protokol [NS78], koji u originalnoj verziji ima sedam poruka; mi ćemo ovdje koristiti skraćenu verziju protokola, adaptiranu iz opisa u [Low96b], koja se sastoji od tri poruke. Protokol se odvija između dva učesnika, nazvana A i B. Korisnik A se naziva i *inicijatorom* (eng. initiator) protokola, a korisnik B *odgovaračem* (eng. responder). Učesnici protokola se ponekad nazivaju i *agentima*.

$$\begin{array}{lll} \text{P1.} & A \rightarrow B: & \{N_A, A\}_{PubK\_B} \\ \text{P2.} & B \rightarrow A: & \{N_A, N_B\}_{PubK\_A}) \\ \text{P3.} & A \rightarrow B: & \{N_B\}_{PubK\_B} \end{array}$$

Slika 2.3: Alice-Bob notacija Needham-Schroeder protokola

Opis protokola je dat na slici 2.3. Na početku, A šalje poruku za B. Sadržaj poruke čini slučajan broj N_A, spojen sa imenom učesnika A, a zatim šifrovan javnim ključem korisnika B. Korisnik B sada može da, koristeći svoj tajni ključ, dešifruje ovu poruku, pročita broj N_A, a zatim i korisniku A pošalje drugu poruku, šifrovanu njegovim javnim ključem. Poruka se sastoji iz pročitanog broja N_A, i novog, slučajno generisanog broja nazvanog N_B. Sada A može da dešifruje ovu poruku svojim tajnim ključem, pročita broj N_B, i vrati ga korisniku B u trećoj poruci, ovaj put šifrovanog javnim ključem korisnika B. Svrhu protokola možemo ugrubo definisati na sljedeći način: A i B trebaju razmijeniti brojeve N_A i N_B, a da oni pri tom ostanu nepoznati potencijalnim napadačima.

2.4 Sigurnosni zahtjevi

Većina zahtjeva koje kriptografski protokoli treba da ispoštuju može se podijeliti u nekoliko vrsta.

- **Povjerljivost (tajnost)** - sprečavanje saznavanja pojedinih informacija od strane neželjenih strana u protokolu (ili spoljnih napadača). Povjerljivost je neophodna za ostvarenje privatnosti korisnika protokola, kao i za zaštitu osjetljivih podataka. Zahtjev tajnosti se obično specifikuje nekom porukom u protokolu i skupom učesnika protokola kojima je dozvoljeno saznavanje ove poruke. Neophodan (ali ne i dovoljan) uslov za ostvarivanje tajnosti je šifrovanje podataka, koristeći sigurne kriptografske algoritme (tj. algoritme koji se ne mogu lako razbiti).

- **Autentifikacija** - utvrđivanje identiteta nekog od učesnika protokola. Ovim se sprečava da se jedna strana u protokolu (ili spoljni napadač) lažno predstavlja kao neka druga strana. Autentifikacija se obično definiše između dva učesnika u protokolu. Cilj uspostavljanja autentifikacije je da se obezbijedi da svi pošteni učesnici u protokolu mogu biti sigurni da zaista komuniciraju sa onim sa kime i misle da komuniciraju. Autentifikacija se obično postiže korišćenjem digitalnih potpisa i/ili jednokratnih, kriptografski sigurnih slučajnih brojeva (eng. nonce).

  Pojam autentifikacije nije sasvim dobro definisan, i ponekad se koristi sa različitim značenjima. Ubrzo ćemo dati i preciznije definicije koje nas interesuju.

- **Integritet podataka** - sprečavanje izmjene poruka u toku njihovog prenosa preko komunikacionih linkova, tj. neovlaštene izmjene podataka. Često se osigurava korišćenjem digitalnih potpisa.

- **Neporecivost** - pružanje dokaza nekoj od strana u protokolu o učešću nekog od drugih učesnika u izvršenju protokola. Nakon što se jedna sesija protokola završi, poželjno je da nijedan učesnik ne može da porekne da je zaista i učestvovao u njoj. I ova osobina se obično ostvaruje pomoću digitalnih potpisa.

Kao što smo već naveli, pojam autentifikacije se ponekad koristi sa različitim značenjima. Lowe u svom radu [Low96a] daje više različitih definicija autentifikacije, i određuje njihovu hijerarhiju. Neke od ovih definicija kombinuju integritet podataka i autentifikaciju. Mi ćemo sada navesti dvije koje nas interesuju. U njima se koncentrišemo na dva učesnika protokola, koje označavamo sa A i B. Svaki od učesnika može da igra neku od *uloga* u protokolu; svaka uloga odgovara jednoj *vrsti* učesnika u protokolu. Tako u Needham-Schroeder protokolu razlikujemo dvije različite uloge: ulogu inicijatora i ulogu odgovarača. U slučaju ovog protokola, moguće je i da isti učesnik igra i više od jedne uloge, tj. da ponekad bude inicijator, a ponekad odgovarač. U nekim drugim protokolima to nije slučaj, tj. jedan učesnik uvijek igra samo jednu vrstu uloge.

Definicija 1. *Kažemo da protokol učesniku A (u ulozi U_1) garantuje* ***neinjektivnu saglasnost*** *sa učesnikom B (u ulozi U_2) na skupu ds (gdje je ds skup slobodnih promjenjivih iz opisa protokola) ako, kada god A (u ulozi U_1) završi jedno izvršavanje protokola, prividno*

sa učesnikom B (u ulozi U_2), tada je B nekad prethodno izvršio protokol u ulozi U_2, prividno sa A (u ulozi U_1), i oba učesnika su se usaglasila oko vrijednosti svih promjenjivih iz skupa ds.

Primijetimo da ova definicija ne garantuje 1-1 odnos između izvršenja protokola od strane A i B; sasvim je moguće da je A dva puta izvršio protokol, dok je B to učinio samo jednom. Otuda i potiče naziv neinjektivna saglasnost. Osim toga, primjećujemo da ova definicija garantuje i integritet podataka u skupu ds.

Definicija 2. *Kažemo da protokol učesniku A (u ulozi U_1) garantuje **saglasnost** (ili, preciznije, **injektivnu saglasnost**) sa učesnikom B (u ulozi U_2) na skupu ds (gdje je ds skup slobodnih promjenjivih iz opisa protokola) ako mu garantuje neinjektivnu saglasnost sa učesnikom B (u ulozi U_2) na skupu ds, i, pri tom, svako izvršenje protokola od strane A odgovara **jedinstvenom** izvršenju protokola od strane B.*

Ponekad se termini autentifikacija i saglasnost koriste kao sinonimi; tada obično kažemo da korisnik A autentifikuje korisnika B na skupu ds. Za neinjektivnu saglasnost se često koristi i termin *slaba autentifikacija*, dok se za injektivnu saglasnost koristi i termin *jaka autentifikacija*. Osim toga, budući da je najčešće jasno koje uloge igraju učesnici u protokolu, obično ih nećemo eksplicitno navoditi.

2.5 Napadi na kriptografske protokole

Sada možemo da definišemo i šta je *napad* na kriptografski protokol: jednostavno, napad je svako kršenje sigurnosnih zahtjeva. Pokazaćemo sada dva primjera napada na protokole. Prvi primjer se odnosi na verziju Needham-Schroeder protokola sa tri poruke, opisanu na slici 2.3.

Da bismo objasnili napad, moramo precizirati i svrhu protokola, odnosno njegove sigurnosne zahtjeve. Protokol služi da korisniku B omogući saglasnost sa korisnikom A, na skupu vrijednosti $\{N_A, N_B\}$; dodatno, možemo zahtijevati i tajnost poruka iz ovog skupa. Naoko, protokol bi trebalo da pruži ovakvu garanciju - niko osim A ne bi trebalo da može da pročita vrijednost N_B koju B šalje u drugoj poruci protokola. Osim toga, A bi trebalo da je saglasan sa korišćenjem vrijednosti N_A, budući da ju je on sam i izabrao.

Napad na ovaj protokol se sastoji od dva paralelna izvršavanja protokola, koja ćemo označiti sa α, odnosno β. Jedno izvršavanje protokola ćemo nazivati i *sesija* protokola. U sesiji α, korisnik A pokušava da izvrši protokol sa nepoštenim učesnikom kojeg ćemo označiti sa I. U sesiji β, nepošteni učesnik izvršava protokol sa korisnikom B, lažno se predstavljajući kao A. Napad je predstavljen na slici 2.4. U opisu napada, oznake poruka se sastoje od oznake sesije i rednog broja poruke, pa tako $\beta.2$ označava drugu poruku sesije β. I_A označava napadača koji se lažno predstavlja kao A, ili presreće poruku koja je namijenjena za A.

U poruci $\alpha.1$ A počinje izvršavanje protokola sa napadačem I, šaljući slučajni broj N_A šifrovan javnim ključem napadača I. I zatim koristi ovu poruku, tako što se lažno predstavlja kao korisnik A i u poruci $\beta.1$ započinje protokol sa korisnikom B, koristeći broj

$$
\begin{array}{lll}
\text{P}\alpha.1. & A \to I: & \{N_A, A\}_{PubK\_I} \\
\text{P}\beta.1. & I_A \to B: & \{N_A, A\}_{PubK\_B} \\
\text{P}\beta.2. & B \to I_A: & \{N_A, N_B\}_{PubK\_A} \\
\text{P}\alpha.2. & I \to A: & \{N_A, N_B\}_{PubK\_A} \\
\text{P}\alpha.3. & A \to I: & \{N_B\}_{PubK\_I} \\
\text{P}\beta.3. & I_A \to B: & \{N_B\}_{PubK\_B}
\end{array}
$$

Slika 2.4: Napad na Needham-Schroeder protokol sa 3 poruke

N_A koji je dobio u prvoj poruci. B odgovara odabirom novog broja N_B i njegovim slanjem u poruci $\beta.2$. I sada ne može da dešifruje ovu poruku, ali može da iskoristi A kao *proročište* (eng. oracle), prosljeđujući mu poruku kao $\alpha.2$. Primijetimo da A očekuje poruku upravo ovakvog oblika. Stoga dešifruje primljenu poruku i šalje broj N_B napadaču u poruci $\alpha.3$. Napadač sada u poruci $\beta.3$ jednostavno prosljeđuje ovaj broj korisniku B, čime se završava sesija β. Sada B vjeruje da je uspješno izvršio protokol sa A.

Ovim je sada narušen zahtjev saglasnosti iz definicija 1 i 2. Mada je A zaista nekada izvršio protokol, i zaista se složio oko vrijednosti za N_A i N_B, i štaviše, jednoj sesiji učesnika B odgovara jedna jedinstvena sesija učesnika A, problem leži u činjenici da A nije izvršavao protokol sa B, čime je prekršena definicija saglasnosti. Osim toga, napadač saznaje i vrijednost N_B, koja je trebala da ostane tajna.

Primijetimo da je ovaj napad izvršen, a da nijedan kriptografski algoritam korišćen u protokolu nije razbijen. Ovo je razlog zbog kojeg kažemo da kriptografija nije dovoljna da garantuje sigurnost.

U ovom slučaju, lako je popraviti protokol i spriječiti ovaj napad. Dovoljno je zahtijevati da se u drugu poruku ubaci identitet odgovarača. Ispravljeni protokol je poznat kao Needham-Schroeder-Lowe protokol i prikazan je na slici 2.5.

$$
\begin{array}{lll}
\text{P1}. & A \to B: & \{N_A, A\}_{PubK\_B} \\
\text{P2}. & B \to A: & \{N_A, N_B, B\}_{PubK\_A} \\
\text{P3}. & A \to B: & \{N_B\}_{PubK\_B}
\end{array}
$$

Slika 2.5: Needham-Schroeder-Lowe protokol

Opisani napad sada neće uspjeti, jer bi poruka $\beta.2$ sada postala:

$$\text{P}\beta.2. \quad B \to I_A: \quad \{N_A, N_B, B\}_{PubK\_A}$$

Napadač sada ne može da proslijedi ovu poruku korisniku A, jer ovaj očekuje poruku koja sadrži identitet napadača I, a ne korisnika B.

Sada ćemo pokazati još jedan napad, ovaj put na punu verziju ispravljenog Needham-Schroeder protokola, koja se sastoji od sedam poruka (ova verzija je takođe adaptirana iz [Low96b]). Protokol je prikazan na slici 2.6. Za razliku od skraćene verzije protokola, u ovoj se pretpostavlja da javni ključevi nisu razmijenjeni prije izvršenja protokola, već se dobijaju upitom ka sertifikacionom tijelu S; u originalnoj terminologiji, S se naziva „server". Tako poruke 1 i 4 predstavljaju upite, a poruke 2 i 5 odgovore sertifikacionog tijela.

P1. $A \to S: \quad B$
P2. $S \to A: \quad sign_S(PubK\_B, B)$
P3. $A \to B: \quad \{N_A, A\}_{PubK\_B}$
P4. $B \to S: \quad A$
P5. $S \to B: \quad sign_S(PubK\_A, A)$
P6. $B \to A: \quad \{N_A, N_B, B\}_{PubK\_A}$
P7. $B \to A: \quad \{N_B\}_{PubK\_B}$

Slika 2.6: Needham-Schroeder-Lowe protokol sa 7 poruka

Ovaj napad spada u jednu posebnu klasu napada, tzv. *napade greškom tipa* (eng. type flaw attacks). Napad greškom tipa je napad na protokol u kojem se polje poruke koje je originalno trebalo imati jedan tip kasnije interpretira kao polje nekog drugog tipa. Napad na originalnu verziju protokola je opisan u [Mea96], ali se isti napad može primijeniti i ovdje (slika 2.7).

Pα.3. $I_A \to B: \quad \{N_I, A\}_{PubK\_B}$
Pα.4. $B \to S: \quad A$
Pα.5. $S \to B: \quad sign_S(PubK\_A, A)$
Pα.6. $B \to I_A: \quad \{N_I, N_B, B\}_{PubK\_A}$
Pβ.3. $I_{(N_B,B)} \to A: \quad \{N_I, (N_B, B)\}_{PubK\_A}$
Pβ.4. $A \to S: \quad (N_B, B)$
Pα.7. $I_A \to B: \quad \{N_B\}_{PubK\_B}$

Slika 2.7: Napad na Needham-Schroeder-Lowe protokol sa 7 poruka

I ovaj napad koristi dvije sesije protokola, čije su poruke označene sa α, odnosno β. Koristimo iste oznake kao u opisu prethodnog napada. Cilj napadača je ponovo da se predstavi kao A, ovaj put u toku sesije α. Kada B pošalje poruku $\alpha.6$, kojom pokušava da autentifikuje A, napadač prosljeđuje ovu poruku upravo korisniku A. Kako je ovo prva poruka koju A dobija, on je interpretira kao početak nove sesije protokola. Stoga uzima polje (N_B, B) kao identitet korisnika i pokušava da od servera zatraži javni ključ „korisnika" (N_B, B) šaljući njegov identitet serveru. Ovo dozvoljava napadaču da sazna N_B, i tako uspješno odgovori na poruku $\alpha.6$, čime B postaje uvjeren da je protokol izvršen sa A.

Većina formalnih modela za verifikaciju protokola ne obrađuje mogućnost ovakvih napada, koristeći tzv. *jaku apstrakciju tipova*. U ovoj apstrakciji, sve poruke se smatraju za dobro tipizovane, a učesnici protokola bez ikakvih informacija mogu odrediti tačne tipove poruka. U ovakvom modelu, ne postoji način da A greškom interpretira polje (N_B, B) kao identitet korisnika, jer on na „magičan" način zna da su to u stvari dva polja. Čini se da ovakvi modeli onda nikako ne mogu biti ispravni, jer ne mogu da otkriju napade poput ovog upravo prikazanog. Srećom, svi napadi greškom tipa se mogu jednostavno eliminisati u toku same implementacije protokola, primjenom odgovarajuće *sheme označavanja poruka* [HLS00]; jedna takva shema će biti opisana u poglavlju 4.3. Ova činjenica nam dozvoljava da se u radu skoncentrišemo na modele verifikacije koji koriste jaku apstrakciju tipova.

S druge strane, postoje i modeli koji ne ovu apstrakciju ne koriste; to su tzv. *netipizovani*

modeli. Protokol za koji se dokaže da je siguran u ovakvom modelu može se bezbjedno implementirati i bez sheme označavanja poruka. Ipak, sigurnost u netipizovanim modelima veoma je krhka. I najmanja promjena u protokolu, poput zamjene redoslijeda elemenata u poruci ili dodavanja novih dijelova u poruku, može da dovede do neke vrste sigurnosnog propusta u protokolu.

Glava 3

Dizajn protokola

Cilj ove glave je da opiše novi sistem za plaćanje, nazvan IPS (Internet Payment System), kao i da objasni razloge nastanka sistema, te odluke koje su donesene pri njegovom kreiranju. Glava započinje kratkim pregledom osobina elektronskih sistema za plaćanje, te daje jednu moguću klasifikaciju ovih sistema. Poglavlje 3.2 zatim daje pregled postojećih sistema za plaćanje, sa nešto detaljnijim opisom tri veoma poznata sistema. Na osnovu ovog pregleda, poglavlje 3.3 zatim prikazuje ciljeve IPS-a. Konačno, glava završava detaljnim opisom IPS-a, sa posebnim akcentom na njegov protokol za plaćanje, koji i čini „kičmu" sistema.

3.1 Elektronski sistemi plaćanja

U svakodnevnom životu, ljudi koriste veliki broj različitih sistema za plaćanje za različite svrhe, poput kupovine namirnica u prodavnici, plaćanja računa u banci ili kupovine knjiga putem Veba. Sredstva plaćanja su takođe raznorodna, od kovanica, papirnog novca i čekova, do kreditnih i debitnih kartica, elektronskih bankovnih transfera i drugih.

Ovdje ćemo se ograničiti na elektronske sisteme plaćanja koji se mogu posmatrati kao distribuirani sistemi. Jedna klasifikacija ovih sistema je data u [PW96]; mi ćemo je ovdje ukratko predstaviti.

3.1.1 Dijelovi sistema

Kao što im i samo ime govori, svrha sistema za plaćanje je upravo plaćanje, ili, opštije, prenos novca od jednog lica ka drugom. Riječ „novac" u ovom slučaju znači stvarni novac, nezavisan od samog sistema za plaćanje. Radi preciznosti, ponekad ćemo stoga govoriti o *eksternom obliku novca*.

Sistem za plaćanje se može predstaviti kao jedan sloj distribuiranog sistema. Ispod ovog sloja, nalazi se sloj za komunikaciju. Na sloju iznad se nalaze entiteti koje nazivamo korisnicima ili učesnicima sistema. Korisnici mogu biti fizička lica, poput kupca koji želi

da nešto kupi svojom kreditnom karticom, ali i drugi elektronski sistemi, poput elektronske prodavnice koja služi za obradu ponude, naručivanja i slanja robe. Logičke tačke na kojima korisnici mogu da pristupe sistemu nazivaju se priključne tačke. Ističu se četiri glavna korisnika i njihove odgovarajuće priključne tačke.

- **Kupac** - korisnik koji želi da potroši novac, obično u zamjenu za neko dobro. Koristićemo i nazive klijent ili vlasnik kartice.
- **Prodavac** - korisnik koji želi da primi novac. Zauzvrat, korisnik obično pruža neko dobro ili uslugu.
- **Banka kupca** - korisnik kod kojeg kupac ima otvoren račun i koji pruža mogućnost kupcu da konvertuje eksterne oblike novca u sistem za plaćanje.
- **Banka prodavca** - služi prodavcu da konvertuje plaćeni iznos u eksterne oblike novca.

Unutar distribuiranog platnog sistema, postoji po jedan entitet koji odgovara svakoj od priključnih tačaka. Ovi entiteti mogu biti računarski programi, fizički uređaji poput pametnih kartica, ali i drugi distribuirani sistemi. Entiteti komuniciraju preko komunikacionih linkova.

- **Entitet klijenta** - takozvani „elektronski novčanik", često prisutan u vidu softvera ili pametne kartice.
- **Entitet prodavca** - najčešće serverski program, ili specijalizovani „point-of-sale" terminal.
- **Entitet banke kupca i entitet banke prodavca** - softverske infrastrukture banaka su često komplikovani distribuirani sistemi.

U mnogim platnim sistemima, entiteti banke kupca i banke prodavca su zamijenjeni jedinstvenim entitetom zvanim *platna kapija* (eng. payment gateway), koja ima pristup zatvorenoj bankarskoj mreži. Osim toga, neki sistemi uključuju i takozvanu *povjerljivu treću stranu* (eng. thrusted third party), koja se uključuje u slučaju sporova o izvršenju protokola.

3.1.2 Događaji u sistemu

Ulaz u sistem se prima sa raznih priključnih tačaka. Osnovni ulazni događaji su sljedeći:

- **plati** - ulaz od strane kupca, kojim on odobrava uplatu. Parametri obično uključuju iznos, opis narudžbe, te ime prodavca;
- **primi** - ulaz od strane prodavca koji služi za prihvatanje uplate, obično povezan sa iznosom i opisom;

- **dozvoli** - ulaz od strane klijentove banke, kojim se dozvoljava prenos određenog iznosa novca sa kupčevog računa na račun prodavca. Banka obično prvo provjerava da li je kupac prethodno konvertovao dovoljan iznos eksternog novca;
- **ospori** - ulaz od strane jedne od tri gore pomenuta entiteta, kojim se osporava iznos, opis, ili učesnici u transakciji. Postoji više vrsta sporova; ubrzo ćemo ih detaljnije objasniti.

Osnovni izlazni događaji su:

- **plaćeno** - signalizira kupcu da je uplata uspješno izvršena;
- **primljeno** - isti signal, samo za prodavca;
- **dodaj** - signal banci prodavca da treba da doda odgovarajući iznos novca na račun prodavca;
- **skini** - signal banci kupca da se iznos treba skinuti sa kupčevog računa;
- **prenesi** - signal u sistemima sa platnom kapijom, koji zamjenjuje prethodna dva signala;
- izlazi povjerljive treće strane vezani za sporove. Sporove mogu započeti kupac i prodavac, oko sljedećih događaja:
  - kupac može osporiti signal **skini**, tvrdeći da nije dao dozvolu banci da skine novac sa računa;
  - kupac može tvrditi da je prodavac primio izlaz **primljeno**, na primjer da bi pokazao da je prodavac sada obavezan da isporuči robu;
  - prodavac može tvrditi da je njegova banka primila signal **dodaj**, odnosno da je bila obavezna da prebaci novac na njegov račun.

  Povjerljiva treća strana zatim daje izlaz **da** ili **ne**, u zavisnosti od vrste spora i pretpostavke da li se osporeni izlaz zaista i desio za ,,poštenog" učesnika u protokolu (tj. učesnika koji se pridržava pravila protokola). Na primjer, u prvom sporu bi odgovor bio **ne** akko bi u toku osporenog izvršavanja protokola ,,poštena" banka zaista primila izlaz **skini**.

Više povezanih događaja u sistemu se obično grupišu u transakcije, kao što su ,,plaćanje" ili ,,spor".

3.1.3 Tipovi sistema

Na osnovu vrste događaja koji pripadaju jednoj transakciji, možemo razlikovati tri vrste sistema za elektronsko plaćanje.

1. **Plati-odmah sistemi** su sistemi kod kojih je jedina transakcija za prenos novca transakcija za plaćanje. Ovdje se izlazi **skini**, **dodaj**, **prenesi**, **plaćeno** i **primljeno** dešavaju nakon odgovarajućih **plati**, **primi** i **dozvoli** ulaza kod kojih su parametri u saglasnosti. Ovakvi sistemi se nazivaju *čekovnim elektronskim sistemima* (eng. check-like), ili *sistemima zasnovanim na računima* (eng. account-based).

2. **Sistemi sa pretplatom** su sistemi koji imaju jednu dodatnu transakciju za prenos novca, koja prethodi transakciji plaćanja. Ova transakcija ima ulaz **podigni**, te izlaze **skini** i **podignuto**. Transakcija plaćanja kod ovih sistema zatim ili rezultuje ostatkom izlaza, ili samo izlazima **plaćeno** i **primljeno**. Prva vrsta sistema se naziva *deponuj-odmah* (deposit-now) sistemima, a druga *deponuj-kasnije* (deposit-later) sistemima. Deponuj-kasnije sistemi imaju još jednu dodatnu transakciju, zvanu *deponovanje*, gdje prodavac zahtijeva izlaze **dodaj** i **deponovano**, koristeći ulaz **deponuj**.

   Sistemi sa pretplatom se još nazivaju i *gotovinskim elektronskim sistemima* (eng. cash-like), ili *sistemima zasnovanim na žetonima* (eng. token-based).

3. **Provizorni sistemi plaćanja** podržavaju transakcije plaćanja koje ne rezultuju sigurnim izlazima. Izlazi **dodaj**, **skini**, **plaćeno** i **primljeno** su odgođeni. Prodavac kompletira plaćanje transakcijom deponovanja. U sistemu sa poništavanjem, kupac ima ulaz **provizorno-plati** u transakciji plaćanja, a u transakciji deponovanja mora da da jedan od ulaza **plati** ili **poništi**.

Pokušaćemo da detaljnije objasnimo razliku između prve dvije vrste sistema analogijom sa ,,fizičkim" sistemima plaćanja (dakle onima koji nisu elektronski), po kojima su uostalom i dobili imena. Kao ilustrativne scenarije iskoristićemo plaćanje čekovima (za plati-odmah sisteme) i plaćanje žetonima (za sisteme sa pretplatom). Naravno, treba imati na umu da je ovo ipak samo analogija - redoslijed ulaza i izlaza u pravim platnim sistemima ne mora da prati ovaj šablon. Za provizorne sisteme pak teško je dati neku analogiju, budući da su, kako im i ime kaže, obično u pitanju sistemi namijenjeni za neke druge svrhe, na koje je kasnije ,,nakalemljen" neki platni sistem.

1. U sistemima sa čekovima:

   (a) Kupac prodavcu daje ček (ulaz **plati**).

   (b) Prodavac nosi ček u svoju banku (ulaz **primi**).

   (c) Banka prodavca šalje ček banci kupca na odobravanje; po odobravanju, banka kupca daje ulaz **dozvoli**.

   (d) Sada se generišu svi izlazi sistema: banka kupca dobija izlaz **skini**, banka prodavca izlaz **dodaj**, prodavac izlaz **primljeno** a kupac izlaz **plaćeno**.

2. U sistemima sa žetonima:

   (a) Kupac odlazi kod blagajnika na kasu sa (eksternim) novcem i traži žetone (ulaz **podigni**); blagajnik uzima novac i predaje mu žetone (izlazi **skini** i **podignuto**). Ovo je transakcija za prenos novca.

(b) Sva kupovina se obavlja isključivo žetonima. Kupac odlazi kod prodavca i predaje mu odgovarajući žeton (moguće i više žetona, a moguća je i upotreba tzv. *djeljivih žetona*, kod kojih kupac može dobiti „kusur"). Ovo su ulazi **plati** i **primi**, koji trenutno rezultuju izlazima **plaćeno** i **primljeno**.

(c) Sada postoje dvije opcije: prodavac može trenutno unovčiti žetone kod blagajnika, što generiše izlaz **dodaj**. Ovo je slučaj u deponuj-odmah sistemima. Druga opcija je da žetone stavi u sef, i da ih kasnije odnese blagajniku (ulaz **deponuj**), koji stavlja novac na račun prodavca (izlazi **deponovano** i **dodaj**). Ovo je transakcija deponovanja u deponuj-kasnije sistemima.

3.2 Postojeća rješenja

U prošlosti je viđeno mnogo prijedloga za razne sisteme plaćanja, kako za čekovne, tako i za gotovinske elektronske sisteme. Istraživanje gotovinskih sistema je započelo radovima Chaum-a ([Cha83], [CFN89]), koji su kasnije komercijalno implementirani u sistemu DigiCash. Drugi poznati sistemi uključuju Brandsov sistem [Bra93], te sistem Okamota i Ohte [OO92], koji je prvi sistem koji nudi djeljivi novac, tj. kod kojeg žetoni mogu biti podijeljeni u više manjih žetona nakon podizanja.

Jedan od prvih čekovnih sistema je opisan u [Cha90], ali postoje i mnogi drugi. Među bitnijim sistemima se nalazi *iKP* familija protokola [Bel+00], dizajnirana od strane IBM-a 1995. godine. Sistem iKP je implementiran u probnoj fazi, i bio je prethodnik platnog sistema *SET*. SET ([MBW98], [Set]) su razvile vodeće kompanije za izdavanje kreditnih kartica, Visa i MasterCard, a razvoju su se priključile i kompanije kao što su IBM, Microsoft i Netscape. Protokol je u toku 90-ih godina prošlog vijeka imao veliki publicitet kao odobreni standard za kreditne kartice, ali nije doživio široku primjenu. Nakon neuspjeha SET-a, Visa je razvila novi protokol nazvan *3-D Secure* [JM03]. Pošto su ova dva protokola razvijena od strane vodećih kompanija u industriji, posvetićemo im posebnu pažnju u poglavljima 3.2.2 i 3.2.3.

Ipak, i pored postojanja ovolikog broja sistema za plaćanje, najrašireniji vid zaštite prilikom plaćanja kreditnim karticama putem Interneta je i dalje *SSL/TLS* - protokol koji je namijenjen zaštiti komunikacionih linkova, a ne elektronskom plaćanju kao takvom.

3.2.1 SSL/TLS

Uobičajen način zaštite komunikacije na Internetu su protokoli *SSL* (Secure Sockets Layer) i *TLS* (Transport Layer Security) [Res01]. SSL je 1994. godine kreirala kompanija Netscape. Primarna namjena protokola je bilo obezbjeđenje sigurne komunikacije između Veb pregledača i Veb servera. Godinu dana kasnije, Internet Engineering Task Force (IETF) je uvela protokol nazvan TLS. Mada između ova dva protokola postoje interne razlike, namjena i mehanizmi su im gotovo identični, pa ćemo ih u radu nazivati zajedničkim terminom SSL/TLS.

SSL/TLS osigurava vezu između dvije tačke, na sloju sesije (u OSI modelu komunikacionih sistema). Skup njegovih podržanih kriptografskih algoritama sadrži kako simetrične, tako i asimetrične mehanizme. Asimetrična kriptografija i potpisi se ostvaruju pomoću RSA, DSA i Diffie-Hellman algoritma za razmjenu ključeva, dok skup algoritama za simetrično šifrovanje uključuje AES, Camelia, DES, Triple-DES, IDEA, RC4 i RC2.

Budući da SSL/TLS pruža sigurnost na nivou sesije, moguće ga je koristiti u kombinaciji sa drugim protokolima višeg nivoa. Jedna od najznačajnijih takvih kombinacija je HTTPS (HTTP + SSL/TLS) protokol, kojim se omogućava sigurna komunikacija putem Veba. U ovom slučaju se vlasništvo nad javnim ključevima, koje je od ključnog značaja za sigurnost SSL/TLS (pa time i HTTPS) sistema, zasniva na mehanizmu sertifikata (poglavlje 2.2). Postoji više korjenih sertifikacionih tijela, čiji su sertifikati već ugrađeni u Veb pregledače. Ova tijela (ili, u pravilu, njihova podtijela) zatim uz naknadu izdaju potpisane sertifikate korisnicima HTTPS sistema koji to žele.

Mada SSL/TLS nije dizajniran kao sistem za plaćanje, sasvim je moguće iskoristiti ga u tu svrhu. Štaviše, plaćanje kreditnim karticama putem HTTPS protokola je i najpopularniji način kupovine danas na Internetu. Glavni razlozi za ovo uključuju:

- transparentnost - budući da SSL/TLS pruža sigurnost na nivou sesije, njegova prisutnost je potpuno nevidljiva, kako za Veb prodavnicu prodavca, tako i za kupca. Ovo je posebno važno za prodavca jer je integrisanje SSL/TLS-a sa postojećim sistemima veoma jednostavno; sve što je potrebno je instaliranje SSL/TLS sertifikata (poglavlje 2.2) i prelazak sa HTTP na HTTPS protokol;
- lakoću korišćenja za kupce - SSL/TLS je već ugrađen u gotovo sve Veb pregledače, pa stoga ne postoji potreba za instalacijom dodatnog softvera;
- pokretljivost kupaca - SSL/TLS ne postavlja nikakva ograničenja na lokaciju kupca; dovoljno je samo da posjeduje računar, ili neki drugi uređaj opremljen Veb pregledačem;
- malu kompleksnost - sistem je relativno jednostavan, zbog čega je njegov uticaj na brzinu izvršenja transakcija minimalan.

Međutim, SSL/TLS ima i neke ozbiljne nedostatke kada se koristi kao protokol za plaćanje. Budući da je zasnovan na nezavisnim konekcijama između dvije tačke, SSL/TLS nije sasvim adekvatan za komunikaciju više učesnika, ili indirektnu komunikaciju. U kontekstu našeg scenarija za plaćanje (slika 1.1), ova neadekvatnost se reflektuje u sljedećim nedostacima:

- prodavac ne može pouzdano utvrditi identitet kupca. U slučaju kada kupac koristi kradenu kreditnu karticu za plaćanje, prodavci su odgovorni za tzv. ,,card not present" povrate novca. Mada SSL/TLS pruža mogućnost dvostrane autentifikacije, pomoću tzv. klijentskih sertifikata, takvi sertifikati nisu obavezni i rijetko se koriste. Štaviše, čak i ukoliko klijent posjeduje sertifikat, on ne mora neophodno imati veze sa njegovom kreditnom karticom, što znači da klijent ne mora biti ovlašćen za korišćenje kartice;

- SSL/TLS osigurava samo vezu između kupca i prodavca. Prodavac može da vidi sve detalje plaćanja (broj kreditne kartice i slično). SSL/TLS ne može da garantuje da prodavac neće zloupotrijebiti ove podatke, niti ih može zaštititi od napada od strane trećih lica dok su smješteni na serveru prodavca;
- SSL/TLS sam po sebi ne može obezbijediti neporecivost (vidjeti poglavlje 2.4).

U smislu klasifikacije iz odjeljka 3.1.3, SSL/TLS je provizorni platni sistem sa poništavanjem. Kupac mora da pregleda svoj bankovni izvod (koji se obično izdaje na kraju svakog mjeseca), a zatim poništi bilo kakve neispravne transakcije.

3.2.2 SET

SET (Secure Electronic Transaction) je otvoreni standard za zaštitu privatnosti elektronskih transakcija, te osiguravanje njihove autentičnosti ([Set], [MBW98]). Za razliku od SSL/TLS-a, SET je upravo dizajniran kao sistem za plaćanje, i to kao čekovni elektronski sistem (u smislu klasifikacije iz odjeljka 3.1.3). SET transakcije plaćanja uključuju tri učesnika, kupca, prodavca i platnu kapiju. SET omogućava autentifikaciju svih učesnika transakcije, korišćenjem digitalnih sertifikata (poglavlje 2.2). Ove sertifikate izdaje povjerljiva treća strana poznata kao *autoritet za sertifikaciju* (eng. Certification Authority, ili skraćeno CA). CA garantuje za identitet vlasnika sertifikata. Budući da i kupac mora posjedovati sertifikat, to znači da se on mora registrovati kod odgovarajućeg CA prije nego što može učestvovati u SET transakcijama.

Ne postoji jedinstven SET protokol; SET je sistem koji se sastoji od više protokola, uključujući protokole za registraciju, zahtjev za plaćanje, odobrenje plaćanja i mnoge druge, manje protokole (npr. za obradu grešaka). Protokoli koriste kako simetričnu kriptografiju, u vidu DES algoritma, tako i asimetričnu, u vidu RSA algoritma.

Najznačajniji dio sistema čine protokoli za zahtjev za plaćanjem i odobrenje plaćanja, koji čine jednu logičku cjelinu koju možemo nazvati SET protokolom za plaćanje. SET specifikacija definiše veći broj varijanti ovog protokola; najznačajnija (tačnije, najsigurnija) je varijanta opisana u [BMP01] koju ćemo ovdje ukratko predstaviti. U ovoj varijanti, prodavac ne saznaje podatke o plaćanju, a platna kapija ne saznaje podatke o narudžbi. Ovo se postiže pomoću tzv. *dvostrukog potpisa*: kupac kreira heš vrijednost za podatke o narudžbi, odnosno podatke o plaćanju, a zatim potpisuje par ovih vrijednosti. Ostalim stranama je zatim dovoljno dati na uvid ovaj potpis, vrijednost koju smiju da vide i heš vrijednosti koja im je skrivena, da bi mogli da se uvjere da barataju istim vrijednostima kao i kupac. Protokol se zatim sastoji od šest osnovnih koraka.

1. Kupac šalje identifikator transakcije i slučajno generisani broj.
2. Prodavac potvrđuje prijem brojeva potpisom, te u poruku uključuje i svoj par identifikatora i slučajnog broja.
3. Kupac prodavcu šalje dvostruke potpise, jedan za prodavca, drugi za platnu kapiju.

4. Prodavac prosljeđuje kapiji podatke o plaćanju.
5. Kapija provjerava podatke i odgovara prodavcu.
6. Prodavac prosljeđuje odgovor kupcu.

Prednosti SET-a su sljedeće:

- ispunjava sljedeće osnovne sigurnosne zahtjeve (vidjeti poglavlje 2.4): povjerljivost, autentifikaciju, te integritet podataka. Ispunjavanje ovih zahtjeva je potvrđeno pomoću više dokaza zasnovanih na formalnim metodama;
- u uobičajenoj varijanti protokola za plaćanje, SET ne dozvoljava prodavcima pristup kupčevim podacima za plaćanje, budući da su ovi podaci šifrovani javnim ključem platne kapije;
- da bi se osigurala privatnost prodavca, SET ne dozvoljava platnoj kapiji pristup podacima o narudžbi.

Uprkos ovim prednostima, SET nije uspio da se izbori za značajniji udio na tržištu. Jedan od najvažnijih, ako ne i najvažniji uzrok za to je što je upotreba SET-a bila relativno komplikovana za potencijalne kupce:

- kupci su morali instalirati dodatni softver, koji je u stanju da izvršava SET transakcije;
- prije učešća u SET transakcijama, kupac je morao da se registruje i dobije ispravan digitalni sertifikat.

Zajedno, ove dvije prepreke su stvarale i treću: jedino mjesto sa kojeg je kupac mogao da obavlja kupovinu je bio njegov lični računar koji je već pripremljen za SET transakcije.

Osim ovih, SET je imao i neke druge mane:

- sistem je bio veoma komplikovan, sa velikim brojem posebnih slučajeva. Dokumentacija je bila izrazito obimna (gotovo 1000 stranica), tako da je implementacija sistema predstavljala veoma težak i komplikovan zahvat. Ova činjenica je loše uticala na prijem SET-a kod banaka i prodavaca;
- integracija SET-a sa postojećim sistemima je za prodavce bila znatno komplikovanija od njihove adaptacije za SSL/TLS. Osim toga, prodavci su morali imati račune u bankama koje su podržavale SET transakcije;
- mada je sistem dizajniran da obezbijedi sigurnost, SET takođe ima određene sigurnosne propuste. Iako sistem u uobičajenoj varijanti protokola za plaćanje ne dozvoljava prodavcu uvid u kupčeve podatke za plaćanje, postoje i varijante u kojima to nije slučaj, baš kao kod SSL/TLS-a. Štaviše, varijantu koja će se koristiti bira sam prodavac [FK00]. Razlog ovakvog dizajna nije sasvim jasan, ali vjerovatno leži u nastojanju da se pruži neka vrsta kompatibilnosti sa postojećim sistemima zasnovanim na SSL/TLS. Osim ovog, postoje i neki drugi, manji propusti [BMP05].

3.2.3 3-D Secure

Nakon što SET nije uspio da ostvari veći udio na tržištu, Visa je promovisala novi sistem, nazvan *3-D Secure*. Ovaj sistem je i danas prisutan i široko rasprostranjen, samo pod nazivom *Verified by Visa*, odnosno *MasterCard SecureCode*. 3-D Secure ima uobičajena četiri osnovna učesnika. Banka kupca se u kontekstu 3-D Secure naziva *izdavač*, a ulogu banke prodavca preuzima platna kapija. Veze između pojedinih učesnika su nazvane *domenima*. U ovoj terminologiji postoje tri domena, od čega i potiče ime 3-D Secure: *domen izdavača* (eng. issuer domain), koji obuhvata vezu između izdavača i kupca, *domen sticanja* (eng. acquirer domain), koji obuhvata vezu između prodavca i platne kapije, te *domen interoperabilnosti* (eng. interoperability domain) koji spaja prethodna dva domena.

U 3-D Secure postoji šest osnovnih koraka.

1. Kupac prosljeđuje broj kartice prodavcu.
2. Prodavac pretražuje tzv. adresar izdavača, i pronalazi odgovarajućeg izdavača kartice.
3. Prodavac preusmjerava korisnika ka izdavaču i u poruci prosljeđuje sadržaj narudžbe.
4. Izdavač korisniku prikazuje detalje narudžbe i traži od njega da unese tajni kod za autentifikaciju.
5. Kupac unosi tajni kod.
6. Ako je prethodni korak bio uspješan, izdavač potpisuje poruku o uspjehu transakcije i preusmjerava kupca ka prodavcu, prosljeđujući poruku potvrde.
7. Prodavac verifikuje i pamti potpis, te obrađuje transakciju.

Pri tom, sve konekcije između pojedinih učesnika sistema su zaštićene SSL/TLS protokolom. Prednosti 3-D Secure sistema su sljedeće:

- za izvršavanje protokola kupcu nije potreban nikakav poseban softver, već je dovoljan običan Veb pregledač koji podržava SSL/TLS;
- sistem pruža autentifikaciju kupca. Verifikacijom potpisa u posljednjem koraku protokola, prodavci su, po ugovoru sa Visa-om, u značajnoj mjeri oslobođeni odgovornosti u slučaju prevare.

Ipak, 3-D Secure ima i značajne nedostatke:

- prodavac nema pristupa tajnom kodu kartice, ali ima pristup njenom broju i drugim detaljima. Mada prisustvo tajnog koda sprečava zloupotrebu kartice putem Interneta, moguća je njena zloupotreba u drugim okruženjima;

- izdavač kartice ima pristup svim detaljima transakcije (uključujući i sadržaj narudžbe);
- implementacija 3-D Secure protokola na strani prodavca zahtijeva instalaciju i licenciranje dodatnog softvera, zvanog Merchant Plug-In, kao i integraciju postojećih rješenja sa ovim softverom;
- preporučeni mehanizmi implementacije protokola su često na meti kritika [MA10], i omogućavaju napade na sistem; pri tom, kupac snosi znatno veću odgovornost u slučaju zloupotrebe njegove kartice nego prilikom korišćenja drugih metoda, poput SSL/TLS.

3.3 Ciljevi predloženog sistema za plaćanje

Na osnovu prednosti i mana postojećih sistema za plaćanje, opisanih u prethodnim poglavljima, definisan je skup ciljeva koje novi platni sistem treba da ispuni. Ovi ciljevi su zasnovani kako na sigurnosnim zahtjevima za efikasan sistem za plaćanje, tako i na zahtjevima za jednostavnost upotrebe od strane krajnjih korisnika. Cilj novog sistema je da se stvori:

- jednostavan, jeftin i siguran sistem za transakcije debitnim i kreditnim karticama između kupaca i prodavaca;
- sistem u kojem kupci ne moraju vršiti posebne pripreme prije nego što mogu učestvovati u transakcijama. Konkretno, ovo znači da kupci:
  - ne moraju instalirati nikakav dodatni softver za sigurno plaćanje
  - ne moraju obezbijediti digitalne sertifikate;
- sistem koji koristi sigurne kriptografske algoritme;
- sistem koji prodavcu ne dozvoljava uvid u podatke za plaćanje;
- sistem koji ne dozvoljava izdavačima kartica uvid u detalje narudžbi, i time štiti privatnost prodavaca i kupaca.

U ovom trenutku, još ne možemo precizno definisati šta podrazumijevamo pod „siguran sistem". Za ovo će biti neophodno da prvo pružimo pregled predloženog sistema.

3.4 Predloženi sistem za plaćanje - IPS

Na osnovu kriterijuma navedenih u prethodnom poglavlju, kreiran je sistem nazvan *IPS* (Internet Payment System), koji ćemo sada detaljno opisati. Transakcije u sistemu se sastoje od dvije faze: faze pripreme i faze IPS protokola. IPS protokol je, logički posmatrano,

jedna transakcija, u kojoj se novac prenosi sa računa kupca na račun prodavca. U smislu klasifikacije iz odjeljka 3.1.3, IPS je čekovni elektronski sistem.

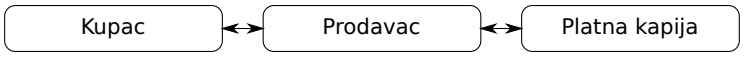
Opis sistema počinje definicijom osnovnih učesnika, kojoj je posvećen odjeljak 3.4.1. Faza pripreme je obrađena u odjeljku 3.4.2, iza čega slijedi detaljan opis samog IPS protokola (odjeljak 3.4.3). Konačno, u odjeljku 3.4.4 su precizirani sigurnosni zahtjevi koje IPS treba da ispoštuje.

3.4.1 Učesnici sistema

IPS je kreiran za upotrebu u tipičnom scenariju elektronske trgovine, kakav je prikazan na slici 1.1. U sistemu dakle postoje četiri učesnika: kupac, prodavac i dvije banke. Kupac je fizičko lice koji posjeduje kreditnu karticu. Sam prodavac je računarski program (Veb prodavnica), koji kupcu pruža usluge izbora robe i brine se za pohranjivanje narudžbi.

Kao što je opisano u poglavlju 3.4.1, svakom od učesnika odgovara po jedan entitet u protokolu. Pri tom, IPS mijenja entitete banke kupca i banke prodavca jedinstvenim entitetom platne kapije.

Entitet kupca je digitalno potpisan računarski program koji se izvršava na računaru kupca (obično digitalno potpisan Java aplet), koji se automatski učitava sa servera prodavca prije izvršetka protokola. Ovaj entitet ćemo nazivati i *platni program*. Entitet prodavca je takođe računarski program, koji se izvršava na serveru prodavca. Entitet platne kapije je distribuirani sistem, koji obuhvata komunikaciju između banke kupca i banke prodavca. Veze između entiteta u protokolu su prikazane na slici 3.1.



Slika 3.1: Entiteti IPS protokola i veze među njima

U daljem tekstu često nećemo praviti razliku između pojedinih učesnika u protokolu i njihovih entiteta, ali bi iz konteksta trebalo biti jasno o kome je riječ. Na primjer, kada kažemo da kupac generiše par asimetričnih ključeva, jasno je da je ovo operacija koju izvršava njegov entitet, bez ikakvog uticaja samog fizičkog korisnika.

3.4.2 Faza pripreme

U svakom elektronskom sistemu za plaćanje potrebno je izvršiti određene pripremne korake (u najmanju ruku, izbor dobara koje kupac želi da kupi) prije nego što otpočne sama transakcija plaćanja. Kod IPS-a, faza pripreme se sastoji od dvije podfaze: faze naručivanja i faze učitavanja platnog programa.

U toku faze naručivanja, komunikacija između kupca i prodavca (odnosno Veb prodavnice) se odvija standardnim protokolima, kao što su HTTP ili HTTPS. Kupac pregleda prodavnicu i odlučuje šta da kupi.

Kada kupac izabere robu i odluči se za plaćanje, započinje faza učitavanja platnog programa. Kao što je već navedeno, ovaj program je digitalno potpisan od strane povjerljive strane, čime se sprečavaju moguće izmjene programa od strane prodavca. Ujedno, platnom programu se dostavljaju i parametri neophodni za izvršenje IPS protokola: opis narudžbe, te sertifikati prodavca i platne kapije.

3.4.3 Opis protokola

Protokol ćemo opisati standardnom Alice-Bob notacijom opisanom u 2.3. Jedina razlika je u metodi šifrovanja. Poruke, koje su u opisu šifrovane javnim ključem, u stvari se šifruju koristeći tzv. metodu *hibridnog šifrovanja*. Hibridno šifrovanje neke poruke znači da je sama poruka šifrovana koristeći simetrični ključ (tzv. *ključ sesije*), koji je pak šifrovan javnim ključem primaoca. Ključ sesije se zatim dalje koristi za svu komunikaciju između pošiljaoca i primaoca poruke u toku trenutne transakcije. Na primjer, poruka:

$$C \rightarrow M: \quad \{PubK\_C\}_{PubK\_M}$$

u stvari znači:

$$C \rightarrow M: \quad (\{Kcm\}_{PubK\_M}, \{PubK\_C\}_{Kcm})$$

Kcm se tako koristi za šifrovanje svih narednih poruka između C i M (umjesto $PubK\_C$ i $PubK\_M$). Zbog konzistentnosti sa notacijom korišćenom u provedenoj formalnoj verifikaciji protokola, entitete ćemo označavati početnim slovima njihovih engleskih naziva. Tako ćemo entitet kupca označavati sa C (od Customer), entitet prodavca sa M (od Merchant), a entitet platne kapije sa PG (od Payment Gateway).

U opisu protokola koristićemo jedan par ključeva (privatni i tajni) kako za šifrovanje, tako i za potpise. Implementacija protokola, s druge strane, može da koristi i dva različita para ključeva za ove namjene (ovo može da bude i neophodno, u zavisnosti od primijenjenih shema šifrovanja i potpisa).

Jedan od glavnih ciljeva IPS-a je bilo kreiranje sistema kod kojeg kupci neće morati prolaziti kroz poseban proces registracije, prije nego što pristupe IPS transakcijama. Međutim, registracija je još uvijek moguća. U kontekstu IPS-a, registracija kupaca podrazumijeva zbavljanje digitalnog sertifikata od strane IPS autoriteta za sertifikaciju.

Stoga, postoje dva osnovna scenarija protokola. Platni program određuje koji će se od dva scenarija izvršiti, tako što pregleda računar kupca u potrazi za IPS sertifikatom. U prvom scenariju, kupac nema sertifikat izdan od IPS autoriteta za sertifikaciju.

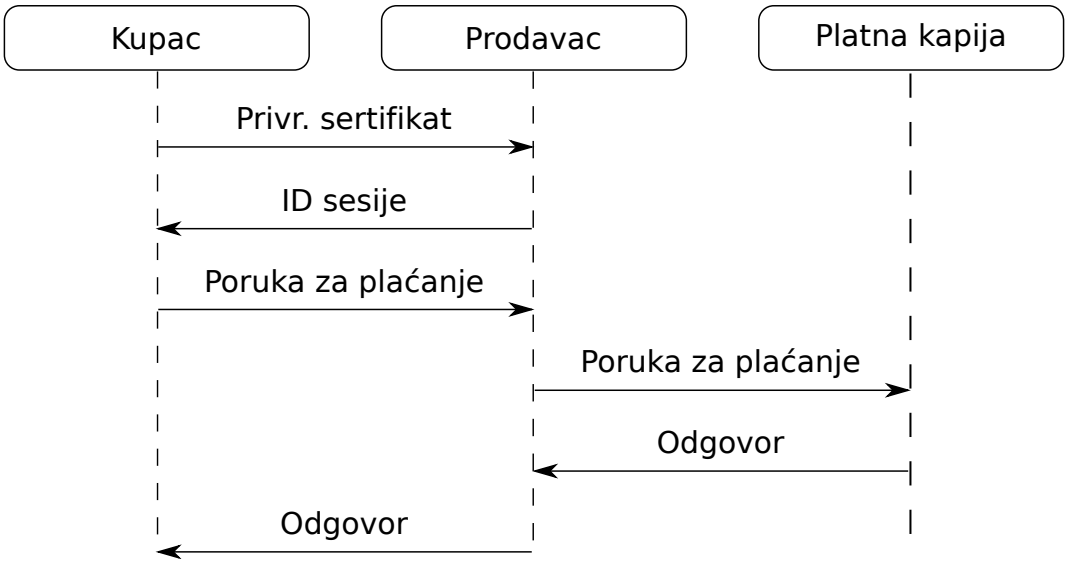
Prvi scenario - bez sertifikata kupca

U ovom scenariju, platni program prvo kupcu predočava podatke o narudžbi, a zatim od njega zahtijeva da unese podatke o plaćanju. Ovi podaci uključuju i tajni kod, koji istovremeno služi da se utvrdi identitet korisnika (autentifikacija), te da se utvrdi da korisnik ima pravo raspolaganja kreditnom karticom (autorizacija). U ovu svrhu se može koristiti

bilo kakav tajni kod (odnosno šifru), izdan od strane banke. U ostatku rada, ovaj kod ćemo nazivati PIN kod; primijetimo da to ne mora biti isti PIN kod kao onaj koji se koristi na bankomatima.

Nakon što korisnik provjeri podatke narudžbe i unese tražene podatke o plaćanju, kupac generiše par ključeva (privatni i javni), na osnovu kojeg se zatim kreira samopotpisani digitalni sertifikat. Primijetimo da ovo nije sertifikat u uobičajenom smislu te riječi; nije potpisan od strane sertifikacionog tijela, te u ovoj fazi protokola ne pruža nikakve garancije o identitetu svoga vlasnika. Vlasništvo nad sertifikatom se može ustanoviti tek kasnije, kao što ćemo opisati pri kraju ovog odijeljka. Nakon što se ovo vlasništvo ustanovi, sertifikat se kasnije može koristiti u razrješenju sporova, slično kao i sertifikati potpisani od strane sertifikacionih tijela, zbog čega ćemo i upotrebljavati termin sertifikat. Bitno je napomenuti da je ovaj sertifikat privremen, i da će se koristiti samo za trenutnu transakciju. Sertifikat se ne instalira na računar korisnika, već se samo šalje prodavcu i platnoj kapiji, kako ćemo kasnije opisati.

Konačno, počinje izvršavanje IPS protokola. Slika 3.2 pruža grubu skicu IPS protokola.



Slika 3.2: Shematski prikaz poruka IPS protokola

Sada ćemo detaljno opisati svaku od poruka sa slike 3.2, koristeći Alice-Bob notaciju (slika 3.3). U opisu ćemo, ponovo zbog konzistentnosti sa notacijom korišćenom u provedenoj formalnoj verifikaciji protokola, koristiti engleske nazive za dijelove poruka.

Izvršavanje protokola inicira kupac, što olakšava implementaciju, budući da je obično lakše ostvariti vezu od kupca ka prodavcu, nego vezu u drugom smjeru. Na početku protokola, kupac šalje svoj privremeni sertifikat prodavcu (slika 3.3, poruka 1):

P1. $C \rightarrow M : \ \{PubK\_C\}_{PubK\_M}$

Ovaj sertifikat je šifrovan javnim ključem prodavca (podsjetimo da se u protokolu koristi hibridno šifrovanje, početak poglavlja 3.4.3). Na ovaj način se korisnik djelimično osigurava da komunicira baš sa prodavcem sa kojim želi, jer jedino on može da dešifruje ovu poruku. Prodavac odgovara sa (slika 3.3, poruka 2):

P1. $C \rightarrow M: \quad \{PubK\_C\}_{PubK\_M}$
P2. $M \rightarrow C: \quad sign_M(Sid)$
P3. $Let:$
$PInf = (CardInf, PIN, Amount, Sid, M, P)$
$PM = \{sign_C(PInf, Non\_CPG)\}_{PubK\_PG}$
$OrderInf = (OrderDesc, Amount, Sid)$
$OrderSig = signOnly_C(OrderInf)$
$C \rightarrow M: \quad (PM, OrderSig)$
P4. $M \rightarrow PG:$
$(PM, \{PubK\_C, signOnly_M(Amount, PubK\_C, Sid)\}_{PubK\_PG})$
P5. $PG \rightarrow M:$
$\{Resp, signOnly_{PG}(Resp, Sid, Amount, Non\_CPG)\}_{PubK\_M}$
P6. $M \rightarrow C:$
$\{Resp, signOnly_{PG}(Resp, Sid, Amount, Non\_CPG)\}_{PubK\_C}$

Slika 3.3: Alice-Bob notacija IPS protokola

P2. $M \rightarrow C: \quad sign_M(Sid)$

Ovdje je Sid identifikator sesije. Ovaj broj treba da bude novi, slučajno generisani broj (nonce), koji bi, osim toga, trebalo da u idealnom slučaju bude jedinstven za svaku sesiju (izvršavanje) protokola. U praksi, ovo znači da Sid mora biti jako veliki broj. Budući da jedinstveno identifikuje sesiju, ovaj broj je od centralne važnosti za protokol; svi sigurnosni zahtjevi vezani za neku transakciju će se uključivati upravo ovaj identifikator. Prodavac zatim potpisuje poruku da bi potvrdio svoj identitet.

Sljedeći korak, kreiranje poruke 3 sa slike 3.3, je nešto komplikovaniji. Kupac, na osnovu korisnikovog unosa, skuplja podatke bitne za izvršenje transakcije, kojima zatim pridružuje ukupan iznos narudžbe, identifikator sesije, te oznake prodavca i platne kapije, M i P:

$PInf = (CardInf, PIN, Amount, Sid, M, P)$

Kupac zatim priprema poruku za plaćanje:

$PM = \{sign_C(PInf, Non\_CPG)\}_{PubK\_PG}$

Poruka za plaćanje se sastoji od podataka za plaćanje i novog, slučajno generisanog broja (Non_CPG), koji se zatim digitalno potpisuju. Poruka je šifrovana javnim ključem platne kapije. Ova činjenica ne dozvoljava prodavcu uvid u podatke o kreditnoj kartici, te osigurava tajnost broja Non_CPG. Ovaj broj će kupac kasnije koristiti da provjeri ispravnost odgovora platne kapije. Kupac zatim skuplja i podatke o narudžbi, i digitalno ih potpisuje.

$OrderInf = (OrderDesc, Amount, Sid)$
$OrderSig = signOnly_C(OrderInf)$

Konačno, kupac kombinuje informacije o plaćanju i potpis narudžbe i šalje ih prodavcu:

P3. $C \rightarrow M: \quad (PM, OrderSig)$

Prodavac sam može da generiše poruku $OrderInf$, budući da su mu poznati i opis narudžbe, i iznos i Sid; zatim se može uvjeriti da je kupac potpisao iste informacije o narudžbi. Ako je potpis ispravan, prodavac ga pamti zajedno sa samim informacijama o narudžbi. Ovo se kasnije može koristiti kao dokaz o sadržaju narudžbe, u slučaju sporova.

Prodavac sada šalje poruku platnoj kapiji (slika 3.3, poruka 4).

P4. $M \rightarrow PG$:
$(PM, \{PubK\_C, signOnly_M(Amount, PubK\_C, Sid)\}_{PubK\_PG})$

Poruka o plaćanju, PM se prosljeđuje platnoj kapiji, budući da je prodavac sam ne može dešifrovati. Prodavac u poruku takođe uključuje i sertifikat kupca. Ovaj sertifikat, ukupan iznos narudžbe, te identifikator sesije se digitalno potpisuju da bi se potvrdio identitet prodavca.

Nakon što primi poruku za plaćanje, platna kapija može da je dešifruje, koristeći svoj tajni ključ. Zatim provjerava potpis podataka za plaćanje, te da li je korisnik autorizovan za korišćenje kreditne kartice. Ova provjera se vrši na osnovu kombinacije broja kreditne kartice i PIN koda. Ako je kombinacija ispravna, to znači da je kupac autorizovan za upotrebu kartice. Osim toga, to znači da je uspostavljeno i vlasništvo nad privremenim sertifikatom - vlasnik sertifikata je ujedno i vlasnik kreditne kartice. To možemo zaključiti iz činjenice da je jedino korisnik koji je vlasnik i kartice i sertifikata u stanju da kompletira podatke o plaćanju, a zatim ih i potpiše. Konačno, budući da je sam sertifikat primljen od strane prodavca, a ne kupca, možemo utvrditi i da se kupac i prodavac slažu oko samog sertifikata. Osim toga, vrši se i provjera da li se slažu i oko iznosa koji treba naplatiti, kao i identifikatora sesije. Konačno, platna kapija u bazi podataka provjerava i da li je Sid već korišćen. Ovo je neophodno da se preduprijede napadi ponavljanjem od strane zlonamjernih prodavaca.

Ukoliko su sve provjere uspješne, podaci o kreditnoj kartici i iznosu se prosljeđuju banci. Banka provjerava preostali iznos novca na računu kupca. Ukoliko novca ima dovoljno, iznos narudžbe se prenosi na račun prodavca, a banka vraća pozitivan odgovor. U suprotnom, odgovor je negativan. Ako je odgovor pozitivan, platna kapija pamti detalje transakcije (uključujući i privremeni sertifikat kupca) u svojoj bazi podataka.

Platna kapija potom prosljeđuje odgovor prodavcu (poruka 5, slika 3.3):

P5. $PG \rightarrow M$:
$\{Resp, signOnly_{PG}(Resp, Sid, Amount, Non\_CPG)\}_{PubK\_M}$

Odgovor se, zajedno sa ostalim podacima, digitalno potpisuje, da se potvrdi identitet kapije. Potpisana poruka sadrži i Non_CPG, koji je nepoznat prodavcu. Stoga klijent može biti siguran da je odgovor stigao od strane platne kapije, i da se odnosi na trenutnu transakciju (tj. da odgovor nije ponovljen od strane zlonamjernog prodavca). Prodavac, s druge strane, može biti siguran da odgovor nije ponovljen jer je kriptovan trenutnim ključem sesije (podsjetimo, koristi se hibridno šifrovanje). Konačno, prodavac prosljeđuje kupcu sadržaj poruke (slika 3.3), čime se IPS protokol završava:

P6. $M \rightarrow C$:
$\{Resp, signOnly_{PG}(Resp, Sid, Amount, Non\_CPG)\}_{PubK\_C}$

Drugi scenario - sa sertifikatom korisnika

U drugom scenariju protokola, kada korisnik posjeduje ispravan IPS digitalni sertifikat, protokol je nešto jednostavniji. U ovom scenariju podrazumijevamo da su svi sertifikati, uključujući i onaj klijenta, već razmijenjeni. Stoga nema potrebe da se isti šalje u prvoj poruci protokola (slika 3.2). Osim toga, informacije o plaćanju, sadržane u trećoj poruci, ne moraju da sadrže PIN kod. Umjesto toga, autorizacija kupca se vrši na osnovu njegovog sertifikata. Ostatak protokola ostaje nepromijenjen. Budući da ova varijanta protokola ne donosi veće prednosti u odnosu na, na primjer, SET protokol, u daljem tekstu je nećemo detaljnije obrađivati.

3.4.4 Sigurnosni zahtjevi za IPS protokol

Sada možemo precizno definisati i sigurnosne zahtjeve za IPS protokol, u smislu definicija iz poglavlja 2.4, a u skladu sa neformalnim ciljevima iz poglavlja 3.3. Od IPS protokola zahtijevamo da se obezbijedi:

- tajnost svih podataka o kreditnoj kartici između kupca i platne kapije;
- tajnost podataka o narudžbi između prodavca i kupca;
- obostrana injektivna saglasnost kupca i prodavca na skupu podataka koji obuhvata iznos narudžbe, opis narudžbe, i identifikator sesije;
- obostrana injektivna saglasnost kupca i platne kapije na skupu podataka koji obuhvata iznos narudžbe, identifikator sesije i podatke o kreditnoj kartici;
- obostrana injektivna saglasnost prodavca i platne kapije na skupu koji obuhvata iznos narudžbe i identifikator sesije.

Napomenimo ovdje da IPS protokol sam po sebi ne može da garantuje platnoj kapiji injektivnu saglasnost ni sa prodavcem ni sa kupcem; za ovo je neophodno da platna kapija po prijemu četvrte poruke protokola (slika 3.3) provjeri da li je identifikator sesije već korišćen.

Glava 4

Implementacija protokola

Kao „dokaz koncepta", kreirana je implementacija za IPS[1]. Poglavlje 4.1 daje opšti pregled ove implementacije. Budući da IPS protokol koristi javne ključeve, neophodno je obezbijediti njihovu infrastrukturu. Implementacija ove infrastrukture je opisana u poglavlju 4.2. U poglavlju 4.3 opisujemo primijenjenu *shemu označavanja poruka*, kojom se sprečavaju ranije opisani napadi greškom tipa (poglavlje 2.5). Konačno, posljednja dva poglavlja pružaju nešto detaljniji pregled implementacije entiteta kupca (poglavlje 4.4), odnosno entiteta prodavca i platne kapije (poglavlje 4.5).

4.1 Pregled implementacije

Implementirani sistem se sastoji iz tri osnovne komponente, od kojih svaka odgovara po jednom entitetu u protokolu:

- kupac;
- prodavac;
- platna kapija.

Komponenta kupca je implementirana koristeći Java 2 Standard Development Kit platformu, u verziji 6.0, a zatim upakovana u vidu Java apleta, potpisanog od strane IPS sertifikacionog tijela. Budući da je Java jezik nezavisan od platforme, te da je podrška za Javu veoma široko rasprostranjena (plug-in za Javu je već instaliran u oko 80% Veb pregledača [OWL10]), korisnik u većini slučajeva ne mora da instalira nikakav softver. Time je postignuto da se IPS može koristiti sa bilo kojeg računara, čime su ispunjeni neki od glavnih ciljeva IPS protokola, naime jednostavnost korišćenja za kupce, te pokretljivost kupaca.

Komponente prodavca i platne kapije su, s druge strane, implementirane koristeći programski jezik Python. Kao i Java, i Python je nezavisan od platforme, čime je olakšana

[1] Implementacija je trenutno dostupna na upit autoru na adresu ognjen.maric@gmail.com pod licencom otvorenog koda, a ubrzo će biti dostupna i putem Veba

instalacija komponente. Osim toga, budući da su komponente kupca i prodavca implementirane u različitim programskim jezicima, ova implementacija je ujedno poslužila i kao test interoperabilnosti.

Cijeli protokol je implementiran povrh HTTP protokola. Ovakva vrsta implementacije je olakšana činjenicom da IPS protokol za plaćanje prati prati šablon zatjev-odgovor. Ovaj šablon je uočljiv i na shematskom prikazu protokola (3.2), a ovdje ćemo ga eksplicitno navesti.

1. Kupac kao zahtjev šalje prodavcu svoj privremeni sertifikat (poruka 1); kao odgovor prodavca dolazi identifikator sesije (poruka 2).
2. Kupac kao zahtjev šalje prodavcu poruku za plaćanje (poruka 3); kao odgovor, dobija poruku da li je plaćanje uspješno izvršeno ili ne (poruka 6).
3. Prodavac kao zahtjev platnoj kapiji prosljeđuje poruku o plaćanju (poruka 4). Platna kapija odgovara da li je plaćanje uspješno izvršeno ili ne (poruka 5).

Pri tom, primjećujemo da je treći par zahtjev-odgovor vremenski ugnježden unutar drugog para poruka.

IPS poruke za plaćanje su zatim implementirane kao binarni sadržaj koji se šalje putem POST HTTP zahtjeva, odnosno odgovora na ove zahtjeve. Sa stanovišta efikasnosti implementacije, HTTP dodaje nepotreban nivo kompleksnosti. Međutim, izbor ovog protokola kao prenosnika za IPS ima značajne prednosti:

- HTTP je obično „imun" na zaštitne zidove (firewall-e), tj. ovi zaštitni zidovi su najčešće konfigurisani ili da propuštaju HTTP saobraćaj, ili pružaju neku vrstu „zaobliaznice" (u vidu posrednika - proxy servera). Samim tim, ovim je omogućeno i korišćenje IPS-a u okruženjima u kojima postoji zaštitni zid;
- prije pokretanja protokola za plaćanje, potrebno je na neki način proslijediti informacije o narudžbi, od Veb prodavnice prodavca do komponenti kupca i prodavca. Uobičajen način za ovakvu vrstu komunikacije je upravo HTTP, koji se danas često koristi kao prenosnik u raznim mehanizmima razmjene informacija između distribuiranih sistema, kao što su SOAP[2] ili REST[3];
- činjenica da kupac već posjeduje neku vrstu Veb prodavnice znači da već posjeduje i konfigurisan Veb server. Instalacija nove Veb aplikacije bi u tom slučaju trebala biti znatno jednostavnija od instaliranja aplikacije koja koristi neki drugi transportni sloj.

Navedene prednosti se odnose na komunikaciju između kupca i prodavca. Ova implementacija koristi HTTP i za komunikaciju između prodavca i platne kapije, ali ni korišćenje nekih drugih mehanizama transporta u te svrhe ne bi imalo nikakvih nedostataka.

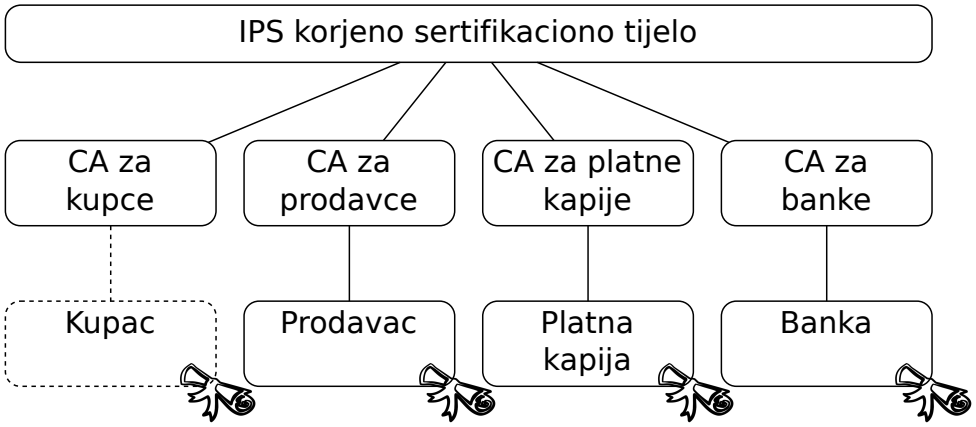
[2]SOAP - *Simple Object Access Protocol*. `http://www.w3.org/TR/soap/`

[3]REST - *Representational State Transfer*. `http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm`

Kriptografske operacije u implementaciji su vršene pomoću tri algoritma, RSA, DSA i AES. AES je korišćen za sve operacije simetričnog šifrovanja, u CBC načinu rada, i to sa 128-bitnom veličinom ključa. Za svaku poruku je korišćen po jedan inicijalizacioni vektor. RSA je korišćen i za šifrovanje, i za potpise, sa 2048-bitnom veličinom ključa, a DSA naravno samo za potpise, sa 1024-bitnim ključevima.

4.2 Infrastruktura ključeva

Budući da se od IPS protokola za plaćanje zahtijeva da obezbijedi autentifikaciju svih učesnika u protokolu, neophodno je postojanje infrastrukture javnih ključeva (poglavlje 2.2). Stoga je kreirano više sertifikacionih tijela (eng. Certificate Authority, ili skraćeno CA). Kao što je prikazano na slici 4.1, ova tijela su posložena u hijerarhiju na čijem se vrhu nalazi korjeni IPS CA, ispod kojeg se (u istom nivou) nalaze tijela za sertifikaciju kupaca, prodavaca, platnih kapija i banaka.



Slika 4.1: Hijerarhija sertifikacionih tijela

Svi učesnici protokola moraju posjedovati sertifikate izdane od odgovarajućeg CA. Izuzetak je kupac, koji, kao što je opisano u poglavlju 3.3, može da pokrene protokol bez posjedovanja sertifikata. U tom slučaju, kao što je već ranije opisano, programska komponenta kupca kreira privremeni sertifikat.

Struktura sa slike 4.1 je implementirana koristeći OpenSSL biblioteku. Jedino odstupanje u implementaciji je nedostatak sertifikacionog tijela za banke, budući da ulogu banke u ovoj implementaciji vrši platna kapija. Svim učesnicima u protokolu je zatim kreiran sertifikat u X.509 formatu, sa 2048-bitnim RSA ključem. Budući da je protokol implementiran u akademske svrhe, sertifikat korjenog tijela je samopotpisan, pa tako nije u hijerarhiji povjerenja za uobičajene ugrađene sertifikate (VeriSign, Thawte i slično). Ovo predstavlja problem za potpis Java apleta kojim je implementirana komponenta kupca, pa je prethodno potrebno uvesti sertifikat ovog tijela u tzv. *Java keystore*[4] na računaru koji se koristi za ku-

[4]Java keystore je baza podataka koja sadrži sertifikate kojima se vjeruje. Svaka instalacija Java okruženja posjeduje ovakvu bazu.

povinu. Kako IPS ne koristi SSL, nije neophodno uvesti sertifikat u bazu sertifikata Veb pregledača.

4.3 Označavanje poruka

Opisaćemo sada implementiranu shemu označavanja poruka, koja se koristi da se spriječe napadi greškom tipa opisani u poglavlju 2.5. Shema označavanja je sistem u kome su sve poruke označene nekim dodatnim informacijama, koje označavaju njihove namijenjene tipove. Ove oznake možemo implementirati pomoću nekoliko bitova koji se stavljaju ispred svakog polja, i označavaju tip odgovarajućeg polja. Na primjer, sa $(nonce, N)$ bismo mogli da predstavimo vrijednost N, označenu tako da sugeriše da treba da se interpretira kao „jednokratni" slučajni broj (nonce). Na sličan način možemo označiti i složene poruke, npr. sa $(pair, ((nonce, N_1), (nonce, N_2)))$ možemo označiti par vrijednosti N_1 i N_2 označenih kao slučajni brojevi. Oznaka za enkripciju treba da sadrži tip ključa korišćenog za enkripciju, te tip tijela enkripcije.

Heather i ostali su tada pokazali sljedeće [HLS00]: ukoliko svi „pošteni" učesnici protokola pravilno označavaju svoje poruke, te provjeravaju ispravnost oznaka u svakoj poruci, onda ukoliko postoji proizvoljan napad na protokol koji koristi shemu označavanja, tada postoji i napad na protokol koji koristi shemu označavanja, i u kome su sva polja ispravno označena. Drugim riječima, ukoliko je protokol siguran pod pretpostavkom jake apstrakcije tipova, tada je siguran i ukoliko se koristi shema označavanja.

Za potrebe IPS protokola, implementirana je jednostavna generička biblioteka za označavanje poruka, kao i „raspakivanje" već označenih poruka. Kako je za realnu implementaciju protokola, da bi se podaci iz nekog polja mogli izvaditi iz poruke, osim poznavanja tipa polja neophodno poznavati i njegovu dužinu, tako se i implementirane oznake sastoje iz dva dijela (za tip i dužinu). Dalje, moguća su i određena uprošćenja predložene sheme. U implementaciji se koristi sljedeće: budući da napadač može proizvoljno promijeniti oznake na vanjskom nivou (oznake koje nisu zaštićene nikakvom enkripcijom), ove oznake možemo eliminisati. Kako se za implementaciju protokola koriste dva programska jezika, Java i Python, biblioteka za označavanje je implementirana u oba.

4.4 Komponenta kupca

Kao što je već navedeno, komponenta kupca je implementirana kao Java aplet, potpisan od strane IPS sertifikacionog tijela. Nakon okončanja kupovine, Veb prodavnica preusmjerava Veb pregledač kupca na stranicu koja sadrži ovaj aplet. Aplet uzima dva parametra: identifikator narudžbe, te adresu na koju se korisnik preusmjerava nakon završetka protokola. Identifikator narudžbe omogućava apletu da preuzme podatke o samoj narudžbi. Ovaj broj nije u vezi sa identifikatorom sesije iz opisa protokola (slika 3.3). Za uspješno izvršavanje IPS protokola nije neophodno da ovaj identifikator bude slučajan ili tajan; međutim, ukoliko se želi postići tajnost sadržaja narudžbe, neophodno je koristiti kripto-

grafski siguran slučajan broj, te je neophodno koristiti HTTPS kao sredstvo prenosa apleta i parametara.

Ukoliko je potpis apleta uspješno verifikovan, Veb pregledač korisniku prikazuje poruku o tome (slika 4.2), i aplet može početi sa izvršavanjem.

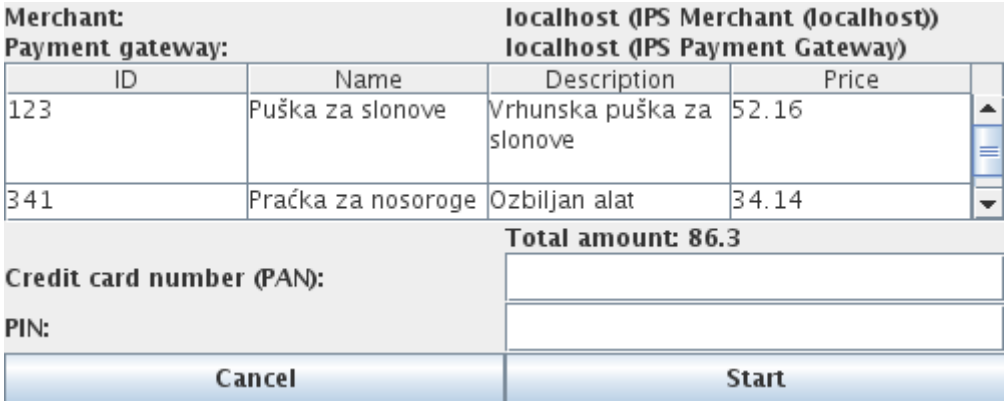
Slika 4.2: Verifikacija potpisa na apletu

Aplet zatim na osnovu ranije opisanog identifikatora narudžbe preuzima podatke o istoj. Podaci su zapisani u vidu XML datoteke. Implementirani format datoteke je jednostavan; svaka stavka narudžbe opisana je sa četiri elementa: identifikatorom, nazivom, opisom i cijenom. Osim toga, aplet preuzima i sertifikate prodavca i platne kapije. Sam aplet već u sebi sadrži sertifikate IPS sertifikacionih tijela za prodavce i platne kapije, na osnovu kojih zatim pokušava da verifikuje sertifikate ostalih učesnika. Ukoliko je verifikacija uspješna, aplet zatim iz sertifikata vadi informacije o drugim učesnicima. Konačno, korisniku se prikazuje dijalog za izvršenje protokola (slika 4.3). Korisnik može da provjeri podatke o narudžbi, prodavcu i platnoj kapiji, a zatim unosi neophodne podatke za plaćanje. U implementiranoj verziji, podaci se sastoje od samo dva polja, broja kartice (*PAN*), i zaštitnog identifikacionog koda (*PIN*).

Nakon završetka izvršavanja protokola, korisnik dobija poruku o tome da li je protokol uspješno izvršen ili ne. U oba slučaja, korisnik se zatim preusmjerava na adresu koja je zadana kao drugi parametar apleta. U trenutnoj implementaciji, ukoliko dođe do greške pri izvršenju protokola, transakciju je potrebno pokrenuti iz početka.

Aplet je implementiran na uobičajen način. „Glavna" klasa, *Applet* je izvedena iz Swing klase *JApplet*. Ostatak implementacije čine klase:

- *IpsPanel*, koja sadrži grafički interfejs apleta;
- *IpsRunner* (apstraktna), koja sadrži glavnu logiku protokola, kao i njena potklasa *IpsHttpRunner*, koja izvršava protokol preko HTTP-a;



Slika 4.3: IPS aplet

- *Message*, koja implementira shemu označavanja opisanu u poglavlju 4.3;

kao i nekoliko manje bitnih pomoćnih klasa.

U implementaciji komponente, korišćene su isključivo klase koje su već ugrađene u J2SE 5.0, tako da za izvršenje apleta nije neophodno učitavanje dodatnih klasa. Implementirani aplet, zapakovan u JAR datoteku, zauzima oko 25 kilobajta.

4.5 Komponente prodavca i platne kapije

Kako su implementirane komponente prodavca i platne kapije međusobno veoma slične, ovdje ćemo ih zajedno opisati.

Kako je već navedeno u poglavlju 4.1, obe komponente su implementirane u programskom jeziku Python, i koriste HTTP protokol za prenos poruka. Budući da je HTTP protokol koji ne čuva stanje (eng. *stateless*), potrebno je na neki način zapamtiti trenutno stanje svakog od učesnika. U implementaciji je u ove svrhe iskorišćen *memcached*[5], jednostavna baza podataka oblika ključ-vrijednost koja sve svoje podatke čuva u memoriji. Za implementaciju su se, osim standardnih Python biblioteka, koristile i još tri dodatne postojeće biblioteke, koje su sve otvorenog koda.

- *web.py*[6] biblioteka za implementiranje serverskih Veb aplikacija. Sama biblioteka je zasnovana na minimalističkim principima i nudi relativno mali broj funkcija, ali je izabrana zbog svoje male veličine (instalacija zauzima oko 300kb) i jednostavnosti podešavanja - za kreiranje nove serverske aplikacije je dovoljna samo jedna datoteka sa Python programom. Osim toga, web.py podržava više protokola za komunikaciju sa Veb serverom, kako standardne CGI, SCGI i FastCGI protokole, tako i one specifične za Python (WSGI i mod_python), čime je omogućena instalacija na gotovo svim Veb serverima.

[5] *Memcached* - A distributed memory object caching system. `http://memcached.org/`
[6] *web.py*: `http://webpy.org/`

- *M2Crypto*[7] biblioteka za kriptografiju. Biblioteka u stvari predstavlja Python interfejs ka poznatoj OpenSSL biblioteci, napisanoj u C-u. Prednost ove biblioteke leži u velikom broju podržanih kriptografskih algoritama, naslijeđenih iz OpenSSL biblioteke. Najveća mana je nedostatak dokumentacije, zbog kojeg je često potrebno tražiti informacije u izvornom kodu.

- *pylibmc*[8] biblioteka za komunikaciju sa memcached bazom.

Osim samog IPS protokola za plaćanje, komponenta prodavca mora da obavi još neke zadatke. Prije početka izvršavanja protokola, mora da primi podatke o samoj narudžbi. Ove podatke šalje Veb prodavnica prodavca, i to u obliku XML dokumenta, čiji je format već opisan u poglavlju 4.4. Osim toga, Veb prodavnica mora da pošalje još dva podatka:

- adresu na kojoj će primiti povratnu informaciju u uspjehu (ili neuspjehu) protokola. Kako u ovom trenutku nije izvršena integracija komponente prodavca sa gotovim Veb prodavnicama, ova adresa se trenutno pamti, ali se ne koristi;

- adresu stranice na koju se korisnik upućuje nakon završetka protokola. Ova adresa se, kako je takođe već opisano u poglavlju 4.4, zatim prosljeđuje komponenti kupca.

Komponenta platne kapije, sa druge strane, treba da izvrši sljedeće dodatne operacije:

- provjeru ispravnosti podataka kupca. U realnijem scenariju, ove informacije bi se provjeravale upitom prema banci. Međutim, u ovoj implementaciji, podaci su direktno dostupni platnoj kapiji, u vidu *SQLite*[9] baze. Podrška za SQLite je ugrađena u Python 2.5 i više verzije;

- provjeru da li u bazi već postoji transakcija sa istim identifikatorom sesije. Ovo je neophodno da bi se spriječili napadi ponavljanjem (eng. replay attacks) od strane zlonamjernih prodavaca;

- provjeru iznosa novca na računu korisnika, te izvršenje same transakcije, tj. prenos novca na račun prodavca. Jasno je da ove operacije moraju biti atomične. Ponovo, ovo su operacije koji bi se u realnijem scenariju vršile u komunikaciji sa bankom.

Kao i za komponentu kupca, i za komponente prodavca, odnosno platne kapije bilo je neophodno implementirati shemu označavanja poruka (poglavlje 4.3). Budući da su obe komponente pisane u istom jeziku, ista implementacija je iskorišćena i za jednu i za drugu.

[7] *M2Crypto* - A Python crypto and SSL toolkit. `http://sandbox.rulemaker.net/ngps/m2/`
[8] *pylibmc* - Quick and small memcached client for Python. `http://pypi.python.org/pypi/pylibmc`
[9] *SQLite*: `http://www.sqlite.org/`

Glava 5

Verifikacija protokola

Svrha svih kriptografskih protokola, pa tako i protokola za elektronsko plaćanje, je da pruže određene sigurnosne garancije učesnicima u protokolu, tj. da ispoštuju određene sigurnosne zahtjeve. Nekoliko uobičajenih vrsta ovakvih zahtjeva smo opisali u poglavlju 2.4. Ova glava daje odgovor na pitanje: kako možemo biti sigurni da neki protokol zaista ispunjava navedene zahtjeve?

Drugim riječima, nakon što su neki kriptografski protokol i njegovi sigurnosni zahtjevi definisani, potreban nam je metod *verifikacije* ovih zahtjeva. Poglavlje 5.1 pruža kratak pregled više predloženih metoda; nešto širi pregled se može naći u doktorskoj disertaciji [Cre06].

Mi ćemo detaljnije opisati dva konkretna alata za verifikaciju, koji koriste različite metode. U poglavlju 5.2 opisujemo programski paket *AVISPA*, koja se može kombinovati sa više metoda verifikacije. Drugi alat je tzv. *interaktivni dokazivač teorema* (eng. interactive theorem prover) nazvan *Isabelle/HOL* (poglavlje 5.3), koji se često koristi u kombinaciji sa metodom nazvanom *Paulsonov induktivni pristup* (poglavlje 5.4).

Potom se posvećujemo konkretnoj verifikaciji IPS protokola, koja je izvedena dvojako. Prvo, upotrijebljen je upravo paket AVISPA. Model IPS protokola u ovom paketu je prezentovan u odjeljku 5.5.1, dok rezultate verifikacije opisuje odjeljak 5.5.2.

Drugi provedeni način verifikacije koristi Isabelle/HOL. Model IPS protokola u Isabelle/HOL je prezentovan u odjeljku 5.5.3, a razlike u odnosu na modele dobijene Paulsonovim induktivnim pristupom razmatra odjeljak 5.5.4. Konačno, poglavlje 5.5.5 daje pregled važnijih dokazanih lema, kao i krajnjih rezultata verifikacije u ovom sistemu.

5.1 Metode formalne verifikacije

Utvrđivanje sigurnosti protokola najčešće nije nimalo jednostavan poduhvat. U ranim danima, pristup ovom poslu je obično bio *ad hoc*. Nakon što bi neki novi protokol bio predložen, tražili bi se propusti u njemu; ukoliko bi se propust pronašao, predlagala bi se ispravljena verzija, i postupak bi se ponavljao. Problem sa ovakvim pristupom je što

propusti načinjeni u kreiranju protokola ponekad mogu biti jako teški za otkrivanje. Čuven primjer je baš Needham-Schroeder protokol za autentifikaciju, opisan u poglavlju 2.3. Napad opisan u 2.5 je pronađen punih 18 godina nakon prvog objavljivanja protokola ([NS78], [Low96b]). Stoga je umjesto ciklusa otkrivanja napada i popravljanja protokola poželjna formalna verifikacija njihove ispravnosti. Postoje dva glavna pristupa ovakvoj verifikaciji.

Kriptografski pristup. Cilj prvog pristupa je da svede problem sigurnosti protokola na problem sigurnosti njegovih kriptografskih elemenata (algoritama šifrovanja, heširanja i slično). Za protokole za koje su izvedeni ovakvi dokazi često se koristi termin *dokazivo sigurni* (eng. provably secure). Ovakvi *kriptografski dokazi* barataju vjerovatnoćama i teorijom kompleksnosti. U osnovi, protokol se smatra sigurnim ako se dokaže da bilo koji efikasan napad na protokol, koji uspijeva sa dovoljno visokom vjerovatnoćom, za sobom povlači neku vrstu efikasnog napada istih karakteristika na kriptografske elemente protokola, a zatim i efikasno rješavanje nekog pretpostavljeno teškog problema (npr. *Diffie-Hellman problema*). Problem sa kriptografskim dokazima je što su prilično komplikovani, a broj slučajeva koji se moraju obraditi je veoma veliki, čak i za najjednostavnije kriptografske protokole. Zbog toga je teško izvesti sveobuhvatne i rigorozne dokaze, pa oni često postaju više nalik na skice, preskačući pojedine slučajeve. Ovo za posljedicu ima dosta česte greške u takvim dokazima ([PW92], [PSW95]).

Formalni pristup. Zbog kompleksnosti kriptografskih dokaza, većina metoda verifikacije pribjegava drugačijem pristupu: izračunavanja se apstrahuju, a kriptografski algoritmi aksiomatizuju. To znači da ove metode podrazumijevaju tzv. *savršenu kriptografiju*, gdje je jedini način da neko pročita sadržaj šifrovane poruke posjedovanje odgovarajućeg ključa za dešifrovanje. Osim toga, iz heš vrijednosti neke poruke ne mogu se izvući nikakve informacije o samoj poruci.

Gotovo svi ovakvi dokazi koriste neku vrstu tzv. *Dolev-Yao modela* [DY83], kojim se definišu mogućnosti napadača. U ovom modelu, napadač može da dešifruje poruke šifrovane nekim ključem ukoliko poznaje inverzni ključ, a može i da sastavi bilo kakvu poruku čiji su mu dijelovi već poznati. Osim toga, napadač ima potpunu kontrolu nad mrežom: može da presretne sve poruke na njoj, kao i da pošalje nove, pri tom se lažno predstavljajući kao neki legitiman učesnik. Još jedna pretpostavka je da su svi javni ključevi poznati svim učesnicima, pa tako i napadaču. Konačno, napadač i sam može da bude učesnik u protokolu, tj. drugi korisnici mogu započeti protokol baš sa napadačem, te prihvataju učešće u izvršavanju protokola koje započne napadač.

Ostaje naravno pitanje: da li se sigurnost dokazana u ovako idealizovanim modelima prenosi i na sigurnost konkretnih kriptografskih realizacija protokola? Drugim riječima, da li su ovakvi modeli *kriptografski opravdani* (eng. cryptographically sound)? Odgovor je, barem u opštem slučaju, negativan. Rad [PSW00] daje kontraprimjer, tj. uvodi protokol koji je siguran pod pretpostavkom savršene kriptografije, ali jedna njegova kriptografska implementacija nije. Međutim, u pitanju je „sintetički" kontraprimjer, koji koristi neobične sheme šifrovanja i rezonovanja, dok zasad još uvijek ne postoji nikakav „realan" kontraprimjer. Nažalost, tačni uslovi pod kojima se sigurnost ovakvih modela prenosi na

sigurnost njihovih kriptografskih realizacija nisu sasvim poznati. U radu [BPW03] je opisan model sličan Dolev-Yao modelu u kojem ovakav prenos sigurnosti dokazano važi (uz određena ograničenja na dozvoljene kriptografske operacije u protokolu), ali je i sam model i dokazi u njemu ipak znatno komplikovaniji od originalnog Dolev-Yao modela. Stoga većina alata i dalje koristi uobičajene Dolev-Yao modele. Kriptografsku opravdanost ovakvih modela za određene (prilično ograničene) klase kriptografskih protokola pokazuju radovi [AR02] i [MW04].

Ipak, čak i kada se zanemare kriptografski detalji dokaza, problem sigurnosti protokola i dalje nije nimalo trivijalan. Distribuirana priroda protokola čini da broj slučajeva koji se moraju uzeti u razmatranje raste veoma velikom brzinom sa povećanjem kompleksnosti protokola. Tako greške nisu rijetkost čak ni u ovako uprošćenim modelima [Low96c]. Međutim, uprošćavanje donosi jednu dodatnu, značajnu prednost: mogućnost primjene formalnih metoda u verifikaciji protokola. Apstrahovanjem se kriptografske operacije svode na simboličke manipulacije, sa kojom se računari mnogo lakše „bore" nego sa teorijom vjerovatnoće iz kriptografskih dokaza. Prvi poduhvat na ovom polju je izveo Gavin Lowe [Low96b]. Naravno, sama činjenica da se računari mogu primijeniti ne mora neophodno značiti da je takva primjena efikasna, budući se broj slučajeva koje je neophodno razmotriti pri analizi protokola ne smanjuje sam od sebe samo zato što upotrebljavamo računar. Štaviše, dokazano je da je problem sigurnosti protokola u opštem slučaju neodlučiv [Dur+99]. Stoga je za algoritamsku provjeru protokola potrebno uvesti određena ograničenja na analizirane slučajeve.

Ukoliko se ograničenja postave na veličinu poruka u protokolu, kao i na broj istovremenih izvršavanja protokola, omogućava se modeliranje protokola kao sistema sa *konačnim brojem stanja* (eng. finite-state systems). Zatim je moguće primijeniti *alate za provjeru modela* (eng. model-checkers) na analizu sigurnosti protokola [MMS]. Drugi pristup je modeliranje napada na protokol u vidu simboličkih ograničenja. U tom slučaju, za analiziranje protokola se za mogu iskoristiti *alati za rješavanje ograničenja* (eng. constraint-solvers) [MS01]; svako rješenje koje zadovoljava ograničenja predstavlja napad na protokol. Za primjenu ovog pristupa nije neophodno ograničiti veličinu poruka u protokolu. Još jedan pristup koji takođe ne zahtijeva ograničenja na veličinu poruka je kombinovanje provjere modela sa *lijenim tipovima podataka* (eng. lazy datatypes) [BMV04], koji omogućavaju modeliranje beskonačnih vrijednosti; u ovom slučaju je onda moguće posmatrati beskonačne *sisteme prelaska stanja* (eng. state-transition systems).

Za razliku od pristupa koji postavljaju ograničenja na veličinu sistema, Paulsonov *induktivni pristup* [Pau98] (eng. inductive approach) posmatra sve moguće *tragove protokola* (eng. protocol traces). Tragovi su nizovi događaja u protokolu, poput primanja, slanja ili memorisanja pojedinih poruka. Zatim se, koristeći indukciju i dokazivač teorema, dokazuje da sigurnosni zahtjevi važe za sve moguće tragove u protokolu. Značajna prednost u odnosu na prethodno nabrojane pristupe je što se sigurnost protokola dokazuje za neograničen broj istovremenih izvršenja i neograničenu dužinu poruka. Nažalost, dokazi u ovom pristupu prilično su komplikovani, i nije moguća njihova potpuna automatizacija (što uostalom slijedi i iz rezultata neodlučivosti). Osim toga, ako protokol nije siguran, odnosno postoje napadi na njega, rekonstrukcija napada u ovom metodu obično nije jednostavna.

Sada slijedi detaljniji prikaz nekoliko različitih pristupa i dva konkretna alata koji ih im-

plementiraju. Prvo, daćemo pregled programskog paketa AVISPA, koji omogućava verifikaciju pomoću više pozadinskih alata. Kako svi ovi alati postavljaju neka ograničenja na veličinu sistema, obično je poželjno pružiti i dokaz koji važi i bez ikakvih ograničenja. Kao što smo već naveli, jedan način za kreiranje takvog dokaza je primjena induktivnog modela. Mada sam metod nije neophodno vezan za neki konkretan alat, gotovo svi dokazi koji koriste ovaj metod napisani su u dokazivaču Isabelle/HOL. Ova veza nije slučajnost - autor ovog dokazivača ujedno je i autor induktivnog metoda. Stoga ćemo dati i kratak uvod u Isabelle/HOL, a zatim i opisati realizaciju induktivnog pristupa u ovom dokazivaču.

5.2 Verifikacija protokola u programskom paketu AVISPA

AVISPA (*Automated Verification of Internet Security Protocols and Applications*) [Arm+05] je paket programa specifično namijenjenih za ispitivanje sigurnosti kriptografskih protokola. AVISPA pruža dvije značajne pogodnosti:

- jednostavna je za korišćenje za dizajnere protokola. Jedino što je potrebno uraditi je modelirati protokol - analiza njegove ispravnosti je dalje potpuno automatizovana;
- isti model protokola se može proslijediti ka više pozadinskih alata u AVISPA paketu, od kojih svaki koristi drugačiju metodu verifikacije. Ovim se postiže veća pouzdanost u rezutate verifikacije.

Kriptografski protokoli se u AVISPA paketu modeliraju koristeći tzv. *High-Level Protocol Specification Language*, ili skraćeno *HLPSL* [Ohe05]. Specifikacije u ovom jeziku su zasnovane na principu *uloga* (eng. roles). Svaka vrsta učesnika u protokolu se modelira jednom ulogom. Svaka uloga je opisana svojim stanjem, te se za svaku definiše više prelazaka stanja, koji u pravilu odgovaraju porukama u protokolu. Zatim se modeliraju željeni sigurnosni zahtjevi za protokol. HLPSL specifikacija se zatim automatski prevodi u takozvani Intermediate Format (IF), koji modelira protokol kao neograničeni sistem prelazaka stanja. Ovaj format se zatim predaje na analizu nekom od pozadinskih alata. Analiza je potpuno automatizovana: alati sami pretražuju prostor stanja u pokušaju da nađu neko stanje koje krši specifikovane sigurnosne zahtjeve protokola. Svi alati koriste uobičajeni Dolev-Yao model.

Sigurnosni zahtjevi za protokol se u HLPSL jeziku navode posebim $secret$, $witness$ i $request$ događajima. Kada učesnik u protokolu šalje poruku za koju želi da ostane tajna dijeljena između nekog skupa učesnika, on generiše događaj oblika $secret(m, id, a, b)$. Značenje ovog događaja je da sadržaj poruke m mora ostati tajna, poznata samo učesnicima iz željenog skupa (u ovom slučaju, to su a i b). Ovom sigurnosnom zahtjevu se zatim dodjeljuje identifikator id. U slučaju da specifikovanim prelascima sistem može doći u stanje u kojem napadač ili neka treća strana saznaje sadržaj poruke m, a pritom se ne nalazi u specifikovanom skupu učesnika kojem se vjeruje, zahtjev tajnosti je prekršen.

Druga vrsta sigurnosnih zahtjeva koji se mogu modelirati u HLPSL jeziku su zahtjevi kako injektivne, tako i neinjektivne saglasnosti. Budući da se u HLPSL-u umjesto ovih termina koriste termini slaba i jaka autentifikacija, i mi ćemo se pridržavati tih naziva u ovom poglavlju. Zahtjevi autentifikacije se u HLSPL-u specifikuju pomoću $witness(a, b, id, m)$ i njihovih odgovarajućih $request(b, a, id, m)$ događaja. Događaj tipa $witness$ označava potvrdu od strane učesnika a da želi da komunicira sa korisnikom b, koristeći vrijednost m, u svrhu identifikovanu sa id. S druge strane, $request$ događaj znači da je b sada uvjeren da komunicira sa a, i, štaviše, da je a saglasan da se vrijednost m koristi u svrhu id. Na ovaj način, zahtjev slabe autentifikacije, koji je identifikovan sa id je prekršen kada god u sistemu postoji događaj $request(b, a, id, m)$ za koji ne postoji prethodni događaj $witness(a, b, id, m)$. Jaka autentifikacija se prevodi u sljedeći dodatni zahtjev: $request$ događaji se ne smiju pojaviti više puta nego odgovarajući $witness$ događaji.

Opisaćemo sada kako se kriptografski protokoli mogu modelirati u HLPSL jeziku, na primjeru Needham-Schroeder protokola (poglavlje 2.3), tačnije ispravljene verzije protokola prikazane u poglavlju 2.5. Podsjetimo, ova verzija se naziva i Needham-Schroeder-Lowe (NSL) protokol. Model NSL protkola je preuzet iz AVISPA biblioteke protokola[1], koja je uključena u instalaciju AVISPA paketa. Prikaz ćemo započeti ulogom inicijatora, vidljivoj na listingu 5.1.

```
role initiator (A, B: agent,
            Ka, Kb: public_key,
            SND, RCV: channel (dy))
played_by A def=

  local State : nat,
        Na, Nb: text

  init State := 0

  transition

    0.  State  = 0 /\ RCV(start) =|>
        State':= 2 /\ Na' := new() /\ SND({Na'.A}_Kb)
                   /\ secret(Na',na,{A,B})
                   /\ witness(A,B,responder_initiator_na,Na')

    2.  State  = 2 /\ RCV({Na.Nb'.B}_Ka) =|>
        State':= 4 /\ SND({Nb'}_Kb)
                   /\ request(A,B,initiator_responder_nb,Nb')

end role
```

Listing 5.1: Uloga inicijatora u HLPSL modelu NSL protokola

[1]AVISPA biblioteka protokola: `http://www.avispa-project.org/library/avispa-library.html`

Definicija uloge započinje ključnom rječju $role$, iza koje slijedi naziv uloge. Svaka uloga posjeduje određene parametre, kojima se opisuje jedno izvršavanje protokola. Parametri uloge inicijatora su identitet inicijatora A, identitet odgovarača B, kao i njihovi pripadajući javni ključevi (Ka i Kb). Za svaki parametar se specifikuje njegov tip (agent, javni ključ i slično). Osim toga, tu su i komunikacioni kanali SND i RCV koje u potpunosti kontroliše Dolev-Yao napadač. Parametri će biti zamijenjeni konkretnim vrijednostima nakon što se uloga instancira. Zatim definišemo koji od agenata zadanih kao parametri igra ovu ulogu (ključna riječ $played\_by$); u ovom slučaju, to je A.

Svaka instanca neke uloge se karakteriše svojim trenutnim stanjem. To stanje se može opisati određenim brojem lokalnih promjenjivih. Ove promjenjive se deklarišu ključnom rječju $local$, iza koje se navode imena promjenjivih i njihovih tipovi. U ovom slučaju, svaka instanca inicijatora će posjedovati lokalne promjenjive $State$, te Na i Nb. Promjenjive Na i Nb će odgovarati istoimenim vrijednostima u opisu protokola, a uloga promjenjive $State$ će biti objašnjena naknadno. Lokalnim promjenjivim pri instanciranju uloge možemo dodijeliti neku fiksiranu početnu vrijednost, pomoću ključne riječi $init$; u ovom slučaju, vrijednost promjenjive $State$ inicijalizujemo na 0.

Ključni dijelovi specifikacije uloge su prelasci između stanja, čiji je početak označen ključnom rječju $transition$. Uloga inicijatora ima dva moguća prelaska, označena sa 0 i 2. Prelazak se može aktivirati („ispaliti", eng. activate, fire) samo kada je ispunjen logički uslov koji se naziva *čuvar prelaska* (ili *okidač prelaska*, eng. guard, trigger). U HLPSL zapisu, čuvar i rezultat prelaska su odvojeni operatorom =|>. Čuvari prelaska su u tipičnom slučaju konjunkcije, u kojima se zahtijeva:

- da je primljena odgovarajuća poruka;
- da je instanca uloge procesirala prethodne poruke, i da se nalazi u odgovarajućem stanju za obradu primljene poruke. Ovo je neophodno kako da bi se osigurali prijem i slanje poruka u ispravnom redoslijedu, tako i da bi se spriječilo da čuvar prelaska neprekidno bude ispunjen. Tipičan način modeliranja ovakvog zahtjeva je postojanje pomoćne promjenjive koja identifikuje trenutno stanje instance uloge. U našem modelu iskorišćena je promjenjiva $State$.

Opišimo sada ukratko prelaske stanja u ulozi inicijatora.

1. Prelazak označen sa 0 odgovara slanju prve poruke protokola. Ovaj prelazak se aktivira samo ako se uloga nalazi u stanju 0, te po prijemu specijalne poruke $start$. Poruka $start$ je specifična za AVISPA modele, i prima se sa mreže (koju, podsjećamo, u potpunosti kontroliše napadač). U novom stanju, vrijednost promjenjive $State$ biće promijenjena na 2; nove vrijednosti promjenjivih se označavaju jednostrukim navodnicima. Pomoću HLPSL funkcije $new()$ generiše se novi slučajan broj, koji se dodjeljuje promjenjivoj Na. Potom se šalje prva poruka protokola i generišu dva događaja. Pomoću prvog od njih, događaja $secret$, zahtijeva se tajnost broja Na, a ovaj sigurnosni zahtjev se označava sa na. Osim toga, generiše se i garancija učesniku B da A zaista želi da komunicira sa njim, u svrhu označenu sa $responder\_initiator\_na$, koristeći vrijednost Na.

2. Prelazak označen sa 2 se aktivira kada je uloga u stanju 2, i kada primi drugu poruku protokola. Vrijednost za Nb' se implicitno ažurira iz sadržaja poruke (budući da je označena jednostrukim navodnikom), dok se zahtijeva da ostatak poruke odgovara postojećim vrijednostima lokalnih promjenjivih. Kao rezultat prelaska, vrijednost promjenjive $State$ se mijenja na 4, šalje se treća poruka protokola, a zatim se generiše $request$ događaj kojim se od učesnika B zahtijeva garancija da je namjeravao da komunicira sa A, i to koristeći broj Nb. Ovaj zahtjev je označen kao $initiator\_responder\_nb$.

Uloga odgovarača je prikazana na listingu 5.2.

```
role responder(A, B: agent,
         Ka, Kb: public_key,
         SND, RCV: channel (dy))
played_by B def=

  local State : nat,
        Na, Nb: text

  init State := 1

  transition

    1.  State  = 1 /\ RCV({Na'.A}_Kb) =|>
        State':= 3 /\ Nb' := new() /\ SND({Na'.Nb'.B}_Ka)
                   /\ secret(Nb',nb,{A,B})
                   /\ witness(B,A,initiator_responder_nb,Nb')

    3.  State  = 3 /\ RCV({Nb}_Kb) =|>
        State':= 5 /\ request(B,A,responder_initiator_na,Na)

end role
```

Listing 5.2: Uloga odgovarača u HLPSL modelu NSL protokola

Početne definicije uloge odgovarača su iste kao kod uloge inicijatora, pa ćemo ovdje samo istaknuti razlike. Promjenjiva $State$ se ovdje inicijalizuje na 1, a ne na 0. Napomenimo da je ovo učinjeno samo radi logičke veze između promjenjive i rednih brojeva poruka u protokolu; promjenjiva je lokalna i, što se HLPSL specifikacije tiče, nema nikakve veze sa promjenjivom $State$ uloge inicijatora. Uloga takođe posjeduje dva prelaska stanja, ali su njihove namjene, shodno svrsi uloge, drugačije.

1. Prelazak označen sa 1 odgovara prijemu prve i slanju druge poruke protokola. U čuvaru prelaska se zahtijeva da je uloga u početnom, odnosno stanju 1, te da je primljena prva poruka. Iz ove poruke se čita i nova vrijednost za Na. Rezultat prelaska je ažuriranje promjenjive $State$ na 3, generisanje vrijednosti za broj Nb, te slanje druge poruke na mrežu. Osim toga, događajem $secret$ se zahtijeva tajnost broja Nb.

Ovaj zahtjev je označen sa nb. Konačno, pruža se i garancija učesniku A da B želi da komunicira sa njim u svrhu označenu sa $initiator\_responder\_nb$, koristeći broj Nb. Primijetimo da inicijator traži upravo ovu garanciju u svom drugom prelasku stanja.

2. Prelazak označen sa 3 odgovara prijemu posljednje, treće poruke protokola. Uslov je uobičajenog oblika, a rezultat je $request$ događaj, kojim se od inicijatora zahtijeva garancija da želi da komunicira baš sa B, u svrhu $responder\_initiator\_na$, koristeći vrijednost Na koju je B primio u svom prvom prelasku stanja. Inicijator, sa svoje strane, ovu garanciju daje odmah po slanju prve poruke protokola, odnosno u svom prvom prelasku stanja.

HLPSL dodatno zahtijeva da se definiše jedna posebna uloga, zvana $session$, koja odgovara izvršavanju jedne sesije protokola. Može se definisati bilo kao osnovna uloga (koja se sastoji od samo jedne uloge), bilo kao složena (koja se sastoji od više drugih uloga). U slučaju NSL protokola, ona se definiše kao kompozicija uloga inicijatora i odgovarača (listing 5.3).

```
role session(A, B: agent, Ka, Kb: public_key) def=

  local SA, RA, SB, RB: channel (dy)

  composition

        initiator(A,B,Ka,Kb,SA,RA)
    /\  responder  (A,B,Ka,Kb,SB,RB)

end role
```

Listing 5.3: Model NSL sesije u HLPSL jeziku

Za analizu NSL protokola možemo koristiti tri paralelne sesije: jednu sa poštenim učesnicima, jednu sa zlonamjernim inicijatorom, te jednu sa zlonamjernim odgovaračem. U HLPSL-u, specifikacija paralelnih sesija se vrši kroz jednu dodatnu posebnu ulogu, koja se obično naziva $environment$ (listing 5.4).

Skup $intruder\_knowledge$ sa listinga 5.4 sadrži sve konstante koje napadač zna prije pokretanja protokola. U ovom slučaju, to su identiteti agenata a i b, njihovi javni ključevi, kao i javni i tajni ključ samog napadača.

Konačno, možemo uključiti ili isključiti provjeru određenih sigurnosnih zahtjeva u odjeljku specifikacije nazvanom $goal$ (listing 5.5). Zahtjeve je naravno potrebno prethodno specifikovati, pomoću $request$ i $secret$ događaja. Pri tom, pri provjeri zahtjeva autentifikacije može se zahtijevati jaka ili slaba autentikacija, komandama $authentication\_on$ i $weak\_authentication\_on$ respektivno.

Po završetku modeliranja, HLPSL specifikacije se automatski prevode u ekvivalentne IF specifikacije pomoću prevodioca nazvanog HLPSL2IF. Potom se za ispitivanje sigurnosti protokola može izabrati jedan od četiri alata iz AVISPA paketa:

```
role environment() def=

    const a, b          : agent,
          ka, kb, ki    : public_key

    intruder_knowledge = {a, b, ka, kb, ki, inv(ki)}

    composition

        session(a,b,ka,kb)
     /\ session(a,i,ka,ki)
     /\ session(i,b,ki,kb)

end role
```

Listing 5.4: Environment uloga u modelu NSL protokola u jeziku HLPSL

```
goal

  secrecy_of na, nb
  authentication_on initiator_responder_nb
  authentication_on responder_initiator_na

end goal
```

Listing 5.5: Specifikovani sigurnosni zahtjevi u HLPSL modelu NSL protokola

OFMC (On-the-fly Model-Checker) [BMV04] je alat za provjeru modela, koji koristi lijene tipove podataka i time omogućava provjeru protokola bez ograničenja na dužinu poruka. Osim toga, alat omogućava provjeru protokola kako u tipizovanim, tako i u netipizovanim modelima, ili, drugim riječima, sa i bez jake apstrakcije tipova (poglavlje 2.5). Provjera u netipizovanim modelima omogućava otkrivanje napadi greškom tipa.

U slučaju kršenja nekog zahtjeva, alat prikazuje niz koraka u protokolu koji su doveli do ovog kršenja, tzv. *trag napada* (eng. attack trace).

CL-AtSe (Constraing-Logic-based Attack Searcher) [Tur06] je alat zasnovan na metodama rješavanja ograničenja. Ni ovaj alat ne uvodi ograničenja na dužinu poruka, a protokole može da posmatra i u tipizovanim i u netipizovanim modelima. Takođe, u slučaju pronalaska napada prikazuje se njegov trag.

SATMC (SAT-based Model-Checker) gradi propozicionu formulu kojom se predstavljaju početno stanje, skup stanja koja predstavljaju kršenje sigurnosnih zahtjeva, te ograničeno razvijanje relacija prelaska stanja specifikovanih u IF formatu. Potom se koristi SAT rješavač kojim se pokušava naći napad na protokol, čiji se trag potom prikazuje. Ovaj alat podržava isključivo tipizovane modele.

TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) provjerava samo zahtjeve povjerljivosti, i to u tipizovanim modelima. Pri tom se znanje napadača modelira *regularnim jezicima nad stablima* (eng. regular tree languages). Najveći problem pri primjeni ovog alata predstavlja činjenica da se protokol ne predstavlja precizno, već svojim tzv. *preaproksimacijama* (eng. overapproximation). Stoga nije jasno da li napadi pronađeni ovim alatom zaista i predstavljaju napad na neki protokol, pogotovo što alat ne proizvodi nikakav trag napada.

Sve analize su potpuno automatizovane. Na primjeru Needham-Schroeder protokola, prva tri alata uspješno pronalaze napad na originalnu varijantu protokola, te utvrđuju sigurnost ispravljene varijante. TA4SP, s druge strane, utvrđuje sigurnost ispravljene varijante protokola, ali se analiza originalne varijante protokola završava bez utvrđenog odgovora (dobija se izlaz *inconclusive*). Postojanje nekog napada se može utvrditi tek nakon određenog „igranja" sa parametrima alata, ali pri tom, kao što je već diskutovano, ne možemo biti sigurni da li je to napad na protokol ili neku njegovu aproksimaciju.

5.3 Isabelle/HOL

Isabelle [NPW02] je tzv. *interaktivni dokazivač teorema* (eng. interactive theorem prover). Interaktivni dokazivači su programi koji podržavaju dokazivanje matematičkih teorema pomoću jedne vrste dijaloga između korisnika i računara. Korisnik specifikuje korake ili strukturu dokaza u notaciji sličnoj programskim jezicima, a sistem provjerava ispravnost svakog koraka i pokušava da dopuni nedostajuće detalje. Isabelle je napisana koristeći LCF pristup, što između ostalog znači da je zasnovana na malom logičkom jezgru, čiju je ispravnost relativno lako provjeriti. Isabelle je generički dokazivač; ovo znači da je u samu

Isabelle ugrađena samo *meta-logika*, nazvana Isabelle/Pure, te algoritmi dokazivanja za ovu logiku. Ova meta logika se zatim koristi za definisanje raznih tzv. *objektnih logika*, kao što su logika prvog reda (First Order Logic - FOL), logika višeg reda (Higher Order Logic - HOL), Zermelo-Fraenkel-ova teorija skupova (ZFC), logika izračunjivih funkcija u logici višeg reda (Higher Order Logic for Computable Functions - HOLCF) i tako dalje. Isabelle je napisana u programskom jeziku SML, i može biti proširena korisničkim SML programima koji sprovode simbolička izračunavanja nad teoremama na logički siguran način. Ovim se korisnicima dokazivača omogućava da sami uvode sopstvene procedure dokazivanja.

Najšire korišćena objektna logika je HOL, pa se i cijeli sistem koji podrazumijeva logiku višeg reda u okviru Isabelle obično naziva Isabelle/HOL. Ovu objektnu logiku možemo posmatrati kao kombinaciju logike i funkcionalnog programiranja. Detaljan opis sintakse i mogućnosti ovog sistema višestruko prevazilazi okvire ove teze; ovdje ćemo dati samo veoma kratak pregled koji će biti koristan za razumijevanje izloženih metoda verifikacije protokola. Pretpostavlja se da je čitalac upoznat sa osnovnim pojmovima funkcionalnog programiranja i matematičke logike.

U Isabelle notaciji, $t :: T$ označava term t tipa T. Oznaka tipa se može koristiti, između ostalog, i da se deklarišu nove konstante, npr. poput **consts** *pi* :: *real*. Tip *real* je tip kojim se, naravno, predstavljaju realni brojevi; tipom *nat* predstavljaju se prirodni brojevi.

Izraz **defs** $c \equiv t$ definiše konstantu c kao term t. Funkcionalna konstanta f se može definisati i kao $f\,x \equiv t$ umjesto $f \equiv \lambda\, x.\ t$. Deklaracija i definicija konstanti se takođe mogu uraditi i u jednom koraku, koristeći komandu **constdefs**.

Deklarisanje konstanti bez njihove definicije je tzv. *podspecifikacija* (underspecification), koja je veoma korisna u apstraktnom modeliranju. Obično se koristi u kombinaciji sa komandom **specification**, kojom se navode osobine deklarisanih konstanti, bez pružanja konkretnih vrijednosti za njih:

consts *K* :: *nat*
specification (*K*) *K-large*: $K > 1024$

Svaku takvu specifikaciju je, međutim, neophodno potkrijepiti dokazom da postoji vrijednost koja je ispunjava (u ovom slučaju, da postoji neko K koje je veće od 1024).

Promjenjive koje označavaju tipove se mogu prepoznati po apostrofu koji im prethodi, poput $'a$. Ako su zadana dva tipa $'a$ i $'b$, $'a \Rightarrow 'b$ označava tip (totalnih) funkcija iz $'a$ u $'b$, $'a \times 'b$ Dekartov proizvod, $'a$ *set* tip skupova čiji su elementi tipa $'a$, dok je $'a$ *list* tip lista čiji su elementi tipa $'a$. Tip funkcije $'a \Rightarrow 'b \Rightarrow 'c$ može se drugačije zapisati i kao $['a, 'b] \Rightarrow 'c$.

Ulazni jezik za Isabelle je bogat, a može se i proširiti dodatnom sintaksom prilikom uvođenja novih teorija. Ova mogućnost je iskorištena u bibliotekama za Isabelle/HOL koje sadrže formalizaciju matematičkih konstrukcija poput skupova i lista da bi se dobio ulazni jezik koji je gotovo identičan standardnoj matematičkoj notaciji, barem u mjeri u kojoj je to moguće, a da se još uvijek održi nedvosmislenost. Tako možemo koristiti konstrukcije poput $\{x.\ P\ x\}$ da označimo skup svih elemenata koji zadovoljavaju predikat P, za uniju dva skupa pišemo $A \cup B$ itd. Napomenimo samo nekoliko manje uobičajenih konstrukcija.

Sa f ` A se označava slika skupa A dobijena funkcijom f, *range f* označava kodomen ove funkcije, dok se sa *insert x G* označava skup dobijen ubacivanjem elementa x u skup G. Prazan skup je označen sa $\{\}$, a prazna lista sa $[]$. Cons operator je predstavljen sa #, pa tako $(x\#xs)$ predstavlja listu čija je glava x, a rep xs.

Postoji više mehanizama za definiciju novih tipova u Isabelle/HOL. Najjednostavniji su sinonimi tipova koji se uvode deklaracijama oblika **types** *T1* = *T2*. Drugi bitan mehanizam je uvođenje novih, tzv. induktivnih tipova podataka, pomoću ključne riječi **datatype**. Tako se na primjer može uvesti tip **datatype** *'a option* = *None* | *Some 'a*. Ovom definicijom se definiše tip *option* koji je polimorfan po promjenjivoj tipa *'a*. Osim toga, ovim se uvode i konstruktori *None* :: *'a option* i *Some* :: *'a* ⇒ *'a option*. Koristeći ovaj tip, možemo modelirati parcijalne funkcije iz *'a* u *'b* kao *'a* ⇒ *'b option*. Definicije tipova mogu biti i rekurzivne, poput sljedeće definicije tipa binarnih stabala.

datatype *'a btree* = *Leaf 'a* | *Node* (*'a btree*) (*'a btree*)

Implementacija za HOL sadrži i paket za rad sa slogovnim tipovima podataka (record). Tako sa:

record *point* =
  pointX :: *nat*
  pointY :: *nat*

možemo definisati novi slogovni tip nazvan *point*. Sada možemo kreirati novu vrijednost tipa *point* koristeći sintaksu ⦇ *pointX* = *1*, *pointY* = *2* ⦈. Osim toga, za svako polje u slogu se automatski definiše i pristupna funkcija, istog imena kao i polje. Tako su u gornjem primjeru automatski definisane funkcije *pointX* i *pointY*, obe tipa *point* ⇒ *nat*. Slogovi takođe podržavaju i ažuriranje pojedinačnih polja, kao u primjeru *p*⦇ *pointX* := *0* ⦈, kojim se kreira nova vrijednost tipa *point*, identična vrijednosti *p* izuzev možda u polju *pointX* (koje je ažurirano na vrijednost 0).

Isabelle/HOL podržava i induktivne skupove, koji se definišu sa jednim ili više monotonih, takozvanih *uvodnih pravila* (eng. introduction rules). Na primjer, ako nam je zadana neka relacija (u vidu skupa uređenih parova), možemo definisati novi skup kojim se definiše njeno tranzitivno zatvorenje.

inductive-set *trans-closure* :: (*'a* × *'a*)*set* ⇒ (*'a* × *'a*)*set*
for *S* :: (*'a* × *'a*)*set*
where
Inj: *p* ∈ *S* ⟹ *p* ∈ *trans-closure S*
|
Trans: ⟦ (*r*, *s*) ∈ *S*; (*s*, *t*) ∈ *trans-closure S* ⟧ ⟹ (*r*, *t*) ∈ *trans-closure S*

Definicija jednog ovakvog skupa se sastoji iz definicije jednog ili više baznih slučajeva, a potom i definicije nula ili više induktivnih koraka. Svakom koraku ili baznom slučaju odgovara po jedno uvodno pravilo, kojem se može dati i ime. U našem primjeru, (jedini) bazni slučaj odgovara uvodnom pravilu nazvanom *Inj*, dok (jedini) induktivni korak odgovara uvodnom pravilu nazvanom *Trans*.

Isabelle tvrđenja

Implementacija logike višeg reda u Isabelle koristi predefinisane konstrukcije meta logike za definiciju konstrukcija objektne logike. Isabelle tvrđenja su predstavljena u sljedećem obliku:

$\bigwedge x\ y\ ...\ [\![A; B; ...]\!] \Longrightarrow C$

Navedeni oblik se koristi kako za već dokazana ili aksiomatski pretpostavljena tvrđenja, tako i za ciljeve dokazivanja. Meta-logika koristi nestandardne oznake za logičke konstrukcije, da bi se izbjegao konflikt sa oznakama za konstrukcije objektne logike. Tako oznaka $\bigwedge$ označava univerzalnu kvantifikaciju u meta-logici, a oznaka $\Longrightarrow$ implikaciju. U gore navedenom primjeru, dakle, x i y su univerzalno kvantifikovane promjenjive, A i B su pretpostavke, a C je zaključak. Ukoliko postoji samo jedna pretpostavka, uglaste zagrade $[\![$ i $]\!]$ se mogu izostaviti. HOL, s druge strane, koristi uobičajene oznake za logičke veznike: $\wedge$, $\vee$ i $\longrightarrow$ za konjunkciju, disjunkciju, odnosno implikaciju, kao i standardne kvantifikatore $\forall$ i $\exists$.

Primijetimo i vezu sa induktivno definisanim skupovima: svako uvodno pravilo za induktivni skup je u stvari jedno tvrđenje (koje nije potrebno dokazivati).

U Isabelle/HOL se mogu definisati i aksiomatska tvrđenja, pomoću ključne riječi **axioms**.

axioms
axiom-of-choice: $(\forall\ x.\ \exists\ y.\ P\ x\ y) \Longrightarrow (\exists\ f.\ \forall\ x.\ P\ x\ (f\ x))$

Isabelle dokazi

Osnovni mehanizam rezonovanja u Isabelle/HOL je prirodna dedukcija. Iz aksioma HOL već su izvedena i dokazana uobičajena pravila zaključivanja, poput pravila *conjI*: $[\![P; Q]\!] \Longrightarrow P \wedge Q$ ili *conjunct2*: $[\![P \wedge Q]\!] \Longrightarrow Q$.

Nakon što dokažemo neko tvrđenje, možemo ga naravno kasnije koristiti u dokazu drugih. Postoji nekoliko uobičajenih načina za to. Tako se, na primjer, pravila *conjI* i *conjunct2* mogu iskoristiti za dokaz činjenice $[\![C \wedge A; D \wedge B]\!] \Longrightarrow A \wedge B$ na tri načina:

- **kao uvodna pravila** (introduction rule). U dokazima „unazad" (od zaključka ka pretpostavkama), možemo iskoristiti pravilo *conjI* da zamijenimo početni cilj sa dva nova cilja, $[\![C \wedge A; D \wedge B]\!] \Longrightarrow A$ i $[\![C \wedge A; D \wedge B]\!] \Longrightarrow B$
- **kao destruktivna pravila** (destruction rule). U dokazima „unaprijed", cilj $[\![C \wedge A; D \wedge B]\!] \Longrightarrow A$ možemo zamijeniti ciljem $[\![A; D \wedge B]\!] \Longrightarrow A$ koristeći pravilo *conjunct2* na $C \wedge A$, što kao rezultat proizvodi A.
- **kao pravilo eliminacije** (elimination rule). Upotrebljavajući pravilo *conjI* na ovaj način, cilj $[\![A; D \wedge B]\!] \Longrightarrow A \wedge B$ možemo zamijeniti novim ciljem $D \wedge B \Longrightarrow B$, koristeći pravilo *conjI*. Ovo je ekvivalentno korištenju pravila *conjI* kao uvodnog pravila, rješavanje prvog podcilja primjenom pretpostavke, te zatim uklanjanjem pretpostavke iz drugog cilja.

Radi preglednosti, osnovni elementi notacije su sumirani u dodatku A.

5.4 Paulsonov induktivni pristup

U radu [Pau98], Lawrence Paulson je predložio pristup formalizaciji kriptografskih protokola zasnovan na analiziranju mogućih *tragova protokola* (eng. protocol traces). Njegov pristup, koji se obično naziva *Paulsonov induktivni pristup* (eng. Paulson's Inductive Approach), kasnije je dalje razvio Giampaolo Bella. Rezultati ovih unapređenja su detaljno opisani u njegovoj doktorskoj disertaciji [Bel00], koja je i poslužila kao osnova za ovaj pregled. Teorije formalizacije protokola koje je G. Bella razvio dolaze uključene uz standardnu Isabelle distribuciju. Sada ćemo dati pregled ovog pristupa i njegove realizacije u Isabelle/HOL.

Počinjemo sa definicijom osnovnih tipova i skupova koji se pojavljuju u modelima protokola u ovom pristupu. Veći dio ovih definicija je nezavisan od protokola, i može se (uz možda manje izmjene) primijeniti na proizvoljan protokol.

U modelu je neophodno na neki način predstaviti učesnike u protokolu. Učesnici se uvijek mogu podijeliti na nekoliko vrsta. Tačan način na koji se ova podjela vrši zavisi od protokola do protokola, ali bi jedna generalna podjela, korisna za mnoge protokole, bila podjela na ,,obične" učesnike, napadača i povjerljivu treću stranu.

U Isabelle/HOL se ovakva podjela može izraziti sljedećim tipom:

datatype
$agent = Server \mid Friend\ nat \mid Spy$

Ovdje, *Server* je jedinstven agent, *Spy* je napadač, dok je *Friend* običan učesnik protokola. Ovi učesnici su parametrizovani prirodnim brojem, čime modeliramo prisustvo potencijalno beskonačno mnogo učesnika u protokolu.

Zatim je potrebno na neki način predstaviti i kriptografske ključeve. Budući da njihov broj nije ograničen, jedan pogodan način predstavljanja ključeva čine prirodni brojevi.

types
$key = nat$

Svakom ključu se pridružuje njegov inverzni ključ pomoću funkcije *invKey*, koja je involucija. Ovdje ćemo iskoristiti mogućnost podspecifikacije u Isabelle/HOL, jer nas interesuje samo osobina involucije.

consts
$invKey \quad :: key \Rightarrow key$

specification $(invKey)$
$invKey$: $invKey\ (invKey\ K) = K$

Sada se simetrični ključevi mogu definisati kao oni koji su inverzni sami sebi.

constdefs

symKeys :: *key set*
symKeys $\equiv \{K.\ invKey\ K = K\}$

U tipizovanim modelima (kakav je i induktivni pristup), svakoj poruci je pridružen odgovarajući tip. Vrste poruka ponovo variraju od protokola do protokola, ali postoji nekoliko tipova koji su zajednički za većinu protokola, kao što su složene ili šifrovane poruke.

Sljedeći tip definiše više ovakvih uobičajenih vrsta poruka.

datatype
msg = *Agent agent* --- Imena agenata
| *Nonce nat* --- Slučajni brojevi (ne mogu se pogoditi)
| *Number nat* --- Brojevi koji se mogu pogoditi
| *Key key* --- Kriptografski ključevi
| *Hash msg* --- Heširane poruke
| *MPair msg msg* --- Složene poruke
| *Crypt key msg* --- Šifrovane poruke (simetrično i asimetrično)

Za lakše čitanje, može se uvesti poseban Isabelle zapis za složene poruke:

translations
$\{\!|\ x, y, z\ |\!\} == \{\!|x, \{\!|y, z|\!\}|\!\}$
$\{\!|x, y|\!\} == MPair\ x\ y$

Slijede tri važna operatora, kojima se izražavaju mnoge definicije i tvrđenja u Paulsonovom modelu. Prvi je operator *parts*, kojim se induktivno definišu svi dijelovi svih poruka iz nekog skupa. Primijetimo da *parts* „prolazi" kroz enkripciju, tj. X je dio poruke *Crypt K X*. S druge strane, ključ K nije dio ove poruke (osim ako nije dio poruke X). Osim toga, ni X nije dio poruke *Hash X*.

Ovim operatorom se modeliraju sve komponente zadanog skupa poruka koje bi se potencijalno mogle saznati analizom njihovog sadržaja, ako bismo imali sve odgovarajuće ključeve. Dokazivanjem da $X \notin parts\ H$ se dokazuje da se X uopšte ne pojavljuje u H, pa se nikako ne može ni saznati iz ovog skupa (bar ne uz pretpostavku savršene kriptografije), čak ni uz pomoć dodatnih ključeva.

Još jednom, bitno je napomenuti da je ova definicija induktivna, tj. dekompozicija skupa poruka se ponavlja sve dok ne dođemo do njihovih sastavnih dijelova. Isto će važiti i za naredne dvije definicije.

inductive-set
parts :: *msg set* $\Rightarrow$ *msg set*
for H :: *msg set*
where
Inj: $X \in H \Longrightarrow X \in parts\ H$
| *Fst*: $\{\!|\ X, Y\ |\!\} \in parts\ H \Longrightarrow X \in parts\ H$
| *Snd*: $\{\!|\ X, Y\ |\!\} \in parts\ H \Longrightarrow Y \in parts\ H$
| *Body*: $Crypt\ K\ X \in parts\ H \Longrightarrow X \in parts\ H$

Drugi bitan operator je operator *analz*. Za zadani skup poruka H, ovaj operator generiše skup svih komponenti poruka koje se mogu saznati koristeći poruke iz H (uključujući i

ključeve). Za razliku od operatora *parts*, sadržaj šifrovane poruke se može saznati samo ako u skupu postoji i odgovarajući ključ za dešifrovanje. S druge strane primjećujemo da je, baš kao i za *parts*, iz heš vrijednosti neke poruke nemoguće saznati bilo kakve informacije o poruci, te da se parovi poruka mogu razdvojiti, baš kao i u *parts*. Važi $analz\ H \subseteq parts\ H$.

Operator *analz* je osnovno sredstvo za modeliranje znanja napadača; dokazivanjem da $X \notin analz\ H$ dokazuje se da ne postoji način da neki napadač sazna poruku X koristeći samo poruke iz skupa H, uz pretpostavku savršene kriptografije.

Ako skup poruka koje napadač poznaje označimo sa *known*, skup *analz known* možemo posmatrati kao neku vrstu kriptografskog zatvorenja ovog skupa znanja „na dole", budući da se saznaju komponente postojećih poruka.

inductive-set
$analz :: msg\ set \Rightarrow msg\ set$
for $H :: msg\ set$
where
Inj: $X \in H \Longrightarrow X \in analz\ H$
| Fst: $\{\!|\ X, Y\ |\!\} \in analz\ H \Longrightarrow X \in analz\ H$
| Snd: $\{\!|\ X, Y\ |\!\} \in analz\ H \Longrightarrow Y \in analz\ H$
| $Decrypt$:
$[\![\ Crypt\ K\ X \in analz\ H; Key\ (invKey\ K) \in analz\ H\ ]\!] \Longrightarrow X \in analz\ H$

Konačno, treći bitan operator je operator *synth*, koji predstavlja zatvorenje „na gore". Njime se modelira skup svih poruka koje se mogu konstruisati koristeći elemente zadanog skupa kao „materijal za gradnju". Dvije poznate poruke se mogu spojiti, a poznata poruka šifrovati poznatim ključem. Imena agenata i brojevi konstruisani sa *Number* se uvijek mogu sintetisati, odnosno pogoditi; međutim, brojevi konstruisani sa *Nonce* i ključevi ne mogu.

inductive-set
$synth :: msg\ set \Rightarrow msg\ set$
for $H :: msg\ set$
where
Inj: $X \in H \Longrightarrow X \in synth\ H$
| $Agent$: $Agent\ agt \in synth\ H$
| $Number$: $Number\ n \in synth\ H$
| $Hash$: $X \in synth\ H \Longrightarrow Hash\ X \in synth\ H$
| $MPair$: $[\![\ X \in synth\ H;\ Y \in synth\ H\ ]\!] \Longrightarrow \{\!|\ X, Y\ |\!\} \in synth\ H$
| $Crypt$: $[\![\ X \in synth\ H;\ Key(K) \in H\ ]\!] \Longrightarrow Crypt\ K\ X \in synth\ H$

Posljednji bitan operator definiše skup ključeva korisnih za dešifrovanje poruka iz zadanog skupa.

constdefs
$keysFor :: msg\ set \Rightarrow key\ set$
$keysFor\ H \equiv invKey\ `\ \{K.\ \exists X.\ Crypt\ K\ X \in H\}$

Paulsonov pristup modelira izvršavanje protokola kao niz događaja. Postoje tri vrste događaja: događaji slanja, primanja i pamćenja poruka.

datatype
event = *Says agent agent msg*
| *Gets agent msg*
| *Notes agent msg*

Jedna od pretpostavki Dolev-Yao modela je da napadač može da učestvuje u izvršenju protokola, kao legitiman učesnik. Taj zahtjev se modelira skupom kompromitovanih korisnika, koje kontroliše napadač. U Paulsonovom modelu ovaj skup je statičan, tj. „dobar" korisnik nikad ne postaje „loš". Ovaj skup sadrži napadača, ali po pretpostavci ne sadrži povjerljivu treću stranu. Osim ovoga, ne postoje druga ograničenja na broj ili vrstu kompromitovanih korisnika.

consts
bad :: *agent set*
specification (*bad*)
Spy-in-bad: *Spy* ∈ *bad*
Server-not-bad: *Server* ∉ *bad*

Funkcija *knows* definiše šta neki učesnik može da sazna iz nekog niza događaja. Definicija funkcije (koju ovdje zbog dužine preskačemo) je takva da napadač saznaje sve poruke na mreži, te da saznaje sve poruke koje pamte kompromitovani korisnici.

consts
knows :: *agent* ⇒ *event list* ⇒ *msg set*

Javne ključeve modeliramo funkcijama koje preslikavaju zadanog agenta u ključ. Zahtijevamo da su ključevi jedinstveni, tj. da je funkcija preslikavanja injektivna. Svaki agent posjeduje dva različita para ključeva: jedan za potpis, i jedan za šifrovanje.

datatype *keymode* = *Signature* | *Encryption*

consts
publicKey :: [*keymode*,*agent*] ⇒ *key*
pubEK ≡ *publicKey Encryption*
specification (*publicKey*)
injective-publicKey:
publicKey b A = *publicKey c A′* ⟹ *b*=*c* ∧ *A*=*A′*

Jedini aksiom koji se uvodi je da nijedan tajni ključ ne može biti jednak nekom javnom ključu. Bez ovog aksioma ne bi se moglo garantovati da su tajni ključevi zaista i tajni.

axioms
privateKey-neq-publicKey: *privateKey b A* ≠ *publicKey c A′*

Nakon ovih uvodnih definicija, možemo prikazati i modeliranje jednog konkretnog protokola. Kao primjer, ponovo ćemo iskoristiti Needham-Schroeder-Lowe protokol. Uopšteno, u Paulsonovom pristupu, protokol P se modelira kao induktivno definisan skup *tragova protokola* (eng. protocol traces), T_P. Induktivni koraci u definiciji ovog skupa odgovaraju slanju poruka učesnika protokola, kao i slanju poruka generisanih od strane napadača. Tragovi su definisani kao liste događaja (tip $event$).

inductive-set *ns-public* :: *event list set*
where
Nil: $[] \in$ *ns-public*
| *NS1*: $[\![$*evs1* $\in$ *ns-public*; *Nonce NA* $\notin$ *used evs1*$]\!]$
$\Longrightarrow$ *Notes A* (*Nonce NA*)
Says A B (*Crypt* (*pubEK B*) $\{\!|$*Nonce NA*, *Agent A*$|\!\}$)
evs1 $\in$ *ns-public*
| *NS2*: $[\![$*evs2* $\in$ *ns-public*; *Nonce NB* $\notin$ *used evs2*;
Gets B (*Crypt* (*pubEK B*) $\{\!|$*Nonce NA*, *Agent A*$|\!\}$) $\in$ *set evs2*$]\!]$
$\Longrightarrow$ *Notes B* (*Nonce NB*)
Says B A (*Crypt* (*pubEK A*) $\{\!|$*Nonce NA*, *Nonce NB*, *Agent B*$|\!\}$)
evs2 $\in$ *ns-public*
| *NS3*: $[\![$*evs3* $\in$ *ns-public*;
Says A B (*Crypt* (*pubEK B*) $\{\!|$*Nonce NA*, *Agent A*$|\!\}$) $\in$ *set evs3*;
Gets A (*Crypt* (*pubEK A*) $\{\!|$*Nonce NA*, *Nonce NB*, *Agent B*$|\!\}$)
$\in$ *set evs3*$]\!]$
$\Longrightarrow$ *Says A B* (*Crypt* (*pubEK B*) (*Nonce NB*))
evs3 $\in$ *ns-public*
| *Reception*:
$[|$ *evsr* $\in$ *ns-public*; *Says A B X* $\in$ *set evsr* $|]$
$==>$ *Gets B X* # *evsr* $\in$ *ns-public*
| *Fake*: $[\![$*evsf* $\in$ *ns-public*; *X* $\in$ *synth* (*analz* (*knows Spy evsf*))$]\!]$
$\Longrightarrow$ *Says Spy B X* # *evsf* $\in$ *ns-public*

Prvo uvodno pravilo (Nil) definiše početni trag, u kojem još uvijek nema događaja. Ovo je ujedno i bazni slučaj indukcije. Pravila označena sa $NS1$, $NS2$ i $NS3$ odgovaraju porukama protokola.

1. Pravilo $NS1$ definiše da se neki postojeći trag $evs1$ može produžiti slanjem prve poruke protokola od učesnika A ka učesniku B, koja se predstavlja događajem oblika $Says\ A\ B$; uslov pravila je da se slučajni broj NA nije nigdje koristio u tragu $evs1$. Pri tom, učesnik A pamti poslani broj (događaj oblika $Notes\ A$).

2. Uslov pravila $NS2$ je nešto komplikovaniji: zahtijeva se da je u postojećem tragu $evs2$ agent B primio prvu poruku (pri čemu se ne zna ko ju je zaista poslao, čime se modelira mogućnost lažnog predstavljanja od strane napadača), te da broj NB nije korišćen. Zaključak pravila je da se trag može produžiti drugom porukom protokola, poslanom od strane agenta B za agenta A, pri čemu će agent B zapamtiti poslanu vrijednost za NB.

3. Pravilo $NS3$ opisuje kada se postojeći trag $evs3$ može produžiti trećom porukom, i to od strane agenta A, za agenta B. Kao prvo, potrebno je da je agent A nekada ranije poslao prvu poruku agentu B. Potom, potrebno je da je A nekad primio poruku koja po obliku odgovara drugoj poruci protokola, te da se vrijednost za NA iz druge poruke slaže sa vrijednosti iz prve.

Pravilo $Reception$ jednostavno specifikuje da se neka poslana poruka može i primiti. Posljednje pravilo, $Fake$, odgovara generisanju lažnih poruka od strane napadača, zvanog

Spy. Lažne poruke se modeliraju pomoću operatora $synth$ i $analz$. Prvo, operatorom $analz$ napadač neprekidno „razbija" sve poruke koje su mu poznate (koje je vidio na mreži ili saznao od kompromitovanih korisnika protokola - funkcija $spies$), dešifrujući složene poruke i poruke šifrovane poznatim ključevima; tokom ovoga, napadač potencijalno doznaje nove ključeve, koje opet može koristiti za dešifrovanje. Proces se nastavlja dok se ne dođe do fiksne tačke, tj. do tačke u kojoj više napadač ne može saznati nove poruke. Tada se koristi operator $synth$, kojim se od ovih sastavnih dijelova generišu sve moguće lažne poruke. Napomenimo da je skup definisan operatorom $synth$ beskonačan, tj. da ne postoji nikakvo ograničenje na potencijalnu dužinu poruke. Iz navedenog, možemo zaključiti da napadač u ovom modelu ima sve mogućnosti Dolev-Yao napadača.

Primijetimo da agenti u ovom modelu ne posjeduju stanje, već se ono na neki način implicitno kodira pomoću postojećih događaja u tragu. Ovakvo predstavljanje (ili nepredstavljanje) stanja agenata ima jednu bitnu posljedicu. Čim je slanje neke poruke jednom omogućeno (tj. uslovi odgovarajućeg induktivnog koraka zadovoljeni), poruka će se poslati beskonačno mnogo puta. Tako je, na primjer, dovoljno da jednom budu zadovoljeni uslovi koraka $NS3$, pa da inicijator A počne da neprekidno šalje jednu te istu poruku. U ovom smislu, Paulsonov model predstavlja *preaproksimaciju* (eng. overapproximation) izvršenja stvarnog protokola, jer bi inicijator koji pamti svoje stanje jednom poslao ovu poruku, a zatim završio izvršavanje protokola. Ovakva preaproksimacija je ispravna u smislu da se osobine autentifikacije i tajnosti dokazane u ovom modelu prenose i na realnu implementaciju protokola. Međutim, kako je u pitanju preaproksimacija, u Paulsonovom modelu nije moguće dokazati sve osobine stvarnog protokola. Tipičan primjer je osobina injektivne saglasnosti, koju nije moguće dokazati koristeći induktivni pristup.

5.5 Verifikacija IPS protokola

U ostatku glave, posvetićemo se konkretnim rezultatima na verifikaciji IPS protokola. Ova verifikacija je sprovedena dvojako: prvo upotrebom AVISPA paketa, a potom i modeliranjem protokola u sistemu Isabelle/HOL.

5.5.1 AVISPA model IPS protokola

U svrhu ispitivanja sigurnosti IPS-a, modeliran je samo protokol za plaćanje, onako kako je opisan u poglavlju 3.4.3. Specifično, ovo znači da je apstrahovana konekcija između platne kapije i banaka, te da se posmatraju samo uloge kupca, prodavca i platne kapije. Osim toga, smatra se da je sadržaj narudžbe već dogovoren, tj. da i kupac i prodavac unaprijed znaju taj sadržaj, kao i ukupan iznos narudžbe. Modelirana su oba scenarija IPS protokola (poglavlje 3.4.3). Međutim, ovdje ćemo pažnju posvetiti samo prvom scenariju, budući da je model za ovaj scenario komplikovaniji.

Kako su osnove sintakse HLPSL specifikacija već obrađene u poglavlju 5.2, smatraćemo da su čitaocu već poznate, te ćemo samo istaknuti specifičnosti vezane za model IPS protokola. Shodno činjenici da u ovom protokolu postoje tri vrste učesnika, modelirane su i

tri različite uloge. HLPSL specifikacija za ulogu kupca prikazana je na listingu 5.6.

Nazivi parametara i lokalnih promjenjivih u svim ulogama su u direktnoj vezi sa nazivima iz opisa protokola. Prelasci stanja takođe odgovaraju porukama u protokolu. Tako prvi prelazak u ulozi kupca odgovara prvoj poruci protokola; drugi odgovara prijemu druge poruke, i slanju treće, dok treći prelazak odgovara prijemu posljednje, šeste poruke protokola. Napomenimo samo da su digitalni potpisi predstavljeni kao poruke koje su heširane, a zatim i šifrovane odgovarajućim tajnim ključevima. Primijetimo da su modelirani sljedeći zahtjevi:

- tajnost opisa narudžbe između kupca i prodavca;
- tajnost iznosa narudžbe između kupca, prodavca i platne kapije;
- tajnost podataka o plaćanju između kupca i platne kapije;
- autentifikacija između kupca i prodavca na skupu koji uključuje identifikator sesije i opis narudžbe;
- autentifikacija između kupca i platne kapije na poruci za plaćanje, kao i na odgovoru platne kapije.

Uloge prodavca i platne kapije su slične. Uloga prodavca je prikazana na listingu 5.7. Kao i uloga kupca, i uloga prodavca ima tri prelaska stanja. Razlika je, naravno, u porukama kojima prelasci odgovaraju:

1. Prvi prelazak stanja prodavca odgovara prijemu poruke 1, odnosno slanju poruke 2.
2. Drugi prelazak odgovara prijemu poruke 3, te slanju poruke 4.
3. Treći prelazak odgovara prijemu poruke 5 i slanju poruke 6.

Zahtjevi definisani u ovoj ulozi su isključivo zahtjevi autentifikacije postavljeni od prodavca, i to u odnosu na:

- kupca, na skupu koji uključuje identifikator sesije i opis narudžbe
- prodavca, na skupu koji uključuje identifikator sesije i iznos, te na odgovor platne kapije.

Uloga kapije je prikazana na listingu 5.8. Za razliku od prethodne dvije, ova uloga posjeduje samo jedan prelazak stanja, shodno ulozi platne kapije u protokolu. Prelazak odgovara prijemu poruke 5, odnosno slanju poruke 6. Uloga definiše dva zahtjeva autentifikacije koje postavlja platna kapija, i to u odnosu na:

- kupca, na skupu koji uključuje podatke o plaćanju;
- prodavca, na skupu koji uključuje identifikator sesije i iznos.

```
role cardholder(C,M,P: agent,
                CardInf: text,
                Amount : nat,
                OrderDesc : text,
                PubK_M,
                PubK_PG : public_key,
                PIN : message,
                Hash : hash_func
               ) played_by C def=
 local State : nat,
       Chall_CPG, Sid, Response : text,
       OI, PI : message,
       Kcm, Kcpg : symmetric_key,
       PubK_C : public_key,
       SND, RCV: channel (dy)
 init State := 0
 transition
 1. State = 0 /\ RCV(start)
    =|>
    State' := 2 /\ PubK_C' := new()
                /\ Kcm' := new()
                /\ SND({Kcm'}_PubK_M.{PubK_C'}_Kcm')
 2. State = 2   /\ RCV(Sid'.{Hash(Sid')}_inv(PubK_M))
    =|>
    State' := 4 /\ Chall_CPG' := new()
                /\ Kcpg' := new()
                /\ PI' := CardInf.Amount.Sid'.Chall_CPG'.PIN.M
                /\ OI' := OrderDesc.Amount
                /\ SND({Kcpg'}_PubK_PG.
                     {PI'.{Hash(PI')}_inv(PubK_C)}_Kcpg' .
                     {Hash(OI'.Sid')}_inv(PubK_C))
                /\ secret(OrderDesc,order,{C,M})
                /\ secret(Amount,order,{C,M,P})
                /\ secret(CardInf,payment,{C,P})
                /\ witness(C,M,cm_deal,Sid'.OI')
                /\ witness(C,P,cp_deal,PI')
 3. State = 4   /\ RCV({Response'.
        {Hash(Response'.Sid.Amount.Chall_CPG)}_inv(PubK_PG)}_Kcm
           )
    =|>
    State' := 11 /\ request(C,M,mc_deal,Sid.OI)
                 /\ request(C,P,pc_deal,PI)
                 /\ request(C,P,pc_response,Response')
end role
```

Listing 5.6: Uloga IPS kupca u HLPSL modelu

```
role merchant  (C,M,P: agent,
                Amount : nat,
                OrderDesc : text,
                PubK_M,
                PubK_PG : public_key,
        Hash : hash_func
               ) played_by M def=
 local State : nat,
       Sid : text ,
       Chall_CPG, Response : text,
       OI, PI : message,
       Kcm, Kmpg : symmetric_key,
       PubK_C : public_key,
       SND, RCV: channel (dy)
 init State := 1
 transition
 1. State = 1 /\ RCV({Kcm'}_PubK_M.{PubK_C'}_Kcm')
    =|>
    State' := 3 /\ Sid' := new()
         /\ OI' := OrderDesc.Amount
         /\ SND(Sid'.{Hash(Sid')}_inv(PubK_M))
 2. State = 3 /\ RCV( PI' . {Hash(OI.Sid)}_inv(PubK_C))
    =|>
    State' := 5 /\ Kmpg' := new()
         /\ SND(PI'.
              {Kmpg'}_PubK_PG.
             {PubK_C.{Hash(Amount.PubK_C.Sid)}_inv(PubK_M)}_Kmpg')
         /\ witness(M,C,mc_deal,Sid.OI)
         /\ witness(M,P,mp_deal,Sid.Amount)
 3. State = 5 /\ RCV({Response'.
    {Hash(Response'.Sid.Amount.Chall_CPG')}_inv(PubK_PG).C}_Kmpg
        )
    =|>
    State' := 10
         /\ SND({Response'.
         {Hash(Response'.Sid.Amount.Chall_CPG')}_inv(PubK_PG)}
             _Kcm)
         /\ request(M,C,cm_deal,Sid.OI)
         /\ request(M,P,pm_deal,Sid.Amount)
         /\ request(M,P,pm_response,Response')
end role
```

Listing 5.7: Uloga IPS prodavca u HLPSL modelu

```
role paymentgateway(C,M,P: agent,
                    PubK_M,
                    PubK_PG : public_key,
          Pin_func : hash_func,
          Hash : hash_func
                 ) played_by P def=
 local State : nat,
       Chall_CPG : text,
       PubK_C : public_key,
       Response : text ,
       OI,PI : message,
       CardInf, Sid : text,
       Kmpg, Kcpg : symmetric_key,
       Amount : nat,
       SND, RCV: channel (dy)
 init State := 6
 transition
 1. State = 6
    /\ RCV(
         {Kcpg'}_PubK_PG.
         {PI'.{Hash(PI')}_inv(PubK_C')}_Kcpg'.
         {Kmpg'}_PubK_PG.
         {PubK_C'.{Hash(Amount'.PubK_C'.Sid')}_inv(PubK_M)}_Kmpg
             ')
    /\ PI' = CardInf'.Amount'.Sid'.Chall_CPG'.
              Pin_func(C.CardInf').M
    =|>
    State' := 9 /\ Response' := new()
         /\ SND({Response'.
         {Hash(Response'.Sid'.Amount'.Chall_CPG')}_inv(PubK_PG)
         .C}_Kmpg')
         /\ request(P,C,cp_deal,PI')
         /\ witness(P,C,pc_deal,PI')
         /\ request(P,M,mp_deal,Sid'.Amount')
         /\ witness(P,M,pm_deal,Sid'.Amount')
         /\ witness(P,C,pc_response,Response')
         /\ witness(P,M,pm_response,Response')
end role
```

Listing 5.8: Uloga IPS platne kapije u HLPSL modelu

```
role session(C,M,P: agent,
             CardInf : text,
             Amount : nat,
             OrderDesc : text,
             PubK_M,
             PubK_PG : public_key,
             PIN : message,
             Pin_func : hash_func,
             Hash : hash_func
            ) def=
 composition
  cardholder(C,M,P, CardInf, Amount, OrderDesc,
             PubK_M, PubK_PG, PIN, Hash) /\
  merchant   (C,M,P, Amount, OrderDesc,
             PubK_M, PubK_PG, Hash) /\
  paymentgateway(C,M,P,
             PubK_M, PubK_PG, Pin_func, Hash)
end role
```

Listing 5.9: Model IPS sesije u jeziku HLPSL

U slučaju IPS protokola, uloga sesije se definiše kao kompozicija uloga kupca, prodavca i platne kapije (listing 5.9).

U analizi protokola, modelirane su tri paralelne sesije: jedna sa poštenim učesnicima, jedna sa zlonamjernim kupcem, i jedna sa zlonamjernim prodavcem. $Environment$ uloga za IPS model je prikazana na listingu 5.10. Pretpostavlja se da je korisnik platna kapija uvijek pošten (napadač i nikada ne igra ulogu ovog korisnika). Smatramo da je ovo razumna pretpostavka, iz dva razloga. Prvo, u svakom protokolu je neophodno uvesti neke pretpostavke o vjerodostojnim učesnicima u protokolu (npr. o sertifikacionim tijelima). Drugo, zlonamjerna platna kapija bi svakako imala pristup osjetljivim podacima, poput brojeva kreditnih kartica, pa u tom slučaju ne bi bilo imalo mnogo smisla govoriti o sigurnosti protokola u takvom scenariju.

Primijetimo da na listingu vrijednosti $amount1$ i $orderDesc1$ ne spadaju u skup $intruder\_knowledge$, tj. pretpostavljamo da se proces kupovine odvijao posredstvom nekog sigurnog kanala, na primjer HTTPS protokola. Modeliran je i drugi scenario, u kome su ovi podaci dostupni napadaču; u ovom modelu onda moramo ukloniti $secret$ događaje koji se odnose na iznos i sadržaj narudžbe.

Kao što se vidi iz modela, platna kapija je opremljena posebnom funkcijom nazvanom pin_func, koju koristimo da modeliramo provjeru dozvole upotrebe kreditnih kartica na osnovu kombinacije broja kartice i PIN koda. Ova funkcija omogućava platnoj kapija da uspostavi vezu između brojeva kreditnih kartica, identiteta korisnika i njihovih PIN brojeva.

```
role environment() def=

    const h: hash_func,
    pin_func : hash_func,
    order,payment,mp_deal,pm_deal,cm_deal,mc_deal : protocol_id,
    cp_deal,pc_deal,pc_response,pm_response : protocol_id,
    c,m,p: agent,
    pubk_m,pubk_pg,pubk_i: public_key,
    cardInf_c,cardInf_i,orderDesc1,orderDesc2,orderDesc3: text,
    orderDesc4 : text, amount4: nat,
    amount1,amount2,amount3 : nat

 intruder_knowledge = {c,m,p,pubk_m,pubk_pg,
        pubk_i,inv(pubk_i),cardInf_i,
        amount2,orderDesc2,amount3,orderDesc3,
        h, pin_func(i.cardInf_i)
 }
 composition
  session(c,m,p,cardInf_c,amount1,orderDesc1,
    pubk_m,pubk_pg,pin_func(c.cardInf_c),pin_func,h)
  /\ session(i,m,p,cardInf_i,amount2,orderDesc2,
    pubk_m,pubk_pg,pin_func(i.cardInf_i),pin_func,h)
  /\ session(c,i,p,cardInf_c,amount3,orderDesc3,
    pubk_i,pubk_pg,pin_func(c.cardInf_c),pin_func,h)
end role
```

Listing 5.10: Environment uloga u modelu IPS protokola u jeziku HLPSL

5.5.2 Rezultati verifikacije u AVISPA paketu

Na osnovu sigurnosnih garancija koje se traže od IPS protokola, navedenih u odjeljku 3.4.4, modelirani su sigurnosni zahtjevi vidljivi na listinzima 5.6, 5.7 i 5.8.

Za analizu protokola je prvo primijenjen tipizovani model, tj. model u kojem se poruke uvijek interpretiraju u skladu sa svojim tipom, i gdje se jedno polje ne može zamijeniti drugim, osim ako su istog tipa. U ovom slučaju, dva pozadinska AVISPA alata za verifikaciju, CL-AtSe i OFMC su verifikovali da je protokol siguran pod pretpostavkom ograničenog broja sesija. Drugim riječima, verifikovali su da protokol ispunjava sve modelirane zahtjeve. Analiza protokola je, po svemu sudeći, bila previše komplikovana za trenutne verzije druga dva alata (SATMC i TA4SP), budući da nijedan od njih nije dao nikakav odgovor ni nakon višečasovnog rada.

Kao što smo već napomenuli, neki od AVISPA alata dozvoljavaju i analizu protokola u netipizovanom modelu. Napadima ovog tipa smo već detaljnije pozabavili u poglavlju 2.5. I ova analiza je izvršena, i ponovo pokazala da je protokol siguran. Međutim, kako je već ranije objašnjeno, sigurnost protokola u ovakvom modelu veoma je krhka. Stoga je preporuka da se u svakom slučaju u implementaciji protokola koristi već pomenuta shema označavanja iz poglavlja 4.3.

Činjenica da je verifikacija u AVISPA-i potpuno automatska je bila od velike pomoći prilikom dizajniranja samog protokola, i omogućavala da se veoma brzo otkriju greške u njegovom razvoju. Međutim, mada je ova verifikacija uspješno izvršena, ona ipak nije u potpunosti zadovoljavajuća. Verifikacija u ovom slučaju znači samo da alati nisu uspjeli da pronađu napad u prostoru stanja koji su pretraživali. Ovo ostavlja otvorenim pitanje: da li je ovaj prostor zaista i dovoljan za utvrđivanje sigurnosti protokola? Odgovor je u ovom slučaju, barem djelomično, negativan. Na primjer, unutar pretraženog prostora, ne postoji napad kojim bi se ugrozio zahtjev da se platnoj kapiji garantuje injektivna saglasnost sa prodavcem ili kupcem, mada takav napad očito postoji: dovoljno je da napadač ponovi poruku 4 koju prodavac pošalje platnoj kapiji. U specifikaciji protokola, ovaj napad se sprečava provjerom jedinstvenosti identifikatora sesije. Takva provjera, međutim, ne može da se izrazi u HLPSL-u, i ni na koji način nije uključena u model protokola. Zaista, ubacivanjem još jedne sesije protokola u ulogu $environment$, oba alata (i OFMC i CL-AtSe) uspješno pronalaze napad. Ukoliko pak dodamo ovu novu sesiju, a zahtjev injektivne saglasnosti oslabimo u zahtjev neinjektivne saglasnosti, ni jedan od alata ne daje nikakav odgovor, ni poslije više sati rada. Čak i ukoliko bi i dali odgovor, ostaje naravno pitanje da li su i ove četiri sesije dovoljne da se pronađu svi mogući napadi na protokol.

Stoga ćemo se u narednim poglavljima okrenuti pokušaju verifikacije IPS protokola za neograničen broj paralelnih sesija.

5.5.3 Model IPS protokola u Isabelle/HOL

Za potrebe verifikacije IPS protokola, razvijen je poseban model u Isabelle/HOL. Ovo je u osnovi Dolev-Yao model: kriptografske operacije su predstavljene simbolički, a napadač ima sve uobičajene osobine Dolev-Yao napadača. Kao takav, model ima dosta sličnosti

sa modelima kreiranim Paulsonovim induktivnim pristupom, ali postoje i određene bitne razlike.

Ponovo, počinjemo sa definicijom osnovnih tipova i skupova koji se pojavljuju u modelu protokola. Ove definicije su mahom prilagođene iz teorija G. Belle, opisanih u 5.4; tačnije, uglavnom su prilagođene teorije razvijene za SET protokol (poglavlje 3.2.2). Ovdje ćemo istaknuti samo bitne razlike specifične za model IPS protokola. Kao i u opisanim modelima, tip učesnika protokola je definisan kao tip *agent*, međutim postoji više različitih tipova učesnika.

datatype
agent = *Cardholder nat* | *Merchant nat* | *PG nat* | *Spy*

Ovdje, *Cardholder* je kupac (vlasnik kartice), *Merchant* je prodavac, a *PG* je platna kapija. Izuzimajući napadača (*Spy*), svi učesnici su parametrizovani prirodnim brojem, čime modeliramo prisustvo potencijalno beskonačno mnogo učesnika u protokolu.

U našem modelu dodajemo još dvije vrste poruka, koje sadrže brojeve kartica i tajne PIN kodove. Napadač neće moći sam generisati poruke ovog oblika (pomoću operatora *synth*).

datatype
msg = *Agent agent* --- Imena agenata
| *Nonce nat* --- Slučajni brojevi (ne mogu se pogoditi)
| *Number nat* --- Brojevi koji se mogu pogoditi (npr. odgovor platne kapije)
| *CardInf nat* --- Informacije o kartici (ne mogu se pogoditi, npr. broj kartice)
| *Pin nat* --- Tajni PIN kodovi (ne mogu se pogoditi)
| *Key key* --- Kriptografski ključevi
| *Hash msg* --- Heširane poruke
| *MPair msg msg* --- Složene poruke
| *Crypt key msg* --- Šifrovane poruke (simetrično i asimetrično)

Po pretpostavci, skup kompromitovanih korisnika ne sadrži ni jednu platnu kapiju.

consts
bad :: *agent set*
specification (*bad*)
Spy-in-bad[*iff*]: *Spy* ∈ *bad*
PG-not-bad[*iff*]: *PG i* ∉ *bad*

Prilikom modeliranja javnih ključeva, nećemo praviti razliku između ključeva za šifrovanje i ključeva za potpis. Stoga će naša funkcija *publicKey* biti nešto jednostavnija.

consts
publicKey :: *agent* ⇒ *key*
invKey :: *key* ⇒ *key*

specification (*publicKey*)
inj-publicKey:
publicKey A = *publicKey A′* ⟹ *A* = *A′*

Potpise modeliramo kao kombinaciju šifrovanja i heširanja.

constdefs
sign :: *key* ⇒ *msg* ⇒ *msg*
sign K X ≡ {|*X*, *Crypt K* (*Hash X*) |}
signOnly :: *key* ⇒ *msg* ⇒ *msg*
signOnly K X ≡ *Crypt K* (*Hash X*)

Za modeliranje IPS protokola, biće nam potrebne još dvije funkcije, kojima vlasnicima kreditnih kartica pridružujemo njihove brojeve.

consts
card-inf-f :: *agent* ⇒ *nat*
pin-f :: *agent* ⇒ *nat*

Budući da su svi brojevi kartica različiti, navodimo i taj zahtjev (predikat *inj* označava injektivnost).

specification (*card-inf-f*)
inj-card-inf-f:
inj card-inf-f

Nakon ovih uvodnih definicija, okrećemo se konkretnom modelu IPS protokola. U modelu, svaki korisnik može da učestvuje u više (tačnije, neograničeno mnogo) paralelnih izvršavanja protokola. Jedno ovakvo izvršavanje ćemo nazivati sesija protokola. Sesije modeliramo prirodnim brojevima.

types
session = *nat*

Veoma bitna osobina našeg modela je da se stanje svakog učesnika protokola, i, štaviše, svake njegove sesije, eksplicitno modelira. Stanje jedne sesije modeliramo slogovnim tipom koji pamti sve ,,lokalne" promjenjive za jednu sesiju učesnika.

record *agent-locals* =
l-State :: *nat*
l-C :: *agent*
l-M :: *agent*
l-P :: *agent*
l-Amount :: *nat option*
l-OrderDesc :: *nat option*
l-CardInf :: *nat option*
l-PIN :: *nat option*
l-Chall-CPG :: *nat option*
l-Sid :: *nat option*
l-Response :: *nat option*
l-PubK-C :: *key option*
l-Kcm :: *key option*
l-Kmpg :: *key option*
l-Kcpg :: *key option*

Imena polja su u direktnoj vezi sa nazivima dijelova poruka IPS protokola (poglavlje 3.4.3). Izuzetak je polje *l-State*, u kojem se pamti dokle je agent stigao sa izvršenjem

trenutne sesije protokola. Ovo polje odgovara promjenjivoj $State$ u AVISPA/HLPSL modelu (poglavlje 5.5.1).

Primijetimo da su sva polja osim polja stanja i polja imena agenata opciona, tj. ne moraju imati vrijednost. Ovakav model je posljedica činjenice da se većina ovih promjenjivih generiše u toku izvršavanja sesije, ili se njihove vrijednosti dobijaju iz poruka drugih učesnika. Osim toga, ovo nam omogućava da koristimo istu vrstu sloga za predstavljanje lokalnog stanja svih učesnika u protokolu, iako se neka polja ne koriste od strane svih učesnika. Tako se, na primjer, polje *l-Kmpg*, u kojem se čuva simetrični ključ dijeljen između prodavca i platne kapije, uopšte ne koristi za sesije kupaca.

Definišemo zatim inicijalna stanja svakog od učesnika u protokolu. Svaka sesija nekog učesnika dobija određene parametre prije početka izvršavanja samog protokola. Za kupce, početni podaci sadrže podatke o narudžbi (uključujući i ukupan iznos) i informacije o kreditnoj kartici (uključujući i PIN kod). Početno stanje kupca je stanje 1, u kojem je kupac spreman za slanje prve poruke.

constdefs
init-locals-c :: *agent* ⇒ *agent* ⇒ *agent* ⇒ *nat* ⇒ *nat* ⇒ *agent-locals*
where
init-locals-c C M P amount order-desc ≡ (| *l-State* = *1*, *l-C* = *C*, *l-M* = *M*, *l-P* = *P*,
l-Amount = *Some amount*,
l-OrderDesc = *Some order-desc*,
l-CardInf = *Some* (*card-inf-f C*),
l-PIN = *Some* (*pin-f C*),
l-Chall-CPG = *None*,
l-Sid = *None*,
l-Response = *None*,
l-PubK-C = *None*,
l-Kcm = *None*,
l-Kmpg = *None*,
l-Kcpg = *None*
|)

Prodavac, sa druge strane, kao parametre dobija iznos i opis narudžbe, ali mu podaci o kartici nisu poznati. Početno stanje prodavca je stanje 2, u kojem on očekuje prijem prve poruke.

constdefs
init-locals-m :: *agent* ⇒ *agent* ⇒ *agent* ⇒ *nat* ⇒ *nat* ⇒ *agent-locals*
where
init-locals-m M C P amount order-desc ≡ (| *l-State* = *2*, *l-C* = *C*, *l-M* = *M*, *l-P* = *P*,
l-Amount = *Some amount*,
l-OrderDesc = *Some order-desc*,
l-CardInf = *None*,
l-PIN = *None*,
l-Chall-CPG = *None*,
l-Sid = *None*,
l-Response = *None*,
l-PubK-C = *None*,

$l\text{-}Kcm = None,$
$l\text{-}Kmpg = None,$
$l\text{-}Kcpg = None$
$\rparr$

Konačno, platna kapija zna sve podatke o karticama svih kupaca, ali ništa više. Drugim riječima, nisu joj unaprijed poznati ni iznos ni opis narudžbe. Početno stanje platne kapije je stanje 5, u kojem kapija očekuje prijem četvrte poruke protokola.

constdefs
init-locals-p :: *agent* ⇒ *agent* ⇒ *agent* ⇒ *nat* ⇒ *nat* ⇒ *agent-locals*
where
init-locals-p P C M amount order-desc ≡ $\lparr$ $l\text{-}State = 5, l\text{-}C = C, l\text{-}M = M, l\text{-}P = P,$
$l\text{-}Amount = None,$
$l\text{-}OrderDesc = None,$
$l\text{-}CardInf = Some\ (card\text{-}inf\text{-}f\ C),$
$l\text{-}PIN = Some\ (pin\text{-}f\ C),$
$l\text{-}Chall\text{-}CPG = None,$
$l\text{-}Sid = None,$
$l\text{-}Response = None,$
$l\text{-}PubK\text{-}C = None,$
$l\text{-}Kcm = None,$
$l\text{-}Kmpg = None,$
$l\text{-}Kcpg = None$
$\rparr$

Sada definišemo i tip sloga kojim predstavljamo stanje cjelokupnog sistema.

record *state* =
sessions :: *agent* ⇒ *session set*
agent-state :: *agent* ⇒ *session* ⇒ *agent-locals*
used :: *msg set*
knowledge :: *msg set*
sid-db :: *nat set*

Za svakog učesnika, definišemo skup svih sesija protokola u kojima učestvuje, ili je učestvovao (polje *sessions*). Potom, za svakog učesnika, i svaku njegovu sesiju, funkcija *agent-state* nam daje slog koji opisuje trenutno stanje sesije učesnika kao vrijednost tipa *agent-locals*, koji smo maloprije obradili. Ova funkcija je parcijalna, tj. nije definisana za sve sesije, već samo za one u kojima je korisnik učestvovao, odnosno sesije iz skupa određenog funkcijom *sessions*. Zbog toga će se u svim definicijama i formulacijama tvrđenja ove dvije funkcije pojavljivati zajedno.

Skup *used* je pomoćni skup koji nema svoj ekvivalent u opisu ili realnoj implementaciji protokola. Uloga mu je da osigura „svježe" slučajne brojeve i nove, generisane ključeve; tačnije, ovaj skup sadrži sve ključeve i slučajne brojeve koji su već iskorišćeni, tako da se uzimanjem neke vrijednosti koja **nije** iz ovog skupa osigurava njena „svježina". Svježina se u realnoj implementaciji naravno ne može garantovati, ali se ostvaruje sa velikom vjerovatnoćom.

Skup *knowledge* modelira trenutno znanje napadača. Budući da koristimo Dolev-Yao model, napadač u potpunosti kontroliše mrežu, što znači da može saznati sve poruke koje su korisnici poslali, te da bilo kojem korisniku može uputiti bilo koju generisanu poruku. Zajedno, ovo znači da je znanje napadača ekvivalentno sadržaju svih komunikacionih kanala, pa ćemo ga u modelu kao takvog i koristiti. Drugim riječima, kada neki korisnik pošalje neku poruku, tu poruku ćemo jednostavno umetnuti u znanje napadača. Obrnuto, ako korisnik čeka da primi neku poruku, za prijem te poruke je dovoljno da se nalazi u znanju napadača (jer napadač ionako može da tu poznatu poruku proslijedi korisniku).

Posljednji element sloga je skup *sid-db*, koji služi kao baza podataka za iskorišćene identifikatore sesije. Ovo je neophodno za modeliranje zahtjeva iz poruke 5 IPS protokola, u kojem se traži da platna kapija provjeri da li je identifikator sesije prethodno već iskorišćen. U ovom modelu zbog jednostavnosti koristimo globalnu bazu ovih identifikatora, zajedničku za sve platne kapije.

Definišemo zatim inicijalno stanje sistema, u kojem ne postoji nijedna sesija. Skup korišćenih poruka se inicijalizuje na skup javnih i tajnih ključeva, čime se osigurava da nijedan od ključeva slučajno generisanih kasnije, u toku izvršavanja protokola, neće biti jednak nekom postojećem tajnom ili javnom ključu. Na početku, jedina informacija koju napadač ima su javni ključevi svih korisnika, te tajni ključevi kompromitovanih agenata. Baza podataka o identifikatorima sesija je prazna.

definition *init-state* :: *state*
where
init-state ≡
(|
sessions = (λ *a*. {}),
agent-state = (λ *a s*. *undefined*),
used = *Key* ` (*range publicKey*) ∪ *Key* ` (*invKey* ` *range publicKey*),
knowledge = *Key* ` (*invKey* ` (*publicKey* ` *bad*)) ∪ *Key* ` (*range publicKey*),
sid-db = {}
|)

Funkcija *insert-known-set-cond* je pomoćna funkcija koja uslovno proširuje znanje napadača za neki skup poruka. Uslov koji se ispituje je da li je zadani agent kompromitovan. Ovu funkciju ćemo koristiti pri generisanju novih ključeva ili slučajnih brojeva, koji će postati poznati ako ih je generisao kompromitovan korisnik.

definition *insert-known-set-cond* :: *agent* ⇒ *msg set* ⇒ *state* ⇒ *state*
where
insert-known-set-cond A new s = (*if A* ∈ *bad*
then
s (‖ *knowledge* := *new* ∪ (*knowledge s*) ‖)
else s
)

Konačno, definišemo i ključni element našeg modela, skup svih dostižnih stanja cjelokupnog sistema. Ovaj skup se definiše induktivno. Baza indukcije je gore definisano inicijalno stanje (*init-state*). Induktivne korake možemo podijeliti u tri vrste:

- korak koji odgovara početku nove sesije protokola;
- koraci koji odgovaraju pojedinim porukama u protokolu;
- korak koji odgovara generisanju lažne poruke od strane napadača.

Svaki korak odgovara jednom prelasku stanja sistema. Uslovi („čuvari") koraka, osim u slučaju prve poruke i „specijalnih" koraka (za početak nove sesije i generisanje lažnih poruka) odgovaraju prijemu poruka očekivanih sa mreže. Tako čuvaru za drugu poruku protokola odgovara prijem prve poruke, itd. Osim toga, svi čuvari naravno uključuju zahtjev da je postojeće stanje dostižno. Drugi dio koraka opisuje novo stanje sistema, koje odgovara rezultatu prelaska.

Radi sticanja bolje slike o modelu, prikazaćemo sada nekoliko induktivnih koraka u definiciji skupa dostižnih stanja (skupa *reach*).

inductive-set *reach* :: *state set* **where**
init: --- Baza indukcije: početno stanje
init-state $\in$ *reach*
|
start: --- Početak jednog izvršavanja protokola (jedne sesije). Čuvar se sastoji od samo jednog uslova: da je sesija nova. U dobijenom stanju sistema, stanja svih učesnika se inicijalizuju pomoću odgovarajućih funkcija, a napadač uslovno saznaje podatke o kartici kupca, te podatke o iznosu i sadržaju narudžbe. Primijetimo da u ovom modelu i iznos i sadržaj modeliramo kao brojeve koje napadač može pogoditi, tj. ne zahtijevamo tajnost ovih brojeva.

$[\![s \in reach;$
$C = Cardholder\ i;$
$M = Merchant\ j;$
$P = PG\ k;$

$new\text{-}session \notin (\bigcup\ range\ (sessions\ s));$

$t = s(|$
$sessions := (sessions\ s)$
$(C := insert\ new\text{-}session\ (sessions\ s\ C),$
$M := insert\ new\text{-}session\ (sessions\ s\ M),$
$P := insert\ new\text{-}session\ (sessions\ s\ P)),$

$agent\text{-}state := (agent\text{-}state\ s)$
$(C := (agent\text{-}state\ s\ C)(new\text{-}session := init\text{-}locals\text{-}c\ C\ M\ P\ amount\ orderDesc),$
$M := (agent\text{-}state\ s\ M)(new\text{-}session := init\text{-}locals\text{-}m\ M\ C\ P\ amount\ orderDesc),$
$P := (agent\text{-}state\ s\ P)(new\text{-}session := init\text{-}locals\text{-}p\ P\ C\ M\ amount\ orderDesc)),$

$knowledge := knowledge\ ($
$insert\text{-}known\text{-}set\text{-}cond\ C$
$\{Number\ amount, Number\ orderDesc, CardInf\ (card\text{-}inf\text{-}f\ C), Pin\ (pin\text{-}f\ C)\}$
$(insert\text{-}known\text{-}set\text{-}cond\ M\ \{Number\ amount, Number\ orderDesc\}\ s))$
$|)$
$]\!] \Longrightarrow t \in reach$

|

msg1: --- Korak koji odgovara slanju prve poruke. U čuvaru prelaska, zahtijevamo da se u trenutnoj sesiji protokola kupac nalazi u stanju označenom sa 1, te da su ključevi *PubK-C*, odnosno *Kcm* svježi, čime simuliramo generisanje novih ključeva.
Rezultat prelaska je novo stanje u kojem napadač sada poznaje prvu poruku i, uslovno, generisane ključeve. Kupac pamti generisane ključeve i prelazi u novo stanje (označeno sa 3), u kojem očekuje prijem druge poruke protokola.

$$
\begin{array}{l}
[\![\ s \in \mathit{reach};\ C = \mathit{Cardholder}\ i;\ \mathit{session} \in \mathit{sessions}\ s\ C; \\
\\
\quad \mathit{l\text{-}State}\ (\mathit{agent\text{-}state}\ s\ C\ \mathit{session}) = 1; \\
\quad \mathit{l\text{-}M}\ (\mathit{agent\text{-}state}\ s\ C\ \mathit{session}) = M; \\
\\
\quad \mathit{PubK\text{-}M} = \mathit{publicKey}\ M; \\
\\
\quad \mathit{PubK\text{-}C} \notin \mathit{symKeys}; \\
\quad \mathit{Key}\ \mathit{PubK\text{-}C} \notin \mathit{used}\ s;\ \mathit{Key}\ (\mathit{invKey}\ \mathit{PubK\text{-}C}) \notin \mathit{used}\ s; \\
\quad \mathit{Key}\ \mathit{Kcm} \notin \mathit{used}\ s;\ \mathit{Kcm} \in \mathit{symKeys}; \\
\\
\quad \mathit{msg1} = \{|\ \mathit{Crypt}\ \mathit{PubK\text{-}M}\ (\mathit{Key}\ \mathit{Kcm}),\ \mathit{Crypt}\ \mathit{Kcm}\ (\mathit{Key}\ \mathit{PubK\text{-}C})\ |\}; \\
\\
\quad t = s(|\ \mathit{agent\text{-}state} := (\mathit{agent\text{-}state}\ s) \\
\qquad (C := (\mathit{agent\text{-}state}\ s\ C)(\mathit{session} := (\mathit{agent\text{-}state}\ s\ C\ \mathit{session}) \\
\qquad\quad (|\ \mathit{l\text{-}State} := 3,\ \mathit{l\text{-}PubK\text{-}C} := \mathit{Some}\ \mathit{PubK\text{-}C},\ \mathit{l\text{-}Kcm} := \mathit{Some}\ \mathit{Kcm}\ |))), \\
\\
\qquad \mathit{used} := (\mathit{used}\ s \cup \{\mathit{Key}\ \mathit{PubK\text{-}C},\ \mathit{Key}\ (\mathit{invKey}\ \mathit{PubK\text{-}C}),\ \mathit{Key}\ \mathit{Kcm}\}), \\
\\
\qquad \mathit{knowledge} := \mathit{insert}\ \mathit{msg1}\ (\mathit{knowledge} \\
\qquad\quad (\mathit{insert\text{-}known\text{-}set\text{-}cond}\ C\ \{\mathit{Key}\ \mathit{Kcm},\ \mathit{Key}\ \mathit{PubK\text{-}C},\ \mathit{Key}\ (\mathit{invKey}\ \mathit{PubK\text{-}C})\}\ s)) \\
\\
\quad |) \\
]\!] \Longrightarrow \\
\quad t \in \mathit{reach}
\end{array}
$$

--- ...

|

msg6-receipt: --- Prijem posljednje poruke protokola. Uslov je da se kupac nalazi u odgovarajućem stanju, te da je primio odgovarajuću poruku *Crypt Kcm* $\{\!|$ *Response*, *Response-Sig* $|\!\}$. Bitno je primijetiti da se ne zahtijeva samo da je oblik poruke odgovarajući, već i da su vrijednosti iz poruke saglasne sa vrijednostima koje je kupac prethodno zapamtio u ovoj sesiji.
Kao rezultat prijema poruke, kupac pamti odgovor platne kapije i završava izvršavanje ove sesije protokola. Završetak sesije se modelira prelaskom u stanje označeno sa 10, u kojem kupac više nije spreman za prijem ili slanje novih poruka (stanje 10 se ne poklapa ni sa jednim čuvarom prelazaka u našem modelu).

$$
\begin{array}{l}
[\![\ s \in \mathit{reach};\ C = \mathit{Cardholder}\ i;\ \mathit{session} \in \mathit{sessions}\ s\ C; \\
\\
\quad \mathit{l\text{-}State}\ (\mathit{agent\text{-}state}\ s\ C\ \mathit{session}) = 7;
\end{array}
$$

$l\text{-}Sid\ (agent\text{-}state\ s\ C\ session) = Some\ SidN;$
$l\text{-}Amount\ (agent\text{-}state\ s\ C\ session) = Some\ AmountN;$
$l\text{-}Chall\text{-}CPG\ (agent\text{-}state\ s\ C\ session) = Some\ Chall\text{-}CPGN;$
$l\text{-}P\ (agent\text{-}state\ s\ C\ session) = P;$
$PubK\text{-}P = publicKey\ P;$
$l\text{-}Kcm\ (agent\text{-}state\ s\ C\ session) = Some\ Kcm;$

$Response\text{-}Sig = signOnly\ (invKey\ PubK\text{-}P)\ \{|\ Response, Sid, Amount, Chall\text{-}CPG\ |\};$
$Crypt\ Kcm\ \{|\ Response, Response\text{-}Sig\ |\} \in knowledge\ s;$

$Response = Number\ n;$
$Sid = Nonce\ SidN;$
$Amount = Number\ AmountN;$
$Chall\text{-}CPG = Nonce\ Chall\text{-}CPGN;$

$t = s(|\ agent\text{-}state := (agent\text{-}state\ s)(C := (agent\text{-}state\ s\ C)$
$(session := (agent\text{-}state\ s\ C\ session)$
$(|\ l\text{-}State := 10, l\text{-}Response := Some\ n\ |)))$
$|)$

$[\![\Longrightarrow$
$t \in reach$
$|$

msg2: --- Uslov za prijem prve poruke je da se prodavac nalazi u stanju označenom sa 2, te da su ključevi u poruci ispravnih tipova (tj. da je *Kcm* simetričan, a *PubK-C* asimetričan). Dodatni uslov za slanje druge poruke je da je identifikator sesije svjež. Rezultat prelaska je novo stanje u kojem je napadaču poznata druga poruka protokola. Primijetimo da nije neophodno ručno ubacivati ključeve *Kcm* i *PubK-C* u znanje napadača ukoliko je prodavac kompromitovan - u tom slučaju napadač već može sam da ih sazna budući da poznaje tajni ključ prodavca.

$[\![s \in reach; M = Merchant\ j; session \in sessions\ s\ M;$

$l\text{-}State\ (agent\text{-}state\ s\ M\ session) = 2;$

$PubK\text{-}M = publicKey\ M;$

$\{|\ Crypt\ PubK\text{-}M\ (Key\ Kcm), Crypt\ Kcm\ (Key\ PubK\text{-}C)\ |\} \in knowledge\ s;$

$PubK\text{-}C \notin symKeys;$
$Kcm \in symKeys;$

$Nonce\ Sid \notin used\ s;$

$msg2 = sign\ (invKey\ PubK\text{-}M)\ (Nonce\ Sid);$

$t = s(|\ agent\text{-}state := (agent\text{-}state\ s)(M := (agent\text{-}state\ s\ M)$
$(session := (agent\text{-}state\ s\ M\ session)$
$(|\ l\text{-}State := 4, l\text{-}Sid := Some\ Sid, l\text{-}Kcm := Some\ Kcm, l\text{-}PubK\text{-}C := Some\ PubK\text{-}C\ |))),$

$knowledge := insert\ msg2$
$(knowledge\ (insert\text{-}known\text{-}set\text{-}cond\ M\ \{Nonce\ Sid\}\ s)),$

$used := insert\ (Nonce\ Sid)\ (used\ s)$

$|)$
$[\![\Longrightarrow$
$t \in reach$
--- ...
|

fake: --- Posljednji korak odgovara generisanju lažne poruke od strane napadača. Lažne poruke se modeliraju pomoću operatora *synth* i *analz*, kao u Paulsonovom modelu.

$[\![s \in reach; known = knowledge\ s; msg \in synth\ (analz\ known)]\!]$
$\Longrightarrow$
$s(|\ knowledge := insert\ msg\ known\ |) \in reach$

Koraci koji odgovaraju porukama 3, 4, 5 i 6 ovdje nisu prikazani, ali je njihova definicija u potpunosti analogna definicijama prikazanih koraka.

Sumirajmo sada glavne osobine našeg modela.

- Model ne postavlja nikakva ograničenja na dužinu poruka u protokolu, budući da poruke generisane operatorom *synth* mogu biti proizvoljne dužine.
- Model ne postavlja nikakva ograničenja na broj paralelnih izvršavanja protokola. Ovo predstavlja poboljšanje u odnosu na model korišćen pri analizi AVISPA paketom.
- Model eksplicitno predstavlja stanje svake sesije svakog od učesnika u protokolu. Kao što ćemo vidjeti, ovo predstavlja poboljšanje u odnosu na induktivni metod.
- Napadač ima potpunu kontrolu nad svim komunikacionim kanalima u modelu. Osim toga, u modelu važi pretpostavka savršene kriptografije. Za stanje s, skup svih poruka koje napadač može da sazna modeliramo kao $analz\ (knowledge\ s)$. Alternativno, uslov tajnosti poruke X izražavamo zahtjevom da $X \notin analz\ (knowledge\ s)$ bude invarijanta u skupu dostižnih stanja (skupu *reach*).

5.5.4 Poređenje sa Paulsonovim induktivnim pristupom

Kao što je već pomenuto u poglavlju 5.4, modeli protokola u Paulsonovom pristupu predstavljaju preaproksimaciju izvršenja stvarnih protokola, pa u takvim modelima nije moguće dokazati sve osobine stvarnih protokola; tipičan primjer je osobina injektivne saglasnosti. Osnovni uzrok ovog nedostatka je činjenica da učesnici u protokolu ne posjeduju stanje.

Naš model, s druge strane, eksplicitno kodira stanje svakog od učesnika, pa tako u odnosu na Paulsonov model omogućava dokazivanje više osobina originalnog protokola. To je ujedno i glavna prednost našeg modela u odnosu na Paulsonov. Druga prednost se sastoji u mogućnosti konciznijeg zapisa induktivnih koraka. Čuvari koji se odnose na prethodno poslane ili primljene poruke nisu neophodni; činjenice da su ove poruke prethodno poslane implicitno slijede iz trenutnog stanja agenta. Ova implicitnost, pak, za sobom povlači i neke negativne posljedice. Potrebno je dokazati veći broj tvrđenja kojima se ove činjenice dokazuju kao invarijante sistema. Osim toga, ponekad nije jednostavno „ubijediti" automatske Isabelle taktike dokazivanja da ove činjenice uzmu u obzir. Posljedica je da su dokazi često mnogo komplikovaniji od dokaza u Paulsonovom modelu. Zaista, trenutni dokaz osobina IPS protokola zahtijeva 2-3 puta više Isabelle koda od dokaza sličnih osobina SET protokola, a provjera naših dokaza zahtijeva 5-6 puta više vremena.

5.5.5 Važnije leme i rezultati

Na početku, neophodno je dokazati veći broj invarijanti vezanih uz lokalne promjenjive svakog učesnika protokola. Primjer ovakve invarijante je sljedeća, koja specifikuje da je vrijednost promjenjive *l-PubK-C*, ukoliko postoji (tj. ukoliko nije *None*) obavezno već poslana preko nekog komunikacionog kanala, tj. već sadržana u nekoj poruci sa mreže. Podsjetimo, mreža je u našem modelu ekvivalentna znanju napadača.

lemma *l-PubK-C-in-sent*: $[\![\ \textit{l-PubK-C}\ (\textit{agent-state}\ s\ A\ \textit{sess}) = \textit{Some PubK-C};$
$\textit{sess} \in \textit{sessions}\ s\ A;\ s \in \textit{reach}\ ]\!] \Longrightarrow$
$\textit{Key PubK-C} \in \textit{parts}\ (\textit{knowledge}\ s)$

Ova lema se dokazuje indukcijom po skupu dostižnih stanja. Budući da je ovaj skup i definisan upravo induktivno, ovo je prirodan, a često i jedini način dokazivanja tvrđenja vezanih za protokol. Nakon ove leme slijedi i veliki broj sličnih, kojim se dokazuju razne, uglavnom trivijalne osobine lokalnih promjenjivih: leme razlučuju slučajeve kada neka promjenjiva ima, a kada nema vrijednost, dokazuju da je vrijednost promjenjive primljena sa mreže, da napadač poznaje ključeve i slučajne brojeve koje generišu ili prime kompromitovani agenti i tako dalje.

Prva značajnija lema karakteriše operatore *parts*, *synth* i *analz* i u osnovi tvrdi da napadač ne može sam da generiše novi ključ, već da može da koristi samo već postojeće ključeve. Primijetimo da tačnost ovog tvrđenja ne predstavlja ograničenje u našem modelu, jer napadač može da iskoristi kompromitovane korisnike za generisanje ključeva.

lemma *key-synth-part-insert*: $[\![\ \textit{Key}\ k \notin \textit{parts}\ H;\ x \in \textit{synth}\ (\textit{analz}\ H);$
$\textit{Key}\ k \in \textit{parts}\ (\textit{insert}\ x\ H)\ ]\!] \Longrightarrow P$

Za uspješnu verifikaciju protokola, neophodan je neki izvor svježih ključeva. Kao što je već opisano, u tu svrhu se u našem modelu koristi skup *used*; ključ koji se ne nalazi u ovom skupu sigurno nikad nije poslan na mrežu, niti je korišćen za šifrovanje poruka na mreži. Prvi dio tvrđenja slijedi kontrapozicijom iz naredne leme, tačnije iz njenog dijela koji tvrdi $\textit{Key}\ k \in \textit{parts}\ (\textit{knowledge}\ s) \longrightarrow \textit{Key}\ k \in \textit{used}\ s$.

Sama formulacija leme je dosta komplikovana, jer je za njen dokaz neophodno da indukciju vršimo istovremeno kako po svim već viđenim ključevima, tako i po svim vrijednostima polja *l-PubK-C* svih agenata. Ovo je neophodno budući da prodavac prima vrijednost ovog ključa u prvoj poruci, a prosljeđuje je tek u četvrtoj. Ovo je primjer jedne leme koja „ispašta" zbog nedostatka eksplicitnosti u definiciji induktivnih koraka skupa dostižnih stanja (za razliku od definicija u Paulsonovom induktivnom pristupu).

lemma *known-Keys-l-PubK-C-are-used*[*rule-format*]: $s \in reach \Longrightarrow$
$\forall\ k\ A\ sess\ PubK\text{-}C.$
$(sess \in sessions\ s\ A \longrightarrow l\text{-}PubK\text{-}C\ (agent\text{-}state\ s\ A\ sess) = Some\ PubK\text{-}C \longrightarrow$
$Key\ PubK\text{-}C \in used\ s) \wedge$
$(Key\ k \in parts\ (knowledge\ s) \longrightarrow Key\ k \in used\ s)$

Drugi dio tvrđenja o svježim ključevima je sadržan u sljedećoj lemi. Podsjetimo, *keysFor* je definisan kao skup ključeva korisnih za dešifrovanje nekog skupa poruka.

lemma *new-keys-not-used* [*rule-format*,*simp*]:
$s \in reach \Longrightarrow$
$Key\ K \notin used\ s \longrightarrow K \notin keysFor\ (parts\ (knowledge\ s))$

Ove leme možemo i generalizovati tako da sadrže argument *KK*, što će nam kasnije olakšati dokazivanje leme o sigurnosti simetričnih ključeva.

lemma *gen-new-keys-not-used*:
$[\![Key\ K \notin used\ s; s \in reach]\!] \Longrightarrow$
$K \notin keysFor\ (parts\ (Key`KK \cup knowledge\ s))$
lemma *gen-new-keys-not-analzd*:
$[\![Key\ K \notin used\ s; s \in reach]\!]$
$\Longrightarrow K \notin keysFor\ (analz\ (Key`KK \cup knowledge\ s))$

Konačno, ove leme nam omogućavaju da nekorišćene ključeve „izvučemo" iz skupa *analz*. Ovo je primjer jedne tehničke leme koja je vrlo korisna prilikom rezonovanja u kasnijim dokazima.

lemma *analz-Key-image-insert-left*:
$[\![Key\ K \notin used\ s; s \in reach]\!]$
$\Longrightarrow analz\ (Key`(insert\ K\ KK) \cup knowledge\ s) =$
$insert\ (Key\ K)\ (analz\ (Key`KK \cup knowledge\ s))$

Konačno, dolazimo i do prvog bitnijeg rezultata naše verifikacije: simetrični ključevi ostaju sigurni. Iskorišćena formulacija ove činjenice je naoko prilično komplikovana, ali je ovakav oblik pogodan za kasniju upotrebu u automatskim taktikama dokazivanja. Suština leme leži u tvrđenju da je napadaču za otkrivanje bilo kojeg simetričnog ključa neophodno poznavanje tajnih ključeva. U osnovi, ovo tvrđenje slijedi iz činjenice da se simetrični ključevi u porukama IPS protokola uvijek pojavljuju šifrovani tajnim ključevima, ali formalan dokaz ovoga nije nimalo jednostavan. Najveći problem za dokazivanje predstavlja četvrta poruka protokola, iz dva razloga:

- Kao dio četvrte poruke prosljeđuje se poruka čiji je sadržaj nepoznat prodavcu, pa ne možemo ništa tvrditi o njenom sadržaju. Međutim, budući da nije dvostruko

šifrovana, ona je već poznata napadaču, tako da njenim saznavanjem napadač ne dobija ništa.

- Kao dio četvrte poruke prodavac prosljeđuje ključ *PubK-C*. U ovom trenutku, porijeklo ovog ključa nije sasvim jasno. Ovdje moramo iskoristiti jednu prethodno dokazanu lemu koja tvrdi da je ovaj ključ različit od tajnih ključeva svih poštenih korisnika.

lemma *symKey-compromise*:
$s \in reach \Longrightarrow$
$(\forall SK\ KK.\ SK \in symKeys \longrightarrow$
$(\forall K \in KK.\ K \notin invKey\ `\ (range\ publicKey)) \longrightarrow$
$(Key\ SK \in analz\ (Key`KK \cup (knowledge\ s))) =$
$(SK \in KK \vee Key\ SK \in analz\ (knowledge\ s)))$

Sada smo spremni da dokažemo i tajnost podataka o kreditnoj kartici. Ukoliko napadač može da sazna podatke o kreditnoj kartici kupca, kupac je obavezno kompromitovan; drugim riječima, napadač ne može da sazna podatke „dobrih" kupaca. Ovo je jedan od sigurnosnih zahtjeva postavljenih pred IPS protokol (poglavlje 3.4.4).

lemma *CardInf-confidentiality*:
$[\![\ CardInf\ (card\text{-}inf\text{-}f\ C) \in analz\ (knowledge\ s);$
$s \in reach\ ]\!] \Longrightarrow C \in bad$

Sada se možemo posvetiti dokazivanju autentifikacije korisnika, odnosno saglasnosti između njih. Prvo ćemo obraditi autentifikaciju između kupca i platne kapije. Neophodan element za dokazivanje ove autentifikacije je dokazivanje tajnosti slučajnog broja *Chall-CPG*, koji kupac šalje tajnoj kapiji.

lemma *Chall-CPG-confidentiality-1*: $s \in reach \Longrightarrow$
$\forall\ Chall\text{-}CPG\ sess\ C\ i.\ C = Cardholder\ i \longrightarrow$
$sess \in sessions\ s\ C \longrightarrow C \notin bad \longrightarrow$
$l\text{-}Chall\text{-}CPG\ (agent\text{-}state\ s\ C\ sess) = Some\ Chall\text{-}CPG \longrightarrow$
$Nonce\ Chall\text{-}CPG \notin analz(knowledge\ s)$

Sljedeći korak je dokazivanje da identifikator sesije (*Sid*) jedinstveno određuje sesiju platne kapije. Ovo je veoma značajno u daljim dokazima autentifikacije, tačnije u prelasku sa neinjektivne na injektivnu saglasnost, a dokaz slijedi iz činjenice da platne kapije čuvaju bazu podataka o iskorišćenim identifikatorima sesije.

lemma *Sid-determines-PG-run*:
$[\![\ l\text{-}Sid\ (agent\text{-}state\ s\ (PG\ i)\ sess1) = Some\ Sid;\ sess1 \in sessions\ s\ (PG\ i);$
$l\text{-}Sid\ (agent\text{-}state\ s\ (PG\ j)\ sess2) = Some\ Sid;\ sess2 \in sessions\ s\ (PG\ j);$
$s \in reach\ ]\!] \Longrightarrow$
$i = j \wedge sess1 = sess2$

Naredna lema nam pruža garanciju da, po prijemu potpisa odgovora platne kapije, kupac može da bude siguran da je platna kapija takođe nekada izvršila protokol, te da je prilikom tog izvršavanja bila saglasna sa svim vrijednostima sadržanim u potpisu. Ovo će biti ključni korak u dokazivanju neinjektivne saglasnosti za kupca u odnosu na platnu kapiju.

U daljem toku verifikacije dokazaćemo više sličnih lema, u kojima će pojedine poruke garantovati učešće njihovih pošiljalaca u protokolu.

Univerzalni kvantifikator vidljiv u formulaciji je neophodan pri dokazu leme.

lemma *Response-Sig-guarantees-PG-aliveness*:
$s \in reach \Longrightarrow$
$\forall$ *Response Sid Amount Chall-CPG i.*
signOnly (*invKey* (*publicKey* (*PG i*)))
$\{\!|$ *Number Response*, *Nonce Sid*, *Number Amount*, *Nonce Chall-CPG* $|\!\}$
$\in$ *parts* (*knowledge s*)
$\longrightarrow$
($\exists$ *sess.*
sess $\in$ *sessions s* (*PG i*) $\wedge$
l-State (*agent-state s* (*PG i*) *sess*) $= 8 \wedge$
l-Response (*agent-state s* (*PG i*) *sess*) $=$ *Some Response* $\wedge$
l-Sid (*agent-state s* (*PG i*) *sess*) $=$ *Some Sid* $\wedge$
l-Amount (*agent-state s* (*PG i*) *sess*) $=$ *Some Amount* $\wedge$
l-Chall-CPG (*agent-state s* (*PG i*) *sess*) $=$ *Some Chall-CPG*)

Iz prethodne dvije leme slijedi sljedeća: ne postoje dva različita potpisa odgovora platne kapije sa istim identifikatorom sesije.

lemma *Response-Sig-Sid-uniqueness*:
$[\![$ *signOnly* (*invKey* (*publicKey* (*PG i*)))
$\{\!|$ *Number Response1*, *Nonce Sid*, *Number Amount1*, *Nonce Chall-CPG1* $|\!\}$
$\in$ *parts* (*knowledge s*);
signOnly (*invKey* (*publicKey* (*PG j*)))
$\{\!|$ *Number Response2*, *Nonce Sid*, *Number Amount2*, *Nonce Chall-CPG2* $|\!\}$
$\in$ *parts* (*knowledge s*);
$s \in reach$ $]\!] \Longrightarrow$
$i = j \wedge Response1 = Response2 \wedge Amount1 = Amount2 \wedge Chall\text{-}CPG1 = Chall\text{-}CPG2$

Sada tvrdimo da je prijem potpisa odgovora platne kapije jedini način da kupac završi sesiju protokola (tj. da dođe u stanje 10). Dokaz je trivijalan i slijedi direktno iz definicije skupa dostižnih stanja.

lemma *State-10-Response-Sig-Receipt*:
$[\![$ *l-State* (*agent-state s C sessC*) $= 10$; *sessC* $\in$ *sessions s C*; $s \in reach$ $]\!] \Longrightarrow$
$\exists$ *Response Sid Amount Chall-CPG.*
signOnly (*invKey* (*publicKey* (*l-P* (*agent-state s C sessC*))))
$\{\!|$ *Number Response*, *Nonce Sid*, *Number Amount*, *Nonce Chall-CPG* $|\!\}$
$\in$ *parts* (*knowledge s*)
$\wedge$ *l-Sid* (*agent-state s C sessC*) $=$ *Some Sid*
$\wedge$ *l-Amount* (*agent-state s C sessC*) $=$ *Some Amount*
$\wedge$ *l-Chall-CPG* (*agent-state s C sessC*) $=$ *Some Chall-CPG*

Iz prethodnih lema slijedi i sledeća, koja tvrdi da je, ukoliko kupac završi izvršavanje protokola, i njegova platna kapija sigurno nekada izvršila ovaj protokol, te da je saglasna oko vrijednosti za identifikator sesije, ukupan iznos i slučajni broj *Chall-CPG*. Nažalost, ovo nam još uvijek nije dovoljno za neinjektivnu saglasnost: ne možemo sa sigurnošću

da tvrdimo da je platna kapija namjeravala da izvrši protokol baš sa ovim kupcem! Ovaj nedostatak ćemo ispraviti nešto kasnije. Bitan detalj koji će nam pomoći u tome je da možemo da utvrdimo da je platna kapija završila sa izvršavanjem protokola (tj. da se nalazi u stanju označenom sa 8).

lemma *non-injective-agreement-for-C-PG-weak*:
$[\![\textit{l-State}\ (\textit{agent-state}\ s\ C\ \textit{sessC}) = 10;\ \textit{sessC} \in \textit{sessions}\ s\ C;$
$\textit{l-Sid}\ (\textit{agent-state}\ s\ C\ \textit{sessC}) = \textit{Some SidN};$
$\textit{l-Amount}\ (\textit{agent-state}\ s\ C\ \textit{sessC}) = \textit{Some AmountN};$
$\textit{l-Chall-CPG}\ (\textit{agent-state}\ s\ C\ \textit{sessC}) = \textit{Some Chall-CPG};$
$\textit{l-P}\ (\textit{agent-state}\ s\ C\ \textit{sessC}) = (\textit{PG}\ k);$
$C = \textit{Cardholder}\ i;$
$s \in \textit{reach}$
$]\!] \Longrightarrow$
$\exists\ \textit{sessP}.$
$\textit{sessP} \in \textit{sessions}\ s\ (\textit{PG}\ k) \wedge$
$\textit{l-State}\ (\textit{agent-state}\ s\ (\textit{PG}\ k)\ \textit{sessP}) = 8 \wedge$
$\textit{l-Sid}\ (\textit{agent-state}\ s\ (\textit{PG}\ k)\ \textit{sessP}) = \textit{Some SidN} \wedge$
$\textit{l-Amount}\ (\textit{agent-state}\ s\ (\textit{PG}\ k)\ \textit{sessP}) = \textit{Some AmountN} \wedge$
$\textit{l-Chall-CPG}\ (\textit{agent-state}\ s\ (\textit{PG}\ k)\ \textit{sessP}) = \textit{Some Chall-CPG}$

Za kupce, vrijednost slučajnog broja *Chall-CPG* koju su izabrali jedinstveno određuje kako samog kupca, tako i njegovu sesiju. Dokaz zavisi od ispravne primjene skupa *used*.

lemma *Chall-CPG-determines-C-run*:
$[\![\textit{l-Chall-CPG}\ (\textit{agent-state}\ s\ C\ \textit{sess}) = \textit{Some Chall-CPG};\ \textit{sess} \in \textit{sessions}\ s\ C;$
$C = \textit{Cardholder}\ i;$
$\textit{l-Chall-CPG}\ (\textit{agent-state}\ s\ \textit{Ca}\ \textit{sessa}) = \textit{Some Chall-CPG};\ \textit{sessa} \in \textit{sessions}\ s\ \textit{Ca};$
$\textit{Ca} = \textit{Cardholder}\ \textit{ia};$
$s \in \textit{reach}\ ]\!] \Longrightarrow$
$C = \textit{Ca} \wedge \textit{sess} = \textit{sessa}$

Sada možemo da utvrdimo i sljedeće: ne postoje dva različita kupca (ili dvije sesije istog kupca) koja su završila izvršavanje protokola, a da pritom koriste isti identifikator sesije. U suprotnom, oba bi imala različite vrijednosti za *Chall-CPG*, a iz prethodnih teorema bismo zatim mogli da zaključimo da postoje dvije platne kapije sa istim identifikatorom sesije i različitom vrijednošću za *Chall-CPG*, što je netačno, budući da identifikator sesije jedinstveno određuje platnu kapiju. Ovo sada znači da je dovoljno dokazati neinjektivnu saglasnost za kupca, pa da automatski dobijemo i injektivnu saglasnost.

Dokaz ove saglasnosti će slijediti iz saglasnosti u drugom smjeru. Za dokaz te saglasnosti, pak, neophodno je pokazati da poruka o plaćanju garantuje učešće kupca u protokolu. Ovo tvrđenje je analogno tvrđenju da potpis odgovora platne kapije garantuje njeno učešće. Primijetimo da je ovu garanciju moguće dati jedino za nekompromitovane kupce (uslov $C \notin \textit{bad}$). Osim toga, ova garancija je nešto čvršća u odnosu na garanciju platne kapije, budući da poruka specifikuje učesnike M i P sa kojima kupac želi da komunicira.

lemma *PM-guarantees-C-aliveness*:
$s \in \textit{reach} \Longrightarrow$
$\forall\ \textit{Kcpg PubK-C i AmountN SidN Chall-CPG M i k}.$

$$
\begin{aligned}
&\mathit{Crypt}\ \mathit{Kcpg}\\
&\ (\mathit{sign}\ (\mathit{invKey}\ \mathit{PubK\text{-}C})\\
&\quad \{\!|\ \mathit{CardInf}\ (\mathit{card\text{-}inf\text{-}f}\ (\mathit{Cardholder}\ i)), \mathit{Number}\ \mathit{AmountN}, \mathit{Nonce}\ \mathit{SidN},\\
&\qquad \mathit{Nonce}\ \mathit{Chall\text{-}CPG}, \mathit{Pin}\ (\mathit{pin\text{-}f}\ (\mathit{Cardholder}\ i)), \mathit{Agent}\ M, \mathit{Agent}\ (\mathit{PG}\ k)\ |\!\})\\
&\quad \in \mathit{parts}\ (\mathit{knowledge}\ s) \longrightarrow\\
&\qquad \mathit{Cardholder}\ i \notin \mathit{bad} \longrightarrow\\
&\qquad\quad \mathit{Kcpg} \in \mathit{symKeys} \longrightarrow\\
&(\exists\ \mathit{sess}.\\
&\qquad \mathit{sess} \in \mathit{sessions}\ s\ (\mathit{Cardholder}\ i) \wedge\\
&\qquad \mathit{l\text{-}Kcpg}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sess}) = \mathit{Some}\ \mathit{Kcpg} \wedge\\
&\qquad \mathit{l\text{-}PubK\text{-}C}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sess}) = \mathit{Some}\ \mathit{PubK\text{-}C} \wedge\\
&\qquad \mathit{l\text{-}Chall\text{-}CPG}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sess}) = \mathit{Some}\ \mathit{Chall\text{-}CPG} \wedge\\
&\qquad \mathit{l\text{-}Sid}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sess}) = \mathit{Some}\ \mathit{SidN} \wedge\\
&\qquad \mathit{l\text{-}Amount}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sess}) = \mathit{Some}\ \mathit{AmountN} \wedge\\
&\qquad \mathit{l\text{-}P}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sess}) = \mathit{PG}\ k \wedge\\
&\qquad \mathit{l\text{-}M}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sess}) = M)
\end{aligned}
$$

Slično kao u dokazu za kupca, možemo pokazati da platna kapija završava svoje učešće u protokolu tek po prijemu poruke o plaćanju. Zajedno sa tvrđenjem koje garantuje učešće (poštenog) kupca koji pošalje ovakvu poruku, imamo dovoljno „dokaznog materijala" da platnoj kapiji garantujemo neinjektivnu saglasnost sa kupcem, i to na skupu $\{\mathit{Amount}, \mathit{Sid}, \mathit{Chall\text{-}CPG}\}$. U kombinaciji sa već dokazanom činjenicom da je *Sid* jedinstven za svako izvršavanje bilo koje platne kapije, ovo u stvari predstavlja dokaz injektivne saglasnosti.

lemma *non-injective-agreement-for-PG-C*:

$$
\begin{aligned}
&[\![\ \mathit{l\text{-}State}\ (\mathit{agent\text{-}state}\ s\ (\mathit{PG}\ k)\ \mathit{sess}) = 8; \mathit{sess} \in \mathit{sessions}\ s\ (\mathit{PG}\ k);\\
&\qquad \mathit{l\text{-}Sid}\ (\mathit{agent\text{-}state}\ s\ (\mathit{PG}\ k)\ \mathit{sess}) = \mathit{Some}\ \mathit{SidN};\\
&\qquad \mathit{l\text{-}Amount}\ (\mathit{agent\text{-}state}\ s\ (\mathit{PG}\ k)\ \mathit{sess}) = \mathit{Some}\ \mathit{AmountN};\\
&\qquad \mathit{l\text{-}Chall\text{-}CPG}\ (\mathit{agent\text{-}state}\ s\ (\mathit{PG}\ k)\ \mathit{sess}) = \mathit{Some}\ \mathit{Chall\text{-}CPG};\\
&\qquad \mathit{l\text{-}PubK\text{-}C}\ (\mathit{agent\text{-}state}\ s\ (\mathit{PG}\ k)\ \mathit{sess}) = \mathit{Some}\ \mathit{PubK\text{-}C};\\
&\qquad \mathit{l\text{-}C}\ (\mathit{agent\text{-}state}\ s\ (\mathit{PG}\ k)\ \mathit{sess}) = \mathit{Cardholder}\ i;\\
&\qquad \mathit{l\text{-}M}\ (\mathit{agent\text{-}state}\ s\ (\mathit{PG}\ k)\ \mathit{sess}) = \mathit{Merchant}\ j;\\
&\qquad \mathit{Cardholder}\ i \notin \mathit{bad};\\
&\qquad s \in \mathit{reach}\\
&\ ]\!] \Longrightarrow\\
&\exists\ \mathit{sessC}.\\
&\qquad (\mathit{sessC} \in \mathit{sessions}\ s\ (\mathit{Cardholder}\ i) \wedge\\
&\qquad \mathit{l\text{-}M}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sessC}) = \mathit{Merchant}\ j \wedge\\
&\qquad \mathit{l\text{-}P}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sessC}) = \mathit{PG}\ k \wedge\\
&\qquad \mathit{l\text{-}Sid}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sessC}) = \mathit{Some}\ \mathit{SidN} \wedge\\
&\qquad \mathit{l\text{-}Amount}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sessC}) = \mathit{Some}\ \mathit{AmountN} \wedge\\
&\qquad \mathit{l\text{-}Chall\text{-}CPG}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sessC}) = \mathit{Some}\ \mathit{Chall\text{-}CPG} \wedge\\
&\qquad \mathit{l\text{-}PubK\text{-}C}\ (\mathit{agent\text{-}state}\ s\ (\mathit{Cardholder}\ i)\ \mathit{sessC}) = \mathit{Some}\ \mathit{PubK\text{-}C})
\end{aligned}
$$

Sada možemo da kompletiramo i dokaz neinjektivne saglasnosti za kupca. Podsjetimo, sve što je nedostajalo je dokaz da je platna kapija namjeravala da komunicira baš sa tim kupcem. Dokaz slijedi iz prethodnog nepotpunog dokaza ove saglasnosti, dokaza saglasnosti u obrnutom smjeru i dokaza da ne postoje dva kupca sa istom izabranom vrijednošću za *Chall-CPG*. Osim toga, ovaj dokaz je ujedno i dokaz injektivne saglasnosti na skupu $\{\mathit{Sid},$

Amount$\}$, budući da smo dokazali da ne postoje dva poštena kupca sa istim identifikatorom sesije.

lemma *non-injective-agreement-for-C-PG*:
$[\![$ *l-State* (*agent-state s C sessC*) = *10*; *sessC* ∈ *sessions s C*;
l-Sid (*agent-state s C sessC*) = *Some SidN*;
l-Amount (*agent-state s C sessC*) = *Some AmountN*;
l-Chall-CPG (*agent-state s C sessC*) = *Some Chall-CPG*;
l-PubK-C (*agent-state s C sessC*) = *Some PubK-C*;
l-P (*agent-state s C sessC*) = (*PG k*);
l-M (*agent-state s C sessC*) = *M*;
C = *Cardholder i*;
C ∉ *bad*;
s ∈ *reach*
$]\!] \Longrightarrow$
∃ *sessP*.
sessP ∈ *sessions s* (*PG k*) ∧
l-State (*agent-state s* (*PG k*) *sessP*) = *8* ∧
l-Sid (*agent-state s* (*PG k*) *sessP*) = *Some SidN* ∧
l-Amount (*agent-state s* (*PG k*) *sessP*) = *Some AmountN* ∧
l-Chall-CPG (*agent-state s* (*PG k*) *sessP*) = *Some Chall-CPG* ∧
l-PubK-C (*agent-state s* (*PG k*) *sessP*) = *Some PubK-C* ∧
l-C (*agent-state s* (*PG k*) *sessP*) = *C* ∧
l-M (*agent-state s* (*PG k*) *sessP*) = *M*

Sada se okrećemo dokazu autentifikacije između platne kapije i prodavca. Osnov za tvrđenje je tajnost ključa *Kmpg* za prodavce koji nisu kompromitovani.

lemma *honest-M-Kmpg-secret*:
$[\![$ *Key Kmpg* ∈ *used s*; *Kmpg* ∈ *symKeys*;
l-Kmpg (*agent-state s M sess*) = *Some Kmpg*;
sess ∈ *sessions s M*; *M* = *Merchant j*; *s* ∈ *reach* $]\!]$
$\Longrightarrow$ *M* ∉ *bad* $\longrightarrow$ *Key Kmpg* ∉ *analz* (*knowledge s*)

Kao i u prethodnim slučajevima, biće nam potrebna neka poruka koja garantuje učešće prodavca u protokolu. U ovom slučaju, ovu ulogu će igrati poruka 4, tačnije potpis prodavca sadržan u ovoj poruci. Kao prvo, neophodno je dokazati da je ovaj potpis tajan.

Potom možemo dokazati da ovaj potpis pruža i garanciju o učešću prodavca u protokolu, kao i o vrijednostima za *Sid*, *Amount* i *PubK-C* sa kojima je saglasan. Nažalost, ovo samo po sebi neće biti dovoljno za utvrđivanje saglasnosti sa platnom kapijom, budući da potpis ne pruža informaciju o platnoj kapiji sa kojom je prodavac želio da komunicira.

lemma *good-M-signature-guarantees-M-aliveness*:
s ∈ *reach* $\Longrightarrow$
∀ *Kmpg PubK-C j AmountN SidN*.
Merchant j ∉ *bad* $\longrightarrow$ *Kmpg* ∈ *symKeys* $\longrightarrow$
Crypt Kmpg {| *Key PubK-C*,
signOnly (*invKey* (*publicKey* (*Merchant j*)))
{| *Number AmountN*, *Key PubK-C*, *Nonce SidN* |}
|} ∈ *parts* (*knowledge s*) $\longrightarrow$

$(\exists\ sess.\ sess \in sessions\ s\ (Merchant\ j) \wedge$
$l\text{-}Sid\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = Some\ SidN \wedge$
$l\text{-}Amount\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = Some\ AmountN \wedge$
$l\text{-}PubK\text{-}C\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = Some\ PubK\text{-}C \wedge$
$l\text{-}Kmpg\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = Some\ Kmpg)$

Tvrđenje možemo ojačati ukoliko šifrovani potpis poruke spojimo sa ključem *Kmpg* koji je korišćen za šifrovanje. Budući da je ovaj ključ tajan, te šifrovan javnim ključem kapije, možemo utvrditi i identitet kapije sa kojom je prodavac želio da komunicira. Dokaz ovog tvrđenja je jedan od komplikovanijih i zahtijeva razmatranje većeg broja slučajeva u induktivnom koraku generisanja lažnih poruka.

lemma *good-M-signature-guarantees-M-aliveness2*:
$s \in reach \Longrightarrow$
$\forall\ Kmpg\ PubK\text{-}C\ j\ k\ AmountN\ SidN.$
$Merchant\ j \notin bad \longrightarrow Kmpg \in symKeys \longrightarrow ($
$Crypt\ (publicKey\ (PG\ k))\ (Key\ Kmpg) \in parts\ (knowledge\ s)$
$\wedge$
$Crypt\ Kmpg\ \{\!|\ Key\ PubK\text{-}C,$
$signOnly\ (invKey\ (publicKey\ (Merchant\ j)))$
$\{\!|\ Number\ AmountN, Key\ PubK\text{-}C, Nonce\ SidN\ |\!\}$
$|\!\} \in parts\ (knowledge\ s)$
$) \longrightarrow$
$(\exists\ sess.\ sess \in sessions\ s\ (Merchant\ j) \wedge$
$l\text{-}Sid\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = Some\ SidN \wedge$
$l\text{-}Amount\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = Some\ AmountN \wedge$
$l\text{-}PubK\text{-}C\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = Some\ PubK\text{-}C \wedge$
$l\text{-}Kmpg\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = Some\ Kmpg \wedge$
$l\text{-}P\ (agent\text{-}state\ s\ (Merchant\ j)\ sess) = PG\ k)$

Sada u dva jednostavna koraka možemo doći do dokaza neinjektivne saglasnosti za platnu kapiju u odnosu na prodavca. Ponovo, ovo možemo tretirati kao dokaz injektivne saglasnosti zbog jedinstvenosti vrijednosti za *Sid* kod platnih kapija.

lemma *non-injective-agreement-for-PG-M*:
$[\![\ l\text{-}State\ (agent\text{-}state\ s\ (PG\ k)\ sess) = 8;\ sess \in sessions\ s\ (PG\ k);\ PubK\text{-}P = publicKey\ (PG\ k);$
$l\text{-}Sid\ (agent\text{-}state\ s\ (PG\ k)\ sess) = Some\ SidN;$
$l\text{-}Amount\ (agent\text{-}state\ s\ (PG\ k)\ sess) = Some\ AmountN;$
$l\text{-}PubK\text{-}C\ (agent\text{-}state\ s\ (PG\ k)\ sess) = Some\ PubK\text{-}C;$
$l\text{-}Kmpg\ (agent\text{-}state\ s\ (PG\ k)\ sess) = Some\ Kmpg;$
$l\text{-}M\ (agent\text{-}state\ s\ (PG\ k)\ sess) = M;$
$M = Merchant\ j;$
$M \notin bad;$
$s \in\ reach\ ]\!] \Longrightarrow$
$(\exists\ sessM.\ sessM \in sessions\ s\ (Merchant\ j) \wedge$
$l\text{-}Sid\ (agent\text{-}state\ s\ (Merchant\ j)\ sessM) = Some\ SidN \wedge$
$l\text{-}Amount\ (agent\text{-}state\ s\ (Merchant\ j)\ sessM) = Some\ AmountN \wedge$
$l\text{-}PubK\text{-}C\ (agent\text{-}state\ s\ (Merchant\ j)\ sessM) = Some\ PubK\text{-}C \wedge$
$l\text{-}Kmpg\ (agent\text{-}state\ s\ (Merchant\ j)\ sessM) = Some\ Kmpg \wedge$
$l\text{-}P\ (agent\text{-}state\ s\ (Merchant\ j)\ sessM) = (PG\ k))$

Početak dokaza u obrnutom smjeru prati ustaljeni šablon. Garanciju učešća platne kapije prodavcu pruža peta poruka. Iz nje možemo sada da garantujemo saglasnost platne kapije na skupu $\{Kmpg, Response, Sid, Amount, Chall\text{-}CPG, C\}$. Nažalost, prodavac iz ove poruke može da sazna samo skup $\{Kmpg, Response, C\}$, te da je platna kapija završila izvršavanje. Dokaz saglasnosti za ostale elemente će ići "okolo", slično kao i dokaz saglasnosti proveden za kupca.

lemma *Response-guarantees-PG-aliveness*:
$s \in reach \Longrightarrow$
$\forall$ *Kmpg ResponseN1 ResponseN2 k SidN AmountN Chall-CPG C*.
Kmpg $\in$ *symKeys* $\longrightarrow$ *Key Kmpg* $\notin$ *analz* (*knowledge s*) $\longrightarrow$
Crypt Kmpg $\{\!|$
Number ResponseN1,
signOnly (*invKey* (*publicKey* (*PG k*)))
$\{\!|$ *Number ResponseN2*, *Nonce SidN*, *Number AmountN*, *Nonce Chall-CPG* $|\!\}$,
Agent C
$|\!\}$ $\in$ *parts* (*knowledge s*)
$\longrightarrow$
($\exists$ *sess*. *sess* $\in$ *sessions s* (*PG k*) $\wedge$
l-Response (*agent-state s* (*PG k*) *sess*) = *Some ResponseN1* $\wedge$
ResponseN1 = *ResponseN2* $\wedge$
l-State (*agent-state s* (*PG k*) *sess*) = *8* $\wedge$
l-Kmpg (*agent-state s* (*PG k*) *sess*) = *Some Kmpg* $\wedge$
l-C (*agent-state s* (*PG k*) *sess*) = *C* $\wedge$
l-Sid (*agent-state s* (*PG k*) *sess*) = *Some SidN* $\wedge$
l-Amount (*agent-state s* (*PG k*) *sess*) = *Some AmountN* $\wedge$
l-Chall-CPG (*agent-state s* (*PG k*) *sess*) = *Some Chall-CPG*)

Lema o saglasnosti koristi već dokazanu saglasnost u obrnutom smjeru, posebno u odnosu na *Kmpg*, te dodatne leme kojima se dokazuje da vrijednost za *Kmpg* jedinstveno određuje sesiju prodavca. Druga lema, koja tvrdi da i *Sid* jedinstveno određuje sesiju prodavca, pretvara ovaj dokaz u dokaz injektivne saglasnosti.

lemma *non-injective-agreement-M-PG*:
$[\![$ *l-State* (*agent-state s* (*Merchant j*) *sessM*) = *9*; *sessM* $\in$ *sessions s* (*Merchant j*);
l-Kmpg (*agent-state s M sessM*) = *Some Kmpg*;
l-Response (*agent-state s M sessM*) = *Some ResponseN*;
l-C (*agent-state s* (*Merchant j*) *sessM*) = *C*;

l-Sid (*agent-state s* (*Merchant j*) *sessM*) = *Some SidN*;
l-Amount (*agent-state s* (*Merchant j*) *sessM*) = *Some AmountN*;
l-PubK-C (*agent-state s* (*Merchant j*) *sessM*) = *Some PubK-C*;
l-Kmpg (*agent-state s* (*Merchant j*) *sessM*) = *Some Kmpg*;
l-P (*agent-state s* (*Merchant j*) *sessM*) = *PG k*;
M = *Merchant j*;
M $\notin$ *bad*;
s $\in$ *reach* $]\!]$ $\Longrightarrow$
($\exists$ *sessP*. *sessP* $\in$ *sessions s* (*PG k*) $\wedge$
l-State (*agent-state s* (*PG k*) *sessP*) = *8* $\wedge$
l-Sid (*agent-state s* (*PG k*) *sessP*) = *Some SidN* $\wedge$

l-Amount (*agent-state s* (*PG k*) *sessP*) = *Some AmountN* ∧
l-C (*agent-state s* (*PG k*) *sessP*) = *C* ∧
l-PubK-C (*agent-state s* (*PG k*) *sessP*) = *Some PubK-C* ∧
l-M (*agent-state s* (*PG k*) *sessP*) = *M* ∧
l-Kmpg (*agent-state s* (*PG k*) *sessP*) = *Some Kmpg*)

Verifikaciju završavamo dokazom tvrđenja koje prodavcu garantuje neinjektivnu saglasnost sa kupcem, na skupu {*Sid*, *Amount*, *PubK-C*}. Tvrđenje se dobija kombinacijom prethodno dokazanih tvrđenja o saglasnosti između prodavca i kupca, odnosno između kupca i platne kapije. Kao i ranije, ova se lema u kombinaciji sa lemom o jedinstvenosti sesije kupca za zadanu vrijednost *Sid* pretvara u tvrđenje o injektivnoj saglasnosti.

lemma *non-injective-agreement-M-C*:
⟦ *l-State* (*agent-state s* (*Merchant j*) *sessM*) = *9*; *sessM* ∈ *sessions s* (*Merchant j*);
l-Kmpg (*agent-state s M sessM*) = *Some Kmpg*;
l-Response (*agent-state s M sessM*) = *Some ResponseN*;
l-C (*agent-state s* (*Merchant j*) *sessM*) = *C*;
C = *Cardholder i*;
C ∉ *bad*;

l-Sid (*agent-state s* (*Merchant j*) *sessM*) = *Some SidN*;
l-Amount (*agent-state s* (*Merchant j*) *sessM*) = *Some AmountN*;
l-PubK-C (*agent-state s* (*Merchant j*) *sessM*) = *Some PubK-C*;
l-Kmpg (*agent-state s* (*Merchant j*) *sessM*) = *Some Kmpg*;
l-P (*agent-state s* (*Merchant j*) *sessM*) = *PG k*;
M = *Merchant j*;
M ∉ *bad*;
s ∈ *reach* ⟧ ⟹
(∃ *sessC*. *sessC* ∈ *sessions s* (*Cardholder i*) ∧
l-Sid (*agent-state s* (*Cardholder i*) *sessC*) = *Some SidN* ∧
l-Amount (*agent-state s* (*Cardholder i*) *sessC*) = *Some AmountN* ∧
l-PubK-C (*agent-state s* (*Cardholder i*) *sessC*) = *Some PubK-C* ∧
l-M (*agent-state s* (*Cardholder i*) *sessC*) = *M*)

Glava 6

Zaključci i dalji rad

U ovom poglavlju izvodimo zaključke iz prezentovanog rada, sumiramo njegove doprinose i navodimo smjernice za potencijalna dalja istraživanja.

6.1 Zaključci

U radu smo posmatrali jedan uobičajen scenario kupovine putem Interneta, u kojem kupac za kupovinu koristi svoju kreditnu karticu. Posmatrajući postojeća rješenja, postavili smo određene zahtjeve za kreiranje novog protokola za plaćanje. Kao što je detaljno obrazloženo u ovoj tezi, možemo zaključiti da predloženo rješenje ispunjava navedene zahtjeve.

Osim toga, rad sproveden pri pisanju teze (kao i drugi rezultati na ovu temu) pokazuje koliko je dizajniranje ispravnih sigurnosnih protokola komplikovan zadatak. Naoko sitne razlike u sadržaju poruka mogu u potpunosti da ugroze sigurnost protokola, a time i sigurnost njegovih učesnika. Nalaženje potencijalnih grešaka je veoma komplikovano, a njihovo uklanjanje metodom pokušaja i pogrešaka je gotovo nemoguće. Umjesto toga, neophodan je sistematski pristup, a pomoć računara pri provjeri rezultata je od izuzetnog značaja.

Redoslijed glava u ovom radu uglavnom odgovara hronološkom redoslijedu u kojem su faze istraživanja sprovedene. Jedino odstupanje je verifikacija protokola u AVISPA paketu, koja je provedena paralelno sa dizajniranjem protokola. Ovo se pokazalo veoma značajnim: propusti u protokolu su veoma brzo uočeni, prije same implementacije, čime je ušteđeno vrijeme koje bi se provelo na implementaciju neispravnog protokola. Osim toga, analiza ovih propusta je omogućila dublji uvid u sigurnosne mehanizme protokola i sticanje bolje intuicije o njegovom radu; kao posljedica, rezultati verifikacije dobijeni u ograničenom okruženju kakvo modelira AVISPA su uspješno preneseni u okruženje bez takvih ograničenja (Isabelle model).

Ovakva metodologija razvoja bi mogla biti korisna za dizajnere nekih budućih sigurnosnih protokola. Ipak, prepreku predstavlja kompleksnost verifikacije koristeći Isabelle. Za upoznavanje sa Isabelle je bilo potrebno nekoliko mjeseci, a zatim i 2-3 mjeseca vrlo inten-

zivnog rada na verifikaciji samog protokola. Konačan rezultat je preko 2500 linija koda potrošenih samo za dokaze specifične za IPS protokol, ne računajući teorije preuzete iz Paulsonovog induktivnog pristupa, kao ni sam model protokola. Mada bi iskusni Isabelle korisnici vjerovatno utrošili znatno manje vremena, i mada bi se dobijeni kod revizijom sigurno mogao znatno skratiti, stiče se utisak da za primjene van akademskih istraživanja ovakav uloženi trud nije uvijek moguće opravdati, te da su potrebni bolji alati.

6.2 Doprinosi

Glava 3 opisuje novi sistem za plaćanje putem Interneta. Protokol je nastao izvlačenjem pouka iz postojećih standardnih sistema za plaćanjem; poređenje osobina ovih sistema i osobina predloženog sistema je dato u tabeli 6.1 (poželjnim osobinama u tabeli odgovara odgovor „da").

Osobina	Protokol			
	SSL/TLS	3-D Secure	SET	IPS
Ne zahtijeva dodatni softver	Da	Da	Ne	Da
Ne zahtijeva sertifikat kupca	Da	Da	Ne	Da
Autorizacija kreditne kartice	Ne	Da	Da	Da
Prodavac ne vidi podatke kartice	Ne	Ne	Da	Da
Nije neophodna treća strana	Da	Ne	Ne	Ne

Tabela 6.1: Poređenje IPS-a sa SSL/TLS, SET i 3-D Secure

Naravno, IPS protokol ima i određeni nedostatke. Prodavci moraju da prilagode svoj postojeći softver IPS protokolu, te moraju imati račune kod banaka sposobnih za izvršenje IPS transakcija. Takođe, i banke moraju instalirati platne kapije, ili iznajmiti ove kapije od trećih lica.

Glava 4 opisuje implementaciju IPS protokola. Bez obzira na sve potencijalne druge osobine protokola, nemoguće ga je koristiti bez realne implementacije. Implementacija je sprovedena koristeći dvije različite platforme, čime je ujedno dokazana i interoperabilnost protokola.

Glava 5 klasifikuje sigurnosne zahtjeve koji se postavljaju pred protokole, a zatim daje i pregled različitih aktuelnih metoda verifikacije. Ova poglavlja mogu poslužiti i kao kratak uvod u formalnu verifikaciju sigurnosnih protokola. Drugi bitan doprinos ovog poglavlja je formalna verifikacija samog IPS protokola; od toga, najveći doprinos leži u verifikaciji protokola pomoću dokazivača Isabelle, koja je sprovedena u modelu koji je izražajniji od standardnog Paulsonovog induktivnog modela.

6.3 Dalji rad

Stečeni uvidi iz razvoja IPS protokola mogli bi se iskoristiti za razvoj novih sigurnosnih protokola. Ideja o privremenom sertifikatu čije se vlasništvo naknadno verifikuje bi mogla potencijalno biti korisna i u drugim okruženjima.

Osim toga, i sam IPS protokol bi se mogao izmijeniti tako da pruži neke dodatne garancije svojim učesnicima. Jedna logična grupa poželjnih osobina su tzv. osobine *pravičnosti* (eng. fairness), koje garantuju učesnicima protokola da će predavanjem nekog dobra (npr. informacije) zauzvrat sigurno dobiti neko drugo dobro. Ovakve garancije su nepraktične i gotovo nemoguće za protokole koji se bave fizičkim dobrima, ali sasvim praktične za one koji se bave elektronskim.

Verifikacija IPS protokola bi se takođe mogla kompletirati sa nekoliko dokaza. Zbog nedostatka vremena, zasada u Isabelle modelu nismo uspjeli dokazati saglasnost kupaca i prodavaca oko narudžbi. Osim toga, dalji rad je potreban za osobinu neporecivosti. Ova osobina nije formalizovana (a samim tim ni dokazana). Sama njena formalizacija bi mogla da bude interesantna.

Dalji napredak je sigurno moguć u brzini i veličini dokaza protokola. Poboljšan model bi potom potencijalno mogao poslužiti kao osnova za neki širi okvir za verifikaciju protokola u Isabelle. Mnoga od dokazanih tvrđenja su trivijalna i mehanička; ovaj okvir bi mogao da pomogne kako njihovim automatskim dokazivanjem, tako i novim, efikasnijim metodama rezonovanja prilagođenim modelu.

Konačno, mada je u radu uspješno proveden dokaz ispravnosti protokola, rezonovanje o osobinama protokola je na veoma niskom nivou, i može da bude veoma naporno. Za širu primjenu su potrebne nove i bolje apstrakcije za rezonovanje o protokolima.

Dodatak A

Isabelle notacija

Oznaka	**Značenje**
pi :: *real*	term pi tipa real
$'a$	promjenjiva koja označava tip
$'a \Rightarrow 'b$	totalna funkcija iz $'a$ u $'b$
$['a, 'b] \Rightarrow 'c$	drugi način za zapis $'a \Rightarrow 'b \Rightarrow 'c$
$'a \times 'b$	tip uređenih parova gdje je prvi element iz $'a$, a drugi iz $'b$
'a set	tip (potencijalno beskonačnih) skupova čiji su elementi tipa 'a
$\{\}$	prazan skup
$\{x.\ P\ x\}$	skup svih elemenata koji zadovoljavaju predikat P
$A \cup B$	unija skupova A i B
f ` A	slika skupa A pod funkcijom f
range f	kodomen funkcije f
'a list	tip (konačnih) lista čiji su svi elementi tipa $'a$
[]	prazna lista
x#xs	lista čija je glava x, a rep xs
$(\!\vert\ x = 1, y = 2\ \vert\!)$	kreiranje novog sloga gdje se vrijednost (predefinisanog) polja x postavlja na 1, a (takođe predefinisanog) polja y na 2
$p(\!\vert\ x := 1\ \vert\!)$	nova vrijednost slogovnog tipa jednaka vrijednosti p, gdje je polje x ažurirano na *1*
$\bigwedge x\ y.\ [\![P; Q]\!] \Longrightarrow R$	tvrđenje sa pretpostavkama P i Q i zaključkom R, gdje su x i y univerzalno kvantifikovani

Tabela A.1: Pregled Isabelle notacije

Bibliografija

[AR02] Martin Abadi and Phillip Rogaway. "Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)". In: *Journal of Cryptology* 20.3 (July 2002), pp. 395–395. ISSN: 0933-2790. DOI: 10.1007/s00145-007-0203-0. URL: http://www.springerlink.com/index/10.1007/s00145-007-0203-0.

[Arm+05] A. Armando et al. "The Avispa Tool for the automated validation of internet security protocols and applications". In: *Proceedings of CAV 2005, Computer Aided Verification, LNCS 3576*. 2005.

[Bel00] Giampaolo Bella. "Inductive Verification of Cryptographic Protocols". PhD thesis. Cambridge University, 2000.

[Bel+00] M. Bellare et al. "Design, implementation, and deployment of the iKP secure electronic payment system". In: *IEEE Journal on Selected Areas in Communications* 18.4 (Apr. 2000), pp. 611–627. ISSN: 07338716. DOI: 10.1109/49.839936. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=839936.

[BMP01] Giampaolo Bella, Fabio Massacci, and Lawrence C. Paulson. "The Verification of an Industrial Payment Protocol The SET Purchase Phase". In: *Symposium A Quarterly Journal In Modern Foreign Literatures* (2001).

[BMP05] Giampaolo Bella, Fabio Massacci, and Lawrence C. Paulson. "An Overview of the Verification of SET". In: *International Journal of Information Security* 4 (2005).

[BMV04] David Basin, Sebastian Mödersheim, and Luca Viganò. "OFMC: A symbolic model checker for security protocols". In: *International Journal of Information Security* 4.3 (Dec. 2004), pp. 181–208. ISSN: 1615-5262. DOI: 10.1007/s10207-004-0055-7. URL: http://www.springerlink.com/index/10.1007/s10207-004-0055-7.

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. "A Composable Cryptographic Library with Nested Operations (Extended Abstract)". In: *10th ACM Conference on Computer and Communications Security (CCS)*. 2003.

[Bra93] Stefan Brands. "Untraceable Online Cash in Wallets with Observers". In: *Advances in Cryptology: CRYPTO '93*. Springer-Verlag, 1993, pp. 302–318.

[CFN89] David Chaum, Amos Fiat, and Moni Naor. “Untraceable Electronic Cash”. In: *CRYPTO*. 1989, pp. 319–327.

[Cha83] David Chaum. “Blind signatures for untraceable payments”. In: *Advances in Cryptology Proceedings of Crypto 82*. 1983, pp. 192–203.

[Cha90] David Chaum. “Online Cash Checks”. In: *Advances in Cryptology EUROCRYPT '89*. 1990, pp. 288–293.

[Cre06] Cas Cremers. “Scyther - Semantics and Verification of Security Protocols”. PhD thesis. Eindhoven, Technische Universiteit, 2006.

[DMG07] Zoran Đurić, Ognjen Marić, and Dragan Gašević. “Internet Payment System : A New Payment System for Internet Transactions”. In: *Universal Journal of Computer Science* 13.4 (2007), pp. 479–503.

[Dur+99] N. A. Durgin et al. “Undecidability of bounded security protocols”. In: *FLOC's Workshop on Formal Methods and Security Protocols*. 1999.

[DY83] D. Dolev and A. Yao. “On the security of public key protocols”. In: *IEEE Transactions on Information Theory* 29.2 (Mar. 1983), pp. 198–208. ISSN: 0018-9448. DOI: 10.1109/TIT.1983.1056650. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1056650.

[FK00] Michael Fritscher and Oliver Kump. “Security And Productivity Improvements – Sufficient For The Success Of Secure Electronic Transaction ?” In: *Proceedings of the 8th European Conference on Information Systems*. 2000, pp. 909–913.

[HLS00] J. Heather, G. Lowe, and S. Schneider. “How to prevent type flaw attacks on security protocols”. In: *Proceedings 13th IEEE Computer Security Foundations Workshop. CSFW-13* (2000), pp. 255–268. DOI: 10.1109/CSFW.2000.856942. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=856942.

[JM03] Pita Jarupunphol and Chris J Mitchell. “Measuring 3-D Secure and 3D SET Against e-Commerce End-user Requirements”. In: *Proceedings of the 8th Collaborative electronic commerce technology and research conference*. 2003, pp. 51–64.

[Kai95] R. Kailar. “Reasoning about accountability in protocols for electronic commerce”. In: *Security and Privacy, IEEE Symposium on* 0 (1995), p. 0236. ISSN: 1540-7993. DOI: http://doi.ieeecomputersociety.org/10.1109/SECPRI.1995.398936.

[Low96a] Gavin Lowe. “A Hierarchy of Authentication Specifications”. In: *CSFW '97: Proceedings of the 10th IEEE workshop on Computer Security Foundations*. IEEE Computer Society, 1996, p. 31.

[Low96b] Gavin Lowe. “Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR”. In: *Tools and Algorithms for the Construction and Analysis of Systems* (1996), pp. 147–166.

[Low96c] Gavin Lowe. “Some New Attacks upon Security Protocols 1 Introduction”. In: *CSFW '96: Proceedings of the 9th IEEE workshop on Computer Security Foundations*. 1996.

[MA10] Steven J Murdoch and Ross Anderson. “Verified by Visa and MasterCard SecureCode: Or, How Not to Design Authentication”. In: *Financial Cryptography and Data Security*. 2010, pp. 336–342.

[MBW98] Mark S. Merkow, Jim Breithaupt, and Ken L. Wheeler. *Building SET applications for secure transactions*. New York, NY, USA: John Wiley & Sons, Inc., 1998. ISBN: 0-471-28305-3.

[Mea96] Catherine A Meadows. “Analyzing the Needham-Schroeder Public Key Protocol : A Comparison of Two Approaches”. In: *European Symposium On Research In Computer Security*. Springer-Verlag, 1996, pp. 351–364.

[MMS] J.C. Mitchell, M. Mitchell, and U. Stern. “Automated Analysis of Cryptographic Protocols Using Murphi”. In: *Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No.97CB36097)*. IEEE Comput. Soc. Press, pp. 141–151. ISBN: 0-8186-7828-3. DOI: 10.1109/SECPRI.1997.601329. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=601329.

[MOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001. URL: http://www.cacr.math.uwaterloo.ca/hac/.

[MS01] Jonathan Millen and Vitaly Shmatikov. “Constraint solving for bounded-process cryptographic protocol analysis”. In: *Proceedings of the 8th ACM conference on Computer and Communications Security - CCS '01* (2001), p. 166. DOI: 10.1145/501983.502007. URL: http://portal.acm.org/citation.cfm?doid=501983.502007.

[MW04] Daniele Micciancio and Bogdan Warinschi. “Soundness of Formal Encryption in the Presence of Active Adversaries”. In: *In Proc. 1st Theory of Cryptography Conference (TCC), volume 2951 of LNCS*. micciancio:soundness:04: Springer, 2004, pp. 133–151.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL --- A Proof Assistant for Higher-Order Logic*. Vol. 2283. LNCS. Springer, 2002.

[NS78] Roger M. Needham and Michael D. Schroeder. “Using encryption for authentication in large networks of computers”. In: *Communications of the ACM* 21.12 (Dec. 1978), pp. 993–999. ISSN: 00010782. DOI: 10.1145/359657.359659. URL: http://portal.acm.org/citation.cfm?doid=359657.359659.

[Ohe05] David Von Oheimb. “The High-Level Protocol Specification Language HLPSL developed in the EU project AVISPA”. In: *Proceedings of APPSEM 2005 Workshop*. 2005.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. “Universal Electronic Cash”. In: *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1992, pp. 324–337. ISBN: 3-540-55188-3.

[OWL10] Stat OWL. *Web Browser Plugin Market Share*. 2010. URL: http://www.statowl.com/plugin_overview.php.

[Pau98] Lawrence C Paulson. “The Inductive Approach to Verifying Cryptographic Protocols”. In: *Journal of Computer Security* (1998), pp. 85–188.

[PSW00] B Pfitzmann, M Schunter, and M Waidner. “Cryptographic Security of Reactive Systems(Extended Abstract)”. In: *Electronic Notes in Theoretical Computer Science* 32 (2000), pp. 59–77. ISSN: 15710661. DOI: 10.1016/S1571-0661(04)00095-7. URL: http://linkinghub.elsevier.com/retrieve/pii/S1571066104000957.

[PSW95] Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. “How to break another "provably secure"payment system”. In: *EUROCRYPT'95: Proceedings of the 14th annual international conference on Theory and application of cryptographic techniques*. Saint-Malo, France: Springer-Verlag, 1995, pp. 121–132. ISBN: 3-540-59409-4.

[PW92] Birgit Pfitzmann and Michael Waidner. “How to Break and Repair a "Provably Secure"Untraceable Payment System (Extended Abstract)”. In: *Proceedings of Advances in Cryptology (CRYPTO '91), volume 576 of LNCS*. Springer, 1992, pp. 338–350.

[PW96] Birgit Pfitzmann and Michael Waidner. “Properties of Payment Systems — General Definition Sketch and Classification —”. In: (1996).

[Res01] Eric Rescorla. *SSL and TLS — Designing and Building Secure Systems*. Addison-Wesley, 2001.

[Set] *SET (Secure Electronic Transaction) specification*. 1997. URL: http://www.cl.cam.ac.uk/research/security/resources/SET/index.html.

[Tur06] Mathieu Turuani. “The CL-Atse Protocol Analyser”. In: *17th International Conference on Term Rewriting and Applications - RTA 2006*. 2006, pp. 277–286.