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

DIPLOMSKI RAD br. 1668

PROŠIRIVI AUTENTIFIKACIJSKI PROTOKOL U IKEv2 OKRUŽENJU

Jelena Vučak

Zagreb, rujan 2007.

Sažetak

U ovom radu opisan je Proširivi autentifikacijski protokol (Extensible authentication Protocol) te programska komponenta WPA_supplicant, kao programsko ostvarenje tog protokola. Također, opisan je protokol za razmjenu ključeva (Internet Key Exchange Protocol). U praktičnom dijelu ovog rada prikazan je način na koji je Proširiv autentifikacijski protokol ostvaren u IKEv2 protokolu. Ostvarena autentifikacija isprobana je na dvije EAP metode te su prikazani dobiveni rezultati.

Abstract

This diploma thesis describes Extensible Authentication Protocol and it's implementation called WPA_supplicant. Also, it describes Internet Key Exchange Protocol. In the practical part of this diploma thesis it is shown how is Extensible authentication protocol implemented into Internet Key Exchange Protocol. Results of this implementation are tested using two EAP methods and are shown at the end of this diploma thesis.

Table of Contents

1 Uvod

Rješavanje problema sigurne komunikacije na Internetu predstavlja velik izazov na području računarstva. Do sada je pronađeno nekoliko rješenja, koja pokušavaju ovaj problem svesti na minimum. SSL (*Secure Sockets Layer*), S-HTTP (*Secure Hyper Text Transfer Protocol*) i IPsec (*Internet Protocol Security*) neka su od mogućih rješenja. Zadnji navedeni, IPsec je skup protokola ostvarenih u jezgri operacijskog sustava, na 3. sloju TCP/IP arhitekture. Skup protokola IPsec-a, čine protokoli za kriptografsku zaštitu paketa (AH i ESP) i protokoli za razmjenu ključeva (IKE i IKEv2). IKEv2 protokol (*engl. Internet Key Exchange protocol*) je protokol za razmjenu ključeva opisan u RFC4306. Ovaj rad se bavi ostvarenjem proširivog autentifikacijskog protokola u IKEv2 protokol. Proširiv autentifikacijski protokol (*engl. Extensible Authentication Protocol – EAP*) opisan u [17] određen je za provedbu autentifikacije u IKEv2 protokolu. EAP podržava nekoliko različitih metoda za provedbu autentifikacije. Protokol i njegove metode poprilično su složeni za ostvarenje od samog početka, te je za ovaj projekt uzet već gotov izvorni tekst programa EAP protokola i njegovih metoda – programska komponenta *WPA_supplicant*. Za uspješno ukomponiranje kôda *WPA_supplicanta* u IKE protokol bilo je potrebno načiniti neke promjene na postojećem IKE kôdu te napisati dodatne funkcije.

Ovaj rad započinje pobližim prikazom proširivog autentifikacijskog protokola. Pod tim prikazom se podrazumijeva opis razmjene poruka EAP protokolom i izgled EAP paketa koji se razmjenjuju prilikom provedbe autentifikacije. Osim općenitih podataka o EAP protokolu, na temelju pročitanih RFC - ova pobliže je opisano nekoliko EAP metoda.

Treće poglavlje donosi opis programske komponente *WPA_supplicant*, kao programskog ostvarenja EAP protokola. Prikazano je okruženje u kojem se *WPA_supplicant* u stvarnosti nalazi te arhitektura *WPA_supplicanta*.

Daljnji tekst objašnjava protokol za razmjenu ključeva (IKEv2 protokol): uspostavu sigurnosne poveznice (SA), sigurnosnu politiku (SP), SAD i SPD baze, IKE *daemon* i njegovu ulogu te automate stanja IKE protokola.

Cilj praktičnog dijela ovog rada je ostvarenje autentifikacije EAP protokolom u IKEv2 protokolu na strani pokretača komunikacije. Za uspješno ostvarenje bilo je potrebno: prilagoditi programsku komponentu *WPA_supplicant*, napisati funkcije za kontrolu automata stanja proširivog autentifikacijskog protokola, promijeniti automat stanja protokola za razmjenu ključeva na strani pokretača komunikacije te složiti odgovarajuće konfiguracijske datoteke za IKEv2 *daemon* i za određenu metodu EAP protokola. Cjelokupan prikaz praktičnog rada, kao i rezultati ispitivanja istog, prikazani su u petom poglavlju.

Za ostvarenje EAP protokola u IKEv2 okruženju, bilo je potrebno načiniti promjene na strani pokretača komunikacije (*inicijatora*) i na strani koja provodi autentifikaciju (*odgovaratelju*). Ovaj rad donosi prikaz promjena nastalih na strani inicijatora.

2 Proširivi autentifikacijski protokol

EAP *(engl. Extensible Authentication Protocol)* je proširiv autentifikacijski protokol koji podržava različite autentifikacijske metode i opisan u [3]. Koristi se na podatkovnom sloju žičnih ili bežičnih mreža. Kod bežičnih mreža implementira se u pristupnim točkama i preklopnicima *(engl. switch)*. Prednost ovog autentifikacijskog protokola je njegova proširivost, koja se očituje u izboru autentifikacijske metode koja će se koristiti. One se mogu naknadno dodavati bez potrebe za promjenom kompletnog kôda pristupne točke. Korištenjem EAP-a nije potrebno da a*utentifikator* podržava sve autentifikacijske metode, nego se koristi autentifikacijski poslužitelj (*engl. Authentication server*). Taj poslužitelj implementira sve (ili samo neke) metode, a autentifikator se onda ponaša kao prijelazni poslužitelj *(engl. pass-through)* za metode. Može se reći kako autentifikacijski poslužitelj obavlja autentifikaciju za autentifikatora tj. obavlja autentifikacijske metode za autentifikatora. Uz autentifikatora i autentifikacijskog poslužitelja, kod EAP-a treba poznavati i pojam *supplicant*, koji se još naziva i partner (*peer*). Kako je autentifikator dio koji pokreće EAP autentifikaciju, supplicant je dio koji odgovara autentifikatoru. Dakle, postoji veza između partnera i autentifikatora i omogućena im je komunikacija. Komunikacija podrazumijeva razmjenu poruka između njih, što čini autentifikaciju.

Prednosti EAP protokola su što, kao što je već rečeno, omogućava upotrebu različitih metoda autentifikacije.

Također, jedna od prednosti je što uređaji za pristup mrežnom poslužitelju (npr. pristupne točke, preklopnici) ne moraju poznavati svaku metodu autentifikacije. Tada se ponašaju kao prijelazni dio za "stražnje" autentifikacijske poslužitelje.

Prednost EAP protokola u brojnim EAP metodama predstavlja ujedno i nedostatak EAP protokola. Razlog tomu je što mnoge EAP metode nisu dobro dokumentirane (ne postoje njihovi RFCovi). To predstavlja teškoće u ostvarivanju tih metoda. Također, tada postaje teško i koristiti te metode, jer ostaju nepoznate informacije potrebne za pokretanje metoda (parametri i slično).

2.1 Razmjena poruka EAP protokolom

Razmjena poruka između partnera i autentifikatora odvija se sljedećim redoslijedom:

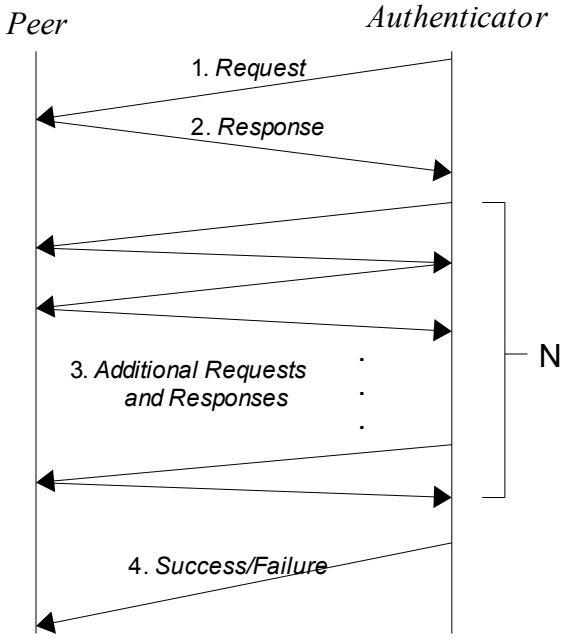
1. Autentifikaciju započinje autentifikator slanjem zahtijeva za autentifikacijom (*engl. Request*) partneru. Ta poruka sadrži polje (*payload*) *Type* koji određuje kakva je vrsta zahtijeva.
2. Zatim slijedi odgovor partnera (*engl. Response*). Kao i zahtijev, odgovor sadrži polje *Type*, koje odgovara polju Type u prije poslanom zahtijevu.
3. Autentifikator zatim šalje dodatne poruke (zahtijeve), a partner odgovara na zahtijeve. Izmjena zahtijeva i odgovora nastavlja se do daljnjega tj. do završetka

autentifikacije (neovisno, radi li se o uspješnoj ili neuspješnoj autentifikaciji). Novi zahtijev ne može se poslati dokle god nije stigao odgovor na prije poslani zahtijev. To vrijedi za sve zahtijeve, osim početnog, kojim autentifikator započinje proces autentifikacije.

4. Uspješnom autentifikacijom se smatra kada autentifikator autentificira partnera i tada može poslati poruku o uspješnoj autentifikaciji (*Success*).

   Ukoliko autentifikator nije primio odgovore na nekoliko odaslanih zahtijeva, autentifikacija partnera se smatra neuspješnom i autentifikator tada šalje poruku o neuspješnoj autentifikaciji (*Failure*).

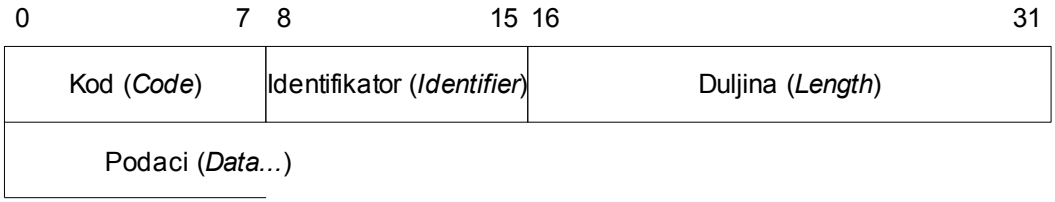
Razmjena EAP paketa prikazana je na slici 2.1.1.



Slika 2.1.1. Razmjena EAP paketa

2.2 Format EAP paketa

Format EAP paketa prikazan je na slici 2.2.1.



Slika 2.2.1. Format EAP paketa

Prvi oktet EAP paketa zauzima polje *Code*, koje sadrži tip (*Type*) EAP paketa. Moguće vrijednosti polja *Code* su prikazane u tablici 2.2.1.

Tablica 2.2.1 Vrijednosti polja Code EAP paketa

1	Request
2	Response
3	Success
4	Failure

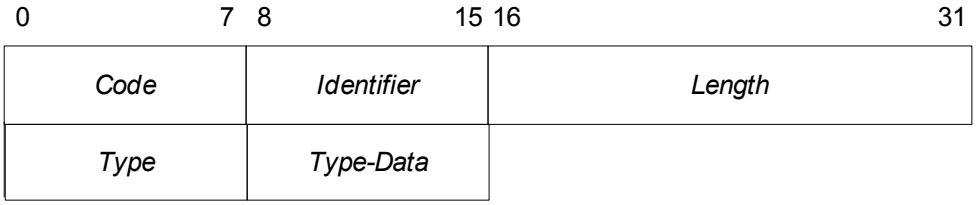
EAP protokol podržava samo gore navedene vrijednosti polja *Code*, te svi paketi s vrijednostima tog polja različitim od gore navedenih vrijednosti trebaju biti odbačeni.

Sljedeći oktet zauzima polje *Identifier*, a zadatak mu je povezivati upite zahtjeva i odgovora na njih.

Polje *Length* sadrži vrijednost koja označava duljinu EAP paketa (u oktetima). Duljina EAP paketa uključuje polja *Code*, *Identifier*, *Length* i *Data*. Svi okteti koji prelaze vrijednost navedenu u polju *Length* se po primitku odbacuju, zanemaruju i smatraju se *padding-om* podatkovnog sloja.

Nakon prva 4 okteta slijede podaci u polju *Data*. Duljina tog polja varira i može poprimiti vrijednosti 0 ili nekoliko okteta. Tip podataka EAP paketa ovisi o vrijednosti u polju *Code*, odnosno o tipu EAP paketa.

EAP paketi se mogu podijeliti u dva “podtipa”: Request/Response tip paketa i Success/Failure tip paketa. Request/Response tip paketa u polju *Code* sadrži vrijednost 1 (Request) tj. 2 (Response). Request šalje autentifikator, a namijenjen je kao upit partneru za autentifikacijom. Response paket je odgovor partnera autentifikatoru. Na slici 2.2.2 je prikazan format Request/Response tipa paketa. U odnosu na općeniti format EAP paketa, ovaj se format razlikuje samo u polju *Type*. To polje je duljine jednog okteta, a sadrži heksadecimalnu vrijednost koja označava različite tipove Request/Response paketa. Različitim EAP paketima, određene su i različite EAP metode, o čemu će biti više riječi kasnije.



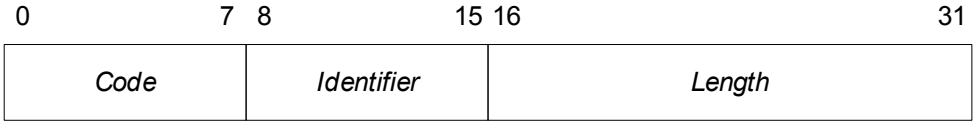
Slika 2.2.2. Format Request/Response paketa

Neke vrijednosti polja *Type* su prikazane u tablici 2.2.2.

Tablica 2.2.2 Vrijednosti polja Type

1	Identity
2	Notification
3	Nak (samo za Response pakete)
4	MD5-Challenge
5	OTP (One Time Password)
6	GTC (Generic Token Card)
13	TLS (Transport Level Security)
26	EAP MSCHAPv2
254	Expanded types
255	Experimental use

Sve EAP implementacije moraju podržavati tipove od 1 do 4, a trebale bi i tip 254. Nakon polja *Type*, a ovisno o njemu, slijedi polje *Type-Data*, koje sadrži podatke paketa tipa *Type*. Drugi tip EAP paketa, Success/Failure paket, je prikazan na slici 2.2.3.



Slika 2.2.3. Format Success/Failure paketa

Taj tip paketa sadrži samo tri polja, koja su ista i s istom funkcijom, kao i u općem prikazu EAP paketa. Treba napomenuti da Success paket u polju *Code* ima oznaku 3, a Failure paket ima oznaku 4.

2.3 EAP metode

Kao što je već spomenuto EAP metode se razlikuju po tipu paketa tj. po polju *Type* u EAP paketu. Također je već spomenuto da sve EAP implementacije moraju podržavati tipove 1, 2, 3 i 4 tj. Identity, Notification, Nak i MD5-Challenge. Uz njih trebale bi podržavati i tip 254 tj. Expanded types. Uz te, već spomenute tipove, u ovom će poglavlju biti prikazani i neki drugi tipovi autentifikacijskih poruka (metode), koje su implementirane u *WPA_supplicant* programu. To su metode TLS, GTC, SIM, PSK, OTP i druge. Njihov tip se numerira počevši s 4.

Identity

Tip Request/Response paketa. Request predstavlja upit autentifikatora za identitetom partnera, a najčešće se šalje kao prvi paket, na početku autentifikacije. Odgovor partnera se šalje u Response paketu (tipa 1). Preporuča se Response paket tipa 1 (Identity) upotrebljavati za dogovor oko korištenja EAP metoda (sadrži broj/brojeve autentifikacijske/ih metode/a, s kojima partner želi provesti autentifikaciju). Identity Request i Identity Response paketi se šalju u čistom (nekriptiranom) obliku.

Notification

Tip RequestResponse paketa. EAP paket tipa 2 služi za prijenos poruke od autentifikatora do partnera.

NAK

Tip Response paketa, koji se koristi kada je potrebno autentifikatora obavijestiti kako se traženi tip autentifikacije (broj sadržan u Nak paketu) ne može prihvatiti tj. metoda provesti. Ukoliko Nak paket sadrži broj 0, tada to znači da ne postoji niti jedna željena metoda za provedbu autentifikacije, te da autentifikator više ne treba slati Request pakete.

EAP-MD5-Challenge

Tip Request/Response paketa. Request paket predstavlja poziv (*challenge*) partneru, a Response odgovor autentifikatoru, s tim da se koristi MD5 funkcija sažimanja.

EAP-OTP

Metoda EAP-OTP (*One Time Password*), tj. tip autentifikacije koja sprječava napad nastao

prisluškivanjem (*engl. Eavesdropping*). Naime, prisluškivanjem mreže mogu se otkriti povjerljive informacije koje služe za autentifikaciju korisnika. Takve informacije su npr. login korisnika, zaporke korisnika i slično. S tim podacima omogućen je napadaču pristup računalnom sustavu, za kojeg nema dopušten pristup. Ova metoda koristi tajni *pass-phrase* korisnika, kojim se generira jednokratna (*single use*) zaporka korisnika. Dodatna sigurnost koju pruža ova metoda je što se ni u kojem trenutku korisnikov tajni pass-phrase ne treba prenositi mrežom i da nikakvi tajni podaci ne trebaju biti pohranjeni na nekom poslužitelju.

EAP-GTC

Metoda EAP-GTC (*Generic Token Card*) je tip koji se koristi s različitim *Token Card* implementacijama, a koje zahtjevaju unose korisnika. Odgovor (Response paket) sadrži Token Card, informacije potrebne za autentifikaciju. Polje *Type* u paketu je postavljeno na vrijednost 6.

EAP-TLS

Metoda EAP_TLS (*Transport Level Security*) za obostranu autentifikaciju omogućava mehanizme za zaštitu integriteta i razmjenu ključeva između dvije krajnje točke. Kao i kod svake autentifikacije EAP-TLS metoda započinje razmjenom Identity Request/Response. Nakon saznavanja identiteta partnera EAP poslužitelj šalje EAP-TLS Start paket, kojim započinje TLS komunikacija (u EAP polju *Type*, je postavljen tip EAP-TLS, vrijednost 13). Zatim slijedi *client_hello_handshake* poruka (unutar EAP Response paketa, također s poljem *Type* postavljenim na EAP-TLS), koja sadržava verziju TLS-a, random broj, skup kriptografskih metoda podržanih od strane klijenta. Poslužitelj odgovara svojim paketom (*server_hello_handshake*) koji sadrži TLS zapise (TLS cerifikat, ključ poslužitelja, zahtjev za certifikatom, itd.). Poslužitelj šalje zahtjev za certifikatom kako bi usporedio identitet partnera kojeg je primio u prvoj poruci (Identity) i identitetom sadržanim u certifikatu. U tu svrhu i partner šalje zahtjev za certifikatom. Svaka uspostava "veze" tj. komunikacije partnera i poslužitelja ima svoj vlastiti SessionID, te skup kriptografskih parametara (metoda, ključeva), koje partner šalje poslužitelju, a ovaj to prihvaća. Ako je postupak autentifikacije uspješno završen autentifikator (poslužitelj) šalje odgovor (Response paket) kojim završava komunikacija s *handshake* porukama. Partner tada šalje TLS paket bez podataka u polju *Data*, a EAP poslužitelj na to odgovara sa Success porukom. U slučaju neuspješne autentifikacije, poslani odgovor sadrži i razloge zbog kojih autentifikacija nije uspjela.

EAP-TLS paketi sadrže i jedno dodatno polje (u odnosu na osnovni EAP paket). To je 8-bitno polje *Flags*, kojim se postavljaju zastavice LMSRRRRR, ovisno o tome da li je u paketu uključeno polje *Length* (L), dodatni fragmenti (M), radi li se o EAP_Start paketu (S), te rezervirani bitovi (RRRRR).

LEAP

LEAP (*Lightweight EAP*) je tip Radius EAP autentifikacijskog protokola, koji se koristi za autentifikaciju bežičnog klijenta (prijenosnog računala ili računala s bežičnom karticom). Autentifikaciju ovom metodom čine poruke (Radius paketi) koje se razmjenjuju između pristupne točke (AP) i Radius poslužitelja. Protokol započinje porukom pristupne točke Radius poslužitelju s imenom korisnika koji je zatražio autentifikaciju. Poslužitelj odgovara porukom Radius Challenge (sadrži random MSCHAP Peer Challenge – PC). Treću poruku šalje pristupna točka poslužitelju s odgovorom na PC poslan prethodnom porukom (PR). Ako je autentifikacija uspjela Radius poslužitelj odgovara Radius Access- Accept porukom. Zatim AP šalje Radius Request poruku koja sadrži APC (Access Point Challenge). Poslužitelj na to odgovara porukom koja sadrži *session* ključ u sljedećem obliku Radius atributa: "leap:session-key=nnnn". Sve poruke su EAP oblika, polje *Type* ima postavljenu vrijednost 17, a polje *Type-Data* je podijeljeno na sljedeće dijelove: 1 oktet s brojem verzije LEAP protokola; 1 oktet postavljen na vrijednost 0x00; oktet s duljinom polja koje slijedi (*Binary-Data*); *m* okteta polja *Binary-Data*; *n* okteta s imenom korisnika koji je zatražio autentifikaciju. LEAP je još poznat i pod nazivom EAP-CISCO, jer se koristi u CISCO-vim usmjerivačima.

EAP-SIM

Metoda, koja za autentifikaciju i raspodjelu ključa koristi GSM SIM modul (*Subscriber Identity Module*). Takva autentifikacija se zasniva na algoritmima SIM modula, koji kao parametre primaju 128-bitni slučajni broj (RAND) i tajni ključ pohranjen na SIM modulu, a kao rezultat vraćaju 32-bitni odgovor (SRES) i 64-bitni ključ K_c, (koji se dalje upotrebljava za enkripciju, ali i za dobivanje drugih ključeva potrebnih za daljnje enkripcije). Format EAP_SIM paketa izgleda kao i osnovni format EAP paketa (polja: *Code*, *Identifier*, *Length*, *Type*-postavljen na 18), nakon kojeg slijede polja *Subtype* i *Reserved*. Polje *Subtype* je određeno EAP_AKA specifikacijama, a ostatak paketa (Reserved) sadrži atribute definirane poljima *Attribute Type*, *Length* i *Value* (definiraju tip atributa, duljinu te podatke tog atributa).

Postoji 9 tipova EAP_SIM poruka, a oni su: EAP-Request/SIM/Start, EAP-Response/SIM/Start, EAP-Request/SIM/Challenge, EAP-Response/SIM/Challenge, EAP-Request/SIM/Re-authentication, EAP-Response/SIM/Re-authentication, EAP-Response/SIM/Client-Error, EAP-Request/SIM/Notification, EAP-Response/SIM/Notification. Poruke se međusobno razlikuju po atributima koje u sebi moraju/mogu sadržavati. Vrste atributa su: AT_VERSION_LIST, AT_SELECTED_VERSION, AT_NONCE_MT, AT_PERMANENT_ID_REQ, AT_ANY_ID_REQ, AT_FULLAUTH_ID_REQ, AT_IDENTITY, AT_RAND, AT_NEXT_PSEUDONYM, AT_NEXT_REAUTH_ID, AT_IV, AT_ENCR_DATA,

AT_PADDING, AT_RESULT_IND, AT_MAC, AT_COUNTER, AT_COUNTER_TOO_SMALL, AT_NONCE_S, AT_NOTIFICATION, AT_CLIENT_ERROR_CODE.

EAP-AKA

Ova se metoda koristi u općim pokretnim telekomunikacijskim sustavima (*UMTS – Universal Mobile Telecommunication System*), a koristi AKA mehanizam (*Authentification and Key Agreement*). Taj se mehanizam zasniva na simetričnoj kriptografiji, a nalazi se na SIM modulima tj. USIM (*UMTS SIM*) modulima, RUIM (*Removable User Identity Module*, slični pametnim karticama). Autentifikacija se temelji na komunikaciji EAP poslužitelja (smještenom na stražnjem autentifikacijskom poslužitelju, koristeći AAA protokol-*Authentication*, *Autorization*, *Accounting*) i autentifikatora. Započinje ramjenom Identity Resquest/Response poruka (kao i kod drugih metoda), a nakon saznavanja identiteta pretplatnika (*subscriber-a*), stvara autentifikacijski vektor, koji služi za daljnju autentifikaciju. Zatim EAP poslužitelj započinje sa AKA algoritmom za daljnju autentifikaciju. Paketi koji se razmjenjuju ovom metodom u polju *Type* imaju zapisanu vrijednost 23.

EAP-PEAP

PEAP (*Protected Extensible Authentication Protocol*) je EAP metoda, s poljem *Type* paketa postavljenim na 25. Paket PEAP metode sličan je paketu EAP-TLS metode. Radi se o metodi koja je vrlo slična TLS metodi, a primjenjuje se kod bežičnih mreža. Ova metoda je podijeljena u dvije faze. U prvoj fazi autentifikacije se uspostavlja sigurnosni tunel između autentifikatora i poslužitelja, koristeći EAP-TLS za autentifikaciju poslužitelja. U drugoj se fazi autentificira klijent, koristeći bilo koju EAP metodu.

PEAP protokol zahtijeva certifikate i to samo za autentifikaciju poslužitelja. To ujedno čini i prvi korak autentifikacije. Zatim klijent treba uspostaviti vezu s autentifikatorom, a autentifikator treba uspostaviti sigurnosni kanal s poslužiteljem. Zatim slijede uobičajene Request/Response poruke između klijenta i autentifikatora te autentifikatora i poslužitelja, a nakon toga EAP-TLS protokol (razmjena poruka EAP-TLS metode). Završetkom EAP-TLS metode uspostavljen je sigurnosni kanal, te završava prva faza autentifikacije. Druga faza započinje razmjenom Identity poruka, a zatim slijedi autentifikacija klijenta nekom od EAP metoda (MD5, CHAP, SIM itd.). Autentifikacija se odvija sigurnosnim kanalom uspostavljenim u prvoj fazi PEAP protokola. Slijedi računanje ključeva (klijent i poslužitelj računaju potrebne ključeve), a zatim te ključeve prima autentifikator, kao i rezultat autentifikacije. Time se završava autentifikacija: klijent i pristupna točka mogu sigurno razmjenjivati poruke.

EAP-MSCHAPv2

Metoda koja omogućuje obostranu autentifikaciju i metode računanja ključeva, kakve se koriste u MPPE kriptiranju (*engl. Microsoft Point to Point Encryption*). Skraćenica je nastala od naziva *Microsoft Challenge Handshake Authentication Protocol version 2*. Koristi se kao druga faza autentifikacije, nakon obavljenog protokola metode EAP-PEAP. Format paketa ove metode se malo razlikuje od osnovnog formata paketa, jer su dodana polja *OpCode*, *MS-CHAPv2-ID*, *MS-Length* i *Data*. *OpCode* polje definira tip EAP-MSCHAPv2 paketa, koji može biti: 1-Challenge, 2-Response, 3-Success, 4-Failure, 7-Change Password. *Type* polje paketa ove metode postavljeno je na vrijednost 26. Ono što razlikuje ove pakete je polje *Data*.

Tako za Challenge tip paketa ono sadrži podatke Challenge.

Za Response tip paketa polje *Data* sadrži polje *Respons*e, koji se dalje dijeli na 4 podpolja: *Peer-Challenge* (random broj), *Reserved* (mora biti 0), *NT-Response* (password), *Flags* (mora biti 0).

Success tip paketa ima dvije podvrste: Success Request i Success Response. Success Request u polju *Data* sadrži poruku (Message) formata: "S=<auth_string> M=<message>". Success Response paket od dodatnih polja, koje imaju paketi EAP-MSCHAPv2 metode, ne sadrži niti jedno, osim polja *OpCode* (s vrijednošću 3).

Failure tip paketa se isto dijeli na dva podtipa: Request i Response. Failure Request paket sadrži polje *Message* oblika: "E=eeeeeeeeee R=r C=cccccccccccccccccccccccccccccccc V=vvvvvvvvvv M=<msg>". Failure response paket, kao i Success Response paket, od dodatnih polja sadrži samo *OpCode*.

Paket tipa Change Password u polju *Data* ima 6 podpolja, a ona su: *Encrypted-Password*, *Encrypted-Hash*, *Peer-Challenge*, *Reserved*, *NT-Response*, *Flags*. Ovom metodom je podržana zaštita integriteta, ali nije podržana zaštita pouzdanosti.

EAP-PSK

Temelji se na simetričnoj kriptografskoj metodi AES-128, što metodu EAP-PSK (*Pre Shared Key*) čini jednostavnom za implementirati. PSK predstavlja 128-bitni ključ, kojeg znaju samo EAP poslužitelj i EAP partner (peer), te služi za računanje daljnjih ključeva (*Authentification Key-AK* i *Key Derivation Key-KDK*). Iz KDK se dalje računaju drugi potrebni ključevi. Uz PSK, metoda koristi i identifikatore poslužitelja i partnera (ID_S i ID_P). Autentifikacija ovom metodom se zasniva na AKEP2 protokolu, a čine ju četiri poruke. U slučaju uspješne autentifikacije EAP-PSK metoda stvara zaštićeni kanal za komunikaciju obiju strana (pri tome koristi EAX metodu). Zaštićeni kanal omogućuje prijenos poruka s kriptiranim tekstom (u daljnjem tekstu - PCHANNEL). Prvu EAP-PSK poruku šalje poslužitelj partneru (server -> partner). Ta poruka ima (uz uobičajena polja EAP

paketa) još dodatna dva polja: *RAND_S* (random broj) i *ID_S* (*server's NAI*). Drugu poruku šalje partner poslužitelju, a njena dodatna polja su: *RAND_S* (isti kao u 1. poruci), *RAND_P* (random broj), *MAC_P* (MAC adresa partnera), *ID_P* (peer's NAI). Treću poruku šalje opet poslužitelj partneru, a čine ju dodatna polja: *RAND_S* (isti kao u prvoj poruci), *MAC_S* (MAC adresa poslužitelja), *PCHANNEL*. *PCHANNEL* polje se sastoji od podpolja: *Nonce N*, *Tag*, *zastavica R*, *zastavica E*, *Reserved*. Četvrtu poruku šalje partner poslužitelju, s poljima RAND_S (isti kao i u 1. poruci) te PCHANNEL.

EAP-PAX

Metoda EAP-PAX (*Password Authenticated Exchange*)za autentifikaciju koristi dijeljeni ključ, a može se podijeliti na dva "podprotokola": PAX_STD i PAX_SEC. PAX_STD obavlja obostranu autentifikaciju koristeći dijeljeni tj. zajednički ključ, a PAX_SEC nadopunjuje prije spomenuti protokol, pružajući zaštitu identiteta koristeći javni ključ poslužitelja. EAP-PAX ima svojstva, koja zadovoljavaju EAP zahtjeve za sigurnošću bežičnih mreža, kao npr.: obostrana autentifikacija; otpornost napadima posrednikom (*engl. man-in-the-middle attack*); zaštita identiteta; autentificirana zaštita podataka itd. Paketi ove metode se prepoznaju po polju *Type* postavljenim na 46.

EAP-FAST

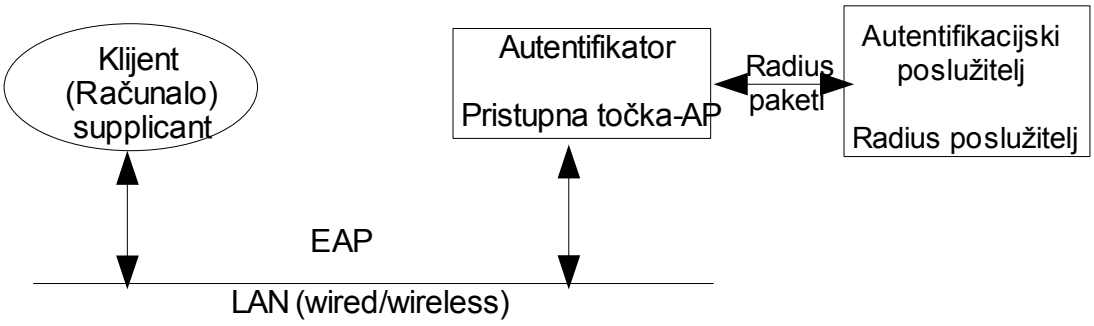
EAP-FAST autentificira i klijenta i autentifikacijskog poslužitelja, koristeći dijeljenu tajnu, poznatu pod nazivom PAC (*Protected Access Credential*). Metoda se može podijeliti na dva dijela: prvi - pripremni i drugi – autentifikacijski. U prvom dijelu se PAC šalje klijentu i poslužitelju i taj se dio obavlja samo jednom (ručno ili automatski). Automatsko slanje stvara tunel, kojim putuju kriptirane poruke i na taj način se osigurava autentifikacija klijenta i sigurna dostava PAC-a klijentu. Taj je mehanizam manje siguran od ručnog slanja, ali je sigurniji od LEAP mehanizma (prepoznaje i onemogućuje napade posrednikom (*engl. man-in-the-middle-attack*). Nakon dobivanja informacija sadržanih u PAC dijeljenoj tajni, slijedi autentifikacija klijenta i poslužitelja, stvaranje kriptiranog tunela, obnavljanje dijeljene tajne (refresh PAC), autentifikacija klijenta koristeći neku EAP metodu te objava autorizacije klijentu (opcionalno).

3 Programska komponenta *WPA_supplicant*

WPA_supplicant je komponenta koja se izvršava u stanicama klijenata i zadužena je za njihovu autentifikaciju kako bi klijenti mogli pristupiti računalnoj mreži. Naziv je sročen od dviju riječi WPA i supplicant. WPA je skraćenica od *Wi-Fi Protected Access* i označava zaštitnu enkripcijsku metodu, a koristi se za zaštitu bežičnih mreža. Ta je metoda razvijena nakon probijanja WEP-a (*engl. Wired Equivalent Privacy*), pa se često spominje i kao WEP2. Bazira se na RC4 algoritmu i pruža bolju enkripciju (od WEP-a) jer se temelji na TKIP (*engl. Temporal Key Integrity Protocol*) protokolu, u kojem se ključ kriptiranja mijenja svakih 10000 paketa. Ova metoda za autentifikaciju koristi EAP (*engl. Extensible Authentication Protocol*). Riječ *supplicant* u nazivu označava komponentu u klijentu, kojoj je zadatak dogovaranje s autentifikatorom oko ključeva koji se upotrebljavaju za zaštitu, te autentifikacija i autorizacija sa bežičnim upravljačkim programima (*engl. wlan drivers*).

Okruženje u kojem se upotrebljava *WPA_supplicant* prikazano je na slici 3.1. Na slici se vidi da se u procesu autentifikacije javljaju tri strane tj. tri sudionika. Prvi sudionik je klijent, kojeg predstavlja računalo (prijenosno ili stolno), a u kojem se nalazi supplicant. Klijent je spojen na lokalnu mrežu (žično ili bežično). Na lokalnu mrežu je spojen i autentifikator, kojeg na slici predstavlja pristupna točka (*engl. access point – AP*). Treći sudionik je autentifikacijski poslužitelj, primjerice RADIUS poslužitelj. EAP protokol se koristi između klijenta i pristupne točke (AP), preko lokalne mreže, dok se između autentifikatora i autentifikacijskog poslužitelja razmjenjuju RADIUS paketi.

Na prikazanoj slici predstavljena je ideja korištenja *WPA_supplicant-a*. Međutim, korištenje



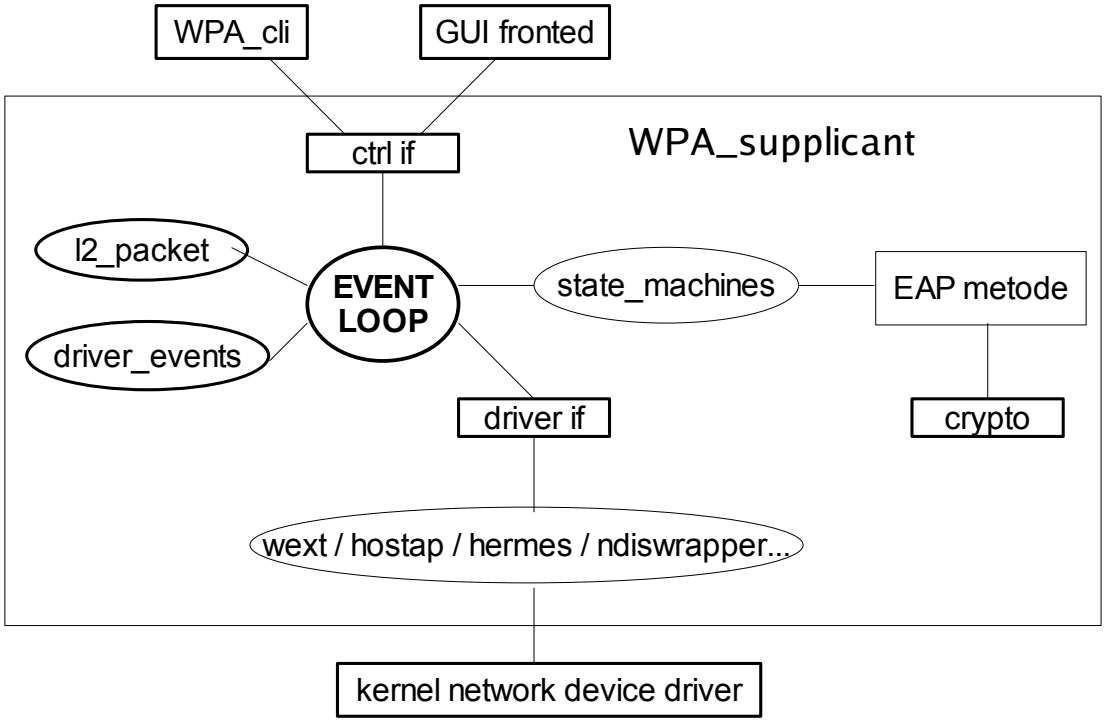
Slika 3.1. Okruženje u kojem se koristi *WPA_supplicant*

WPA_supplicant-a je znatno složenije, a programska realizacija poprilično složena. Razlog tomu je složenost proširivog autentifikacijskog protokola (EAP - a), čije je *WPA_supplicant* programsko ostvarenje. Zato se *WPA_supplicant* sastoji od nekoliko, međusobno povezanih dijelova. Više riječi o tome bit će u idućem poglavlju.

3.1 Arhitektura *WPA_supplicant-a*

Glavni dio programskog ostvarenja *WPA_supplicant-a* čini autentifikacijski protokol EAP tj. automat stanja EAP protokola (EAP state machine). EAP koristi mnoge metode autentifikacije, a one pak koriste mnoge kriptografske algoritme (EAP metode, crypto, TLS). Uz automat stanja EAP-a, tu su

još i automat stanja WPA te EAPOL-a (*engl. EAP over LAN*). Za komunikaciju s vanjskim programima *WPA_supplicant* koristi kontrolno sučelje, kojeg upotrebljavaju programi za konfiguraciju. Tako se razlikuju `wpa_cli` i `wpa_gui`. Kontrolnim sučeljem omogućava se vanjskim programima dohvat informacija o rezultatima operacija koje se u *WPA_supplicant-u* događaju (proces autentifikacije). Uz do sada spomenute module, važan modul predstavlja i sučelje prema mrežnim sučeljima (*engl. Driver wrapper implemetation*). Spomenuti modul sadrži skup manjih datoteka s implementiranim generičkim kôdom, kojim se zamijene dijelovi upravljačkih programa (*driver-a*) ovisni o raspoloživom sklopovlju, programskoj podršci ili operacijskom sustavu. Posljednji modul važan za spomenuti jest l2_layer packet, koji predstavlja sučelje za razmjenu paketa putem 2. sloja. Arhitektura programske komponente *WPA_supplicant* prikazana je na slici 3.1.1.



Slika 3.1.1. Arhitektura programske komponente *WPA_supplicant-a,* [REF12]

3.2 Funkcije automata stanja proširivog autentifikacijskog protokola

Pod pojmom funkcija automata stanja proširivog autentifikacijskog protokola podrazumijevaju se sve funkcije kojima se mogu kontrolirati EAP automati stanja, npr. stvaranje i uništavanje EAP automata stanja. Sve su funkcije ostvarene u programskoj komponenti *WPA_supplicant*.

Autentifikacija na strani inicijatora započinje inicijalizacijom EAP automata stanja. To se ostvaruje funkcijom `eap_sm_init` koja se nalazi u datoteci `eap.c`. Prototip funkcije je sljedeći:

```
struct eap_sm *eap_sm_init(void *, struct eapol_callbacks *,
struct eap_config *);
```

Prvi argument funkcije je pokazivač na podatke potrebne za pozive eapol_cb (eapol callbacks). Svaka callback funkcija, prilikom poziva, će dobiti ovaj argument. Time je olakšano ostvarivanje više konkurentnih automata stanja. Drugi argument funkcije je pokazivač na strukturu `eapol_callbacks`, koja sadrži pokazivače na callback funkcije koje koriste EAP automati stanja prilikom pristupa dodatnim kontrolnim varijablama. Treći argument je također pokazivač i to na strukturu koja sadrži konfiguracijske parametre za EAP automate stanja i EAP metode. Radi se o strukturi `struct eap_config` (definiranoj u `eap.h`), a prototip joj je prikazan na slici 3.2.1.

```
struct eap_config {
         /**
          * opensc_engine_path - OpenSC engine for OpenSSL
          * engine support
          *
          * Usually, path to engine_opensc.so.
          */
          const char *opensc_engine_path;
         /**
          * pkcs11_engine_path - PKCS#11 engine for OpenSSL
          * engine support
          *
          * Usually, path to engine_pkcs11.so.
          */
          const char *pkcs11_engine_path;
         /**
          * pkcs11_module_path - OpenSC PKCS#11 module for
          * OpenSSL engine
          *
          * Usually, path to opensc-pkcs11.so.
          */
  const char *pkcs11_module_path;
};
```

Slika 3.2.1. Prototip strukture `struct eap_config`

Kad je EAP automat stanja inicijaliziran, prelazak između njegovih stanja ostvaruje se funkcijom `eap_sm_step`, prototipa:

```
int eap_sm_step(struct eap_sm *)
```

Funkcija prima jedan argument – pokazivač na EAP automat stanja, koji treba obraditi neki zahtijev. Funkcija vraća vrijednost 1 ukoliko EAP automat stanja promijeni stanje, a u protivnom vraća 0. Po završetku autentifikacije EAP automat stanja treba izbrisati iz memorije, a za to se koristi funkcija:

```
void eap_sm_deinit (struct eap_sm *);
```

Ovom se funkcijom oslobađaju svi reursi zauzeti inicijalizacijom EAP automata stanja.

Za dohvat odgovora iz automata stanja koristi se funkcija `eap_get_eapRespData` prototipa:

```
u8 * eap_get_eapRespData(struct eap_sm *, size_t *);
```

Funkcija vraća pokazivač na polje s odgovorom, dobivenog iz EAP automata stanja, a kojeg treba proslijediti autentifikatoru. Prvi argument funkcije je pokazivač na EAP automat stanja, koji je inicijaliziran funkcijom `struct eap_sm *eap_sm_init`, a drugi argument je pokazivač na varijablu u koju će se spremiti duljina odgovora.

U slučaju kada treba poništiti proces autentifikacije (a time i automat stanja), npr. u slučaju kada se predugo čeka odgovor od strane odgovaratelja i kada istekne vremensko ograničenje

(*timeout)*, koristi se funkcija `eap_sm_abort` prototipa:

```
void eap_sm_abort(struct eap_sm *);
```

Neke EAP metode po uspješnom završetku autentifikacije generiraju glavni ključ sjednice (*master session* key, MSK). IKEv2 protokol zahtijeva da se završna autentifikacija obavi s tim ključem i stoga je potrebno dohvatiti ključ iz automata stanja. Za to se koristi funkcija:

```
const u8 *eap_get_eapKeyData(struct eap_sm *, size_t *);
```

Funkcija vraća ključ i veličinu ključa, koju sprema u drugi argument, dok se prvim argumentom funkciji predaje pokazivač na EAP automat stanja inicijaliziran s `eap_sm_init()`. Prije poziva funkcije za dohvaćanje ključa, moguće je provjeriti da li je ključ uopće dostupan i to pozivom funkcije:

```
int eap_key_available(struct eap_sm *);
```

Funkcija vraća 1 u slučaju dostupnosti ključa, a 0 u slučaju kad ključ nije dostupan.

U funkcije automata stanja proširivog autentifikacijskog protokola ubrajaju se i c*allback* funkcije za kontrolu automata stanja. To su funkcije kojima dohvaćaju ili postavljaju vrijednosti pojedinih varijabli automata stanja. Ako se EAP automat stanja zamisli kao crna kutija, onda se ovim varijablama kontrolira EAP automat stanja. Varijable za kontrolu EAP automata stanja prikazane su u tablici 3.2.1.

Tablica 3.2.1. Varijable za kontrolu EAP automata stanja

Variable	***Description***
EAPOL_eapSuccess	Postavlja ju automat stanja; obavijest o uspješnoj autentifikaciji
EAPOL_eapRestart	Postavlja ju korisnik (inicijator); zahtijev za resetiranjem
EAPOL_eapFail	Postavlja ju automat stanja; obavijest o neuspješnoj autentifikaciji
EAPOL_eapResp	Postavlja ju automat stanja; obavijest o postojećem odgovoru
EAPOL_eapNoResp	Postavlja ju automat stanja; obavijest o nepostojećem odgovoru
EAPOL_eapReq	Postavlja ju korisnik (inicijator); obavijest o postojećem zahtijevu kojeg treba obraditi
EAPOL_portEnabled	Postavlja ju korisnik (inicijator); obavijest o početku autentifikacije
EAPOL_altAccept	
EAPOL_altReject	

Način na koji varijable kontroliraju EAP automat stanja bit će pojašnjen u sljedećem primjeru. Primjer pokazuje prelaske između stanja u EAP automatu stanja, a detaljno je opisan u [18]. Kako bi automat stanja mogao obrađivati zahtijeve, varijabla

`EAPOL_portEnabled` treba biti postavljena na TRUE. Zatim slijedi poziv funkcije `eap_sm_step` te automat stanja prelazi u stanje *INITIALIZE*, a potom u stanje *IDLE*. Po primitku EAP request paketa postavlja se varijabla `EAPOL_eapReq` na TRUE, poziva funkcija `eap_sm_step` s kojom se prelazi u stanje *RECEIVED*. Iz tog stanja postoji nekoliko mogućnosti za daljnje prelaske. U slučaju primljenog ispravnog paketa prelazi se u stanje *METHOD*, gdje paket obrađuje odgovarajući automat stanja ovisno o odabranoj EAP metodi. Prilikom obrade primljenog ispravnog EAP paketa, automat prolazi i kroz sljedeća stanja: *GET_METHOD, IDENTITY, NOTIFICATION, RETRANSMIT*. U slučaju primljenog neispravnog EAP paketa prelazi se u stanje *DISCARD*, koji zanemaruje primljeni paket. U slučaju neispravne autentifikacije automat prelazi u *FAILURE* stanje. Iz primjera se vidi, kako se postavljenjem kontrolnih varijabli utječe na daljnji rad EAP automata stanja.

Callback funkcije *WPA_supplicant-a* kojima se postavlja i dohvaća vrijednost varijabli prikazane su u tablici 3.2.2.

Tablica 3.2.2. Callback funkcije EAP automata stanja

Function	***Description***
get_config	Poziva ju EAP automat stanja; dohvaća konfiguracijsku strukturu
get_bool	Dohvaća vrijednost *boolean* varijable
set_bool	Postavlja vrijednost *boolean* varijable
get_int	Dohvaća vrijednost *integer* varijeble
set_int	Postavlja vrijednost *integer* varijable
get_eapReqData	Poziva ju EAP automat stanja za dohvat zahtijeva
set_config_blob	Nije korišteno u IKEv2 *daemonu*
get_config_blob	Nije korišteno u IKEv2 *daemonu*
notify_pending	Nije korišteno u IKEv2 *daemonu*

3.3 Konfiguracijske datoteke EAP metoda

Pri korištenju EAP protokola kao autentifikacijskog protokola potrebna je i konfiguracijska datoteka za određenu EAP metodu. Svaka EAP metoda zahtijeva određene, sebi svojstvene parametre. U nastavku će biti prikazani parametri dviju konfiguracijskih datoteka – za dvije EAP metode, EAP-MD5 i EAP-TLS.

Svaka konfiguracijska datoteka za EAP metodu započinje riječju `network`. Za EAP-MD5 metodu, mrežni (`network`) dio konfiguracijske datoteke sadržava sljedeće parametre:

- `key_mgmt` (key_management) – navodi se jedan ili više (lista) autentifikacijskih protokola za razmjenu ključeva

- `eap` – naziv EAP metode koja će se koristiti
- `identity` – identitet klijenta
- `password` – zaporka klijenta

Konfiguracijska datoteka za EAP-TLS metodu sadrži više parametara u mrežnom dijelu, jer TLS metoda zahtijeva i certifikate klijenta. Stoga u konfiguracijskoj datoteci treba biti naveden put (*path*) do certifikata. Parametri EAP-TLS konfiguracijske datoteke su sljedeći:

- `key_mgmt` (key_management) – navodi se jedan ili više (lista) autentifikacijskih protokola
- `pairwise` *algoritam*– enkripcijski algoritam
- `group` *algoritam* – enkripcijski algoritam
- `eap` *metoda* – naziv EAP metode koja će se koristiti
- `identity` *ime* – identitet klijenta
- `password` – zaporka klijenta
- `ca_cert` *path* – navodi se put (*path*) do CA certifikata
- `client_cert` *path* navodi se put (*path*) do klijetovog certifikata
- `private_key` *path* – navodi se put *(path)* do datoteke sa klijentovim tajnim ključem
- `private_key_passwd` – navodi se zaporka za datoteku s klijentovim tajnim ključem

4 Internet protokol za razmjenu ključeva

IKEv2 (*Internet Key Exchange protocol version 2*) je protokol za razmjenu ključeva između dviju strana koje žele komunicirati. Te dvije strane se zovu *inicijator* (*initiator*) i *odgovaratelj* (*responder*), a da bi započeli svoju komunikaciju moraju uspostaviti IKE sigurnosnu poveznicu (*IKE security association* – IKE_SA). IKE sigurnosnom poveznicom dvije strane dogovaraju način kojim će zaštiti svoju komunikaciju, što podrazumijeva dogovor oko skupa kriptografskih algoritama (potrebnih za zaštitu prometa) te razmjenu dijeljenog ključa. Promet se razmjenjuje sigurnosnom poveznicom CHILD_SA, koja nastaje odmah nakon uspostave IKE_SA. Između inicijatora i odgovaratelja može postojati više CHILD_SA sigurnosnih poveznica, a sve su one spremljene u bazi koja se zove *Security Association Database (SAD)* i nalazi se u jezgri operacijskog sustava.

4.1 Sigurnosna politika

Svi parametri kojima je određena pojedina sigurnosna poveznica nazivaju se jednom riječju sigurnosna politika – SP (*Security Policy*) i unose se u bazu SPD (*Security Policy Database*). Pod parametrima koji čine sigurnosnu politiku smatraju se: izvorišna i odredišna IP adresa, tip protokola višeg sloja tj. promet koji se želi zaštititi, smjer komunikacije, način rada i dr. Kao i sigurnosne poveznice i SP – ovi dolaze u parovima (za svaki smjer po jedan SP zapis u SPD bazi). Unos zapisa u SPD bazu obavlja se naredbom `spdadd` u datoteku `setkey.cf`, a to izgleda ovako (slika 4.1.1):

```
#!/sbin/setkey -f

spdflush;
spdadd 192.168.177.128 192.168.177.1 icmp
      -P in ipsec ah/transport//require;

spdadd 192.168.177.1 192.168.177.128 icmp
      -P out ipsec ah/transport//require;
```

Slika 4.1.1. Unos zapisa u SPD bazu

Gornji primjer pokazuje unos dva zapisa u SPD bazu. Prvom naredbom `spdflush` brišu se svi do tada zapisani unosi u SPD bazi. Naredbom `spdadd` unose se sigurnosne politike (SP) u bazu. Prvi unos odnosi se na sve dolazne (postavljen je parametar – *in*) IP pakete protokola ICMP, s izvorišnom adresom 192.168.177.128 , a odredišnom 192.168.177.1. Tražena je zaštita prometa *ipsec*-om, sigurnosni protokol je *ah* u *transportnom* načinu rada uz *require* nivo zaštite. Drugi zapis je sličan prvom; razlika je u zamijenjenoj izvorišnoj i odredišnoj adresi te promjenjenom postavljenom smjeru (*out*). Znači, drugi zapis odnosi se na sve

odlazne IP pakete protokola ICMP.

4.2 Stvaranje sigurnosnih poveznica

Uspostava IKE sigurnosne poveznice potrebna je za odvijanje sigurne komunikacije između inicijatora i odgovaratelja. IKE sigurnosna poveznica određuje način na koji se želi zaštiti promet.

Kao i IPsec sigurnosne poveznice i IKE sigurnosnu poveznicu određuju sljedeći parametri:

- izvorišna i odredišna IP adresa – IP adrese krajnjih točaka (inicijatora i odgovaratelja);
- tip IPsec protokola – sigurnosni protokol koji osigurava autentičnost, integritet i pouzdanost (AH ili ESP);
- kriptografski algoritam i tajni ključ kojeg koristi IPsec;
- SPI (*Security Parameters Index*) – broj pridružen svakoj sigurnosnoj poveznici.

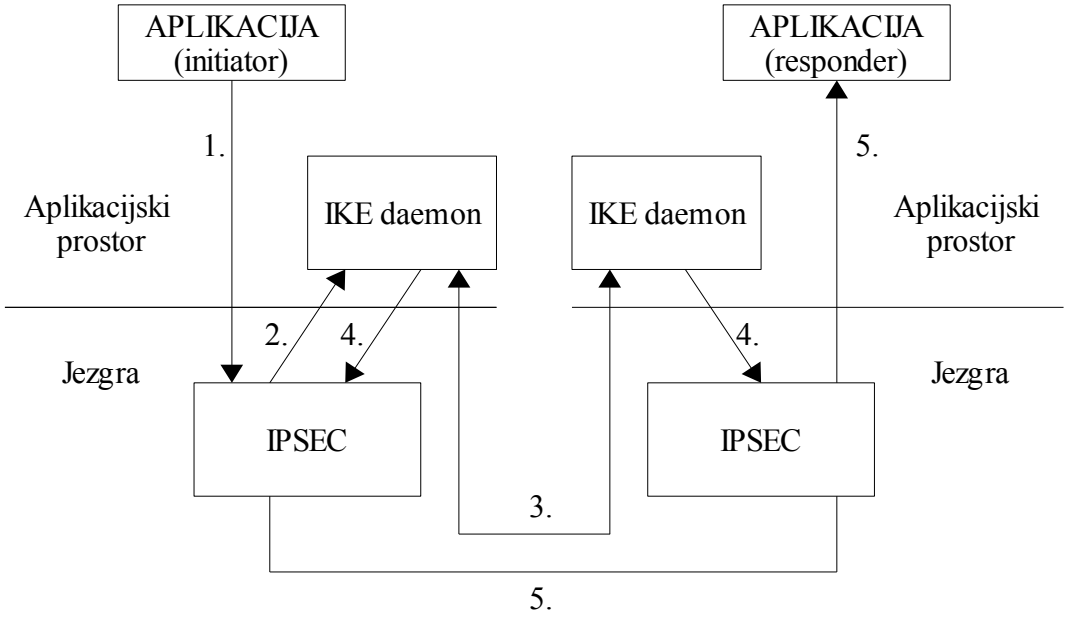
Kako je svaka sigurnosna poveznica određena izvorišnom i odredišnom adresom, a da bi se zaštitio promet u oba pravca (od inicijatora k odgovaratelju i obrnuto) potrebne su dvije sigurnosne poveznice, s obrnutim izvorišnim i odredišnim adresama. Može se reći da sigurnosne poveznice dolaze u parovima. Između inicijatora i odgovaratelja može postojati nekoliko različitih sigurnosnih poveznica, a one se razlikuju po prometu kojeg štite ili po načinu na koji ga štite te po SPI broju. Sve sigurnosne poveznice zapisane su u *Security Association Database* – SAD bazi, koja se nalazi u jezgri operacijskog sustava. Unos zapisa u SAD bazu obavlja se naredbom `add`:

```
add 192.168.177.128 192.168.177.1 esp 15701 -E 3des-cbc
"TajniKljuc";
add 192.168.177.1 192.168.177.128 esp 15702 -E 3des-cbc
"TajniKljuc";
```

Prva sigurnosna poveznica IKE_SA u gornjem primjeru za izvorišnu adresu postavlja 192.168.177.128, a za odredišnu 192.168.177.1, sigurnosni protokol je ESP, SPI broj je 15701, enkripcijski algoritam je 3des-cbc, a tajni ključ je “TajniKljuc”. Postoji i sigurnosna poveznica u suprotnom smjeru: razlika je u zamijenjenoj izvorišnoj i odredišnoj IP adresi te u SPI broju. Gore prikazan način unosa sigurnosnih poveznica u bazu se zove ručni način unosa. Postoji i automatski način, kada taj unos obavlja program.

4.3 IKEv2 *daemon*

Kao što je već rečeno, do sada prikazan unos zapisa u SAD bazu naziva se ručni unos. Takav unos je lakše za implementirati, ali je nesiguran tj. podložan različitim napadima. Radi veće sigurnosti koristi se aplikacija za razmjenu ključeva, *daemon*, koji pregovara razmjenu ključeva između dviju strana koje žele uspostaviti komunikaciju. Dakle, *daemon* je zaslužan za uspostavu sigurnosne poveznice između dviju strana i njen unos u SAD bazu, a njegov rad definiran je prokokolom IKE [18].



Slika 4.3.1. Uspostava IKE_SA pomoću IKE *daemona*

IKEv2 generira glavni ključ na temelju dijeljene tajne dobivene Diffie Hellman razmjenom i nekog izvora slučajnosti. Iz glavnog se ključa generiraju svi potrebni autentifikacijski i enkripcijski ključevi. Da bi se uspostavila sigurnosna poveznica između dviju strana, jedna strana npr. inicijator ne mora odgovaratelju ponuditi točno jedan enkripcijski ili autentifikacijski algoritam. U konfiguracijskog datoteci *daemona* inicijator tada sadrži skup enkripcijskih, autentifikacijskih algoritama te duljine ključeva koji su njemu (inicijatoru) prihvatljivi. Odgovaratelj iz ponuđenih inicijatorovih sigurnosnih parametara odabire one koji i njemu odgovaraju. Nakon dogovaranja s odgovarateljom i uspješno uspostavljenog dogovora oko sigurnosnih parametara *daemon* će načiniti sigurnosnu poveznicu i unijeti je u SAD bazu. Uspostavljanje sigurnosne poveznice tj. sigurne komunikacije između inicijatora i odgovaratelja prikazana je na slici 2.1.1. Inicijator ima u svojoj jezgri zapisano da se sav promet koji se šalje prema odgovaratelju treba zaštititi. Način zaštite je naveden u SP zapisima SPD baze.

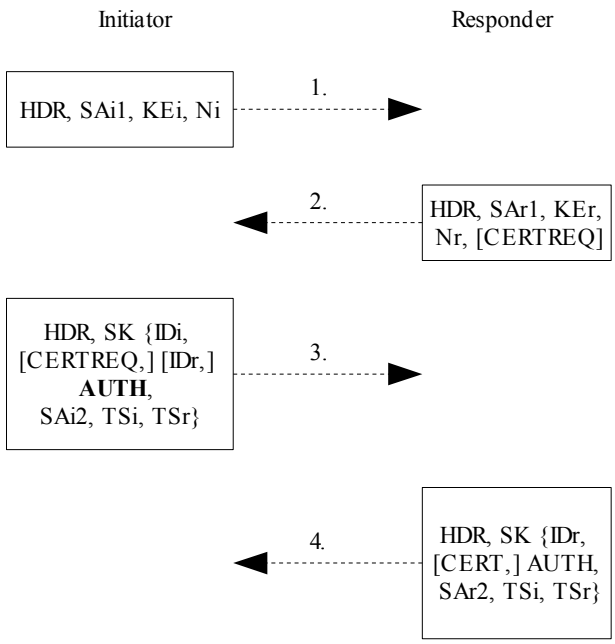
1. Kako u jezgri inicijatora postoji definiran način zaštite prometa (SP), ali ne postoji sigurnosna poveznica (SA), inicijator aktivira svoj IKE *daemon*.

2. IKE *daemon* je zadužen za uspostavu sigurnosne poveznice između inicijatora i odgovaratelja. On će to obaviti zajedno sa IKE *daemonom* odgovaratelja. Ta dva *daemona* dogovorit će ključeve i sigurnosne algoritme potrebne za zaštitu komunikacije.
3. Nakon što su dva *daemona* dogovorila kriptografske algoritme i razmijenili ključeve svaki zapisuje odgovarajući zapis u SAD bazu – inicijatorov *daemon* kod inicijatora, a IKE *daemon* odgovaratelja kod odgovaratelja. Zapisi su određeni izvorišnim i odredišnim IP adresama, SPI brojevima te parametrima potrebnim za zaštitu prometa. Time je uspostavljena sigurnosna poveznica IKE_SA između inicijatora i odgovaratelja.
4. Nakon uspostave sigurnosne poveznice komunikacija može početi: kroz nesigurnu mrežu prenosi se zaštićen promet.

IKE *daemon*, osim uspostave IKE_SA poveznice, obavlja i automatsku promjenu ključeva (*rekeying*). Sve nove sigurnosne poveznice koje nastanu nakon stvorene IKE_SA nazivaju se CHILD_SA poveznice i služe za prijenos prometa. CHILD_SA poveznice su i poveznice koje nastaju prilikom promjene ključeva (*rekeying – a*).

4.4 Razmjena poruka IKE_SA_INIT i IKE_AUTH

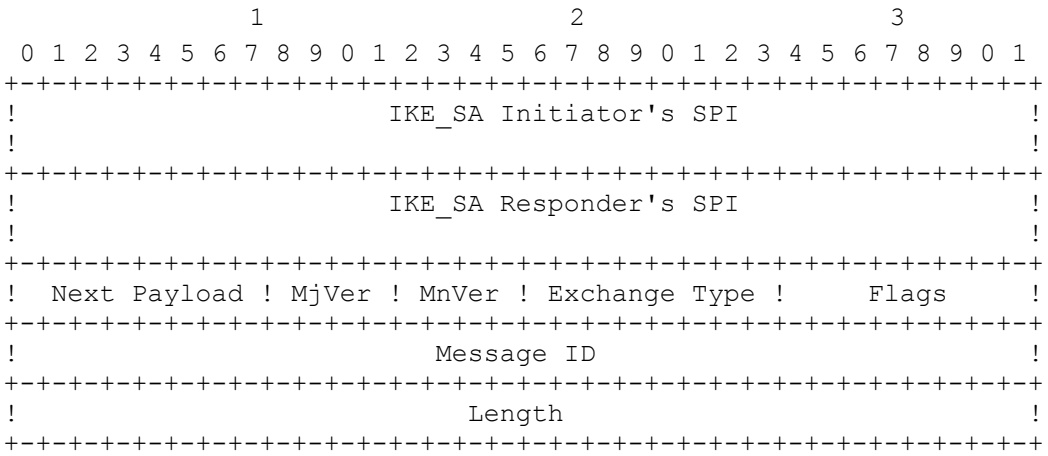
IKE poruke se razmjenjuju korištenjem UDP protokola, a razmjena se obavlja u parovima. Sigurnosna poveznica IKE_SA se uspostavlja dvjema razmjenom poruka IKE_SA_INIT, a razmjenom poruka IKE_AUTH uspostavlja se CHILD_SA sigurnosna poveznica. S prvim parom poruka dvije se strane dogovaraju oko kriptografskih algoritama i ostalih parametara koji su potrebni za zaštitu IKE komunikacije, razmjenjuju slučajne brojeve (*nonce*) i obavljaju Diffie – Hellman razmjenu. To čini prvu fazu uspostave sigurnosnih poveznica. S drugim parom poruka autentificiraju se poruke razmijenjene u prvoj fazi. Njima se dvije strane predstavljaju jedna drugoj slanjem vlastitih ID-eva i certifikata (ukoliko je to potrebno). S te dvije faze razmjene poruka sigurnosna poveznica IKE_SA je uspostavljena i time je razmjena poruka zaštićena. Razmjena IKE_SA_INIT i IKE_AUTH poruka prikazana je na slici 4.4.1.



Slika 4.4.1. Razmjena IKE_SA_INIT i IKE_AUTH poruka bez EAP-a

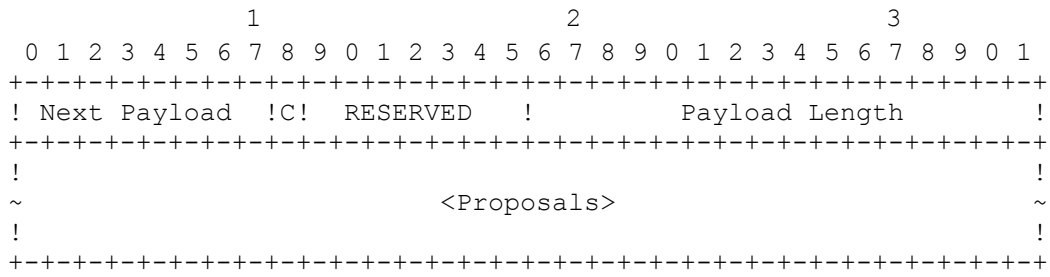
Sve gore prikazane poruke razlikuju se po poljima koja ih tvore. Prva poruka u razmjeni IKE_SA_INIT tvori se od polja: HDR, SAi1, KEi i Ni. Zaglavlje, polje HDR (*header*), sadrži svaka poruka. To polje tvore sljedeći parametri: SPI brojevi inicijatora i odgovaratelja, brojevi verzije te različite zastavice, a izgled zaglavlja je sljedeći, slika 4.4.2.

Sljedeće polje prve poruke je SA (*security association*) prikazano na slici 4.4.3. U tom polju



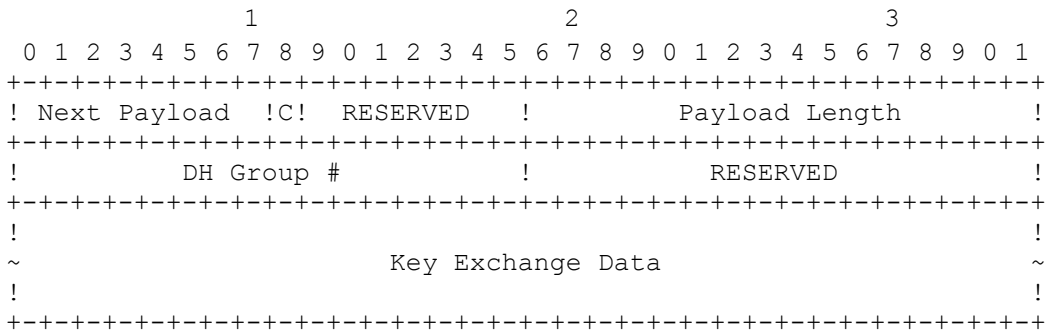
Slika 4.4.2. Sadržaj polja HDR (Header)

nalaze se ponuđeni algoritmi za zaštitu komunikacije te za generiranje slučanih brojeva i provjeru integriteta.



Slika 4.4.3. Sadržaj polja SA (Security Association)

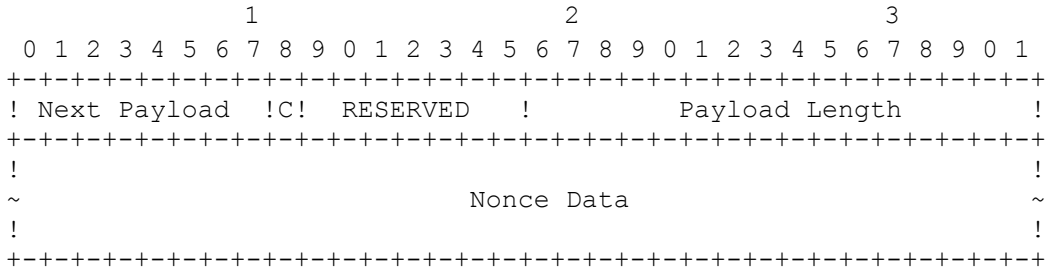
KEi polje sadrži inicijatorovu Diffie – Hellman vrijednost, slika 4.4.4.



Slika 4.4.4. Sadržaj polja KE (Key exchange)

Posljednje polje prve poruke čini inicijatorov *nonce* broj, slika 4.4.5.

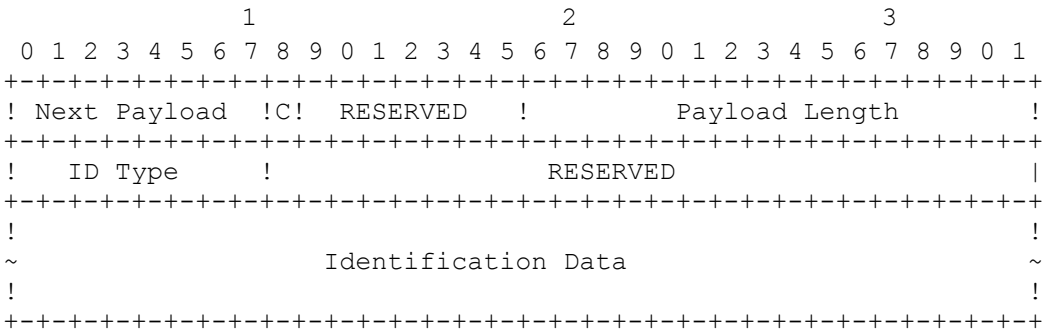
S gore navedena 4 polja stvorena je prva poruka koju inicijator šalje odgovaratelju i s kojom



Slika 4.4.5. Sadržaj N polja (Nonce)

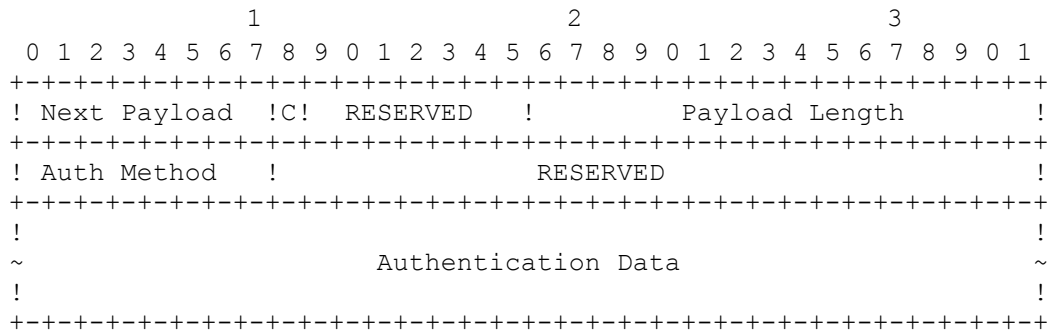
započinje IKE_SA_INIT razmjena. Zatim, u IKEv2 razmijeni poruka, slijedi poruka odgovaratelja. Ona je istog oblika kao i prva poruka koju je slao inicijator, samo u svojim poljima sadrži vrijednosti specifične za odgovaratelja. Odgovaratelj je iz ponuđenih kriptografskih parametara (algoritama koje je inicijator naveo kao moguće) odabrao one koji njemu odgovaraju i pohranio ih u SAr1 polje. Također ima i svoju Diffie – Hellman vrijednost, *nonce* broj itd. Opcionalno, ova poruka može sadržavati koje certifikacijske centre (*Certificate Authorities*) podržava odgovaratelj, što u poruci označava polje CERTREQ.

Nakon obavljene IKE_SA_INIT razmjene slijedi IKE_AUTH razmjena poruka. Sva polja tih poruka, osim zaglavlja (HDR polja) su sada kriptirana te im je zaštićen integritet. To se označava oznakom SK, unutar čijih vitičastih zagrada se navode polja. Tako za treću poruku koja se razmijenjuje u razmjeni poruka, unutar SK dijela poruke dolaze polja: IDi, AUTH, SAi2, TSi, TSr, a opcionalno mogu doći polja: CERT, CERTREQ, IDr. Poljem IDi (Identification) inicijator predstavlja sebe, slika 4.4.6.



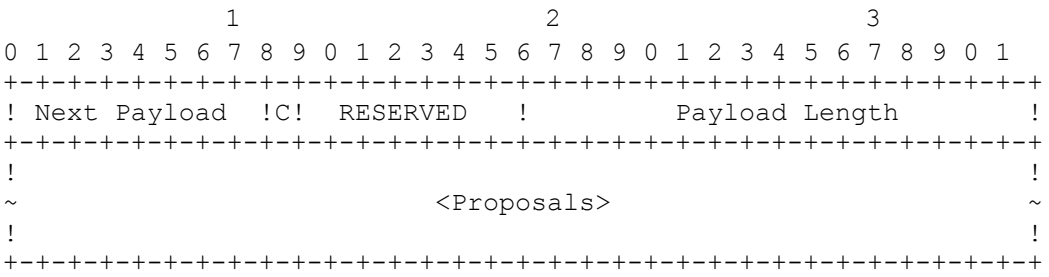
Slika 4.4.6. Sadržaj polja ID (Identification)

Osim s ID poljem, inicijator se može predstaviti i dodatnim parametrima, navedenim u opcionalnim poljima poruke (npr. certifikatom). S poljem IDr, inicijator određuje s kojim točno odgovarateljom želi razgovarati, a to se polje koristi ukoliko odgovaratelj ima više različitih identiteta. Slijedi polje AUTH (Authentication Payload) s određenom autentifikacijskom metodom i podacima potrebnim za autentifikaciju, slika 4.4.7.



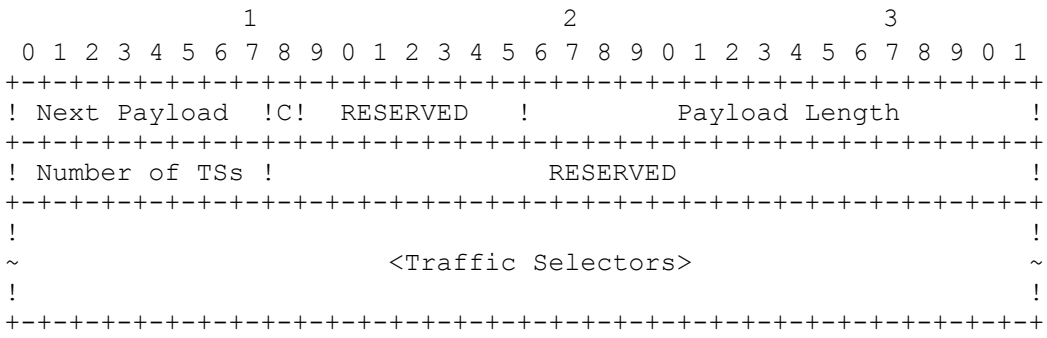
Slika 4.4.7. Sadržaj polja AUTH (Authentication Payload)

Polje SA (Security Association Payload) sadrži parametre (*Proposals*) potrebne za dogovaranja oko atributa sigurne poveznice – SA, slika 4.4.8. Svi parametri navedeni u SA polju ove (treće) poruke u IKEv2 razmjeni su parametri koje podržava inicijator (SAi).



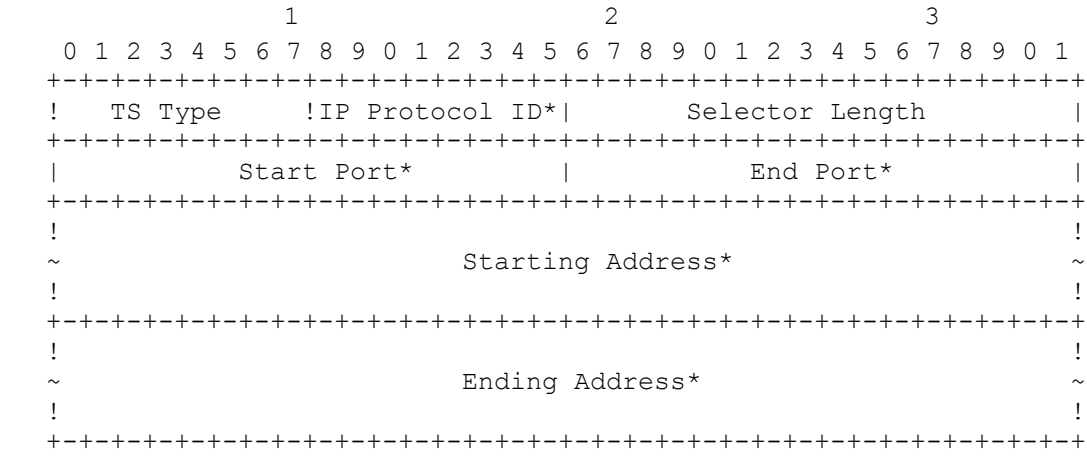
Slika 4.4.8. Sadržaj polja SA (Security Association Payload)

Zadnja dva polja treće poruke su TSi i TSr (*Traffic Selector Payload*), a sadrže prometne selektore koji će biti zaštićeni s IPsec sigurnosnom poveznicom. TS polje tvori IKE generičko zaglavlje te odvojeni TS, slika 4.4.9.



Slika 4.4.9. Polje TS (Traffic Selector Payload)

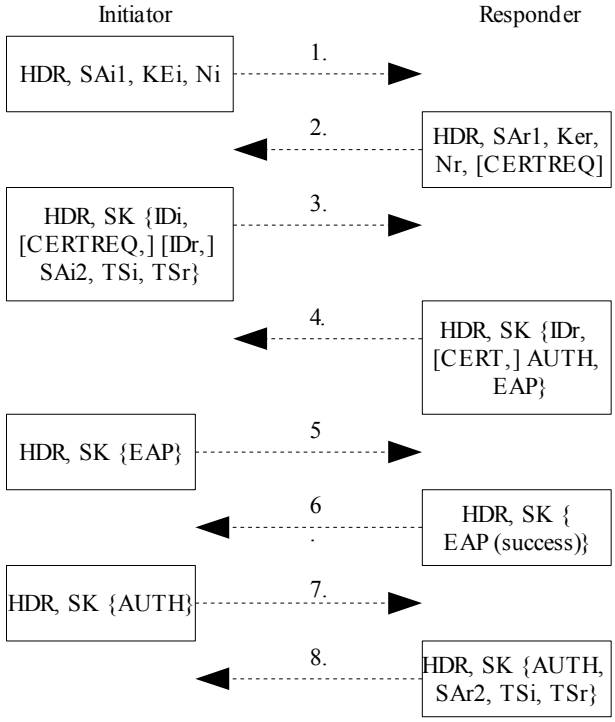
Prometni selektor je složena podatkovna struktura koja se dijeli na dva dijela. Jedan dio određuje odredište, a drugi dio izvorište. U svakom dijelu nalazi se prioritetna lista selektora. Ta lista se tvori upotrebom polja prikazanih na slici 4.4.10. Prometni selektori definiraju IP adrese, pistupe (*ports*), protokole koji će se prenositi CHILD_SA sigurnosnom poveznicom.



Slika 4.4.10. Sadržaj polja TS (Traffic Selector)

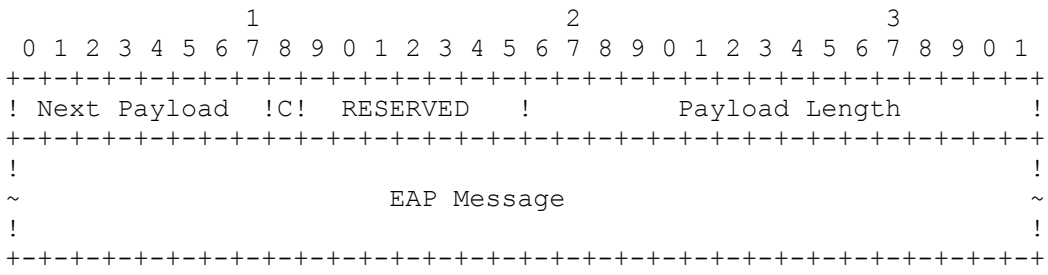
Posljednja poruka u IKEv2 razmjeni, poruka koju šalje odgovaratelj istog je oblika kao i treća poruka, poslana sa strane inicijatora. Polja u poruci su ista: IDr, AUTH, SAr2, TSi, TSr. Razlika je samo u sadržaju polja, koja u ovoj poruci odgovaraju odgovaratelju (odgovarateljov ID, odgovarateljovi odabrani parametri za SA itd.). Inicijator je ponudio vrijednosti u SA i TS poljima svoje poruke, a odgovaratelj odabire što mu od ponuđenog odgovara i u tim istim poljima, ali svoje poruke, to vraća.

Kao što je na slici 4.4.1 prikazano, sigurnosna poveznica se uspostavlja razmjenom četiri poruke. No, to je slučaj samo kada se proširiv autentifikacijski protokol (EAP) ne koristi za autentifikaciju. U slučaju kada se EAP koristi za autentifikaciju, u drugoj fazi (fazi razmjene IKE_AUTH poruka) razmjeni se više parova poruka. Po IKEv2 protokolu, inicijator zahtijeva EAP autentifikaciju izostavljanjem AUTH sadržaja u 3. poruci koja se razmjenjuje (druga koju šalje inicijator na slici 4.4.1). Na tako primljenu poruku odgovaratelj odgovara sa porukom u kojoj izostavlja parametre potrebne za stvaranje sigurnosne poveznice. Sve sljedeće poruke koje se razmjenjuju sadrže samo EAP polje. Razmjena takvih poruka traje dok se autentifikacija ne završi. U slučaju uspješne autentifikacije, sigurnosna poveznica je stvorena i sigurna komunikacija između inicijatora i odgovaratelja može započeti. U suprotnom, slučaju neuspješne autentifikacije, odgovaratelj prekida sve daljnje uspostave IKE_SA i CHILD_SA. Razmjena IKE_SA_INIT i IKE_AUTH poruka kada se proširiv autentifikacijski protokol koristi za autentifikaciju, prikazana je na slici 4.4.11.



Slika 4.4.11. Razmjena IKE_SA_INIT i IKE_AUTH poruka kod korištenja EAP-a

Struktura IKE_SA_INIT poruka ne razlikuje se u ova dva navedena slučaja: s EAP protokolom i bez EAP protokola. Razlike se vide u strukturi IKE_AUTH poruka. U slučaju kada se EAP koristi kao autentifikacijski protokol poruke u IKE_AUTH razmijeni sadrže EAP pakete tj. EAP polje (Extensible Authentication Protocol (EAP) Payload). Struktura EAP paketa prikazana je u poglavlju 2.2, a EAP polje u IKEv2 poruci je prikazano na slici 4.4.12. EAP polje u IKEv2 poruci je uvijek kriptirano i ima zaštićen integritet. Razmjena poruka s EAP poljem traje dok odgovaratelj ne pošalje poruku s EAP success paketom.



Slika 4.4.12. EAP polje u IKEv2 poruci

4.5 Automati stanja u implementaciji protokola IKEv2

Protokol za razmjenu ključeva ima svoj automat stanja. Točnije, postoje automati stanja inicijatora i odgovaratelja, a dijele se na dvije vrste: automat stanja 1. stupnja (First level state machine) i automat stanja 2. stupnja (Second level state machine). Automat stanja prvog stupnja opisuje ponašanje sigurnosnih poveznica (IKE_SA), a njegova stanja mogu se podijeliti u 3 skupine: stanja koje ima samo inicijator, stanja koja ima samo odgovaratelj te zajednička stanja. Stanja kroz koja prolazi samo inicijator su:

IKE_SMI_INIT	- početno stanje u koje ulazi automat stanja inicijatora kada jezgra operacijskog sustava (kernel) zahtijeva uspostavu prve CHILD poveznice (CHILD SA). Inicijator pošalje IKE_SA_INIT zahtijev i prelazi u IKE_SMI_AUTH stanje gdje čeka IKE_SA_INIT odgovor odgovaratelja.
IKE_SMI_INVALIDKE	- stanje u koje automat ulazi u slučaju da odgovaratelj pošalje IKE_SA_INIT odgovor odbijajući predloženu Diffie-Hellman grupu. Tada inicijator šalje novi IKE_SA_INIT zahtijev (s novom Diffie – Hellman grupom) i potom prelazi u IKE_SMI_AUTH stanje.
IKE_SMI_COOKIE	- stanje u koje inicijator automat stanja prelazi ukoliko od odgovaratelja primi kolačić (cookie). Tada se ponavlja slanje IKE_SA_INIT zahtijeva sa uključenim kolačićem i automat prelazi u IKE_SMI_AUTH stanje.
IKE_SMI_AUTH	- stanje u kojem se izvode ključevi za zaštitu IKE_SA, stvara i šalje IKE_AUTH zahtijev. Nakon slanja IKE_AUTH zahtijeva prelazi se u IKE_SMI_INSTALLCSA stanje.
IKE_SMI_INSTALLCSA	- posljednje stanje u koje ulazi automat stanja. IKE_SA je već stvorena i nalazi se u jezgri operacijskog sustava. Potom automat stanja prelazi u IKE_SM_MATURE.

Od gore navedenih stanja dva su stanja kroz koja prolazi odgovaratelj. To su stanja IKE_SMI_INIT i IKE_SMI_AUTH, s tim da su im za automat stanja odgovaratelja promijenjena imena i ona sada glase:

- IKE_SMR_INIT
- IKE_SMR_AUTH

U tim se stanjima odvijaju sve radnje identično kao i kod automata stanja inicijatora.

Stanja zajednička za oba automata stanja su:

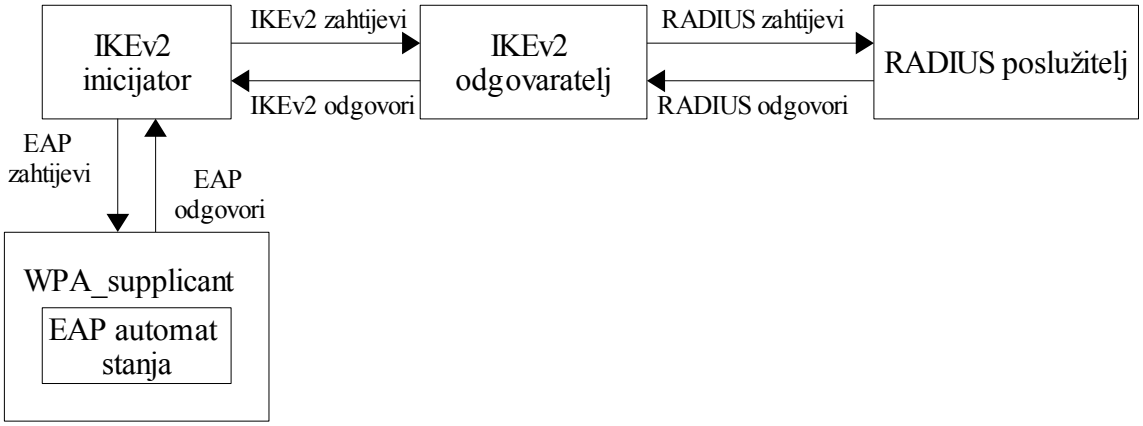
- IKE_SM_MATURE – glavno stanje svake IKE_SA poveznice
- IKE_SM_REAUTH – stanje u koje automat ulazi kada je IKE_SA u procesu reautentifikacije
- IKE_SM_DYING – automat prelazi u ovo stanje u slučaju da se IKE_SA treba izbrisati, zajedno sa svojim CHILD_SA poveznicama. Šalje se zahtijev za brisanjem IKE_SA, koja prelazi u DEAD stanje.

Nakon što je automatom stanja 1. stupnja stvorena IKE_SA poveznica, automatom 2. stupnja se održava IKE_SA. To znači da automat 2. stupnja radi *rekeying* IKE_SA, stvara nove CHILD_SA poveznice te radi *rekeying* postojećih CHILD_SA poveznica. Moguća stanja kroz koja automat prolazi su:

CSA_STATE_INIT	- početno stanje
CSA_STATE_I_REKEY	– stanje u kojem CHILD_SA poveznica šalje zahtijev za *rekeyingom* i prelazi u CSA_STATE_I_REKEY_RESP
CSA_STATE_R_REKEY	– stanje u koje ulazi CHILD_SA poveznica kad je primljen zahtijev za njenim *rekeyingom*. Po završetku *rekeyinga* CHILD_SA poveznica prelazi u CSA_STATE_DEAD.
CSA_STATE_I_REKEY_RESP	– stanje u kojem čeka odgovor na prije poslan zahtijev za *rekeyingom*
CSA_STATE_I_CREATE	– kad jezgra operacijskog sustava zahtijeva novu sigurnosnu poveznicu – SA, novostvorena CHILD_SA se smješta u ovo stanje i šalje se zahtijev partneru (*peer*) za stvaranje nove SA
CSA_STATE_I_CREATE_RESP	– u ovom stanju CHILD_SA stvorena u CSA_STATE_I_CREATE čeka na odgovor partnera o slaganju s parametrima nove sigurnosne poveznice (SA). U slučaju pozitivnog odgovora stvara se nova sigurnosna poveznica.
CSA_STATE_IKESA_REKEY	– stanje u kojem se nalazi csa struktura kada je potreban *rekeying* IKE_SA. Iz ovog se stanja priprema i šalje zahtijev partneru.
CSA_STATE_IKESA_REKEY_RESP	– nakon što je u stanju CSA_STATE_IKESA_REKEY poslan zahtijev za *rekeyingom* IKE_SA, u ovom se stanju čeka odgovor partnera. Kad stigne odgovor partnera stvara se novi *session*, a stari se ukljanja.
CSA_STATE_R_CREATE	– stanje u kojem se ne rasponderu stvara nova CHILD_SA poveznica i šalje se odgovor onoj strani koja je zahtijevala stvaranje nove poveznice
CSA_STATE_R_IKESA_REKEY	– stanje u kojem se na odgovaratelju obavilo rekeying IKE_SA poveznice
CSA_STATE_DEAD	– stanje u kojem je CHILD_SA pripremljena za brisanje, a zahtijev za brisanjem CHILD_SA je poslan partneru (*peer*)
CSA_STATE_REMOVE	– stanje u kojem se CHILD_SA briše iz memorije
CSA_STATE_MATURE	– stanje u kojem jezgra operacijskog sustava koristi CHILD_SA

5 Praktični rad – Ostvarenje EAP protokola u IKEv2 okruženju

U specifikaciji Internet protokola za razmjenu ključeva (IKEv2 protokola), proširiv autentifikacijski protokol (EAP) je predložen kao autentifikacijski protokol. Ostvarenje EAP-a i njegovih metoda od početka bio bi dug i nimalo jednostavan posao. Stoga je iskorišteno već postojeće ostvarenje EAP protokola – programska komponenta *WPA_supplicant*. Kako se radi o komponenti sastavljenoj od većeg broja manjih modula, koji se mogu zasebno koristiti, korištenje ove komponente se ispostavilo učinkovitim. Programska komponenta *WPA_supplicant* se koristi kao biblioteka, a neki njeni dijelovi iskorišteni su na strani inicijatora. Druga strana, odgovaratelj, komunicira s RADIUS poslužiteljem putem RADIUS podsustava za razmjenu poruka. Na strani inicijatora stvoren je modul koji se zove *supplicant*, a uključuje već postojeći kôd IKEv2 *daemona* na strani inicijatora, dijelove preuzete iz *WPA_supplicant - a* te dodatno napisane funkcije potrebne za komunikaciju IKEv2 *daemona* i EAP automata stanja. *Supplicant* modulom omogućeno je povezivanje IKEv2 automata stanja sa *WPA_supplicant* komponentom. Cjelokupan pregled razmjene poruka u komunikaciji inicijatora, odgovaratelja i RADIUS poslužitelja dan je na slici 5.1.



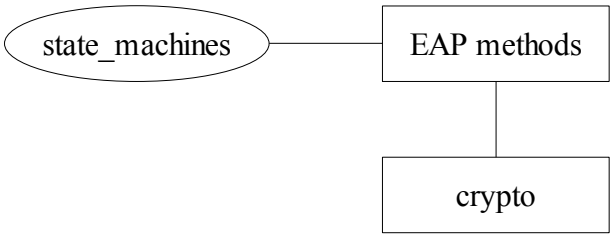
Slika 5.1. Razmjena poruka između tri strane

IKEv2 inicijator razmjenjuje EAP poruke sa *WPA_supplicant - om* tj. s EAP automatom stanja. EAP poruke (paketi) ubacuju se u IKE poruke, koje se onda razmjenjuju između inicijatora i odgovaratelja. Iz tih se poruka izvlači EAP paket i ubacuje u RADIUS poruke, koje se onda šalju RADIUS poslužitelju. U obrnutom toku poruka – iz primljenih RADIUS poruka, na odgovaratelju se izvlači EAP paket, koji se ubacuje u IKEv2 poruku i šalje inicijatoru. inicijator EAP automatu stanja prosljeđuje samo EAP paket, izvađen iz IKEv2 poruke.

Odgovaratelj komunicira s RADIUS poslužiteljem, a za to koristi RADIUS podsustav. RADIUS podsustav je modul čiji je glavni zadatak pružanje podrške prilikom razmjene RADIUS paketa između odgovaratelja i RADIUS poslužitelja. Modul sadrži funkcije za primanje/slanje RADIUS paketa od/k RADIUS poslužitelju, kao i funkcije za slaganje poruka, izvlačenje EAP paketa iz poruka, parsiranje poruka i slično.

5.1 Prilagodba programske komponente *WPA_supplicant*

Iz arhitekture programske komponente *WPA_supplicant*, kakva je opisana u 3. poglavlju preuzeti su neki dijelovi kako bi se na strani inicijatora olakšala implementacija proširivog autentifikacijskog protokola, slika 5.1.1. Ti su dijelovi pretvoreni u dinamičku biblioteku i uključeni u IKEv2 *daemon*.



Slika 5.1.1. Dijelovi preuzeti iz programske komponente *WPA_supplicant*

Prije uključivanja *WPA_supplicant - a* u IKEv2 *daemonu* bilo je potrebno prilagoditi i instalirati najnoviju verziju programske komponente *WPA_supplicant*. Napravljeno je nekoliko *Makefile* datoteka. To su datoteke koje opisuju koje datoteke iz izvornog kôda *WPA_supplicant - a* je potrebno uzeti te način na koji ih je potrebno prevesti kako bi se pretvorile u biblioteku. Također, Makefile datoteke definiraju što i gdje instalirati nakon prevođenja. Popis *Makefile* datoteka nalazi se u Dodatku.

U Dodatku se nalazi i popis datoteka iz *WPA_supplicant-a* koje su uključene u IKEv2 *daemon*. Datoteke su podijeljene u nekoliko grupa ovisno o sličnosti njihove funkcionalnosti:

1. Datoteke EAP automata stanja i EAP metoda - implementiraju EAP automate stanja i EAP metode te definiraju strukture potrebne za konfiguracijske parametre konfiguracijskih datoteka.
2. Datoteke kriptografskih funkcija
3. Konfiguracijske datoteke - datoteke potrebne za čitanje i parsiranje konfiguracijskih datoteka

5.2 Dodatne funkcije za kontrolu automata stanja

Kako bi se ispravno uključio EAP automat stanja u *supplicant* modul na strani inicijatora, bilo je potrebno napisati dodatne funkcije za postavljanje i dohvaćanje vrijednosti kontrolnih varijabli kada EAP automat stanja prelazi iz stanja u stanje. Zadatak tih funkcija je komunikacija između EAP automata stanja i *WPA_supplicant-a* na inicijatoru tj. IKEv2 *daemona*. Supplicant modul ima dva sučelja: prema EAP automatu stanja te prema IKEv2 *daemonu*. Sučelje prema EAP automatu stanja složeno je od funkcija koje služe za komunikaciju između *WPA_supplicant - a* i *supplicant* modula. *WPA_supplicant* ostvaruje EAP automat stanja i EAP metode, a kako bi se razmijenile poruke između supplicanta i EAP automata stanja, supplicant mora postaviti određene varijable, za što su potrebne sljedeće funkcije:

```
void set_bool (void *, enum eapol_bool_var, Boolean)
   Boolean get_bool (void *, enum eapol_bool_var)
  unsigned int get_int(void *, enum eapol_int_var)
void set_int(void *, enum eapol_int_var, unsigned int)
```

Funkcija `set_bool` postavlja TRUE ili FALSE vrijednost EAPOL varijablama EAP automata stanja, dok funkcija `get_bool` obavlja suprotnu radnju, dohvaća vrijednost (TRUE ili FALSE) tih varijabli. Funkcije `get_int` i `set_int` dohvaćaju, odnosno postavljaju, vrijednost varijable `EAPOL_idleWhile`.

Za inicijalizaciju struktura potrebnih EAP automatu stanja, koriste se sljedeće funkcije:

```
struct wpa_ssid *get config (void *)
void set_config_blob (void *, struct wpa_config_blob *)
const struct wpa_config_blob *get_config_blob (void *, const
char *)
```

Funkcija `*get_config` vraća strukturu `wpa_ssid`, dok funkcije `set_config_blob` i `*get_config_blob` postavljaju tj. dohvaćaju konfiguracijski blob.

Drugim sučeljem, onim između supplicanta i IKEv2 *daemona*, prihvaćaju se EAP odgovori od automata stanja i prosljeđuju se EAP zahtijevi poslani od IKEv2 *daemona* k automatu stanja. Funkcijom `*supplicant_new` inicijalizira se supplicantov automat stanja.

```
struct supplicant_data *supplicant_new(const char *);
```

Funkcija kao argument prima pokazivač na datoteku s konfiguracijskim parametrima te nakon čitanja datoteke instancira EAP automat stanja. Kada postoji zahtijev kojeg treba proslijediti EAP automatu stanja, npr. od RADIUS poslužitelja, poziva se funkcija:

```
gint supplicant_process_request(struct supplicant_t *, void
*, gint)
```

Funkcija prima tri argumenta i to: pokazivač na supplicant strukturu instanciranu funkcijom `struct supplicant_data *supplicant_new`, pokazivač na zahtjev te ukupnu duljinu zahtijeva.

Osim prosljeđivanja zahtijeva EAP automatu stanja potrebno je i prihvatiti odgovor automata i proslijediti ga autentifikatoru. Odgovor se dohvaća funkcijom:

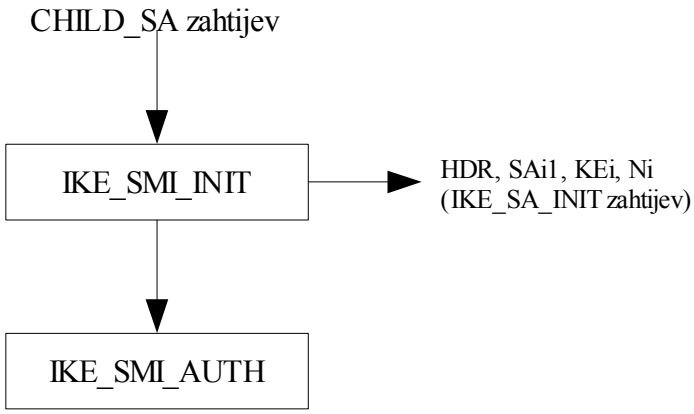
```
gint supplicant_get_response(struct supplicant_t *,void **)
```

Funkcija će vratiti pokazivač na polje koje sadržava odgovor automata stanja.

5.3 Automat stanja IKEv2 inicijatora

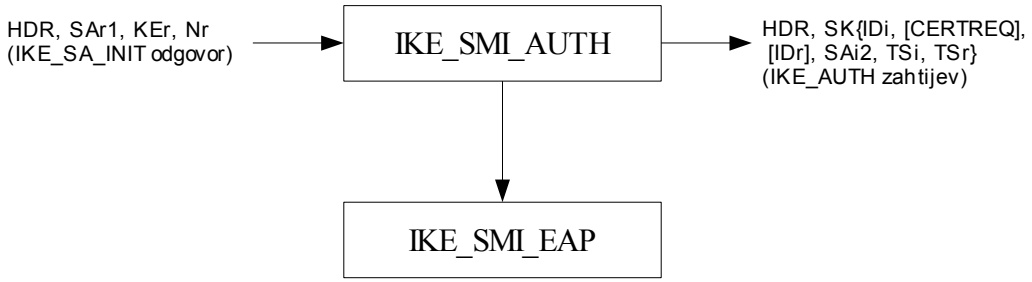
Prilikom ostvarivanja proširivog autentifikacijskog protokola u IKEv2 *daemonu* bilo je potrebno načiniti neke izmjene u IKE automatu stanja na strani inicijatora. Točnije, uvedeno je jedno novo stanje – IKE_SMI_EAP.

Kada inicijator želi uspostaviti novu CHILD_SA poveznicu stvara novi automat stanja i na početku se nalazi u IKE_SMI_INIT stanju. U tom se stanju stvara IKE_SA_INIT zahtijev, šalje odgovaratelju i prelazi u IKE_SMI_AUTH stanje, u kojem se čeka IKE_SA_INIT odgovor odgovaratelja. IKE_SA_INIT zahtjev je prva poruka koja se šalje u IKE_INIT razmjeni. Njen sadržaj objašnjen je u poglavlju 4.2 (slika 5.3.1).



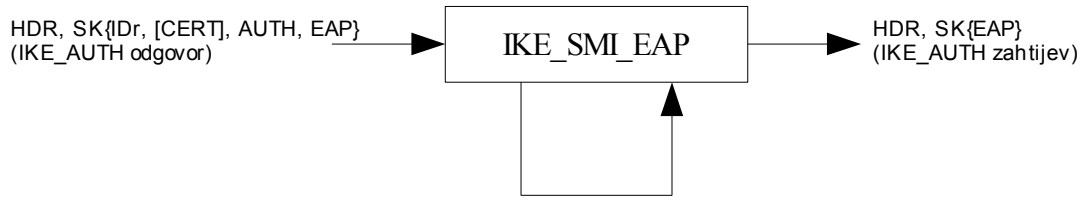
Slika 5.3.1. Stanje IKE_SMI_INIT

Po primitku IKE_SA_INIT odgovora, IKEv2 *daemon* ga obrađuje te kreira IKE_AUTH zahtijev. U IKE_AUTH zahtijevu nema AUTH polja, čime se naznačuje da se autentifikacija želi obaviti nekom EAP metodom. Odmah po završetku slanja IKE_AUTH zahtijeva automat inicijatora prelazi u IKE_SMI_EAP stanje (slika 5.3.2). U tom stanju ostaje dok ne primi odgovor odgovaratelja.



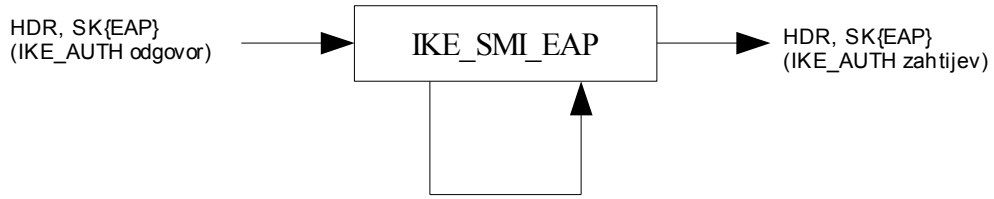
Slika 5.3.2. Stanje IKE_SMI_AUTH

U slučaju da se odgovaratelj slaže sa zatraženom EAP autentifikacijom, odgovor odgovaratelja (IKE_AUTH odgovor) trebao bi sadržavati upit za identitetom inicijatora. Tada se inicijalizira EAP automat stanja i slaže se odgovor koji se prosljeđuje odgovaratelju. inicijator i dalje ostaje u IKE_SMI_EAP stanju (slika 5.3.3).



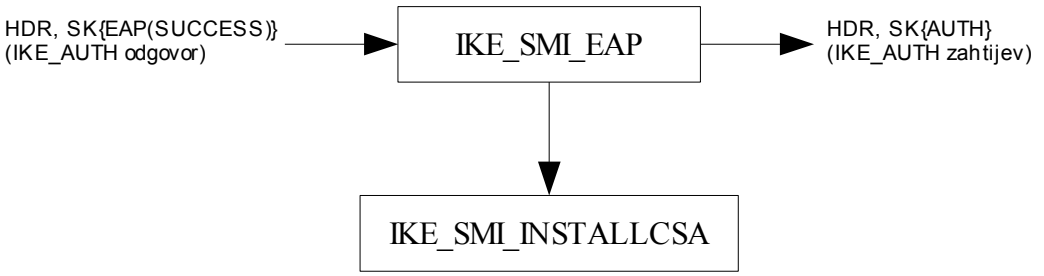
Slika 5.3.3. Stanje IKE_SMI_EAP

Nakon što je odgovor EAP automata stanje proslijeđen odgovaratelju čeka se odgovarateljov odgovor. Ova razmjena poruka između EAP automata stanja i odgovaratelja obavlja se sve dok odgovaratelj ne pošalje SUCCESS ili FAILURE poruku.



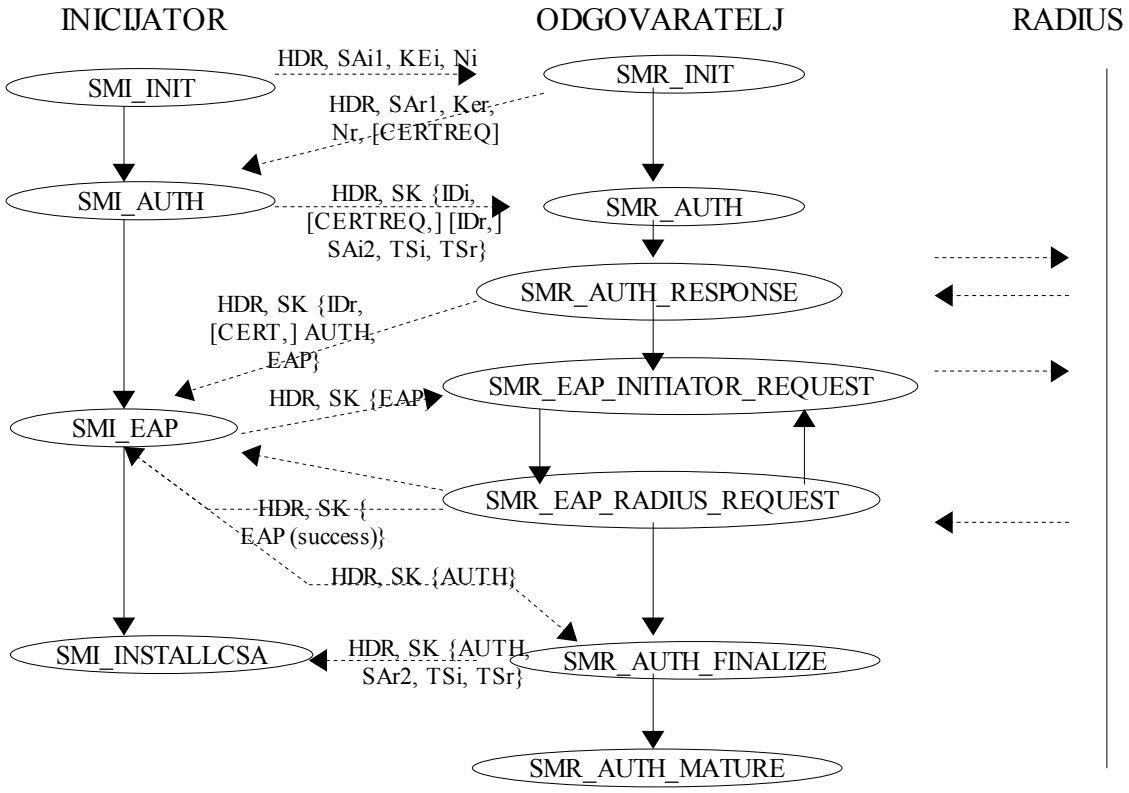
Slika 5.3.4. Stanje IKE_SMI_EAP - razmjena poruka s odgovarateljom

U slučaju kada odgovaratelj pošalje SUCCESS poruku automat stanja IKEv2 inicijatora šalje završni IKE_AUTH zahtijev te prelazi u IKE_SMI_INSTALLCSA stanje (slika 5.3.5). U IKE_SMI_INSTALLCSA stanju odgovaratelj čeka odgovarateljov IKE_AUTH odgovor. U slučaju da u IKE_SMI_EAP stanju primi FAILURE poruku poslanu od odgovaratelja, automat prelazi u IKE_SMI_DEAD stanje i sjednička struktura (*session* struktura) za inicijatora se makne iz memorije.



Slika 5.3.5. Stanje IKE_SMI_INSTALLCSA

Ovim je pregledom ukratko objašnjen rad inicijatorovog automata stanja. Sve poruke koje se razmijenjuju između inicijatora i odgovaratelja prilikom prelaska iz stanja u stanje, objašnjene su u poglavlju 4.2. Radi se o razmijeni IKE_SA_INIT i IKE_AUTH poruka kada se kao autentifikacijski protokol koristi EAP. Razmjena svih poruka između inicijatora, odgovaratelja i RADIUS poslužitelja te prelasci između stanja kod automata stanja inicijatora i odgovaratelja prikazani su na slici 5.3.6.



Slika 5.3.6. Razmjena EAP poruka te automati stanja inicijatora i odgovaratelja

5.4 Pokretanje IKEv2 *daemona*

5.4.1 Ispitna okolina

Okolinu u kojoj se ispituje ikev2 čine tri strane: računalo na kojem se nalazi inicijator, računalo na kojem se nalazi odgovaratelj i RADIUS poslužitelj.

Računala koja predstavljaju inicijatora i odgovaratelja su dva virtualna stroja (VMWare stroja). Na stroju na kojem se nalazi odgovaratelj mora se nalaziti i RADIUS poslužitelj, jer odgovaratelj komunicira sa RADIUS poslužiteljem. Prilikom ostvarivanja proširivog autentifikacijskog protokola u IKEv2 *daemonu* korišten je FreeRADIUSposlužitelj.

Svako računalo ima Linux operacijski sustav i postavljen IKEv2 *daemon*. Računalo koje predstavlja inicijatora sadrži i potrebne dijelove programske komponente *WPA_supplicant* (kao što je i navedeno u poglavlju 5.2).

Kako bi se IKEv2 *daemon* pokrenuo i ispravno radio, potrebne su konfiguracijske datoteke. Koriste se dva tipa konfiguracijskih datoteka: jednu skupinu skupinu konfiguracijskih datoteka čine one napisane za IKEv2 *daemon*, a drugu konfiguracijske datoteke za rad proširivog autentifikacijskog protokola.

5.4.2 Konfiguracijska datoteka IKEv2 *daemona*

Konfiguracijska datoteka potrebna za pokretanje IKEv2 *daemona* mora se nalaziti na oba računala na kojima se pokreće IKEv2 *daemon* tj. i na inicijatoru i na odgovaratelju. Sadržaj IKEv2 konfiguracijske datoteke podijeljen je u nekoliko dijelova, a oni su sljedeći: *general*, *logging*, *radius_server*, *remote* i *peer*. Svaki dio konfiguracijske datoteke sadrži vlastite parametre i pripadajuće vrijednosti.

- **general**
  - `rand_device` -

    Generator slučajnih brojeva
  - `dos_treshold` *broj* - broj djelomično uspostavljenih poveznica nakon čega se počnu slati kolačići (*cookies*)
  - `sm_threads` *broj* - broj dretvi za procesiranje sjednica (sessiona)
  - `mode` *responder/initiator/unspec*- način na koji se *daemon* ponaša
  - `sanity_check` *off/warning/error* - opcija koja treba biti uključena ako se želi provjeriti konfiguracijska datoteka nakon parsiranja
  - `ikesa_max` *broj* - broj koji određuje koliko će cjelokupno uspostavljenih

IKE_SA poveznica iste IP adrese IKE *daemon* dozvoliti prije nego počne ignorirati daljnje zahtijeve

- `ikesa_max_halfopened` *broj* - broj koji određuje koliko će djelomično uspostavljenih IKE_SA poveznica iste IP adrese IKE *daemon* dozvoliti prije nego prestane primati nove IKE_SA_INIT zahtijeve
- `psk_file` *ime datoteke* – ime datoteke u kojoj se nalaze PSK ključevi za različite identifikatore
- `kernel_spd` *flush/rosync/generate* – određuje što je potrebno učiniti s sigurnosnom politikom kernela u trenutku pokretanja

- **logging** – konfiguracijska opcija kojom se određuje koji će podsustavi biti logirani o određenu datoteku (logfile datoteku)
- **radius_server** – parametri RADIUS poslužitelja
  - `nas_identifier` *ime* – ime radius poslužitelja
  - `timeout` *broj* – broj koji određuje koliko traje *timeout*
  - `retries` *broj* – broj ponovnih pokušaja
  - `auth_server {address, shared_secret}` – opcija koja sadrži adresu RADIUS poslužitelja te dijeljenu tajnu
- **remote IP adresa** – remote sekcija koja određuje parametre za IKE prvu fazu; ukoliko je umjesto IP adrese navedeno *any* parametri odgovaraju svim IP adresama.
  - `nonce_size` *broj* – određuje vrijednost "nonce" u byte-ovima
  - `natt` *on/off/passive* – određuje da li je podržan NAT (*passive*) ili ga *daemon* inicira (*on*)
  - `response_timeout` *broj* – *broj* određuje vrijeme čekanja prije slanja odgovora
  - `response_retries` *broj* – *broj* određuje broj pokušaja slanja odgovora
  - `remote_id` *ime* – određuje identifikator koji se šalje drugoj strani
  - `proposal` – posebni dio remote sekcije, a sadržava zahtijeve koji se traže/predlažu drugoj strani
    - `encryption_algorithm` *naziv algoritma* - naziv enkripcijskog algoritma, u prvoj fazi pregovora
    - `auth_algorithm` *naziv algoritma* – naziv autentifikacijskog algoritma

    - `pseudo_random_function` *algoritam* – naziv algoritma sažetka
    - `dh_group` *vrijednost* – određuje grupu Diffie-Hellmann eksponencija
- **peer any**
    - `authlimit time/allocs/octets`
    - `rekeylimit time/allocs/octets`
    - `ike_max_idle` *broj* -
    - `auth_method` – autentifikacijska metoda (npr. pre shared key)
    - `peer_auth_method` – autentifikacijska metoda klijenta (npr. eap)
    - `my_identifier` *ime* – identifikacija klijenta
    - `radius_server` *ime* - naziv autentifikacijskog poslužitelja kod kojeg se autentificira klijent
    - `sainfo` – određuje parametre druge faze IKEv2 protokola (IKE_AUTH faza)
        - `hardlimit time/allocs/octets` - "tvrde" granice druge faze IKEv2 protokola
        - `softlimit time/allocs/octets` - "meke" granice druge faze IKEv2 protokola
        - `encryption_algorithm` – enkripcijski algoritam druge faze IKEv2 protokola
        - `auth_algorithm` – autentifikacijski algoritam druge faze IKEv2 protokola

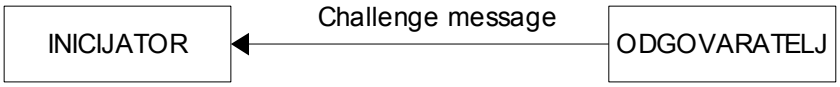
Uz konfiguracijske datoteke IKEv2 *daemona*, na oba računala na kojima se pokreće IKEv2 *daemon* treba se nalaziti i psk.txt datoteka. To je datoteka koja sadrži PSK (*Pre Shared Key*) ključeve za različite identifikatore. Jedan zapis u psk.txt datoteci izgleda:

```
mail:client@zemris.fer.hr      somenotveryverysecretword
```

5.5 Ispitivanje EAP-MD5 i EAP-TLS metoda u IKEv2 *daemonu*

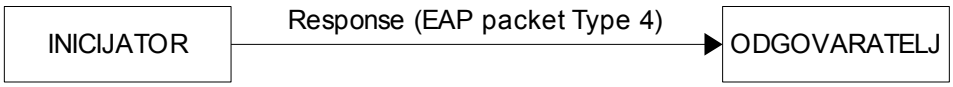
Nakon što je proširiv autentifikacijski protokol ostvaren u IKEv2 *daemonu*, njegov je rad isproban korištenjem dvije EAP metode: MD5 i TLS.

EAP-MD5 metoda autentificira klijenta primjenom metode trostrukog rukovanja (*3-way handshake*). Taj je oblik autentifikacije temeljen na CHAP protokolu (Challenge – Handshake Authentication Protocol) [15]. Razmjena poruka započinjem slanjem odgovarateljove *Challenge* poruke inicijatoru, slika 5.1.1.



Slika 5.5.1. EAP-MD5: Prva poruka

Slijedi odgovor inicijatora s EAP paketom koji ima Type 4 (označava MD5 metodu). Također, poruka sadrži vrijednost izračunatu MD5 funkcijom sažimanja, slika 5.5.2.



Slika 5.5.2. EAP-MD5: Druga poruka

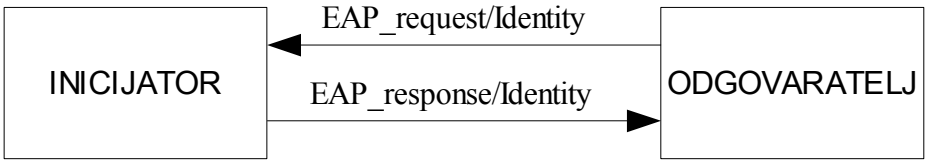
Po primitku inicijatorovog odgovora odgovaratelj provjeri vrijednost sadržanu u poruci, računajući vlastiti sažetak. Ukoliko se primljeni i izračunati sažetci podudaraju, klijent je autentificiran. Odgovaratelj šalje SUCCESS poruku i daljna komunikacija može započeti. U suprotnom, slučaju neuspješne autentifikacije, odgovaratelj šalje FAILURE poruku te veza između inicijatora i odgovaratelja se prekida, slika 5.5.3.

Slika 5.5.3. EAP-MD5: Treća poruka

Sve poruke koje se razmjenjuju prilikom EAP-MD5 autentifikacije imaju format uobičajenog EAP paketa, samo je polje Type postavljeno na vrijednost 4.

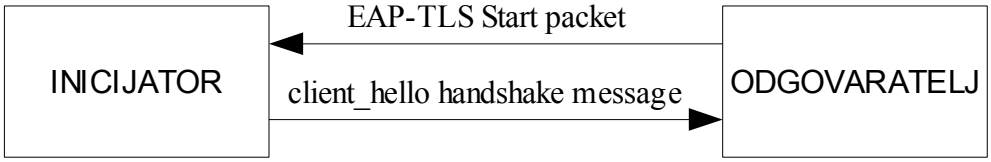
Za razliku od metode EAP-MD5, metoda EAP-TLS prilikom procesa autentifikacije razmijeni više poruka. Razlog tomu je što je autentifikacija složenija.

Autentifikacija započinje porukom odgovaratelja inicijatoru sa zahtjevom za njegovim identitetom. inicijator na to odgovara odgovarajućom porukom, slika 5.5.4.



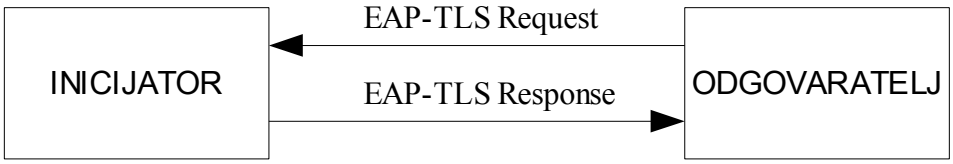
Slika 5.5.4. EAP-TLS: Početak autentifikacije

Odgovaratelj potom šalje EAP-TLS Start poruku u kojoj je polje Type postavljeno na TLS slika 5.5.5. Inicijator odgovara porukom u kojoj se nalazi polje *TLS client_hello handshake message*. To polje sadrži broj TLS verzije, *sessionId*, slučajni broj te skup kriptografskih parametara koje podržava inicijator.



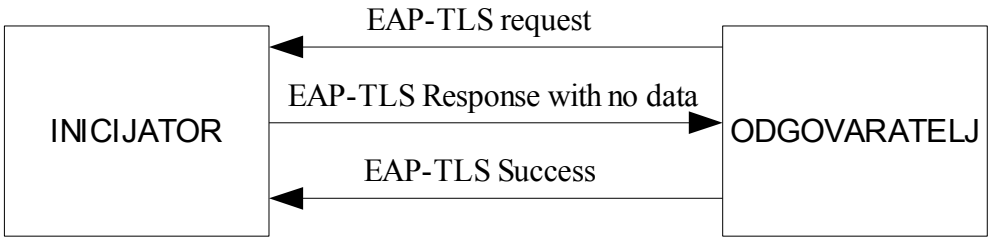
Slika 5.5.5. EAP-TLS: Autentifikacija

Odgovaratelj odgovara s TLS server_hello handshake message slika 5.5.6. Ta poruka sadržava: broj TLS verzije, slučajni broj, *sessionID* te kriptografske parametre. Nakon ove poruke mogu slijediti: TLS certifikat, *server_key_exchange*, zahtjev za certifikatom, *server_hello_done* poruka i/ili TLS *change_chiper_spec* poruka i/ili završna poruka rukovanja (*finished handshake message*). Inicijator odgovara porukom koja sadržava odgovore na sve zahtjeve primljene u odgovarateljovoj poruci.



Slika 5.5.6. EAP-TLS: Autentifikacija

U slučaju uspješne autentifikacije odgovaratelj šalje *TLS change chiper spec* poruku koja sadržava zahtjev za promjenom kriptografskih parametara te završnu poruku rukovanja. Završna poruka rukovanja sadržava poruku inicijatoru o uspješnoj autentifikaciji. inicijator tada provjerava sažetak poruke, kako bi mogao autentificirati poslužitelja. Kod uspješne autentifikacije inicijator EAP-TLS odgovor bez *Data* polja u poruci, a odgovaratelj odgovara sa *Success* porukom, slika 5.5.7.



Slika 5.5.7. EAP-TLS: Završetak autentifikacije

U slučaju neuspješne autentifikacije odgovaratelj šalje EAP-TLS zahtjev sa odgovarajućom porukom o neuspješnoj autentifikaciji. inicijator odgovara sa *client_hello_handshake* porukom, u kojoj zahtjeva ponovnu autentifikaciju ili ne šalje nikakvu poruku. U slučaju da inicijator ne zahtjeva ponovnu autentifikaciju, odgovaratelj šalje Failure poruku i komunikacija se završava.

Za ispitivanje ovih dviju metoda složene su i konfiguracijske datoteke, za svaku metodu posebno. EAP konfiguracijska datoteka pri pokretanju IKEv2 *daemona* treba se nalaziti na računalu inicijatora. Objašnjena parametara datoteka dana su u poglavlju 3.3, a u nastavku su navedene konfiguracijske datoteke s postavljenim vrijednostima parametara.

Konfiguracijska datoteka za metodu EAP-MD5 prikazana je na slici 5.5.8.

```
network={

key_mgmt=IEEE8021X
eap=MD5
identity="client@zemris.fer.hr"
password="clientverysecret"
eapol_flags=0

}
```

Slika 5.5.8. Konfiguracijska datoteka za metodu EAP-MD5

Konfiguracijska datoteka za metodu EAP-TLS prikazana je na slici 5.5.9.

```
network={

key_mgmt=WPA-EAP
pairwise=CCMP TKIP
group=CCMP TKIP
eap=TLS
identity="client@zemris.fer.hr"
ca_cert="/root/ikev2_i/eap-ca.crt"
client_cert="/root/ikev2_i/eap-client.crt"
private_key="/root/ikev2_i/eap-client.key"
private_key_passwd="clientsecret"

}
```

Slika 5.5.9. Konfiguracijska datoteka za metodu EAP-TLS

5.6 Rezultati ispitivanja

U prethodnim poglavljima objašnjeno je što je sve potrebno za pokretanje IKE *daemona* te način na koji se to radi. Nakon što se *daemon* pokrene na oba računala, inicijatoru i odgovaratelju, pokrene se program `ping` s jednog na drugo računalo. Rezultati tog pokretanja zapisuju se u datoteku *logfile*. Logfile datoteku imaju i računalo inicijatora i računalo odgovaratelja, ali s različinitim sadržajem. U tu se datoteku zapisuju rezultati *logiranja*, a oni uključuju sve promjene koje se na ta dva računala odvijaju: promjene stanja automata stanja, ulazak i izlazak iz funkcija, pročitani zapisi iz konfiguracijskih datoteka, kriptirane poruke koje se razmjenjuju između inicijatora i odgovaratelja itd.

Isječak ispisa iz `logfile` datoteke odgovaratelja (slika 5.6.1):

```
1187765419.521 sm DEBUG - sm_ike_thread:2920: Processing
IKEV2_EXT_IKE_SA_INIT or IKEV2_EXT_IKE_AUTH
1187765419.521 sm DEBUG - Entering sm_ike_r_thread:5010
1187765419.521 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765419.521 crypto DEBUG - Entering rand_bytes:  75
1187765419.521 radius DEBUG - Entering
radius_send_packet:1626
1187765419.521 crypto DEBUG - Entering rand_bytes:  75
1187765419.521 radius DEBUG - Entering radius_get_attribute:
814
1187765419.521 radius DEBUG - Leaving radius_get_attribute:
834
1187765419.521 radius DEBUG - Entering
radius_create_access_request_packet: 366
1187765419.521 radius DEBUG - Entering
radius_eap_dump_packet: 278
1187765419.521 radius DEBUG - radius_eap_dump_packet: 289:
EAP packet:>>>
1187765419.521 radius DEBUG - radius_eap_dump_packet: 290:
Code=2 (Response)
1187765419.521 radius DEBUG - radius_eap_dump_packet: 291:
 Identifier=2
1187765419.521 radius DEBUG - radius_eap_dump_packet: 292:
 Length=106
1187765419.521 radius DEBUG - radius_eap_dump_packet: 298:
1187765419.521 radius DEBUG - radius_eap_dump_packet: 305:

Type=13 (TLS)
1187765419.521 radius DEBUG - radius_eap_dump_packet: 321:

Type-
Data=0x00160301005f0100005b030146cbdc74be912fbe3c2593736d6ed
ec4b30fdb8e251103ebdca59eff26eaa0b6000034003900380035001600 1
3000a00330032002f006600050004006300620061001500120090065006
400600014001100080006000301 00
1187765419.521 radius DEBUG - radius_eap_dump_packet: 329:
EAP packet:<<<
1187765419.521 radius DEBUG - Leaving
radius_eap_dump_packet: 331
1187765419.521 radius DEBUG - Entering radius_get_attribute:
814
1187765419.521 radius DEBUG - Leaving radius_get_attribute:
834
1187765419.521 crypto DEBUG - Entering rand_bytes:  75
1187765419.522 radius DEBUG - Leaving
radius_create_access_request_packet:
646
1187765419.522 network DEBUG - Entering network_send_packet:
500
1187765419.522 network DEBUG - network_send_packet: 504:
data=0x87bda00,
len=210, daddr=127.0.0.1:1812
```

Slika 5.6.1. Isječak zapisa iz datoteke *logfile*

Zapisi u logfile datoteci imaju određeni format, koji će biti prikazan na sljedećem primjeru iz gornjeg izvatka logfile datoteke:

```
1187765419.521 radius DEBUG – Entering  radius_get_attribute:
814
```

Prvi zapis – broj 1187765419.521 – govori vrijeme u sekundama, brojano od 1.1.1970., a u kojem je obavljeno navedeno logiranje. Sljedeći parametar – radius - je ime modula koji je generirao zapis (radius.c), a parametar DEBUG govori da se radi o zapisu DEBUG nivoa. Osim DEBUG nivoa, postoje i niz drugih, primjerice ERROR, NOTICE, TRACE nivoi logiranja. Posljedni paramater u ovom primjeru je ime funkcije koja se logira i linija u kojoj se nalazi u datoteci s izvornim kôdom. Posljedni parametar nije kod svakog logiranja isti. Ukoliko se radi o logiranju pročitanog zapisa iz konfiguracijske datoteke ili kriptirane poruke koja se razmjenjuje izmedju inicijatora i odgovaratelja, zapis u datoteci je drugačiji. Tada zapis sadrži i ispis pročitanih podataka, kao što je slučaj na kraju gornjeg primjera, kada je u `logfile` datoteci zapisana i kriptirana poruka.

Gornji isječak ispisa iz logfile datoteke uzet je s računala odgovaratelja. Ovaj isječak sadrži nekoliko podataka: stanje u kojem se nalazi respoderov automat stanja je STATE IKE_SMR_EAP_INITIATOR_REQUEST; EAP paket sadrži zahtjev za EAP-TLS metodom autentifikacije. Ukoliko se na računalu inicijatora pokrene naredba:

```
[root@LR ~]# grep STATE logfile
```

dobiva se ispis (slika 5.6.2):

```
1187765364.234 sm DEBUG - sm_ike_i_thread:3370: STATE
IKE_SMI_INIT
1187765364.288 sm DEBUG - sm_ike_i_thread:3806: STATE
IKE_SMI_AUTH
1187765364.360 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765364.406 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765364.556 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765364.640 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765364.699 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765364.744 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765365.266 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765365.344 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765365.444 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765366.027 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765366.039 sm DEBUG - sm_ike_i_thread:4287: STATE
IKE_SMI_EAP
1187765366.072 sm DEBUG - sm_ike_i_thread:4368: STATE
IKE_SMI_INSTALLCSA
```

Slika 5.6.2. Ispis stanja kroz koja prolazi inicijator

Ispis iz logfile inicijatora pokazuje kako je inicijatorov automat stanja prošao kroz sva stanja, pretpostavljena za taj automat, a koja su opisana u poglavlju 5.4.

Pokrene li se ista naredba na računalu odgovaratelja, dobiva se ispis (slika 5.6.3):

```
[root@misko source]# grep STATE logfile
1187765419.297 sm DEBUG - sm_ike_r_thread:5034:
STATE IKE_SMR_INIT
1187765419.377 sm DEBUG - sm_ike_r_thread:5588:
STATE IKE_SMR_AUTH
1187765419.382 sm DEBUG - sm_ike_r_thread:6485:
STATE IKE_SMR_AUTH_RESPONSE
1187765419.409 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765419.456 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765419.521 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765419.531 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765419.610 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765419.618 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765419.696 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765419.704 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765419.781 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765419.786 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765420.317 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765420.325 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765420.398 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765420.407 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765420.482 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765420.492 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765420.542 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765421.047 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765421.081 sm DEBUG - sm_ike_r_thread:5447:
STATE IKE_SMR_EAP_INITIATOR_REQUEST
1187765421.086 sm DEBUG - sm_ike_r_thread:5484:
STATE IKE_SMR_EAP_RADIUS_REQUEST
1187765421.103 sm DEBUG - sm_ike_r_thread:6217:
STATE IKE_SMR_AUTH_FINALIZE
```

Slika 5.6.3. Ispis stanja kroz koja prolazi odgovaratelj

Ispis pokazuje sva stanja koja su zapisana u datoteci `logfile` na računalu odgovaratelja. Vidi se kako je automat stanja odgovaratelja prošao kroz sva stanja, kao što je i prikazano u poglavlju 5.4, na slici 4.4.8.

Osim zapisa u logfile-u ispravan rad IKE *daemona* provjerava se i provjerom zapisa u SAD i SPD bazi na oba računala. Ispis SAD baze na računalu inicijatora dobiva se naredbom `setkey -D`, a izgleda ovako:

```
[root@LR ~]# setkey -D
192.168.177.1 192.168.177.128
        esp mode=transport spi=3403987481(0xcae4ba19) reqid=0(0x00000000)
        E: 3des-cbc  342568a4 df19385d e30d072a b06880bc b53d6d2f adcd98b0
        A: hmac-md5  2a233fdb 93c045f5 90302e71 42bc5a85
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Aug 22 19:52:17 2007   current: Aug 22 19:52:39 2007
        diff: 22(s)     hard: 12000(s)  soft: 6000(s)
        last: Aug 22 19:52:17 2007      hard: 0(s)      soft: 0(s)
        current: 320(bytes)     hard: 12582912(bytes)   soft:
1048576(bytes)
        allocated: 5    hard: 0 soft: 0
        sadb_seq=1 pid=2541 refcnt=0

192.168.177.128 192.168.177.1
        esp mode=transport spi=300258285(0x11e593ed) reqid=0(0x00000000)
        E: 3des-cbc  104a45a2 190b86c1 13324a02 58b60e02 df5e2638 1c73f279
        A: hmac-md5  b51721e7 738db5f1 6f7efedb 670c0505
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Aug 22 19:52:16 2007   current: Aug 22 19:52:39 2007
        diff: 23(s)     hard: 12000(s)  soft: 6000(s)
        last: Aug 22 19:52:16 2007      hard: 0(s)      soft: 0(s)
        current: 720(bytes)     hard: 12582912(bytes)   soft:
1048576(bytes)
        allocated: 6    hard: 0 soft: 0
        sadb_seq=0 pid=2541 refcnt=0
```

Slika 5.6.4. SAD baza inicijatora

Gornji ispis pokazuje kako je na računalu inicijatora stvorena IKE sigurnosna poveznica. Točnije, u SAD bazi inicijatora nalaze se dvije sigurnosne poveznice s obrnutim IP adresama i različitim SPI brojem. SAD baza sadrži broje podatke, a između ostalog i: način rada (`esp mode=transport`), sigurnosni protokol (`esp`), enkripcijski algoritam (`3des-cbc`), autentifikacijski algoritam (`hmac-md5`).

Ispis SPD baze na računalu inicijatora dobiva se naredbom `setkey -DP`, a izgleda ovako (slika 5.6.5):

```
[root@LR ~]# setkey -DP
192.168.177.1[any] 192.168.177.128[any] icmp
        in prio def ipsec
        esp/transport//require
        created: Aug 22 19:37:01 2007
        lastused: Aug 22 19:52:21 2007
        lifetime: 0(s) validtime: 0(s)
        spid=48 seq=2 pid=2542
        refcnt=2

192.168.177.128[any] 192.168.177.1[any] icmp
        out prio def ipsec
        esp/transport//require
        created: Aug 22 19:37:01 2007
        lastused: Aug 22 19:52:21 2007
        lifetime: 0(s) validtime: 0(s)
        spid=65 seq=1 pid=2542
        refcnt=2

192.168.177.1[any] 192.168.177.128[any] icmp
        fwd prio def ipsec
        esp/transport//require
        created: Aug 22 19:37:01 2007  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=58 seq=0 pid=2542
        refcnt=1
```

Slika 5.6.5. SPD baza inicijatora

SPD baza sadrži SP zapise unijete naredbom `setkey -f`, zapisanih u datoteci `setkey.cf`, a objašnjeni su u poglavlju 4.1.

SAD baza odgovaratelja također sadrži SA zapise (slika 5.6.6):

```
[root@misko source]# setkey -D
192.168.177.1 192.168.177.128
        esp mode=transport spi=3403987481(0xcae4ba19)
reqid=0(0x00000000)
        E: 3des-cbc  342568a4 df19385d e30d072a b06880bc b53d6d2f
adcd98b0
        A: hmac-md5  2a233fdb 93c045f5 90302e71 42bc5a85
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Aug 22 19:50:56 2007   current: Aug 22 19:51:12 2007
        diff: 16(s)     hard: 12000(s)  soft: 6000(s)
        last: Aug 22 19:50:56 2007      hard: 0(s)      soft: 0(s)
        current: 600(bytes)     hard: 12582912(bytes)   soft:
1048576(bytes)
        allocated: 5    hard: 0 soft: 0
        sadb_seq=1 pid=5764 refcnt=0

192.168.177.128 192.168.177.1
        esp mode=transport spi=300258285(0x11e593ed) reqid=0(0x00000000)
        E: 3des-cbc  104a45a2 190b86c1 13324a02 58b60e02 df5e2638
1c73f279
        A: hmac-md5  b51721e7 738db5f1 6f7efedb 670c0505
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Aug 22 19:50:56 2007   current: Aug 22 19:51:12 2007
        diff: 16(s)     hard: 12000(s)  soft: 6000(s)
        last: Aug 22 19:50:56 2007      hard: 0(s)      soft: 0(s)
        current: 320(bytes)     hard: 12582912(bytes)   soft:
1048576(bytes)
        allocated: 5    hard: 0 soft: 0
        sadb_seq=0 pid=5764 refcnt=0
```

Slika 5.6.6. SAD baza odgovaratelja

SAD baza odgovaratelja sadrži dvije IKE sigurnosne poveznice kao i SAD baza inicijatora. Izvorišne i odredišne IP adrese u sigurnosnim poveznicama su obrnute, ali su iste kao i u SAD bazi inicijatora. Također, SPI brojevi sigurnosnih poveznica inicijatora podudaraju se s SPI brojevima sigurnosnih poveznica zapisanih u SAD bazi odgovaratelja. Također, podudaraju se i parametri zapisani u SAD bazama (enkripcijski i autentifikacijski algoritmi, način rada ...)

SPD baza odgovaratelja također sadrži zapise, a ispis izgleda ovako (slika 5.6.7):

```
[root@misko source]# setkey -DP

192.168.177.128[any] 192.168.177.1[any] icmp
        in prio def ipsec
        esp/transport//require
        created: Aug 22 19:35:52 2007
        lastused: Aug 22 19:51:00 2007
        lifetime: 0(s) validtime: 0(s)
        spid=24 seq=2 pid=5862
        refcnt=1

192.168.177.1[any] 192.168.177.128[any] icmp
        out prio def ipsec
        esp/transport//require
        created: Aug 22 19:35:52 2007
        lastused: Aug 22 19:51:00 2007
        lifetime: 0(s) validtime: 0(s)
        spid=41 seq=1 pid=5862
        refcnt=1

192.168.177.128[any] 192.168.177.1[any] icmp
        fwd prio def ipsec
        esp/transport//require
        created: Aug 22 19:35:52 2007  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=34 seq=0 pid=5862
        refcnt=1
```

Slika 5.6.7. SPD baza odgovaratelja

Postojanje zapisa u SAD bazama inicijatora i odgovaratelja potvrđuje da je uspostavljena sigurnosna poveznica između inicijatora i odgovaratelja. U SAD bazu zapise je unio IKE *daemon* te je komunikacija inicijatora i odgovaratelja sigurna.

6 Zaključak

Mnogi projekti otvorenog kôda za svoj razvoj koriste već postojeće biblioteke s nekih drugih, također, projekata otvorenog kôda. Razlog tome je ušteda na vremenu koje je potrebno za razvoj projekta i veća interoperabilnost novog kôda. Takav je slučaj primjenjen i prilikom ostvarivanja Proširivog autentifikacijskog protokola (EAP) u Protokolu za razmjenu ključeva (IKEv2).

Proširiv autentifikacijski protokol je složen protokol, koji, kao što je i pokazano, podržava brojne EAP metode. Ostvarivanje EAP protokola i svih metoda bio bi dug i nimalo lagan proces. Stoga je iskorištena već postojeća programska komponenta – *WPA_supplicant*, kao programsko ostvarenje EAP protokola i EAP metoda. Odabrana je ova programska komponenta jer se pokazala kao najbolje dokumentirana i sa cjelovitim ostvarenjem EAP – a i brojnih EAP metoda. Također, kôd za EAP metode napisan je kao biblioteke pa su kao takve i preuzete iz *WPA_supplicant – a* i ubačene u IKEv2 protokol. Dakako, bilo je potrebno načiniti neke prilagode na strani IKEv2 automata stanja, ali i na strani *WPA_supplicant – a*. Kako iz *WPA_supplicant - a* nije preuzet cijeli kôd već samo jedan njegov dio, napisan je *supplicant* modul, koji sadrži funkcije potrebne za komunikaciju između *WPA_supplicant – a* i IKEv2 *daemona*. Promjene na strani IKEv2 automata stanja uključuju novo stanje koje je uvedeno za EAP autentifikaciju.

Nakon uspješno obavljenog uključivanja kôda *WPA_supplicant - a* u IKEv2 *daemon* autentifikacija je isprobana na dvije EAP metode te se pokazalo kako je ideja uspješno ostvarena. Pozitivne strane korištenja gotovog kôda programske komponente *WPA_supplicant* očituju se u činjenicama da je vrijeme izrade skraćeno jer EAP i njegove metode nisu razvijane od samog početka, a daljnje "održavanje" preuzetog kôda prepušteno je njegovom autoru. Za daljnji rad ostaje ispitati i isprobati rad ostalih EAP metoda. Tu se i nazire jedan od mogućih problema, a to je slaba dokumentiranost EAP metoda, što bi moglo dovesti do poteškoća u ispitivanju rada nekih EAP metoda u IKEv2 protokolu.

7 Literatura

[1.] L. Budin, M. Golub, *Operacijski sustavi 2*, predavanja iz predmeta Operacijski sustavi 2, Zagreb 2005.

[2.] V. Glavinić, *Mreže računala*, (radni materijal za predavanja iz predmeta Mreže računala), Zagreb 2004.

[3.] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz Ed., Extensible Authentication Protocol (EAP), 2004., dostupno na Internet adresi: http://www.faqs.org/rfc/rfc3748.txt

[4.] B. Aboba, D. Simon, PPP EAP TLS Authentication Protocol, 1999., dostupno na Internet adresi: http://www.faqs.org/rfc/rfc2716.txt

[5.] J. Arkko, H. Haverinen, Extensible Authentication Protocol Method for 3rd Generation, Authentication and Key Agreement (EAP-AKA), 2006., dostupno na Internet adresi: http://www.faqs.org/rfc/rfc4187.txt

[6.] M. Bauer, Paranoid Penguin - Securing WLANs with WPA and FreeRADIUS, Part I, 2005., dostupno na Internet adresi: http://www.linuxjournal.com/article/8017

[7.] M. Bauer, Paranoid Penguin - Securing Your WLAN with WPA and FreeRADIUS, Part II, 2005., dostupno na Internet adresi: http://www.linuxjournal.com/article/8095

[8.] F. Bersani, H. Tschofenig, The EAP-PSK Protocol: a Pre-Shared Key EAP Method, draft-bersani-eap-psk-07, 2005., dostupno na Internet adresi: http://www.tschofenig.com/drafts/draft-bersani-eap-psk-07.html

[9.] T. Clancy, W. Arbaugh, EAP Password Authenticated Exchange, draft-clancy-eap-pax-07, 2006., dostupno na Internet adresi: http://www.ietf.cnri.reston.va.us/internet-drafts/draft-clancy-eap-pax-07.txt

[10.] N. Haller, C. Metz, P. Nesser, M. Straw, A one time password (OTP), 1998., dostupno na Internet adresi: http://www.faqs.org/rfc/rfc2289.txt

[11.] H. Haverinen, Ed., J. Salowey, Ed., Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM), 2006., dostupno na Internet adresi: http://www.faqs.org/rfc/rfc4186.txt

[12.] Jouni Malinen, Linux WPA/WPA2/IEEE 802.1X Supplicant, dostupno na Internet adresi: http://hostap.epitest.fi/wpa_supplicant/

[13.] J. Vollbrecht, P. Eronen, N. Petroni, Y. Ohba, State Machines for Extensible Authentication Protocol (EAP), Peer and Authenticator, 2005., dostupno na Internet adresi: http://www.faqs.org/rfc/rfc4137.txt

[14.] G. Zorn, Microsoft PPP CHAP Extensions, Version 2, 2000., dostupno na Internet adresi: http://www.faqs.org/rfc/rfc2759.txt

[15.]W. Simpson, PPP Challenge Handshake Authentication Protocol (CHAP), 1996, dostupno na Internet adresi: http://www.faqs.org/rfcs/rfc1994.html

[16.]B. Aboba, P. Calhoun, RADIUS support for EAP, dostupno na Internet adresi: http://www.ietf.org/rfc/rfc3579.txt

[17.]B. Aboba, et all, Extensible Authentication Protocol (EAP), 2004., dostupno na Internet adresi: http//www.ietf.org/rfc/rfc3748.txt

[18.]J. Vollbrecht, P. Eronen, N. Petroni, Y. Ohba, State Machines for Extensible Authentication Protocol (EAP) – Peer and Authenticator, 2005., dostupno na Internet adresi: http://www.ietf.org/rfc/rfc4137.txt

[19.]C. Kaufman, Ed., Internet Key Exchange Protocol (IKEv2), 2005., dostupno na Internet adresi: http://www.ietf.org/rfc/rfc4306.txt

[20.]P. Eronen, P. Hoffman, IKEv2 Clarifications and Implementation Guidelines, 2006., dostupno na Internet adresi: http://www.ietf.org/rfc/rfc4718.txt

[21.]S. Groš, J. Vučak, Supplicant design document, FER, Zagreb, 2006., neobjavljeni rad

[22.]J. Vučak, L. Jelenković, M. Golub, Implementation of EAP authentication into IKEv2 protocol, MIPRO 2007

Dodatak A – Popis kratica

AAA – Authentication, Autorization, Accounting

AES – Advanced Encryption Standard

AKA – Authentication and Key Agreement

AP – Access Point

EAP – Extensible Authentication Protocol

EAPOL – EAP Over LAN

ESP – Encapsulating Security Protocol

GTC- Generic Token Card

IKE – Internet Key Exchange

LEAP – Lightweight Extensible Authentication Protocol

MSCHAP – Microsoft Challenge Handshake Authentication Protocol

OTP – One Time Password

PAC – Protected Access Credential

PAX – Password Authenticated Exchange

PEAP – Protected Extensible Authentication Protocol

PPP – Point to Point Protocol

PSK – Pre Shared Key

RFC – Request For Comments

SIM – Subscriber Identity Module

SSID – Service Set Identifier

TKIP – Temporal Key Integrity Protocol

TLS – Transport Level Security

WEP – Wired Equivalent Privacy

Wi-Fi – Wireless - Fidelity

WPA – Wi-Fi Protected Access

Dodatak B – Popis Makefile datoteka

Naziv *Makefile* datoteke	Uloga *Makefile* datoteke
Makefile.libeloop	datoteka smještena u hostapd/ direktoriju *wpa_supplicant-a*. Ona prevodi upravljačke djelove (*scheduler parts*) wpa_supplicant-a.
Makefile.libhostapd	datoteka za prevođenje i postavljanje/instaliranje autentifikatorskog dijela programa wpa_supplicant, a smješta se u hostapd/ direktorij.
Makefile.libsupplicant	datoteka za prevođenje kôda EAP metoda i EAP automata stanja, a smješta se u wpa_supplicant/ poddirektorij wpa_supplicant-a.
Makefile.libwpa_utils	datoteka koja uključuje uslužne funkcije (*utility functions*) u *util* biblioteku, koja je također povezana sa IKEv2 *daemonom*.

Dodatak C – Popis datoteka WPA_supplicant-a uključenih u IKEv2 *daemon*

Vrsta datoteka	Popis datoteka
Datoteke EAP automata stanja i EAP metoda	`eap.h`, `eap.c`, `eap_i.h`, `eap_defs.h`, `eap_methods.h`, `eap_methods.c` `eap_aka.c`, `eap_fast.c`, `eap_gtc.c`, `eap_leap.c`, `eap_md5.c`, `eap_mschapv2.c`, `eap_otp.c`, `eap_pax.c`, `eap_pax_common.h`, `eap_pax_common.c`, `eap_peap.c`, `eap_psk.c`, `eap_psk_common.h`, `eap_psk_common.c`, `eap_sake.c`, `eap_sake_common.h`, `eap_sake_common.c`, `eap_sim_common.c`, `eap_sim_common.h`, `eap_sim.c`, `eap_tls.c`, `eap_tls_common.c`, `eap_tls_common.h`, `eap_tlv.c`, `eap_tlv.h`, `eap_ttls.c`, `eap_ttls.h`
Datoteke kriptografskih funkcija	`crypto.h`, `crypto.c`, `md5.c`, `md5.h`, `rc4.c`, `rc4.h`, `sha1.c`, `sha1.h`, `aes_wrap.c`, `aes_wrap.h`, `aes.c`, `crypto_gnutls.c`, `ms_funcs.c` and

Vrsta datoteka	Popis datoteka
	ms_funcs.h, tls.h, tls_none.c, tls_openssl.c, tls_gnutls.c
Konfiguracijske datoteke	config_ssid.h, config.h, config.c, config_file.c

Dodatak D – Konfiguracijska datoteka IKEv2 *daemona*

```
#
#This is the configuration file for Initiator. Initiator will
#authenticate using EAP-MD5 method while responder is expected
#to authenticate using shared secret.
#Note that this is actually forbidden combination, i.e. when
#initiator authenticates through EAP, responder must
#authenticate with certificates. But, for testing purposes it
#is better this way...

general {
        rand_device "/dev/urandom";

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

        # Number of concurent threads for processing sessions
        sm_threads 2;

        # How many fully established IKE SAs from the same IP
        # address will IKE daemon allow before ignoring new
        # requests...
        ikesa_max 50;

        # How many half opened IKE SAs from the same IP address
        # before IKE daemon will stop accepting new IKE SA INIT
        # requests
        ikesa_max_halfopened 50;
        psk_file "./psk.txt";
        kernel_spd rosync;
}

logging {
        file "logfile" {
                # The following configuration directive
                # determines if time is prepended to each log
                # entry

                level trace;
                subsystem main, aes_xcbc, config, crypto, csa,
                message,payload, network, pfkey, proposals,
                session, netlib, sockaddr, sm, transform, ts,
                parser, radius, supplicant, timeout, cfg, auth,
                cert;
        }
```

```
}

remote any
{
        nonce_size 32;
        remote_id rfc822_addr responder@zemris.fer.hr;
        response_timeout 300 s;
        response_retries 3;

        proposal {
                encryption_algorithm 3des;
                auth_algorithm md5_96;
                pseudo_random_function md5;
                dh_group modp768;
        }

        proposal {
                encryption_algorithm aes128_cbc;
                auth_algorithm md5_96;
                pseudo_random_function md5;
                dh_group modp1024;
        }
}

peer any {
        auth_method eap;
        peer_auth_method pre_shared_key;
        my_identifier rfc822_addr jelena@zemris.fer.hr;
        wpa_conf "./eap-tls.conf";

        authlimit {
                time 18 days;
                allocs 128;
                octets 1 MB;
        }

        rekeylimit {
                time 18 days;
                allocs 128;
                octets 1 MB;
        }

        ike_max_idle 1 hour;

        sainfo local 192.168.177.128 remote 192.168.177.1 proto
icmp {

                hardlimit {
                        time 12000 s;
                        allocs 1200;
                        octets 12 MB;
```

```
                }

                softlimit {
                        time 6000 s;
                        allocs 1000;
                        octets 1 MB;
                }

                encryption_algorithm 3des;
                auth_algorithm md5_96, sha1_96;
        }
}
```