ZAVOD ZA ELEKTRONIKU, MIKROELEKTRONIKU, RAČUNALNE I INTELIGENTNE SUSTAVE
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
SVEUČILIŠTE U ZAGREBU

Uvod u IPsec standard i IKEv2 protokol

Ana Kukec

SEMINARSKI RAD IZ PREDMETA *OPERACIJSKI SUSTAVI 2*

Zagreb, 2006.

Sadržaj

1 IPsec (IP security) standard

IPsec (*Internet Protokol security*) je standard i skup protokola (opcionalan za IPv4, a obavezan za IPv6) koji obuhvaća mehanizme za zaštitu prometa na razini trećeg sloja OSI modela - kriptiranjem i/ili autentifikacijom IP paketa.
IPsec osigurava ispunjenje sljedećih sigurnosnih zahtjeva:

- tajnost (*confidentiality*; isključivo ovlaštena osoba može pristupiti podacima),
- besprijekornost (*integrity*; nemogućnost promijene podataka od strane neovlaštene osobe),
- autentičnost (*authentication*; verifikacija identiteta pošiljaoca),
- raspoloživost (*availability*; dostupnost podacima unatoč neočekivanim događajima, npr. DOS napad i sl.).

Cilj ovog dokumenta je objasniti na koji način IPsec osigurava ispunjenje ovih zahtjeva te zašto i u kojim situacijama su oni uopće potrebni.

Spomenuti zaštitni mehanizmi IPsec-a zasnivaju se na:

- Sigurnosnim protokolima: *Encapsulating Security Payload* (**ESP**) i *Authentication Header* (**AH**)),
- Specifičnoj arhitekturi: *Security Association* (**SA**) unosi u *Security Association Database* (**SAD**) u jezgru OS-a i *Security Policy* (**SP**) unosi u *Security Policy Database* (**SPD**) u jezgru OS-a.
- Algoritmima za autentifikaciju i kriptiranje (npr. DES, 3DES, RSA, DH, SHA1, MD5 i dr).
- Protokolima za razmjenu ključeva: Internet Key Exchange (**IKE**), Internet Key Exchange v2 (**IKEv2**), Kerberized Internet Negotiation of Keys (**KINK**), Just Fast Keying (JFK) i dr.

Literaturu o protokolima i svim ostalim detaljima vezanim uz IPsec najbolje je potražiti među RFC-ovima i draftovima. Razvoj IPsec-a započeo je s RFC1825[33] (*Security Architecture for the Internet Protocol*, Kolovoz 1995) i RFC2401[34] - RFC2412[35] (različiti RFC-ovi, Studeni 1998). Trenutno, razvoj je vrlo dinamičan, a zadnji objavljeni RFC i draft vezani su uz IKEv2 protokol: RFC4306[5] (*Internet Key Exchange (IKEv2) Protocol*, Prosinac 2005) i draft-hoffman-ikev2-1-00.txt[36] (Siječanj 2006). Činjenice iznesene u ovom dokumentu su u skladu su dokumentima RFC4306 u kombinaciji s RFC4301[6] (poznatijim kao draft RFC2401-bis koji je RFC4301 od prosinca 2005). Razlog tome je taj što ovaj dokument ima naglasak na IKEv2 protokolu, a on se implementira nad RFC2401-bis.
Zanimljiv draft, posebice za početnike (koji nažalost miruje od 2003) je draft: *Understanding IKEv2: Tutorial and rationale for decisions*[37]. Uz ovaj, svakako je korisno spomenuti: *IKEv2 Clarifications and Implementation Guidelines*[38] (Listopad 2005) u kojem su opisane sve nedoumice oko razumijevanja originalnog RFC-a IKEv2. Nakon velikog broja suhoparnih RFC-ova o IPsec-u, svakako pročitajte i šaljiv RFC2410[11]: *The NULL Encryption Algorithm and Its Use With IPsec*.

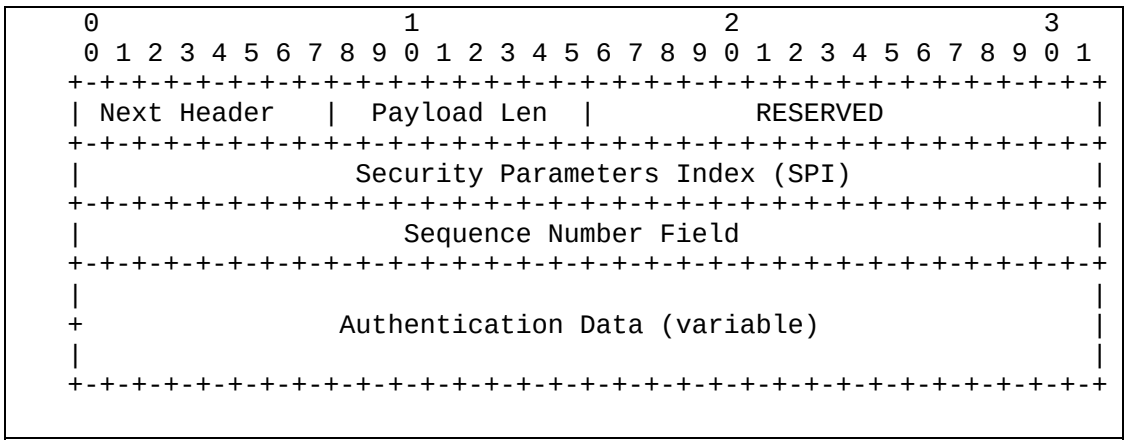
1.1 Sigurnosni protokoli AH i ESP

Sigurnosni protokoli AH i ESP su protokoli trećeg sloja OSI modela i osiguravaju zaštitu toka IP paketa. AH protokol je Internet (IP) protokol s brojem 51 koji osigurava besprijekornost i autentičnost IP datagrama. ESP protokol je Internet (IP) protokol s brojem 50 koji uz spomenutu autentifikaciju podataka koja je opcionalna, primarno osigurava tajnost IP datagrama. AH ili ESP protokol je zapravo jedan od kriterija "politike" unesene u jezgru operacijskog sustava. Uspješna komunikacija između Alice i Boba će biti uspostavljena samo onda kada i Alice i Bob za tu specifičnu komunikaciju imaju u jezgri postavljeni zahtjev za istim sigurnosnim protokolom: AH ili ESP. Uobičajeno je da se koriste nezavisno, no postoji i mogućnost njihovog istovremenog korištenja. RFC4301[6] ne spominje pregovaranje ESP i AH protokola istovremeno (u jednoj sigurnoj vezi), što ima za posljedicu veliku vjerojanost da većina implementaciju neće podržati takav zahtjev.

AH zaglavlje (Tablica 1) osigurava besprijekornost i autentičnost IP paketa[3]. Dodatno, na temelju klizećeg prozora (*sliding window*) i odbacivanja starih paketa, osigurava zaštitu od napada ponavljanjem paketa (*reply attacks*). Dijelovi AH zaglavlja su:

- *Next Header* - identificira protokol podatkovne jedinice koja slijedi (ovisno o načinu rada (*transport, tunnel*) to može biti protokol prijenosnog sloja ili mrežnog sloja),
- *Payload Length* - veličina AH paketa,
- RESERVED - rezervirano za buduće potrebe (sada je popunjeno nulama),
- *Security Parameters Index* (SPI) - jednoznačno pripada sigurnosnim parametrima veze u jednom smjeru (zajedno s odredišnom adresom i tipom sigurnosnog protokola (AH ili ESP) jednoznačno identificira zaštićenu vezu (*Security Association*) od Alice prema Bobu ili obrnuto),
- *Sequence Number* - redni (monotono rastući) broj paketa koji štiti od napada ponavljanjem,
- *Authentication Data* - svi podaci potrebni za osiguravanje autentičnosti IP paketa.

U *Authentication Data* polju AH zaglavlja nalazi se ICV vrijednost (*Integrity Check Value*).

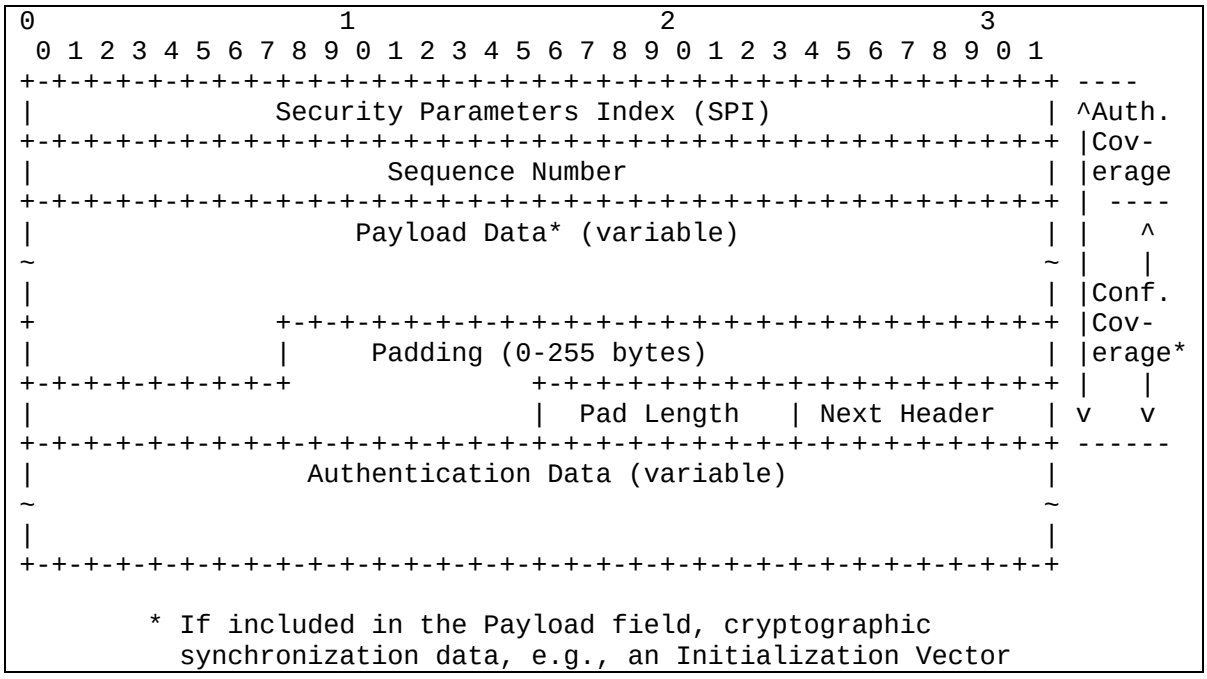


Tablica 1 AH zaglavlje

Nad većinom polja AH IP paketa izvodi se zaštita pomoću ICV vrijednosti (kao rezultat HMAC[28] algoritma). Za razliku od "obične" funkcije sažimanja, ICV vrijednost kao rezultat HMAC algoritma računa se i na temelju podataka koji se štite i na temelju dijeljene

tajne. Kao takav, pruža zaštitu besprijekornosti i autentičnosti podataka prema čemu je sličan digitalnom potpisu. Za razliku od digitalnog potpisa, ne osigurava zaštitu od poricanja.
Jasno je, izračunavanje ICV vrijednosti na temelju sadržaja i ključa je mnogo sigurnije nego koristiti "čistu" jednosmjernu sumu tj. izračunati sažetak samo na temelju podataka koji se štite, ali ne i ključa. Budući da se u IPsec-u kao osnova MAC algoritma upotrebljava funkcija sažimanja, MAC se naziva HMAC[27][29]. Uobičajena terminologija IPsec-a je: zaštita autentičnosti i besprijekornosti na temelju HMAC-MD5 i HMAC-SHA1 algoritama (preostali su opcionalni). Kada Bob primi poruku zaštićenu nekim od tih algoritama obavlja provjeru ICV vrijednosti. Izračunava HMAC ICV vrijednost[28] nad unaprijed dogovorenim poljima IP paketa dogovorenim autentifikacijskim algoritmom te provjerava da li su dobivena i izračunata vrijednost jednake.
AH štiti gotovo sve dijelove IP paketa, isključeni su samo oni koji se pri svakom skoku kroz mrežu (u usmjernicima) mijenjaju - TTL (*Time To Live*) i suma zaglavlja (*header cheksum*) te još i TOS (*Type Of Service*), *flgs* i *frag offset*. Važno je uočiti da AH štiti i polja s izvorišnom i odredišnom adresom što za posljedicu ima "probleme" kod AH+NAT funkcionalnosti.

ESP (Tablica 2) osigurava autentičnost, besprijekornost i tajnost paketa[4]. Na desnoj strani uz zaglavlje obilježeno je područje koje je zaštićeno s obzirom na zahtjev tajnosti (*Conf. Coverage*) i područje koje je zaštićeno s obzirom na zahtjev autentičnosti (*Auth. Coverage*).



Tablica 2 ESP paket

Dio koji je kriptiran (dakle, zaštićen po pitanju tajnosti) obuhvaća sam sadržaj (*Payload Data*; *payload* IP paketa je npr. TCP ili UDP segment (segment prijenosnog sloja)), *Padding* i polje koje u sebi nosi tip protokola poruke višeg sloja koja je enkapsulirana u tom IP paketu. Na ovaj način, osim što su podaci kriptirani, napadač čak ni ne zna što je enkapsulirano unutar paketa (jer je polje *Next header* također zakriptirano).
Kriptiranje se obavlja nekim od simetričnih algoritama oko kojih se Alice i Bob moraju dogovoriti. Tako Alice kada traži komunikaciju s Bobom, uz ostale zahtjeve, od Boba zahtjeva i korištenje specifičnog simetričnog algoritma (DES, 3DES, Blowfish, AES ili dr). Komunikacija će biti uspostavljena ukoliko Bob prihvati zahtjev i osigura ispunjenje tog

zahtjeva (dakle, ako se Alice i Bob dogovore oko korištenja istog algoritma, uz sve ostale parametre). Moguće je koristiti ESP i bez kriptiranja što je označeno kao ENCR_NULL algoritam. Koristiti ESP s NULL algoritmom ne zadovoljava zahtjev tajnosti (njime se ESP-om, ukoliko ga koristimo u kombinaciji s AH, ostvaruje autentičnost i besprijekornost, ali samo zaglavlja) i iz toga aspekta najčešće nema nikakvog smisla koristiti ga osim u ispitne svrhe (ovo je opisano na prikladan (šaljiv) način u RFC2410[11]).
Odabir simetričnog algoritma potpuno je proizvoljan, važno je jedino da se Alice i Bob dogovore koje će koristiti. Svojedobno se na IPsec mailing listi razvila diskusija o tome kako RFC-ovi o IPsec arhitekturi (npr. RFC4301[6]) čak niti ne spominju da li koristiti u implementaciji primjerice ECB ili CBC što bi moglo izazvati probleme u interoperabilnosti. Odgovor se uvijek podrazumijeva - potrebno je razmisliti što je sigurnije. ECB način kriptiranja ima karakteristiku da bez inicijalizacijskog vektora (IV) algoritam daje ponovljene instance iste poruke kriptirane istim ključem. CBC nema okvakvu karakteristiku i stoga se upotrebljava u implementacijama. Detaljnije informacije mogu se naći u RFC2451[9]: *The ESP CBC-Mode Cipher Algorithms*.
Zaštita s obzirom na zahtjev autentičnosti je opcionalna i sadržana je u polju *Authentication Data* koje se prema izboru dodaje na kraj zaglavlja. Za razliku od AH protokola, u ovom slučaju, autentificiraju se samo ESP zaglavlje i kriptirani sadržaj (*Payload*), a ne cijeli IP paket.

Uz dva spomenuta protokola IPsec nudi mogućnost korištenja još dva IP protokola: ipcomp (*IP Payload Compression Protocol*; IP protokol s brojem 108) i encap (*IP encapsulation*; IP protokol s brojem 98). Spomenute protokole nije moguće koristiti zasebno za neku komunikaciju. Primjerice, nije moguće u jezgru unijeti zahtjev samo za kompresijom nekog toka IP paketa. IPcomp za neku komunikaciju može biti zatražen dodatno uz ESP ili AH sigurnosni protokol (prema RFC4306[6] implementiranim nad RFC4301[5]).

1.2 Transport, tunnel i road-warrior načini rada

Dva osnovna IPsec načina rada su *transport* i *tunnel*. Jednako kao i sigurnosni protokol (AH ili ESP) i način rada je parametar politike koji se unosi o jezgru operacijskog sustava, a Alice i Bob se prije komunikacije moraju dogovoriti oko korištenja istog načina rada (uspješna komunikacija će se upostaviti samo ako i Alice i Bob imaju u jezgri unesen isti način rada).

Prvi spomenuti način rada, *transport* način podrazumijeva *host-to-host* (*end-to-end*) komunikaciju. Ukoliko Alice i Bob žele na ovaj način ostvariti sigurnu vezu, tada i Alice i Bob moraju imati IPsec implementaciju. Podatkovna jedinica koju razmjenjuju enkapsulira samo IP *payload*[2]. Posljedica ove činjenice je ta da u zaštićenom paketu u ovom načinu rada postoji samo jedna izvorišna i odredišna adresa IP adresa budući da se između Alice i Boba ne formira sigurni tunel (ovo je jedna od osnovnih razlika između *transport* i *tunnel* načina rada). U AH *transport* načinu rada originalni je IP paket tek neznatno promijenjen - između originalnog IP zaglavlja i *payload-a* IP paketa (transportnog segmenta) umetnuto je AH zaglavlje. Jasno je, u originalnom zaglavlju polje *Next header* više ne pokazuje na protokol transportnog sloja nego na AH protokol, dok polje *Next header* u AH zaglavlju sada pokazuje na prijenosni protokol. Slika IP paketa s umetnim AH zaglavljem (AH *transport* IP paket) i označenim zaštićenim poljima IP datagrama nalazi se u Dodatku A. Na slici je vidljivo da je autentificiran i zaštićen po pitanju integriteta gotovo cijeli IP paket uključujući izvorišnu i odredišnu adresu. Jasno je da zbog toga AH autentifikacija ne može funkcionirati ukoliko se na putu IP paketa od Alice prema Bobu događa prepisivanje adresa (NAT; *Network Address*

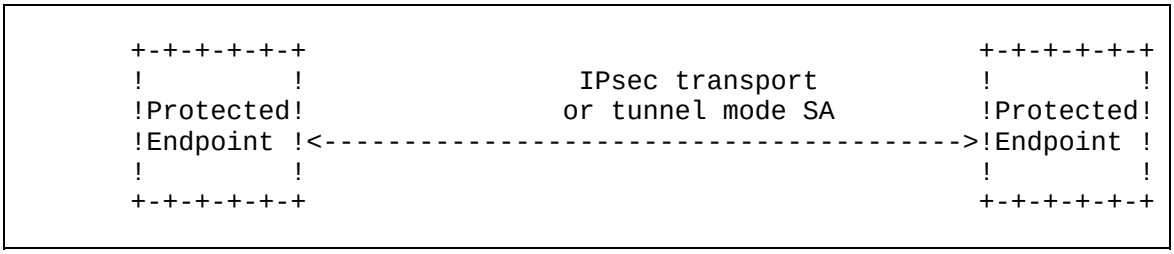
Translation) - ako neki mrežni uređaj prepiše ili izvorišnu ili odredišnu adresu, zaštitna suma više neće biti jednaka. Ovo je svojstvo nekompatibilnosti AH i NAT-a neovisno o načinu rada, isto se događa i u AH *tunnel* načinu rada.
Kod ESP *transport* načina rada također se između originalnog IP paketa i IP *payload*-a dodaje ESP zaglavlje, a IP *payload* se kriptira. Opcionalno se na kraj IP paketa dodaju autentifikacijski podaci. Na slici IP paketa s ESP kriptiranim *payload*-om (Dodatak A) vidljivo je da je autentificiran i zaštićen po pitanju integriteta samo kriptirani sadržaj. Izvorišna i odrediša adresa se ne nalaze pod zaštitnom sumom.

Drugi način rada, *tunnel* način podrazumijeva *net-to-net* komunikaciju (dakle, za računala iza prilaza *(gateway)*) što znači da se mora obaviti enkapsulacija cijelog IP paketa. Kako bi se formirao IPsec tunel, u zaglavlju enkapsuliranog IP paketa (*inner header*) postoji jedan par izvorišna - odredišna IP adresa, a u vanjskom IP zaglavlju (*outer header*) drugi par. U ovom slučaju se Alice i Bob nalaze iza prilaza između kojih se ostvaruje siguran tunel - Alice i Bob nemaju implementiran IPsec, a prilazi su ti koji imaju implementiran IPsec. Uobičajeno je takve prilaze u IPsec terminologiji nazivati sigurnosnim prilazima (SGW; *Security Gateways*). [5][6] Enkapsulacija cijelog IP paketa u ovom načinu rada znači da, primjerice kod AH protokola, nakon originalnog IP zaglavlja (*outer header*) i AH zaglavlja slijedi cijeli IP paket (unutarnje IP zaglavlje (*inner header*) te prijenosni segment), a ne samo prijenosni segment kao što je to slučaj kod *transport* načina rada (vidi Dodatak A). Jedina polja koja nisu zaštićena nalaze se u vanjskom zaglavlju (TTL, ToS, *flgs*, *frag offset*, *header cheksum* i naravno *Authentication Data* koji je rezultat AH protokola). Analogna je ovome i razlika između ESP *transport* i ESP *tunnel* podatkovne jedinice trećeg sloja OSI modela - kriptiran nije samo prijenosni segment nego cijeli IP paket (vidi Dodatak A).

Upotreba jednog od dva spomenuta načina rada ovisi o topologiji mreže i zahtjevima korisnika. Postoji nekolicina scenarija (ovisno o topologiji mreže tj. dijela mreže koja se nalazi između Alice i Boba) na temelju kojih se odlučuje koji način rada odabrati.
Može se raditi o komunikaciji između[5]:

1. krajnjih korisnika (s kraja na kraj) - *transport* ili *tunnel* način rada,
2. dva sigurnosna prilaza (*Security Gateways*) - *tunnel* način rada,
3. krajnjeg računala i sigurnosnog prilaza - *tunnel* način rada.

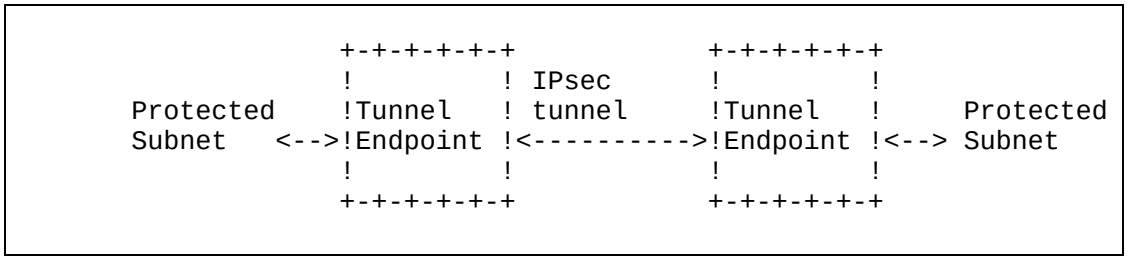
U prvom je slučaju uobičajeno upotrebljavati već opisani *transport* način rada (Tablica 3). No, moguće je upotrijebiti i *tunnel* način rada. Ukoliko se odabere *tunnel* način rada, izvorišna i odredišna adresa i u vanjskom i u unutrašnjem zaglavlju (ukoliko postoje oba zaglavlja) su jednake.



Tablica 3 Endpoint-to-Endpoint komunikacija[5]

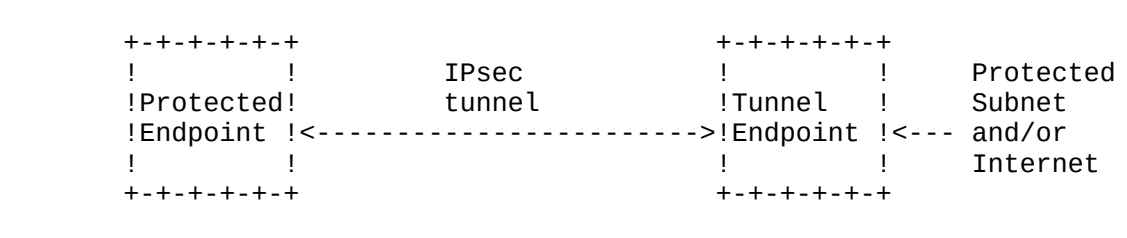
U drugom slučaju, kod komunikacije između dva sigurnosna prilaza (Tablica 4), nužno je upotrijebiti *tunnel* način rada - u svrhu formiranja tunela kroz nesigurnu mrežu, u vanjskom zaglavlju postoji jedan par izvorišne i odredišne adrese, a u unutarnjom zaglavlju drugi par.

Sigurnosni prilazi implementiraju IPsec i iza njih se nalazi zaštićena mreža računala koja nemaju implementiran IPsec. Unutrašnja adresa je ta koja pripada krajnjem korisniku (Bobu ukoliko Alice šalje paket preko dva SGW-a).



Tablica 4 Security Gateway-to-Security Gateway tunel[5]

Treći način rada naziva se *road-warrior* način rada (Tablica 5). *Road-warrior*-i su klijenti koji mijenjaju IP adrese (primjerice, prijenosna računala koja se preko nesigurne mreže prijavljuju u matičnu tvrtku). Između klijenta i SGW-a ostvaruje se IPsec tunel (u oba smjera) - radi se o *tunnel* načinu rada. Kako bi *road-warrior* klijent i SGW ostvarili tunel, SGW mora klijentu dodijeliti IP adresu (bilo statički, bilo dinamički). Pojasnimo na primjeru Alice i Bob. Alice je *road-warrior* klijent koji ima svoju trenutnu adresu i još jednu, adresu koju dobiva od SGW-a. Bob se nalazi u zaštićenoj mreži iza SGW-a i za razliku od Alice i SGW-a nema implementiran IPsec. Ukoliko se Alice i Bob (zapravo Alice i SGW) dogovore oko korištenja istog sigurnosnog protokola (AH ili ESP) te istog načina rada (*tunnel*) i drugih parametara politike koje imaju unesene u jezgru, uspješno se ostvaruje komunikacija. SGW će Alice dodijeliti IP adresu iz mreže u kojoj se nalazi Bob i ta se adresa upisuje na mjesto vanjske izvorišne adrese. U unutarnjem zaglavlju na mjestu izvorišne adrese ostaje zapisana trenutna IP adresa od Alice.
U ovakvom scenariju, *roadwarrior* klijent je u politici u jezgri SGW-a predstavljen s IP adresom 0.0.0.0. Kada u SGW pristigne paket od klijenta s nepoznatom adresom, smatra se da je njegova adresa 0.0.0.0 i poštuje se pripadno pravilo iz jezgre. Ako se paket uspješno autentificira, SGW čita iz paketa trenutnu adresu klijenta te na temelju preostalih potrebnih parametara iz paketa i pročitane trenutne adrese klijenta u jezgru unosi pravilo za komunikaciju s tim *roadwarrior* klijentom. Uobičajeno je reći da SGW u ovakvom scenariju mora imati uključenu opciju generiranja politike (*generate policy on*).



Tablica 5 Endpoint-to-Security Gateway tunel[5]

1.3 IPsec arhitektura

Kada Alice i Bob žele ostvariti sigurnu vezu, njihova IPsec implementacija mora znati dvije stvari:

1. Što zaštiti?
2. Kako to zaštiti?

Komunikacija između Alice i Boba definirana je izvorišnom i odredišnom IP adresom, tipom protokola višeg sloja (promet koji želimo zaštiti), smjerom komunikacije, tipom sigurnosnog protokola (AH ili ESP, opcionalno IPcomp ili encap), načinom rada i dr. Navedeni skup parametara naziva se sigurnosna politika (SP; *Security Policy*) i zapisuje se u jezgru operacijskog sustava. Jednom smjeru komunikacije pripada jedan takav zapis u bazi (SPD; *Security Policy Database*). SP-ovi uvijek dolaze u parovima - za dvosmjernu komunikaciju potrebna su dva SP unosa u SPD bazu.

Opisali smo kako definirati što želimo zaštiti, no kako definirati na koji način to želimo zaštiti? Način na koji određenu komunikaciju želimo zaštiti zapisan je u sigurnosnim poveznicama (SA; *Security Association*). SA je također jednosmjerna veza, a pripada joj i jedinstven pridružen broj - SPI (*Security Parameters Index*) koji je zapisan u zaglavlju AH ili ESP paketa [6, Appendix A]. Kada Alice želi komunicirati s Bobom, određenu SA vezu prepoznaje na temelju SPI-a za taj smjer (kojeg je odabrao Bob za svoj smjer), na temelju Bobove IP adrese i IPsec protokola (AH ili ESP). Vrijedi i za obrnuti smjer, od Boba prema Alice. Ukoliko između Alice i Bob postoji više SA-ova (štite različite prometa ili promet štite na različite načine) tada se one razlikuju na temelju SPI-a. SA-ovi kao i SP uvijek dolaze u parovima (po jedan za svaki smjer), a zapisani su u jezgri operacijskog sustava, u bazi koja se naziva SAD (*Security Association Database*).
Koncept SA-a temelj je IPsec arhitekture i predstavlja sigurnu poveznicu između Alice i Boba kojoj pripada dijeljena tajna informacija te skup kriptografskih algoritama koji štite promet koji se tom poveznicom prenosi. Alice i Bob će upsješno uspostaviti komunikaciju samo ako se uspiju dogovoriti oko skupa kriptografskih algoritama, ali i drugih parametara kojima će štiti promet koji žele razmijeniti. Kriptografski se algoritmi pri tome dijele na:

- enkripcijske: DES-CBC[8][9][10], 3DES-CBC[8][9][10], CAST-128[12], RC5, RC5-CBC, RC5-CTS[13], AES[14] i dr.
- autentifikacijske: MD5[15][16], HMAC[17], HMAC-MD5-96[18], HMAC-SHA1-96[19], HMAC-RIPEMD-160-96[20], AES-XCBC-MAC-96[21].

Zamjetimo još samo da je broj SA-ova barem jednak broju SP-ova ili veći. Naime, više SA-ova može pogađati jedno pravilo, a uz to SAD baza je podložna promijenama i u nekom trenutku može postojati više poveznica (SA-ova) između Alice i Boba nego na početku ostvarivanja komunikacije (jedan od razlog ovome je *rekeying* SA-ova, drugi je primjerice koncept tzv. *traffic* selektora što će detaljnije biti objašnjeno kod IKEv2 protokola).

Uz SAD bazu koja sadrži parametre pridružene već uspostavljenoj SA vezi (*keyed session*) i SPD bazu koja sadrži sigurnosne politike koje definiraju razmještaj IP prometa ulaznog ili izlaznog za računalo (*host*) ili sigurnosni prilaz (SGW) koji implementiraju IPsec, sadrži i PAD bazu. PAD baza (*Peer Authorization Database*) ima unose koji su poveznica protokola za automatsku razmjenu ključeva i SPD unosa.

Kada Alice (10.0.0.3) želi komunicirati s Bobom (10.0.0.4), IP paket sa vlastitom izvorišnom adresom i Bobovom odredišnom adresom. U IP paketu je enkapsuliran transportni segement odnosno protokolna podatkovna jedinca aplikacijskog sloja. Ovaj će paket na temelju izvorišne i odredišne adrese te tipa prometa za zaštitu (protokola višeg sloja) u jezgri pogoditi sigurnosnu politiku (SP) koja definira koju akciju želimo nad njim izvesti. Za tako definiranu politiku s tim točno određenim parametrima, u SAD bazi pronalazi se pripadni SA i prema Bobu šalje AH ili ESP zaštićeni paket. Kada Bob primi paket, u jezgri se detektira da je

pristigao kriptirani paket. Za paket se provjerava koju sigurnosnu politiku pogađa te koji je pripadni SA. Tada je i Alice spremna za slanje ESP zaštićenog paketa prema Bobu.

Alat kojim se SA i SP unosi unose u jezgru naziva se `setkey` (iz `ipsec-tools` paketa). Pokažimo jednostavni primjer - *transport* način rada na konfiguracijskoj datoteci za `setkey` (`/etc/setkey.conf`) u Tablici 6. Podsjetimo se, SP definira što želimo zaštiti, a SA kako to želimo zaštiti. SP unosi, unose se naredbom `spdadd`, a SA unosi naredbom `add`. Pretpostavimo komunikaciju računala 10.0.0.3 i 10.0.0.4, a prikazujemo konfiguracijske datoteke na računal 10.0.0.3.

```
# /etc/setkey.conf na racunalu 10.0.0.3
#!/usr/sbin/setkey -f

# brisemo sve unose u SAD bazi
flush;
# brisemo sve unose u SPD bazi
spdflush;

# ESP SA unosi za rucno podesavanje (bez ikev2 daemona)
add 10.0.0.3 10.0.0.4 esp 15701 -E 3des-cbc "MySecretKey";
add 10.0.0.4 10.0.0.3 esp 15702 -E 3des-cbc "MySecretKey";

# ESP SP unosi
spdadd 10.0.0.3 10.0.0.4 icmp -P out ipsec
           esp/transport//require;
spdadd 10.0.0.4 10.0.0.3 icmp -P in ipsec
           esp/transport//require;
```

Tablica 6 Primjer konfiguracijske datoteke za ala setkey

Prvi SA određen je izvorišnom IP adresom (10.0.0.3) i odredišnom IP adresom (10.0.0.4), tipom sigurnosnog protokola (ESP), SPI-em (15701), tipom enkripcijskog algoritma (3DES-*Cipher Block Chain*) nakon opcije -E i simetričnim ključem. Kada bi htjeli dodati i autentifikacijsku zaštitu, dodali bi opciju -A i autentifikacijski protokol (HMAC-SHA1, HMAC-MD5 ili dr.).
Drugi SA opisuje suprotan smjer, od 10.0.0.4 prema 10.0.0.3. Kao što je spomenuto, svaki smjer tj. svaki SA ima svoj SPI (tako je ovdje vrijednost 15702). Ukoliko želimo da komunikacija bude uspješno uspostavljena, enkripcijski algoritmi u oba smjera moraju biti jednaki (uobičajeno je reći da su se Alice i Bob dogovorili oko algoritma). Običaj je također u oba smjera upotrebljavati jednaki simetrični ključ, iako to nije nužno.

Prvi SP unos definira slijedeće:

- Smjer od 10.0.0.3 prema 10.0.0.4 je za računalo 10.0.0.3 izlazni smjer (`out`),
  Preostale mogućnosti odabira smjera se: `in`, `out` i `fwd`.
- Protokol višeg sloja koji želimo zaštiti je `icmp`,
  Mogućnosti odabira prometa koji želimo zaštiti su još i: `icmp`, `tcp`, `any` i dr.
- Tražena akcija je zaštita prometa `ipsec`-om,
  Preostale mogućnosti akcija su: `discard`, `none`, `bypass` i `entrust`.
- Tip sigurnosnog protokola je `esp` u `transport` načinu rada uz `require` nivo zaštite.
  Mogućnosti odabira protokola su: `esp`, `ah`, `ipcomp` i `entrust`.
  Mogućnosti načina rada su: `transport` i `tunnel`.

Mogućnosti nivoa zaštite su: `require`, `default` i `use`.
Drugi SP unos je gotovo identičan. Razlika je u definiranju smjera: smjer od 10.0.0.4 prema 10.0.0.3 je za računalo 10.0.0.3 ulazni smjer. Konfiguracijske datoteke Boba (10.0.0.4) gotovo su identične - dovoljno je zamijeniti tek smjer (`in` i `out`).

Naredbom `setkey -f /etc/setkey.conf` učitavaju se definirani SA i SP unosi u jezgru. SAD bazu možemo pregledati naredbom `setkey -D`, a SPD bazu naredbom `setkey -DP`. Ispisom SAD baze dobivamo mnoštvo podataka pridruženih toj vezi - među njima možemo uočiti spominjane: enkripcijski algoritam nakon oznake E, autentifikacijski algoritam nakon oznake A, SPI, sigurnosni protokol i način rada i dr. Ispisom SPD baze, za pojedini SPD unos možemo između ostalog uočiti: akciju zaštite, sigurnosni protokol, nivo zaštite i dr. Primjeri ispisa SAD i SPD baza nalaze se u nastavku dokumenta. Pokušamo li pingati između dva računala (budući da smo zatražili zaštitu `icmp` prometa), uočiti ćemo drugačiji ispis od onog na koji smo navikli. U ispisu (Tablica 7) prepoznajemo podatke iz zaglavlja AH ili ESP paketa: SPI i redni broj:

```
13:45:39.886113   10.0.0.3 > 10.0.0.4:    ESP(spi=0x000002b5,seq=0x118)
13:45:39.887436   10.0.0.4 > 10.0.0.3:    ESP(spi=0x000002b3,seq=0x10a)
13:45:40.898972   10.0.0.3 > 10.0.0.4:    ESP(spi=0x000002b5,seq=0x119)
13:45:40.900134   10.0.0.4 > 10.0.0.3:    ESP(spi=0x000002b3,seq=0x10b)
13:45:41.908735   10.0.0.3 > 10.0.0.4:    ESP(spi=0x000002b5,seq=0x11a)
13:45:41.909912   10.0.0.4 > 10.0.0.3:    ESP(spi=0x000002b3,seq=0x10c)
```

Tablica 7 Primjer ESP kriptiranih IP paketa tcpdump alatom

Budući da se SA-ovi i dijeljeni ključ unose ručno u jezgru, ovakva se metoda razmjene ključeva naziva ručna metoda. Ona je zapravo najjednostavnija - Alice i Bob generiraju simetrične ključeve, razmijene ih i ručno upišu u SAD bazu zajedno sa SA-ovima. Zašto *daemon* ne unosi sigurnosnu politiku u jezgru ne treba dodatno objašnjavati - svaki adminstrator zasigurno želi sam odrediti i unijeti sigurnosnu politiku u jezgru.
Metoda je jednostavnija, naravno, samo gledajući s implementacijske strane - nije potrebno implementirati aplikaciju koja će dogovarati ključeve i uspostavu SA-ova. Takva opisana ručna metoda je nesigurna[7] - mnogo je sigurnije koristiti *daemon* koji će pregovarati razmjenu ključeva i uspostavljati SA veze u određenim vremenskim razmacima. Protokol koji opisuje jedan od takvih *daemona* naziva se IKE protokol (*Internet Key Exchange*) i to specifično IKEv2 (verzija 2) koji je s obzirom na svog prethodnika brži i jednostavniji (u smislu razmjene manjeg broja razmijenjenih poruka između Alica i Boba za uspostavu SA veze).

2 IKEv2 protokol

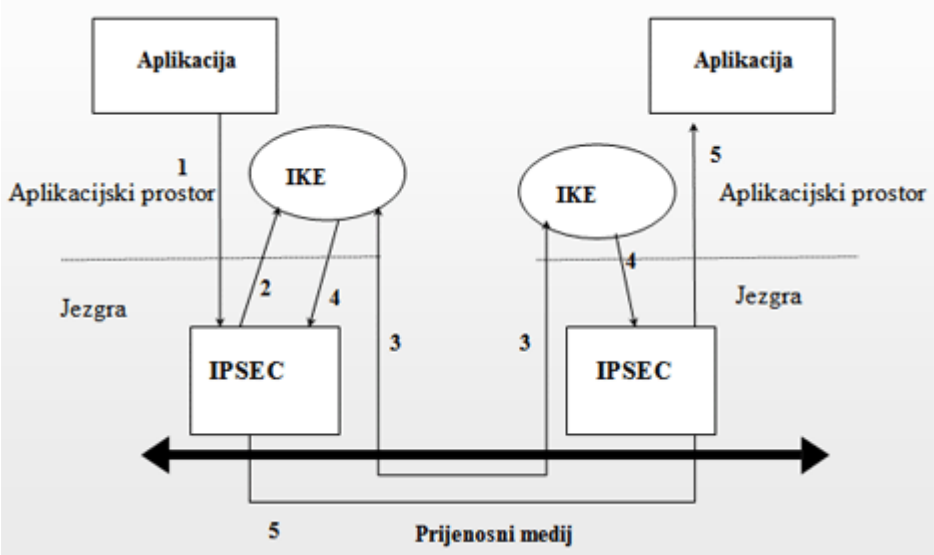
Podsjetimo se problema ručne razmjene ključeva:

- dijeljeni ključ je potrebno na siguran način razmijeniti između Alice i Boba preko nesigurne mreže,
- za komunikaciju sa svakim udaljenim partnerom, Alice mora čuvati drugačiji dijeljeni ključ (veliki broj unaprijed dodijeljenih ključeva),
- postoji mogućnost napada prisluškivanjem i otkrivanja dijeljenog ključa na temelju prikupljenih podataka (nema *perfect forward secrecy* zaštite),
- SA poveznica se u jezgru unosi ručno s točno definiranim enkripcijskim i autentifikacijskim algoritmom te duljinama ključeva.

Neki od protokola za razmjenu ključa su:

- IKE (*Internet Key Exchange*) [23],
- IKEv2 (*Internet Key Exchange version 2*) [5],
- KINK (*Kerberized Internet Negotiation of Keys*) [24],
- JFK (*Just Fast Keying*)[25].

Najčešće je upotrebljavani IKEv2 koji će jedini ovdje biti objašnjen.
Navedene probleme riješava automatska razmjena ključeva koja se obavlja prema nekome od protokola za razmjenu ključa. *Daemon* za razmjenu ključeva slučajno generira simetrične ključeve (na temelju dijeljene tajne i nekog izvora slučajnosti (*source of randomnes*)) i obavlja automatsku promjenu ključeva u definiranim periodima (*rekeying*), a nakon generiranja razmjenjuje na principu asimetričnog kriptosustava. Alice ne mora točno navesti enkripcijski i/ili autentifikacijski algoritam i njihove ulazne duljine ključeva, već Bobu nudi skup protokola koji su njoj prihvatiljivi (na kraju oni će se dogovoriti za jedan od svake vrste - enkripcijski ili autentifikacijski algoritam, zatim dogovaraju i Diffie-Helman grupu te pseudo-slučajnu funkciju koja je izvor *keying* materijala za kriptografske algoritme). Alice u konfiguracijskoj datoteci *daemona* za automatsku razmjenu ključeva sadrži zapise koje slikovito možemo pročitati primjerice na sljedeći način: "Za komunikaciju s tobom Bob, meni je prihvatljivo koristiti enkripcijski algoritam ili DES ili 3DES ili AES128_CTR s minimalno sljedećom duljinom ključa...". Nakon uspješnog dogovora *daemon* za automatsku razmjenu ključeva će formirati SA poveznicu i unijeti je u jezgru operacijskog sustava u SAD bazu.
Pogledajmo na slici 2.1 kako funkcionira uspostavljanje IPsec komunikacije između Alice i Boba.



Slika 2.1Uspostava SA poveznice pomoću IKE daemona

1. Alice (aplikacija) šalje prema Bobu IP paket. Alice ima u jezgri zapisano da se svi paketi za Boba moraju zaštiti (tj. Alice ima u jezgri u SPD bazi SP unos koji definira zaštitu tog prometa).
2. Budući da Alice ima u jezgri samo sigurnosnu politiku (SP), a ne i SA poveznicu, aktivira se njezin IKE daemon.
3. Alice-in IKE daemon inicira komunikaciju s Bobovim IKE daemonom kako bi razmijenili dijeljene ključeve i dogovorili se oko skupa algoritama kojima će zaštiti SA poveznicu.
4. *Daemoni* su razmijenili dijeljene ključeve i dogovorili se oko kirptografskih algoritama te možemo reći da istovremeno zapisuju taj ključ i algoritme u jezgru operacijskog sustava - formiraju novi unos u SAD bazi određen IP adresama Alice i Boba, SPI brojem, akcijom, tipom protokola i razinom zaštite, dogovorenim kriptografskim algoritmima i dr. Takav SA koji je IKE uspostavio, naziva se IKE SA, a uz njega se u inicijalnom koraku stvara i CHILD SA - sigurnosna poveznica kojom se razmjenjuje traženi promet.
5. Nakon koraka 4, ukoliko se ne događaju nikakve nepredvidive situacije (npr. napad) daemon može mirovati (sve do promijene ključeva). U tom koraku, između uspostave SA nakon generiranja ključeva i uspostave novog SA nakon generiranja novih ključeva, kroz nesigurnu se mrežu prenosi zaštićen traženi promet (sigurnom poveznicom koja će se stvoriti kroz uspostavljenu IKE SA - naziva se, kao što je rečeno, CHILD SA). Bob na temelju pridruženog SP unosa u jezgri, prepoznaje koji promet treba zaštiti, a prema SPI broju i drugim parametrima pronalazi pripadni SA.

IKE SA je prva poveznica koja se uspostavlja između Alice i Boba. Sve SA poveznice koje bivaju stvorene kroz IKE SA (prilikom promijene ključa (*rekeying*)) nazivaju se CHILD SA. Zahtjev za *rekey*-ingom Alice može primiti od jezgre (kod isteka tzv. *soft expiry* parametra) ili od Boba (ukoliko je njemu istekao *softlimit time* parametar i pošalje Alice zahtjev za *rekeyingom*). IKE SA ostaje aktivna još neko vrijeme (do isteka tzv. *hard expiry* parametra). *Hard expiry* kod IKE SA naziva se vrijeme reautentifikacije (budući da se mora ponovo proći kroz proces autentifikacije). Kod rekey-inga CHILD SA, redoslijed zbivanja uslijed istekta *soft* i *hard* vremena trajanja SA, događaju se analogna zbivanja.

2.1 IKE poruke, IKE SA i CHILD SA

IKE poruke razmijenjuju se UDP protokolom, a *daemon* se vrti na portu 500 (ili 4500 u nešto kompliciranijoj situaciji koju ovdje nećemo obrađivati). Zamjetimo za početak da je kod IPsec komunikacije s *daemon*-om za razmjenu ključeva potrebno zaštiti i traženi promet, ali i IKE promet (poruke koje se razmijenjuju između dva *daemona*).
IKE poruke se uvijek razmjenjuju u parovima (*exchanges*) pri čemu se za Alice i Boba upotrebljavaju termini *initiator* i *responder* [5]. IKE SA i prva CHILD SA uspostavlja se kroz četiri poruke:

1. IKE_SA_INIT *exchange* i
2. IKE_AUTH *exchange.*

Prvim parom poruka (IKE_SA_INIT), Alice i Bob se dogovaraju oko kriptografskih algoritama i drugih parametara kojima će štiti traženi promet (uobičajeno je za skup takvih protokola upotrijebiti pojam *cryptographic suite*, a za cijeli skup parametara *proposal*), razmijenjuju *nonce* brojeve i obavljaju Diffie-Helman razmijenu (asimetričnim kriptosustavom razmijenjuju dijeljenu tajnu). IKE_SA_INIT par poruka je kriptiran (zaštićen je enkripcijskim algoritmom definiranim za zaštitu IKE prometa), no nisu još autentificirane jer se podaci potrebni za autentifikaciju tek prenose ovim porukama, a Alice i Bob će se međusobno predstaviti jedan drugome tek u sljedećem paru poruka.

Drugi par poruka (IKE_AUTH) služi autentifikaciji prethodno razmijenjenih poruka (izvodi se HMAC na temelju sadržaja iz prethodnih poruka i dijeljene tajne). Ovim se parom poruka Alice i Bob predstavljaju jedan drugome (šalju svoje identitete i/ili certifikate) i uspostavlja se prva CHILD SA poveznica kojom se prenosi traženi promet. IKEv2 protokol dozvoljava različite metode autentifikacije[5]:

- PSK: *pre-shared key* - na temelju dijeljenog ključa i HMAC algoritma te prf funkcije (*pseudo random function*),
- RSA: RSA-potpisan PKCS-nadopunjen MAC (*Message Autentification Code*)[30],
- DSS (*Digital Signature Standard*): DSS-potpisan sažetak [31],
- ECDSS (*Elliptic Curve Cryptography Digital Signature Standard*): ECC baziran DSS[32].

IKE_AUTH poruke su i kriptirane i autentificirane na temelju ključeva koji su razmijenjeni prvim parom poruka (IKE_SA_INIT). Postoje situacije u kojima je za IKE SA potrebno razmijeniti više od četiri poruke, npr:

- U slučaju detekcije prevelikog broja poluotvorenih veza (što se tipično događa kod DOS napada) potreban je dodatan par poruka zbog razmijene tzv. *cookie* vrijednosti - ovo je princip zadovoljavanja sigurnosnog zahtjeva raspoloživosti,
- U slučaju EAP autentifikacije [*Extensible Authentication Protocol*] [26] također je potrebno razmijeniti dodatne poruke.

Navedene četiri poruke moraju se razmijeniti točno ovakvim navedenim redoslijedom. Ukoliko je Alice inicijator, Alice šalje IKE_SA_INIT zahtjev, a Bob odgovara IKE_SA_INIT odgovorom. Ukoliko su te dvije poruke uspješno razmijenjene, Alice šalje IKE_AUTH zahtjev, a Bob odgovara IKE_AUTH odgovorom (ukoliko sve prođe u redu). Nakon razmijene ove četiri poruke, između Alice i Bob počinje se razmjenjivati traženi promet zaštićen SA poveznicom (kriptiran i autentificiran).

Sljedeću IKE poruku, nakon što je IKE SA jednom uspostavljen, može poslati i Alice i Bob. Poruke nakon četiri inicijalne poruke mogu biti:

- CREATE_CHILD_SA zahtjev ili odgovor,
- INFORMATIONAL poruke.

CREATE_CHILD_SA zahtjev je zahtjev za stvaranjem nove SA poveznice (uslijed *rekey*-inga). Očito je, ovu će poruku poslati bilo Alice bilo Bob - onaj kome prvome istekne *soft expiry*. Ovim se parom poruka opcionalno može obaviti dodatna Diffie-Hellman razmjena (kako bi se omogućila jača *perfect forward secrecy* zaštita). Razmijenjuju se *nonce* brojevi (kako bi se uspostavio jedinstveni ključ za taj CHILD SA) i opcionalno se dogovaraju tzv. prometni selektori (*traffic selectors*) koji definiraju IP adrese, pristupe (*ports*) i protokole koji će se prenositi tom CHILD SA. PFS (*perfect forward secrecy*) jedna je od karakteristika automatske razmijene ključeva - ukoliko napadač uspije otkriti trenutni dijeljeni ključ, ne može proračunati niti do tad upotrebljavane djeljene ključeve, niti one koji će se tek generirati. Ova karakteristika zasniva se na Diffie-Hellman razmijeni. Upravo je PFS svojstvo razlog zbog kojeg se uz DH eksponente razmijenjuju i *nonce* brojevi. Ukoliko bi se svaki puta (za svaki novi SA - u CREATE_CHILD_SA porukama) upotrebljavao jedinstveni privatni DH broj, potreba za *nonce* brojevima (koji moraju biti jedinstveni za svaki SA) zapravo uopće ne bi postojala. No, kako bi se izbjegli "skupi" DH eksponent DH brojevi se ponekad ponovo iskorištavaju (*concept of reusing DH exponents*).

INFORMATIONAL razmjena može se dogoditi nakon inicijalne četiri poruke i obično detektira neočekivana stanja. Primjeri neočekivan stanja i NOTIFY *payload*-a koji se umeću u INFORMATIONAL poruku su:

- kod DOS napada - INVALID_COOKIE,
- nepoznati SPI broj - INVALID_IKE_SPI,
- neodgovarajući zatraženi skup kriptografskih algoritama - NO_PROPOSAL_CHOSEN
- poruke za brisanje SA - DELETE INFORMATIONAL.

3 IPsec implementacija s IKEv2 daemonom

Trenutno postoje tri OSS IKEv2 implementacije: **racoon2**, **OpenIKEv2** i **IKEv2**. Ovdje ćemo pokazati kako konfigurirati ikev2 *daemon*. IKEv2 je pisan u jeziku C, za operacijski sustav Linux, pod GNU General Public licencom (GPL).

Alati potrebni za IPsec implemetaciju su:

1. **`setkey`** - rad sa SAD i SPD unosima u jezgru.
   SA parametre ćemo samo pregledavati (naredbom `setkey -D`) (ne unosimo ih ručno - to obavlja ikev2 *daemon*).
   SP unose ćemo dodavati u SPD tablicu naredbom `spdadd` i pregledavati naredbom `setkey -DP`.
2. **IKE *daemon***.

Pretpostavimo komunikaciju računala 10.0.0.3 i 10.0.0.4. Pokazati ćemo konfiguraciju na računalu 10.0.0.3.

3.1 Alat setkey

U konfiguracijsku datoteku (Tablica 8) za setkey (`/etc/setkey.conf`) potrebno je unijeti samo sigurnosnu politiku (SP unose):

```
#!/usr/sbin/setkey -f

# brisanje unosa u SAD i SPD bazi
flush;
spdflush;

# ESP SA-ove ce u SAD bazu unijeti ikev2 daemon

# ESP SP unosi
spdadd 10.0.0.3 10.0.0.4 icmp -P out ipsec
           esp/transport//require;
spdadd 10.0.0.4 10.0.0.3 icmp -P in ipsec
           esp/transport//require;
```

Tablica 8 Primjer konfiguracijske datoteke za alat setkey uz IKEv2 razmjenu ključeva

Konfiguracijska je datoteka slična onoj kod ručnog dogovaranja SA-ova i ključeva i ovdje neće biti ponovo objašnjavana. Za učitavanje konfiguracijske datoteke potrebno je upisati: **`setkey -f /etc/setkey.conf`**.

Sada se u SPD nalaze SP unosi, dok je SAD baza prazna budući da komunikacija još nije uspostavljena; SA unose u SAD bazi pratiti ćemo nakon pokretanja daemona. Uvjerimo se da su stanja SAD i SPD baza zaista takva ((Tablica 9)):

```
host% setkey -DP
      10.0.0.4[any] 10.0.0.3[any] icmp
        in ipsec
        esp/transport//require
```

```
        created: Dec 28 10:23:01 2005  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=112 seq=12 pid=3216
        refcnt=1
      10.0.0.3[any] 10.0.0.4[any] icmp
        out ipsec
        esp/transport//require
        created: Dec 28 10:23:01 2005  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=129 seq=11 pid=3216
        refcnt=1

      host% setkey -D
      No SAD entries.
```

Tablica 9 Ispis SPD i SAD baze prije uspostavljanja komunikacije

3.2 ikev2 daemon

ikev2 *daemon* dostupan je na http://sourceforge.net/projects/ikev2. Nakon izvođenja `aclocal`, `automake`, `autoconf`, `./configure`, `make` (potrebni su i `log4c` (za logiranje), `flex`, `bison v1.875` i sl.) treba urediti tri konfiguracijske datoteke. *Daemon* ima dvije konfiguracijske datoteke:

1. konfiguracijska datoteka ikev2 (**`ikev2.conf`**) i
2. konfiguracijska datoteka za log4c (**`log4crc`**) - primjer konfiguracijske datoteke (XML tipa) za zapisivanje logova u datoteku `logfile` koja se automatski stvara u direktoriju gdje se nalazi i konfiguracijska datoteka nalazi se u Dodatku B.

Uz to, budući da ćemo pokazati primjenu *daemona* uz psk (*pre-shared key*) autentifikaciju (koja je za sada jedina implementirana u *daemonu*) potrebna je i:

3. datoteka **`psk.txt`**.

Sve tri datoteke potrebno je smjestiti u direktorij u kojem se nalazi izvršna ikev2 datoteka dobivena prevođenjem. Primjer `log4crc` datoteke nalazi se u dodatku X.

Minimalna **`ikev2.conf`** konfiguracijska datoteka (Tablica 10):

```
# ikev2.conf
# This is a configuration file for IKEv2 daemon and as such it is a
# combination of SAD, SPD and PAD. For details of those databases see
# RFC4301.
#

random device "/dev/urandom";
path pre_shared_key "psk.txt";

# Number of half-opened connections above which we start to send
# cookies
dos_treshold 50;

# Number of concurent threads for processing sessions
sm_threads 5;

remote anonymous
{
        nonce_size 32;
```

```
        # Time after which IKE daemon reauthenticates of IKE SA
        authlimit time 1 week;

        # Maximum idle time after which IKE daemon starts dead peer
        # detection
        ike_max_idle 1000 s;

        # If timeout or retries is set to zero then retransmissions are
        # disabled...
        response_timeout 3 s;
        response_retries 3;

        my_identifier rfc822_addr mojId;

        proposal {
                limit time 10 s;

                encryption_algorithm 3des;
                auth_algorithm md5_96;
                pseudo_random_function md5;
                dh_group modp768;

                authentication_method pre_shared_key;
        }
}

sainfo anonymous
{
        hardlimit time 1000 s;
        softlimit time 6 s;

        auth_algorithm sha1_96;
        encryption_algorithm 3des;
}
```

Tablica 10 Konfiguracijska datoteka za IKEV2 daemon

`ikev2.conf` sastoji se od tri dijela:

1. direktive unutar `remote` bloka štite IKE promet tj. IKE SA,
2. direktive unutar `sainfo` bloka štite promet za koji smo postavljenom politikom u jezgri (pomoću `setkey`-a) zatražili zaštitu tj. štiti CHILD SA,
3. općenite direktive izvan navedenih blokova (npr. staza do `psk.txt`, broj paralelnih dretvi i sl).

Unutar **remote** bloka nalazi se već spomenuti `proposal` blok. Parametri od kojih se sastoji su slijedeći:

- *soft* IKE SA *expiry* (`limit time`): vremenski period nakon kojeg se obavlja *rekeying*,
- *cryptographic suite* (u slučaju IKE_SA sastoji se od četiri algoritma):
    1. enkripcijski algoritam,
    2. autentifikacijski algoritam,
    3. pseudo-slučajna funkcija (generira tzv. *keying* materijal za sve kriptografske algoritme),
    4. Diffie-Hellman grupa (za sigurnu razmjenu informacija simetričnog kriptosustava i ostvarenje *perfect forward secrecy* zaštite).

- metoda autentifikacije: PSK.

Popis svih algoritama koji se koriste u IKEv2 protokolu naveden je u RFC dokumentu[22].

Direktive unutar **sainfo** bloka štite traženi promet (u našem primjeru, s obzirom na dani SP u `/etc/setkey.conf`) to je ICMP promet. Parametri koji štite traženi promet tj CHILD SA su:

- *soft* CHILD SA *expiry* (`softlimit time`): vremenski period nakon kojeg se obavlja *rekeying* (ekvivalent je parametru `limit time` za IKE SA),
- *hard* CHILD SA *expiry* (`hardlimit time`): vremenski period nakon kojeg se obavlja brisanje SA-a i stvaranje novog (ekvivalent je parametru `authlimit time` za IKE SA tj. reautentifikaciji kod IKE SA),
- kriptografski algoritmi koji štite traženi promet:
  1. enkripcijski algoritam,
  2. autentifikacijski algoritam.

Postavka na koju je važno obratiti pažnju je `my_identifier` kod koje je umjesto `mojID` potrebno navesti vlastiti ID. ID je podatak koji se nalazi zapisan u `psk.txt` datoteci i njemu je pridružen tajni ključ. Odredište s kojim komuniciramo također mora imati analogne zapise u **`psk.txt`** datoteci (Tablica 11). Pokažimo na primjeru:

```
# psk.txt
mojId                   sharedSecretKey
remoteId                sharedSecretKey
```

Tablica 11 psk.txt datoteka za PSK metodu autentifikacije

Budući da je politika unesena u jezgru, a sve konfiguracijske datoteke uređene, možete pokrenuti *daemon*. Provjerimo da se `ikev2` zaista vrti (na portu 500):

```
host% netstat -uanp
Active Internet connections (servers and established)
Proto R-Q S-Q Local Address Foreign Address State         PID/Program
name
udp    0    0   0.0.0.0:500      0.0.0.0:*                 4364/ikev2
```

Tablica 12 Ispis netstat naredbe za port 500

Pokušajmo sada pingati s jednog računala na drugo (podsjetimo se, uneseni SP u jezgri definira zaštitu ICMP prometa) te provjeriti stanje SAD baze:

```
host% setkey -DP

10.0.0.4 10.0.0.3
esp mode=transport spi=224162611(0x0d5c7333) reqid=0(0x00000000)
E: 3des-cbc  5d421c1b d33b2a9f 4e9055e3 857db9fc 211d9c95 ebaead04
A: hmac-sha1  c5537d66 f3c5d869 bd736ae2 08d22133 27f7aa99
seq=0x00000000 replay=4 flags=0x00000000 state=mature
created: Nov 11 12:28:45 2002  current: Nov 11 12:29:16 2002
diff: 31(s)    hard: 600(s)   soft: 480(s)
last: Nov 11 12:29:12 2002    hard: 0(s)     soft: 0(s)
current: 304(bytes)    hard: 0(bytes) soft: 0(bytes)
allocated: 3   hard: 0 soft: 0
sadb_seq=1 pid=17112 refcnt=0
```

```
10.0.0.3 10.0.0.4
esp mode=transport spi=165123736(0x09d79698) reqid=0(0x00000000)
E: 3des-cbc  d7af8466 acd4f14c 872c5443 ec45a719 d4b3fde1 8d239d6a
A: hmac-sha1  41ccc388 4568ac49 19e4e024 628e240c 141ffe2f
seq=0x00000000 replay=4 flags=0x00000000 state=mature
created: Nov 11 12:28:45 2002  current: Nov 11 12:29:16 2002
diff: 31(s)    hard: 600(s)   soft: 480(s)
last:                         hard: 0(s)     soft: 0(s)
current: 231(bytes)    hard: 0(bytes) soft: 0(bytes)
allocated: 2   hard: 0 soft: 0
sadb_seq=0 pid=17112 refcnt=0
```

Tablica 13 Ispis SAD baze nakon uspostavljanja komunikacije

Među mnogobrojnim podacima zapisanim u bazi, uočimo one koje smo sami postavili unošenjem politike u jezgru: način rada (`transport`), sigurnosni protokol (`esp`), enkripcijski algoritam (`3des`), autentifikacijski algoritam (`sha1`).

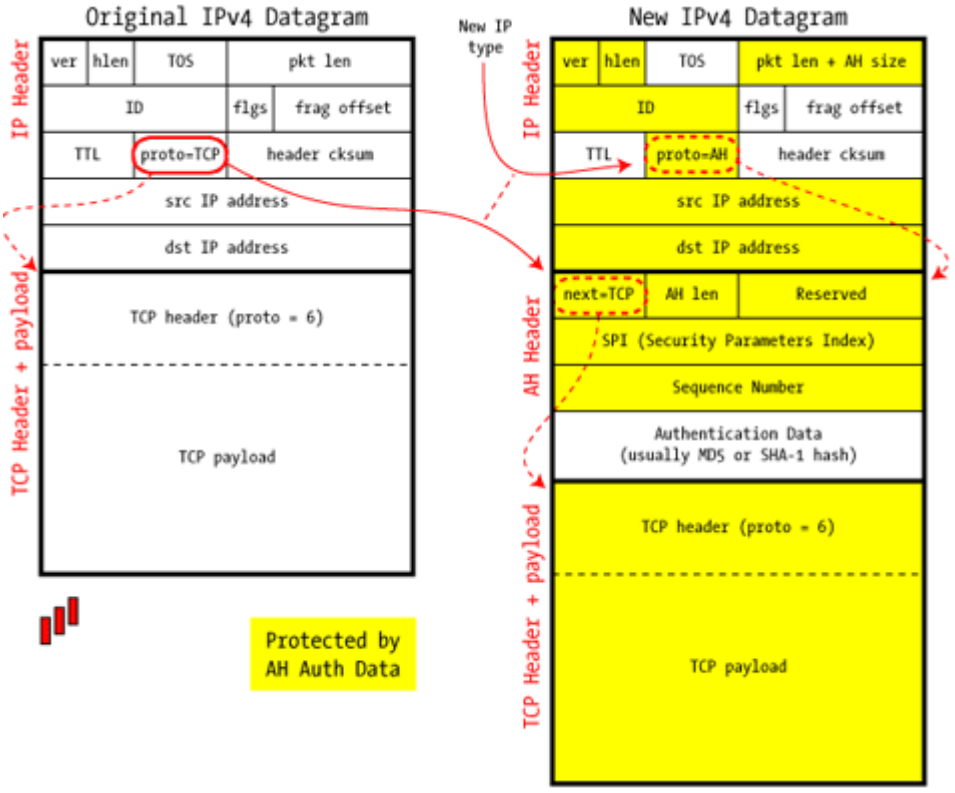
Pokušamo li primjerice `tcpdump`-om pratiti ping promet, rezultat je kriptirani promet.

4 Literatura

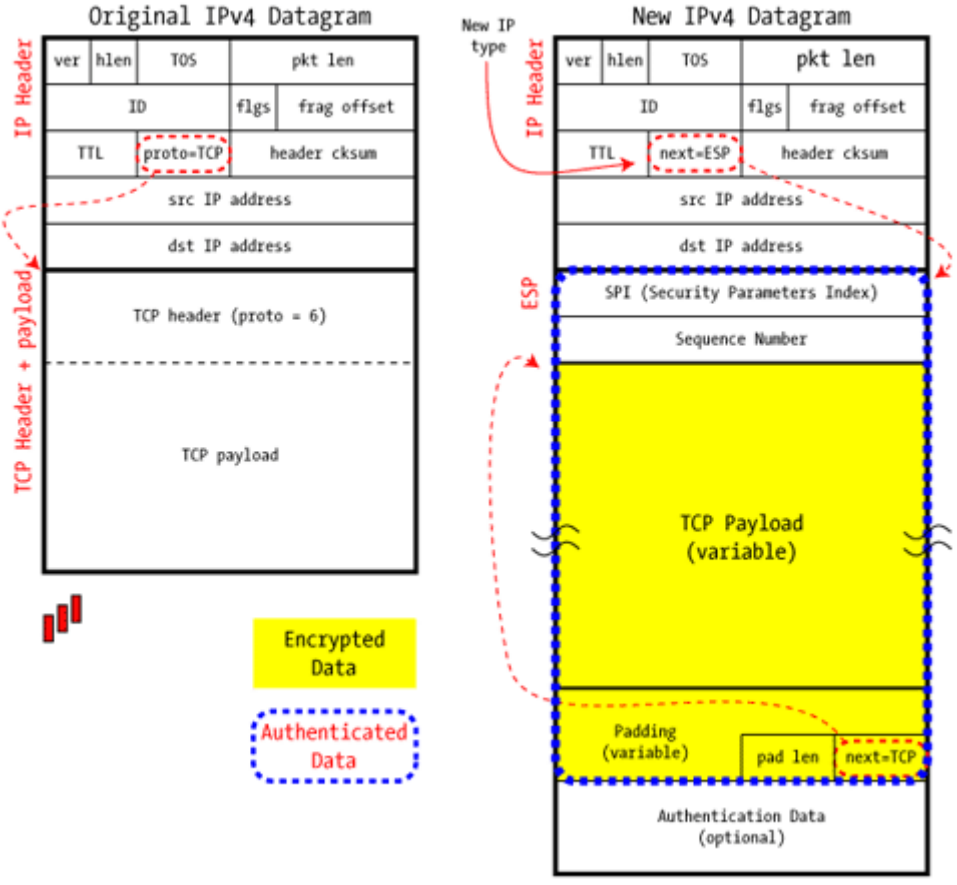
1. http://www.freeswan.org/freeswan_snaps/CURRENT-SNAP/doc/glossary.html: Glossary for the Linux FreeS/WAN project.
2. http://www.unixwiz.net/techtips/iguide-ipsec.html: An Ilustrated Guide to IPsec.
3. http://www.faqs.org/rfc/rfc2402.txt: IP Authentication Header.
4. http://www.faqs.org/rfc/rfc2406.txt: IP Encapsulating Payload (ESP).
5. http://www.faqs.org/rfc/rfc4306.txt: Internet Key Exchange (IKEv2) Protocol.
6. http://www.faqs.org/rfc/rfc4301.txt: Security Architecture for the Internet Protocol.
7. http://www.freeswan.org/freeswan_snaps/CURRENT-SNAP/doc/adv_config.html#man-auto: IPsec configuration possibilities.
8. http://www.faqs.org/rfc/rfc1829.txt: The ESP DES-CBC Transform.
9. http://www.faqs.org/rfc/rfc2451.txt: The ESP CBC-Mode Cipher Algorithms.
10. http://www.faqs.org/rfc/rfc2405.txt: The ESP DES-CBC Cipher Algorithm.
11. http://www.faqs.org/rfc/rfc2410.txt: The NULL Encryption Algorithm and Its Use With IPsec.
12. http://www.faqs.org/rfc/rfc2144.txt: The CAST-128 Encryption Algorithm.
13. http://www.faqs.org/rfc/rfc2040.txt: The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms
14. http://www.faqs.org/rfc/rfc3602.txt: The AES-CBC Cipher Algorithm and Its Use with IPsec
15. http://www.faqs.org/rfc/rfc1828.txt: IP Authentication using Keyed MD5.
16. http://www.faqs.org/rfc/rfc2085.txt: HMAC-MD5 IP Authentication with Replay Prevention.
17. http://www.faqs.org/rfc/rfc2104.txt: HMAC: Keyed-Hashing for Message Authentication.
18. http://www.faqs.org/rfc/rfc2403.txt: The Use of HMAC-MD5-96 within ESP and AH
19. http://www.faqs.org/rfc/rfc2404.txt: The Use of HMAC-SHA-1-96 within ESP and AH
20. http://www.faqs.org/rfc/rfc2857.txt: The Use of HMAC-RIPEMD-160-96 within ESP and AH.
21. http://www.faqs.org/rfc/rfc3566.txt: The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec.
22. http://www.faqs.org/rfc/rfc4307.txt: Cryptographic Algorithms for Use in the Internet Key Exchange 2 (IKEv2).
23. http://www.faqs.org/rfc/rfc2409.txt: The Internet Key Exchange (IKE)
24. http://www.faqs.org/rfc/rfc3129.txt: Requirements for Kerberized Internet Negotiation of Keys.

25. http://www3.ietf.org/proceedings/03mar/I-D/draft-ietf-ipsec-jfk-04.txt: Just Fast Keying (JFK).
26. http://www.faqs.org/rfc/rfc3748.txt: Extensible Authentication Protocol (EAP).
27. http://www.faqs.org/rfc/rfc2104.txt: HMAC: Keyed-Hashing for Message Authentication
28. http://en.wikipedia.org/wiki/Hmac: Hashed Message Authentication Code.
29. http://www.faqs.org/rfc/rfc2202.txt: Test Cases for HMAC-MD5 and HMAC-SHA-1
30. http://en.wikipedia.org/wiki/Message_authentication_code: Message Authentication Code.
31. http://www.itl.nist.gov/fipspubs/fip186.htm: FIPS PUB 186, Digital Signature Standard.
32. http://www.w3.org/PICS/DSig/ECC-1_0.html: ECDSS Signature Suite - Version 1.0.
33. http://www.faqs.org/rfc/rfc1825.txt: Security Architecture for the Internet Protocol.
34. http://www.faqs.org/rfc/rfc2401.txt: Security Architecture for the Internet Protocol.
35. http://www.faqs.org/rfc/rfc2412.txt: The OAKLEY Key Determination Protocol.
36. http://www.ietf.org/internet-drafts/draft-hoffman-ikev2-1-00.txt: Internet Key Exchange Protocol: IKEv2.1 (draft-hoffman-ikev2-1-00.txt).
37. http://www.tatsuyababa.com/internet-drafts/draft-ietf-ipsec-ikev2-tutorial-01.txt: Understanding IKEv2: Tutorial, and rationale for decisions (draft-ietf-ipsec-ikev2-tutorial-01.txt).
38. http://www.ietf.org/internet-drafts/draft-eronen-ipsec-ikev2-clarifications-06.txt: IKEv2 Clarifications and Implementation Guidelines (draft-eronen-ipsec-ikev2-clarifications-06.txt).

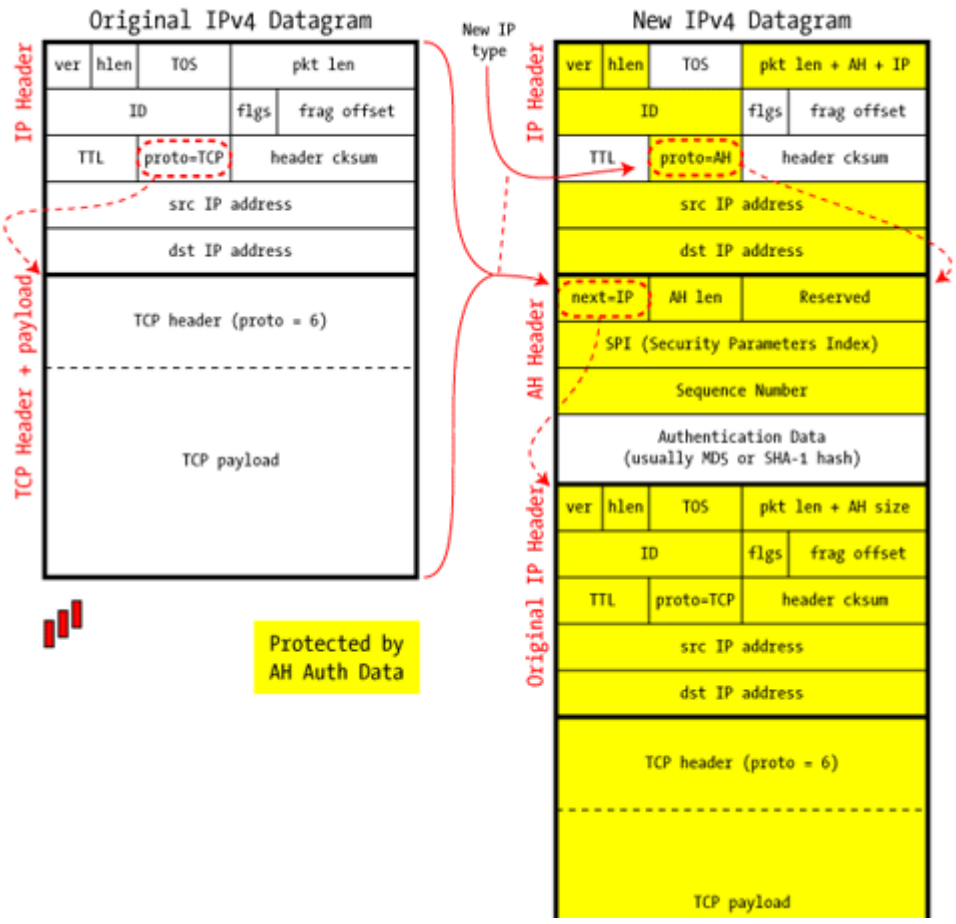
5 Dodatak A



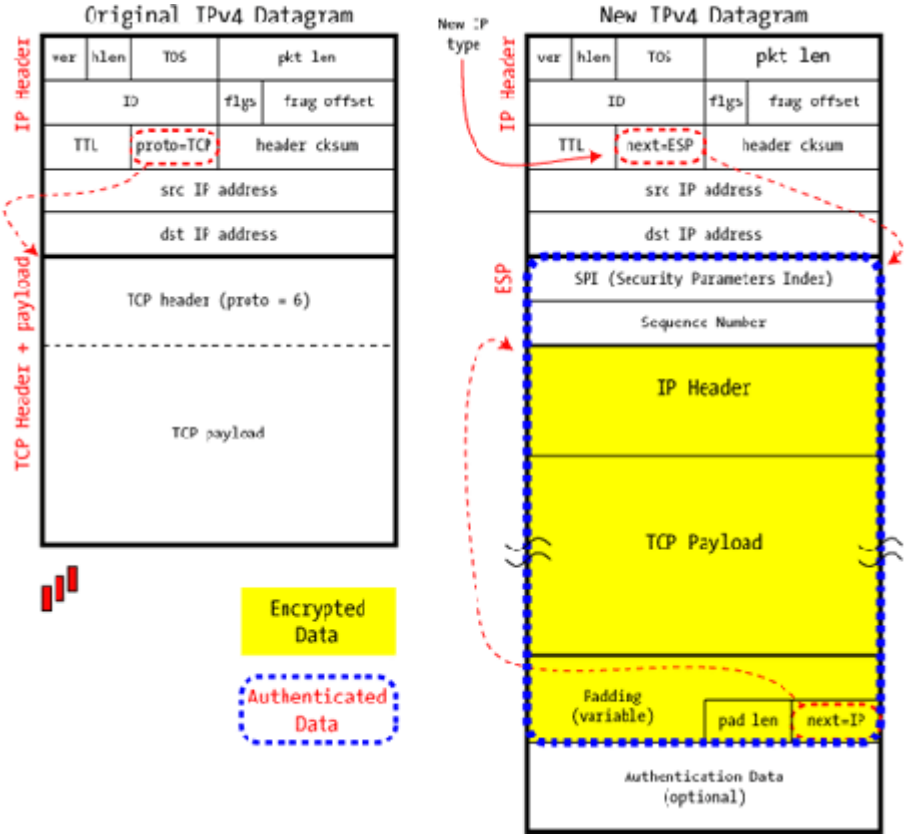
Slika 5.1 AH paket u transport načinu rada[2]



Slika 5.2 ESP paket u transport načinu rada[2]



Slika 5.3 AH paket u tunnel načinu rada[2]



Slika 5.4 ESP paket u tunnel načinu rada[2]

6 Dodatak B

`log4crc` konfiguracijska datoteka `ikev2` *daemona* za logiranje u datoteku `logfile`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE log4c SYSTEM "">

<log4c version="1.0.10">

        <config>
                <bufsize>0</bufsize>
                <debug level="100"/>
                <nocleanup>0</nocleanup>
        </config>

        <category name="main" priority="unknown" appender="logfile"/>
        <category name="aes_xcbc" priority="unknown" appender="logfile"/>
        <category name="config" priority="unknown" appender="logfile"/>
        <category name="crypto" priority="unknown" appender="logfile"/>
        <category name="csa" priority="unknown" appender="logfile"/>
        <category name="message" priority="unknown" appender="logfile"/>
        <category name="network" priority="unknown" appender="logfile"/>
        <category name="payload" priority="unknown" appender="logfile"/>
        <category name="pfkey" priority="unknown" appender="logfile"/>
        <category name="proposals" priority="unknown" appender="logfile"/>
        <category name="session" priority="unknown" appender="logfile"/>
        <category name="sm" priority="unknown" appender="logfile"/>
        <category name="sockaddr" priority="unknown" appender="logfile"/>
        <category name="transforms" priority="unknown" appender="logfile"/>
        <category name="ts" priority="unknown" appender="logfile"/>
        <category name="parser" priority="unknown" appender="logfile"/>

        <!-- default appenders ==================================== -->
        <appender name="logfile" type="stream" layout="dated"/>
        <appender name="stdout" type="stream" layout="basic"/>
        <appender name="stderr" type="stream" layout="dated"/>
        <appender name="syslog" type="syslog" layout="basic"/>

        <!-- default layouts ======================================= -->
        <layout name="basic" type="basic"/>
        <layout name="dated" type="dated"/>
</log4c>
```

Tablica 14 Primjer log4crc konfiguracijske datoteke

7 Dodatak C

`ikev2` *manual* stranica:

```
IKEV2(8)                                                    IKEV2(8)

NAME
       ikev2 - automatic key exchange daemon

SYNOPSIS
       ikev2 [ -h|--help ] [ -p|--pidfile file ] [ -c|--config file ]


DESCRIPTION
       ikev2(8) is daemon which performs automatic key exchange.

OPTIONS
       This  program  follows  the  usual  GNU  command line syntax, with long
       options starting with two dashes (`--'). A summary of the options  sup-
       ported by ikev2 is below.

       -h, --help
              Show summary of options and exit.

       -p, --pidfile file
             Option  defines the file in which the daemon writes the pidfile.

       -c, --config file
              Option defines ikev2 configuration file.

CONFIGURATION FILE
       ikev2.conf is organized in three parts: non-section parameters, one  or
       more  remote  sections and one or more sainfo sections with parameters.
       Section parameter is a statement which is a list of clauses or it is  a
       block.  Statements are semi-colon terminated and clauses are comma ter-
       minated.  Statements inside of block and those inside  of  the  section
       are enclosed by brackets.  Comments are lines beginning with '#'.

       Meta Syntax

       File  and  path are double quoted strings.  Number is a decimal number.
       Time_unit is s, sec, secs, second, seconds, min, mins, minute, minutes,
       h,  hour,  hours, day, days, week or weeks.  Octet_unit is byte, bytes,
       B, KB, kB, MB, GB or TB.  IPaddress is  IP  address  in  dotted-decimal
       notation, NETaddress is IP address in CIDR notation, email is email and
       fqdn is Fully Qualified Domain name.

       Non-section parameters

       random device file
              Specified file is source of randomness used for generating  ran-
              dom data (i.e. for pre-shared key data derivation).

       path pre_shared_key file
              Specified  file  consists of shared keys and identifiers used to
              authenticate the peer.   Related  directive  is      authentica-
              tion_method (set to pre_shared_key).

              pre_shared_key  file  (psk.txt)  holds  pairs: identifier (IPad-
              dress, email or fqdn) and shared key separated by whitespaces.

              Example of psk.txt:

                   my_ikev2@domain.com       someveryverysecretword
                   your_ikev2@domain.com     someveryverysecretword
```

```
                192.168.0.15              anotherverysecretword
```

path certificate path
Parameter defines the path to the certificate received from the other peer. Related directive is authentication_method (set to rsasig).

dos_treshold number
Maximum number of half-opened IKE_SAs (i.e. in purpose of Denial Of Service attack protection).

sm_threads number
Number of concurrent threads for processing sessions.

Remote section

Remote section parameters define IKE_INIT and IKE_AUTH exchange and protect the IKE traffic with the remote peer. It consists of parameters inside the proposal block and parameters outside of the proposal block. There can be one or more proposal blocks. Parameters outside of the proposal block are default parameters for all proposal suites inside the remote section.

remote anonymous | IPaddress | NETaddress | IPaddress - IPaddress | fqdn { statements }
IP addresses and fqdn are used to specify designated peer to match the remote statements. For others, anonymous is used.

Examples:
remote anonymous { statements }
remote 192.168.1.10 { statements }
remote 192.168.1.0/24 { statements }
remote 192.168.1.1 - 192.168.1.10 { statements }

nonce_size number
Nonce_size number define input to to cryptografic functions and it has to be beetwen 16 and 256.

authlimit time number time_unit
Authlimit time is maximum authentication time after which for active sessions reauthentication is performed and for others removal.

limit time number time_unit | allocs number | octets number octets_unit
Limit statemet defines limits for IKE traffic (maximum time, CHILD_SA allocations or number of transfered octets) after which IKE SA rekeying is performed. These are default parameters for proposals if no other limits are specified in proposals.

ike_max_idle number time_unit
Parameter define maximum idle time after which dead peer detection process for session is started.

response_timeout number time_unit
Parameter defines time after which session retransmission is started.

response_retries number
Parameter defines number of retransmission retries.

my_identifier id_type email | address IPaddress | fqdn fqdn
Parameter define identifier for the peer that is sent to the other peer. rfc822addr, email, address and fqdn are supported types of identifiersi (id_type).

certificate_type x509 cert_file private_key_file

Parameters define name of the certificate file and file with private key.

proposal { statemets }
Cryptographic proposals sent to the other peer for negotiation and parameters specific for that proposal suite negotiation. Following statements have to be inside of the proposal block:

limit time number time_unit | allocs number | octets number octets_unit
Limit statemet defines limits for IKE traffic for specific proposal suite (maximum time, CHILD_SA allocations or number of transfered octets) after which IKE SA rekeying is performed.

encryption_algorithm enc_algs
enc_algs is a list of chosen encryption algorithms to protect IKE traffic. Supported are: des, 3des, null, aes_cbc and aes_ctr. Algorithms are delimited by comma.

auth_algorithm auth_algs
auth_algs is a list of chosen authentication (integrity) algorithms to protect IKE traffic. Supported are: md5_96, sha1_96, aes_xcbc_96. Algorithms are delimited by comma.

pseudo_random_function prf
prfs is a list of chosen pseudo random functions (prf HMAC functions) to protect IKE traffic. Supported are: md5, sha1, aes_cbc. Algorithms are delimited by comma.
dh_group dh_groups
dh_groups is a list of chosen Diffie-Hellman groups to protect IKE traffic. Supported are: modp768, modp1024, modp1536, modp2048, modp3072, modp4096, modp6144. Groups are delimited by comma.

authentication_method pre_shared_key | rsasig
Parameter defines authentication method used to perform IKE traffic protection. Supported are: pre_shared_key and rsasig. Directives related to pre_shared_key are: path pre_shared_key and to rsasig: path certificate, cert_type.

Sainfo section

Sainfo section parameters define CREATE_CHILD_SA exchange and protect the designated IPsec traffic with the remote peer as specified in kernel security policy database by setkey(8).

sainfo anonymous from from_type from_id { statements }

sainfo anonymous from from_type from_id proto (number | protocol | any) { statements }
sainfo anonymous proto (number| protocol | any) icmp-type (any | icmp_type | number | number - number) icmp-code (any | icmp_code | IPaddress | IPaddress - number) { statements }

sainfo src ip_list dst ip_list { statements }

For from_type supported types are: rfc822addr and fqdn (related from_id types are as described in Meta Syntax paragraph). For protocol tcp and icmp are supported.

Examples:

sainfo anonymous from
rfc822_addr my.email@my.email { statements }

sainfo src 192.168.0.10 sport any
dst 192.168.1.0/24 dport 111-120

```
                    proto tcp {statements }

              sainfo src 192.168.2.10, 192.168.0.1-192.168.0.10,
                    192.168.5.0/24 sport 100-200
                    dst 192.168.1.1-192.168.1.15 dport 112-120
                    proto tcp { statements   }
       hardlimit time number time_unit |  octets number  octets_unit
              Hardlimit statements define limits for CHILD SA (maximum time or
              number  of  transfered octets). If limit is exceeded, removal of
              CHILD SA is performed and new SA has to be created.

       softlimit time number time_unit |  octets number  octets_unit
              Softlimit statements define limits for CHILD SA (maximum time or
              number  of transfered octets). If limit is exceeded, rekeying of
              CHILD SA is performed.

       auth_algorithm auth_algs
               auth_algs is a list of authentication (integrity) algorithms to
              protect  designated traffic as defined in kernel policies.  Sup-
              ported algorithms are: md5_96, sha1_96, aes_xcbc_96.

       encryption_algorithm enc_algs
               enc_algs is a list of encryption algorithms to  protect  desig-
              nated  traffic  as  defined in kernel policies.  Supported algo-
              rithms are: des_iv64, des, 3des, rc5, idea, 3idea,  cast,  blow-
              fish, des_iv32, aes128_cbc, null, aes128_ctr.


SIGNAL HANDLING
       SIGINT  and SIGTERM stop the daemon and cleans all established IKE SAs.
       SIGUSR1 dumps  current  informations  about  IKE  SAs  into  log  file.
       SIGUSR1 dumps current informations about IKE SAs into log file.  SIGHUP
       causes configuration file reloading.  SIGPIPE is followed by the daemon
       exit.

LOGGING
       C  library log4c is used for logging to logfile or to stdout.  Configu-
       ration   file   for   log4c   is   log4crc   with    possible    paths:
       ${LOG4C_RCPATH}/log4rc,    ${HOME}/.log4crc    or    ./log4crc   (where
       LOG4C_RCPATH holds the prefix used for installation).

VERSION
       Current version is

BUGS
       The daemon is not prepared to issue request peer and in the  same  time
       receive and process request from the same peer.

       If  IKEv2 daemon receives in proposal two protocols to create/rekey, it
       will ignore one of them.

       If SADB_ACQUIRE is received during/or immediatley after pfkey  protocol
       is registered, the daemon will segfault.
       will ignore one of them.

       If SADB_ACQUIRE is received during/or immediatley after pfkey  protocol
       is registered, the daemon will segfault.

       If retransmission fails, in some cases we should not terminate session.

       If daemon receives very fast two or more duplicate requests for new IKE
       SA, it will become confused.

SEE ALSO
       racoon(8), racoon.conf(5), setkey(8)

       http://sourceforge.net/projects/ikev2              <URL:http://source-
       forge.net/projects/ikev2>
```

AUTHORS

Stjepan Gros <stjepan.gros@fer.hr>, Marko Cupic <marko.cupic@fer.hr>, Udo Schilcher <udo.schilcher@edu.uni-klu.ac.at>, Ana Kukec <ana.kukec@fer.hr>