Capitolul 6

Securitate pe Internet ($IPSec$)

6.1 Introducere

În 1994, *Internet Architecture Board* (IAB) a emis un raport intitulat "*Securitatea în arhitectura de Internet*" (RFC 1636). Raportul a declarat că internetul are nevoie de mai multă securitate şi a identificat domenii - cheie pentru mecanismele de protecţie. Printre acestea, necesitatea de a securiza infrastructura de reţea: de la monitorizarea neautorizată şi controlul traficului până la necesitatea de a asigura comunicarea între oricare doi utilizatori folosind mecanisme de autentificare şi criptare.

Exemplul 6.1. *Raportul din* 1998 *emis de Computer Emergency Response Team (CERT) prezintă* 1.300 *raportări de incidente de securitate care au afectat aproape* 20.000 *site-uri. Cele mai grave tipuri de atac includ IP spoofing, în care diverşi intruşi creează pachete cu adrese IP false şi exploatează aplicaţiile care utilizează autentificări bazate pe adresă de IP, precum şi diverse forme de interceptare a pachetelor sniffing, în care atacatorii citesc informaţiile trimise, inclusiv informaţii de log on şi conţinuturi ale bazelor de date.*

Practic, $IPSec$ (**I**nternet **P**rotocol **Sec**urity) este o secvenţă de protocoale având ca scop securizarea protocoalelor de comunicare prin Internet (**I**nternet **P**rotocol – IP), autentificând şi criptând fiecare pachet IP de şir de date.

$IPSec$ include de asemenea protocoale de autentificare reciprocă între parteneri la începutul sesiunii şi stabilirea cheilor criptografice care vor fi folosite pe durata sesiunii de comunicare.

$IPSec$ poate fi utilizat de asemenea pentru protejarea fluxurilor de date dintre două entităţi (de exemplu un calculator şi un server), între două gateway-uri (porţi) de securitate[1] (de exemplu routere sau firewalls), sau între un gateway de securitate şi un utilizator.

[1]Un *gateway de securitate* este un sistem intermediar care implementează diverse protocoale $IPSec$. A se vedea Definiţia 6.1.

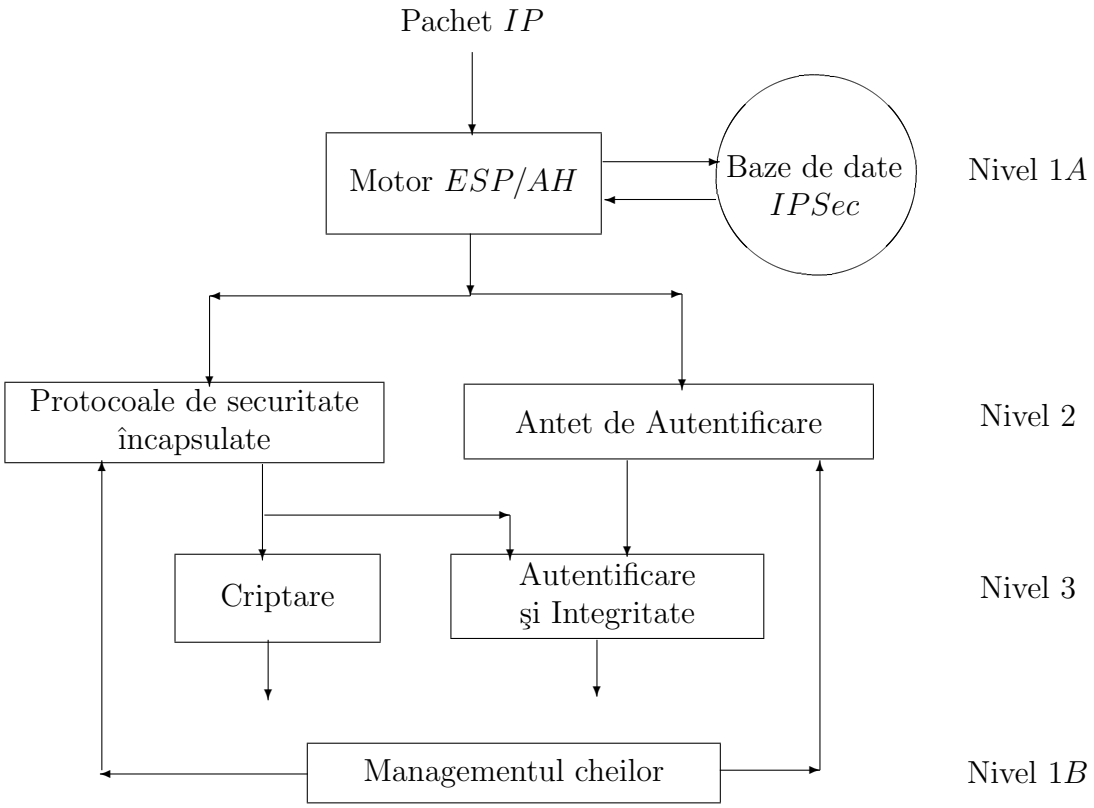
De multe ori *IPSec* este utilizat pentru protejarea traficului pe Internet.

Istoric, *IPSec* este un succesor al standardului *ISO NLSP* (**N**etwork **L**ayer **S**ecurity **P**rotocol). *NLSP* era definit pe baza protocolului *SP*3 publicat de *NIST*, care desemna un proiect de securitate a datelor prin reţea aparţinând *NSA* (**N**ational **S**ecurity **A**gency).

Oficial, standardele *IPSec* sunt specificate de *IETF* (**I**nternet **E**ngineering **T**ask **F**orce) printr-o serie de cereri de comentarii relative la diverse componente şi extensii (inclusiv definirea termenilor şi a nivelurilor de securitate).

6.2 Arhitectura *IPSec*

Aşa cum s-a stablit, *IPSec* este compus dintr-un set de protocoale interconectate, al căror scop este asigurarea unor servicii de securitate.
În termeni generali, arhitectura unui *IPSec* este descrisă de figura de mai jos:



- **Bazele de date** *IPSec*:

  De obicei sunt trei baze de date standard la acest nivel:

  1. *SPD* (**S***ecurity* **P***olicy* **D***atabase*): Specifică controlul de trafic *IP* al ambelor părţi implicate (expeditor şi destinatar).

2. *SAD* (**S***ecurity* **A***ssociation* **D***atabase*): Conţine parametrii fiecărei asocieri de securitate stabilite (*SA*). Fiecare *SA* are un element în *SAD*.
   În general, un *SAD* conţine:
   - Indexul parametrului de securitate (*SPI*).
   - Serviciile de securitate încapsulate (*ESP* – **E***ncapsulated* **S***ecurity Payload/***P***rotocols*), cu datele curente transmise: algoritmi de criptare şi integritate, modul de generare al cheilor, valorile iniţiale (*IV*) etc.
   - Antetul de autentificare (*AH*), cu algoritmul de autentificare, cheile *MAC* etc.
   - Timpul de valabilitate al *SA*.
   - Tipul de protocol *IPSec* ("*tunel*" sau "*transport*") aplicat lui *SA*.
3. *PAD* (**P***eer* **A***uthorisation* **D***atabase*): asigură legătura între *SPD* şi un protocol de gestiune a securităţii (de exemplu *IKE*).

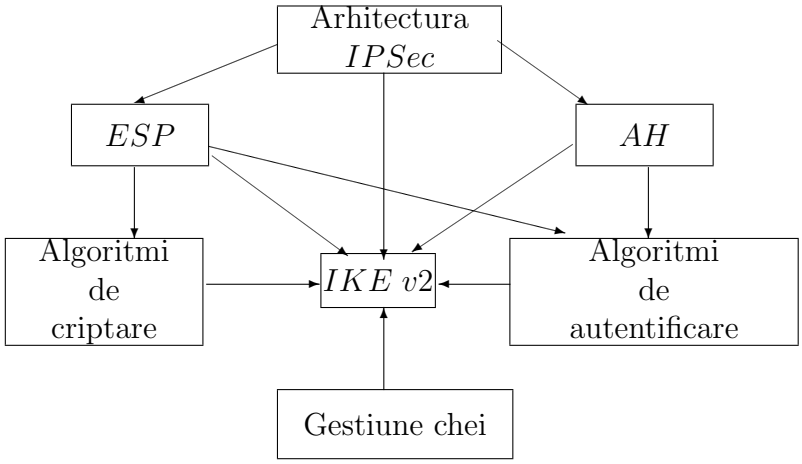
- **Nivel** 2: Cele două protocoale de securitate de la acest nivel (care au sub control şi Nivelul 3) sunt:
  1. *AH* (*IP* **A**uthentication **H**eader): folosit pentru autentificare, este bazat pe standardul *RFC* 4302.
  2. *ESP* (bazat pe standardul *RFC* 4303): este folosit pentru criptare şi autentificare.

- **Nivel** 3:

  Fiecare algoritm criptografic pentru autentificare şi criptare este definit de un standard *RFC* specific. În mod uzual, pentru criptare se folosesc *AES* şi 3*DES*, iar pentru autentificare şi integritate – funcţii de dispersie cu cheie.

- **Protocoale de gestiune a cheilor**:
  Sunt descrise în *IKEv*2 (**I**nternet **K**ey **E**xchange, version 2), bazate pe un standard din 2005 (*RFC* 4306).

O relaţie între aceste protocoale este redată de schema următoare:



6.3 Negociere $IPSec$

În momentul când un pachet de date este transmis între doi utilizatori, are loc un protocol de negociere lansat de expeditor (notat *Alice*), respectiv destinatar (*Bob*).

6.3.1 Pachetul de ieşire

Serverul *Alice* vrea să trimită pachetul de date α spre serverul *Bob*. Pentru aceasta se deschide o aplicaţie $IPSec$, care funcţionează după următorul protocol (numit "negociere"):

1. Aplicaţia apelează stiva TCP/IP pentru a prelua α.
2. α este preluat de un algoritm de negociere $\mathcal{AN}$, care construieşte un "înveliş" de protecţie.
3. $\mathcal{AN}$ caută pachetul α în baza de date a protocoalelor de securitate şi decide dacă este necesară o protejare a sa sau doar un permis de trecere $IPSec$. În final, α poate fi
   (a) protejat folosind serviciile $IPSec$;
   (b) eliminat;
   (c) asociat cu un permis de trecere $IPSec$.
4. Dacă α trebuie protejat, $\mathcal{AN}$ trimite aplicaţiei adresa lui *Bob*, care o caută în SA şi SPI din bazele sale interne de date.
5. Dacă nu a fost negociat încă un SA pentru adresa respectivă, atunci aplicaţia lansează o negociere IKE cu adresa respectivă, pentru crearea unui SA.
6. După terminarea negocierii, aplicaţia trimite SPI şi SA spre $\mathcal{AN}$; acesta va construi o protecţie pentru α folosind cheia negociată de aplicaţie.

6.3.2 Pachetul de intrare

Dacă *Bob* primeşte un pachet de date α în portul rezervat negocierii IKE (de obicei acest port este 500 sau 4500), atunci:

1. Dacă pentru adresa primită nu a fost negociat încă nici un SA, atunci $\mathcal{AN}$ va transmite pachetul α aplicaţiei de negociere.

2. Dacă α are un SPI asociat, atunci este căutat SA-ul corespunzător în baza de date $IPSec$. Dacă SPI nu are corespondent în baza de date, pachetul α este respins.

3. Dacă α nu conţine nici un SPI, atunci $\mathcal{AN}$ poate deduce că α nu are asociat nici un SA şi îl respinge.

6.4 Asocierile de securitate (SA)

O asociere de securitate ataşează parametri de securitate pachetelor de date care sunt trimise în trafic. Un SA descrie parametrii de securitate acceptaţi de *Alice* şi *Bob*.

În această fază se stabileşte o conexiune între o sursă şi o destinaţie, în care cei doi parteneri trebuie să cadă de acord – printre altele – asupra algoritmilor de criptare şi autentificare, asupra cheilor de criptare (mărimea, durata, modul de trimitere), valorile de iniţializare, precum şi alţi parametri de securitate.

După ce s-a definit SA-ul unei conexiuni, lui i se atribuie un index (SPI – *Security Parameter Index*) şi este stocat în baza de date SPD. În interiorul acestei baze de date, la SA se adaugă adresele IP ale sursei şi destinaţiei.

Deci, toată informaţia dintr-un SA este grupată în:

- **Indexul parametrului de securitate** (SPI). Scopul lui este de a asocia un SA cu o conexiune particulară.

- **Adresa IP de destinaţie**: de obicei adresa unui user sau a unui gateway (router sau firewall).

- **Protocolul pentru securitatea ID**: Indică dacă protocolul de securitate este ESP sau AH.

Asocierea de securitate corespunzătoare unei conexiuni poate fi un ESP sau un AH, dar nu amândouă. Dacă o conexiune are asociate ambele protocoale, atunci sunt create cel puţin două SA-uri pentru a-i asigura protecţia. O comunicare securizată bidirecţională tipică între două entităţi necesită două asocieri de securitate (câte una pentru fiecare direcţie).

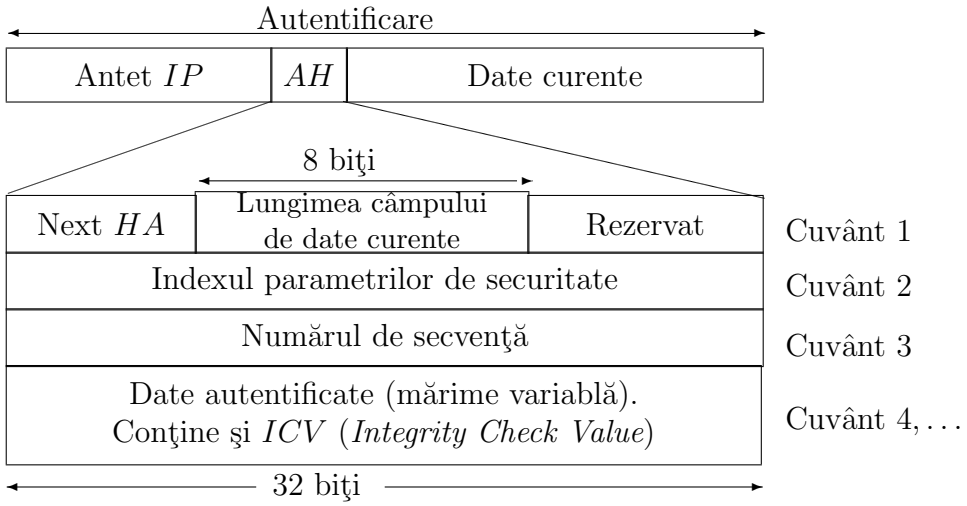
Atât protocolul de securitate ESP cât şi AH suportă două moduri de operare: modul transport şi modul tunel.

6.4.1 Antetul de autentificare (AH)

Protocolul AH (RFC 4302) defineşte formatul pachetelor $IPSec$; el asigură integritatea conexiunii, autentificarea originii datelor şi – opţional – un serviciu anti-replay. De remarcat că el nu criptează porţiunea de date din pachet.

AH poate fi folosit singur, combinat cu ESP, sau într-o formă internă a modului de utilizare tunel.

Structura unui AH poate fi reprezentată astfel:



unde:

- **Next** HA: zonă de 8 biţi care identifică tipul antetului pe care îl are următorul câmp de date curente (payload).
- **Lungimea câmpului de date curente**: Dacă AH este format din s cuvinte, conţintul acestui câmp este $s - 2$. În cazul standard (ca în figură) al unui AH de 4 cuvinte, tot antetul de autentificare are 6 cuvinte, deci conţinutul acestui câmp este valoarea 4 (scrisă pe 8 biţi).
- **Rezervat**: Câmp de 16 biţi, rezervat pentru o utilizare ulterioară.
- **Indexul parametrilor de securitate** (SPI): indică protocoalele de securitate, algoritmii şi cheile folosite. Împreună cu adresa IP a destinatarului şi protocolul de securitate (AH), el identifică în mod unic asocierea de securitate pentru o comunicare. Valoarea SPI este selectată de (sistemul) destinatar în momentul stabilirii SA, din intervalul $[1, 255]$ (codificarea este asigurată de $IANA$ – *The* **I***nternet* **A***ssigned* **N***umbers* **A***uthority*).
- **Numărul de secvenţă**: spune câte pachete au fost trimise şi asigură protecţie non-reply. Câmpul conţine un contor monoton crescător.

  - Prezenţa lui este obligatorie, chiar dacă destinatarul *Bob* nu alege o opţiune non-replay pentru un anumit SA.
  - La iniţierea unei asocieri de securitate, contoarele lui *Alice* şi *Bob* sunt setate pe 0. Primul pachet trimis folosind un SA fixat va avea numărul 1.
  - Dacă este activat un non-replay (lucru realizat de obicei by default), numărul de secvenţă nu se alocă niciodată într-un ciclu.
  - Numărul maxim de pachete transmise într-un SA este 2^{32}. Cele două contoare trebuie resetate (stabilind un nou SA şi deci o nouă cheie) înainte ca ele să ajungă la această valoare.

- **ICV**: Este valoarea de control a integrităţii pachetului de date transmis. Este o variabilă întreagă scrisă ca un multiplu de 32 biţi (pentru $IPv4$) sau 64 biţi ($IPv6$), variante selectate printr-un câmp suplimentar, inclus explicit. Toate implementările trebuie să asigure acest câmp suplimentar.

  Pentru a calcula ICV-ul unui AH se ţine cont de:

  - Câmpurile antetelor IP, care sunt sau neschimbate în trafic, sau sunt predictibile ca valoare (pentru un AH) în momentul recepţiei.
  - Toate datele din antetul AH.
  - Datele oferite de nivelul de protocol şi setul de date curente – despre care se presupune că sunt neschimbate în trafic.

  Algoritmii folosiţi pentru implementarea antetului de autentificare sunt:

  - $HMAC - SHA1 - 96$; obligatoriu.
  - $AES - XBCB - MAC - 96$; recomandabil.
  - $HMAC - MD5 - 96$; opţional.

6.4.2 *ESP*

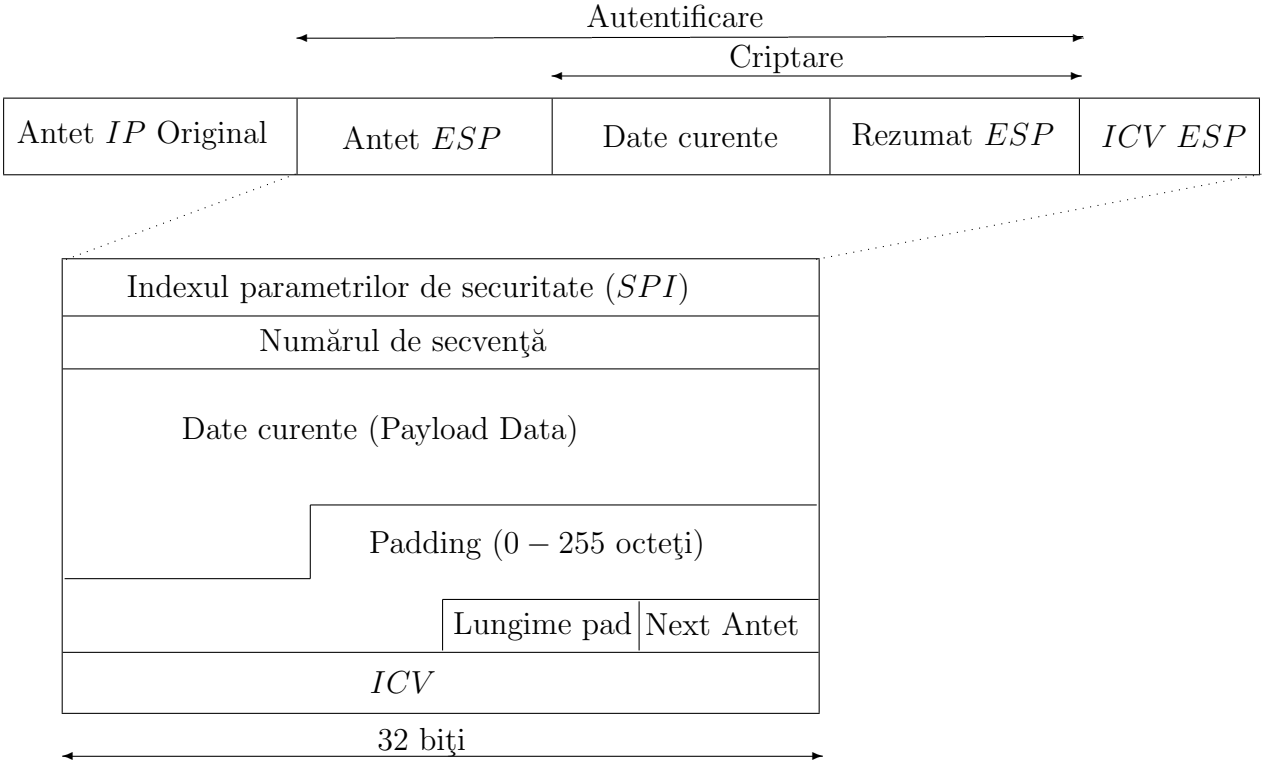
Protocolul ESP (RFC 4303) oferă – pe lângă serviciile asigurate de un AH – confidenţialitatea datelor, şi – într-o formă limitată – a traficului pe canal (deci criptarea datelor).

Principala diferenţă dintre autentificarea asigurată de ESP şi cea oferită de AH constă în extinderea ariei de acoperire: ESP nu protejează nici un antet IP care nu este încapsulat prin ESP (modul tunel).
Setul de servicii ESP depinde de opţiunile selectate în momentul stabilirii asocierii de securitate şi de locul implementării.

Autentificarea originii datelor şi corectitudinea informaţiilor de conectare sunt servicii auxiliare numite "*integritate*". Pentru protocol trebuie selectat cel puţin unul din serviciile de confidenţialitate sau integritate.

Serviciul non-replay poate fi ales doar împreună cu cel de integritate, iar această alegere este în totalitate la dispoziţia destinatarului.

Confidenţialitatea fluxului din trafic necesită alegerea modului tunel şi ea este realizată efectiv dacă se include în securitatea la nivel de gateway – unde agregarea traficului poate masca pachetele de informaţie sursă - destinaţie.



ESP asigură criptarea unui IP la nivel de pachet, folosind algoritmi de criptare simetrici. Sunt admişi diverşi algoritmi cel recomandat de standard fiind AES.

Să dăm o descriere a câmpurilor dintr-o încapsulare ESP (conform figurii de mai sus):

- **SPI**: Identic cu formatul AH.
- **Numărul de secvenţă**: Identic cu formatul AH.
- **Date curente** (Payload data): Câmp de lungime variabilă care conţine datele descrise de câmpul *Next Antet*.
  ESP foloseşte algoritmi de criptare simetrici şi – deoarece pachetele IP pot veni într-o ordine arbitrară – fiecare pachet va conţine şi informaţii de sincronizare criptografică (de exemplu un vector de iniţializare).
  Algoritmii de criptare utilizaţi în implementări sunt:
  - $3DES - CBC$: obligatoriu.

  - $AES - CBC$: recomandabil.
  - $AES - CTR$: recomandabil.

  Se pot folosi şi alţi algoritmi de criptare, cum ar fi $RC5$, $IDEA$, $CAST$, $Blowfish$, pentru care **DO**meniul de **I**nterpretare (DOI) are deja asignaţi identificatori.

- **Zona anexă (Padding)** – de maxim 255 octeţi – este necesară pentru următoarele motive:
  - Unii algoritmi de criptare folosesc blocuri de text clar de lungime fixată. Zona anexă este folosită pentru a completa textul clar până la lungimea solicitată. Textul clar constă din datele curente, anexa, lungimea zonei anexe şi câmpul rezervat pentru "*Next Antet*".
  - Anexa este adăugată la textul criptat, pentru ca în final numărul de biţi al textului criptat să fie multiplu de 32.
  - Zone anexe adiţionale se pot folosi şi pentru a ascunde lungimea reală a anexei, cu scopul de a creşte confidenţialitatea fluxului de date în trafic.

- **Lungime pad**: Indică numărul de octeţi din zona anexă. Valoarea este un număr din intervalul $[0, 255]$, unde 0 înseamnă că zona anexă nu este prezentă.

- **Next antet**: Câmp de un octet care identifică tipul de date din zona datelor curente. Codificarea folosită este indicată pe pagina $IANA$ (*http://www.iana.org/numbers/*). De exemplu, valoarea 4 indică $IPv4$, valoarea 41 indică $IPv6$ etc.
  De asemenea, se poate identifica un protocol de nivel superior; de exemplu, valoarea 6 indică protocolul TCP.

- **ICV**: Valoarea de control a integrităţii este identică cu cea de la AH, cu singura specificaţie că aici ea este calculată pentru 3 câmpuri: *Antet ESP, Rezumat ESP* şi *Date curente*. Dacă s-a selectat serviciul de criptare, ultimele două câmpuri sunt sub formă criptată (criptarea fiind aplicată înainte de autentificare)

6.4.3 Modurile de operare AH şi ESP

$IPSec$ poate fi implementat pe două tipuri de echipament: pe server gazdă sau pe poartă (gateway) de securitate.

Definiţia 6.1. *Un "gateway de securitate" este un sistem intermediar între două reţele. Canalul de intrare într-un gateway este nesigur, iar canalul de ieşire este securizat.*

Un gateway implementează $IPSec$ pe interfaţa nesigură, pentru a permite comunicaţii securizate între servere gazdă de pe ambele laturi.
Astfel, serviciile de securitate pot fi asigurate:

1. între orice două servere care comunică între ele;
2. între o pereche de gateway de securitate;
3. între un gateway şi un server.

AH şi *ESP* suportă două moduri de operare: modul transport şi modul tunel.

- Un *SA* în **mod transport** este o asociere de securitate între două servere.
  Când un gateway de securitate lucrează în mod transport, el acţionează ca un server: traficul îi este destinat lui.
- Un *SA* în **mod tunel** este o asociere de securitate în care cel puţin unul din parteneri este un gateway.

Modul transport este folosit pentru protejarea protocoalelor de comunicare, pe când modul tunel are ca principal scop protejarea pachetelor *IP* (tot pachetul *IP* este încapsulat în alt pachet *IP* şi este adăugat un nou antet *IP* între antetele interne şi externe).

1. În modul transport, antetul protocolului de securitate apare imediat după antetul *IP* originar şi înaintea datelor curente. Pentru *ESP* implementat în modul transport, *SA* asigură servicii de securitate numai pentru protocoale de nivel înalt, nu şi pentru antetul *IP* originar sau alte antete extinse care preced antetul *ESP*.
   Pentru *AH*, protecţia este extinsă la antetul *IP* originar.
2. Pentru *SA* în modul tunel, apar încă două antete:
   - unul extern – care specifică destinaţia finală *IPSec* şi detalii legate de această destinaţie;
   - un antet intern care specifică ultima destinaţie (aparentă) a pachetului.

   Antetul protocolului de securitate apare după antetul *IP* exterior şi înaintea celui interior. Dacă *AH* este utilizat în modul tunel, unele porţiuni din antetul *IP* exterior sunt asignate protecţiei, împreună cu toate pachetele *IP* interne. Asta înseamnă că toate antetele *IP* interioare sunt protejate, împreună cu toate protocoalele specificate în datele curente.
   Dacă se foloseşte *ESP*, protecţia este asigurată numai pentru pachetele interne, nu şi pentru antetul exterior.

6.5 Schimbul de chei Internet (*IKE v2*)

Deoarece serviciile de securitate *IP* folosesc sisteme simetrice de criptare, este necesar ca ambele părţi (*Alice* şi *Bob*) să cadă de acord asupra unor mecanisme care să le permită să stabilească chei secrete pentru criptare, autentificare şi integritate. *IPSec* permite o distribuţie a cheilor atât manual cât şi automat.

Definiţia 6.2. *Protocolul IKE (**I**nternet **K**ey **E**xchange) stabileşte proceduri şi formate de pachete de date care stau la baza definirii, negocierii, modificării şi eliminării asocierilor de securitate (SA). El asigură cadrul de securitate pentru transferul de chei şi autentificarea datelor, independent de mecanismul de generare al cheilor, algoritmii de criptare sau mecanismele de autentificare.*

Protocolul IKE este utilizat free în două versiuni: $IKE\ v1$ (definit de RFC 2409) şi $IKE\ v2$ (definit în prima variantă în decembrie 2005 cu RFC 4306, iar forma completă în septembrie 2010, prin RFC 5996). Ambele au fost dezvoltate şi cu drept de copyright de $ISOC$ (*The* **I***nternet* **SOC***iety*).

Datele curente ale unui SA propun un set de algoritmi de criptare, mecanisme de autentificare şi algoritmi de stabilire a cheilor care să fie utilizaţi în IKE, ESP şi/sau AH. Protocolul IKE nu se limitează la anumiţi algoritmi; el permite construcţii pentru accesarea upgradărilor sau a altor mecanisme care pot fi preferate de utilizatori. Aceste noi obiecte cu care va lucra $IPSec$ nu necesită dezvoltarea unui nou IKE sau înlocuirea celui vechi.

Vom prezenta în această secţiune elementele principale din $IKE; v2$, , urmaând ca ulterior să abordăm şi $IKE\ v1$.

În mod obişnuit, o asociere de securitate include obligatoriu următorii parametri (fără a se limita doar la aceştia):

1. Tipul de protecţie folosit (ESP sau AH).
2. Algoritmul de autentificare folosit cu AH.
3. Cheile folosite cu algoritmul de autentificare din AH.
4. Algoritmul de criptare şi modul, utilizate cu ESP.
5. Cheia utilizată de algoritmul de criptare în ESP.
6. Vectorul de iniţializare pentru algoritmul de criptare utilizat în ESP.
7. Algoritmul de autentificare şi modul, utilizate cu ESP.
8. Cheia de autentificare utilizată cu algoritmul de autentificare în ESP.
9. Durata de valabilitate a cheii utilizate sau data când ea trebuie schimbată.
10. Algoritmii de dispersie pentru crearea amprentei care va fi semnată.
11. Informaţii despre grupul peste care se va defini schimbul de chei Diffie - Hellman.
12. Perioada de valabilitate a asocierii de securitate curente.
13. Adresa sursei care a construit asocierea de securitate.

6.5.1 Selectarea algoritmilor

IKE foloseşte următoarele recomandări de implementare (care trebuie negociate de *Alice* şi *Bob* pentru construirea asocierii de securitate $IPSec$):

- Algoritmii de criptare care protejează datele:
  - $3DES$ (obligatoriu)
  - $AES-CBC-128$ (recomandat)
  - $AES-CTR-128$ (recomandat)
- Algoritmi de protecţie a integrităţii datelor, care produc amprente:
  - $HMAC-SHA1-96$ (obligatoriu)
  - $AES-XCBC-96$ (recomandat)
  - $HMAC-MD5-96$ (opţional)
- Informaţii despre grupul peste care operează protocolul Diffie - Hellman:
  - Grupul 2 – unde se calculează logaritmi discreţi pe 1024 biţi (obligatoriu)
  - Grupul 14 – unde se calculează logaritmi discreţi pe 2048 biţi (recomandat)
  - Curbe eliptice peste $GF(2^{155})$ sau $GF(2^{185})$ (opţional)
- Generatori de numere pseudoaleatoare:
  - $PRF-HMAC-SHA1$ (conform standardului RFC 2104) (obligatoriu)
  - $PRF-AES-XCBC-PRF-128$ (conform standardului RFC 3664) (recomandat)
  - $PRF-HMAC-MD5$ (conform standardului RFC 2104) (opţional)

6.5.2 Schimbul de mesaje IKE

Alice (care iniţiază comunicarea) propune un set de algoritmi pe care poate să îi suporte sistemul său; *Bob* va selecta din ei algoritmii pe care îi agrează, creînd astfel un $IKE-SA$ (asocierea de securitate corespunzătoare IKE). Acesta este utilizat pentru protejarea negocierii pentru formarea protocolului SA.

În general, două entităţi (de exemplu două servere $IPSec$) pot negocia mai multe SA-uri.

Comunicările IKE constau din perechi de mesaje: o cerere urmată de un răspuns. Primul schimb de mesaje constă totdeauna din două cereri/răspuns: $IKE-SA-INIT$ şi $IKE-AUTH$.

IKE-SA-INIT

În acest prim schimb de mesaje, iniţiatorul *Alice* şi destinatarul *Bob* negociază utilizarea algoritmilor de criptare (definind un $IKE-SA$) şi schimbă informaţiile necesare stabilirii unui schimb de chei (trimiţând nonce-uri şi valorile Diffie - Hellman). Cheile alese sunt utilizate pentru protejarea schimbului următor $IKE-AUTH$.

- La primul pas (Pas 1), *Alice* trimite un mesaj de forma

$$HDR\|SA_A1\|KE_A\|N_A$$

unde

  - HDR – Zonă care conţine SPI, numărul versiunii IKE, şi alţi identificatori (cum ar fi datele curente pentru KE_A şi SA_A1).
  - SA_A1 – Informaţia propusă de *Alice* pentru crearea $IKE-SA$: generatorul de numere pseudo-aleatoare şi algoritmii criptografici agreaţi, precum şi grupul de bază pentru protocolul Diffie - Hellman.
  - KE_A – Cheia Diffie - Hellman g^a a lui *Alice*.
  - N_A – un nonce generat de *Alice* (pentru protejarea contra unui atac reply).

- *Bob* răspunde (Pas 2) cu mesajul

$$HDR\|SA_B1\|KE_B\|N_B\|[CERTREQ]$$

unde

  - Conţinutul HRD, SA_B1, KE_B sunt similare cu informaţiile analoge ale lui *Alice*. Şirul algoritmilor propuşi de *Bob* sunt o selecţie din intersecţia listei trimise de *Alice* şi proprii săi algoritmi agreaţi. Cheia Diffie - Hellman g^b şi nonce-ul N_B completează informaţia curentă.
  - Opţional, *Bob* poate solicita un anumit tip de certificat (de exemplu $X.509$), trimiţând această cerere în $CERTREQ$.

După acest prim schimb de mesaje, partenerii au stabilit un $IKE-SA$ conţinut în SA_B1, neautentificat încă.
Similar, după schimbul de chei Diffie-Hellman, ei au generat o cheie de sesiune neautentificată $SKEYSEED$ din care vor deriva ulterior toate cheile necesare pentru $IKE-SA$:

SK_e – cheia de criptare (câte una pentru fiecare direcţie);

SK_a – cheia pentru autentificarea şi verificarea integrităţii mesajului (câte una pentru fiecare direcţie);

SK_d – cheia pentru derivarea cheilor necesare asocierilor de securitate derivate;

SK_p – cheia pentru crearea datelor curente din schimbul de mesaje următor.

De remarcat că în acest moment *Alice* şi *Bob* au stabilit algoritmii care vor fi utilizaţi în comunicările ulterioare, dar încă nu s-au autentificat reciproc.

IKE-SA-AUTH

În acest al doilea schimb de mesaje *Alice* şi *Bob* se autentifică reciproc folosind diverse mecanisme de autentificare (semnături digitale, certificate, EAP – **E**xtensible **A**uthentication **P**rotocol, chei partajate). Acum se crează primul $IKE - SA$ şi asocierea sa de securitate $IPSec$; ele se numesc generic "$child - SA$".

- *Alice* trimite (Pas 3) lui *Bob*

$$HDR \| SK\{ID_A, [CERT], [CERTREQ], [ID_B], AUTH, SA_A2, TS_A, TS_B\}$$

unde:

  - HDR – antetul; include SPI-urile celor doi parteneri, numărul versiunii IKE, identificatorii de mesaje folosiţi în $IKE - SA - INIT$.
  - $SK\{\ldots\}$ – se asigură criptarea şi integritatea datelor folosind SK_e şi SK_a.
  - $CERT$ – un certificat al lui *Alice*, eliberat de o entitate PKI.
  - $CERTREQ$ – O listă de autorităţi de certificare de încredere.
  - ID_A, ID_B – identificatorii celor doi parteneri (*Alice*, *Bob*).
  - $AUTH$ – autentificarea (un mesaj semnat).
  - SA_A2 – Informaţia propusă de *Alice* pentru crearea primului $child-SA$; poate conţine de exemplu antetul de autentificare (AH sau ESP), protocoalele utilizate etc.
  - TS_A, TS_B – selectori de trafic. Transmit celor doi parteneri adresele de contact, porturile şi protocolul IP folosit.

$CERT$, $CERTREQ$ şi ID_B sunt opţionale.

Exemplul 6.2. *Dacă Alice trimite* $TS_A = \{192.0.1.0 - 192.0.1.255\}$, $TS_B = \{192.0.2.0 - 192.0.2.255\}$*, înseamnă că ea doreşte ca toată informaţia să îi fie trimisă la o adresă* IP *din domeniul* $\{192.0.1.0 - 192.0.1.255\}$ *şi solicită ca tot ce transmite pe adresa lui Bob să fie la o adresă* IP *din domeniul* $TS_B = \{192.0.2.0 - 192.0.2.255\}$.

- *Bob* răspunde (Pas 4) cu mesajul

$$HDR \| SK\{ID_B, [CERT], AUTH, SA_B2, TS_A, TS_B\}$$

unde

  - HDR include SPI-ul lui *Alice* şi *Bob*, versiunea IKE şi identificatorul de mesaj trimis de *Alice*.

- $CERT$ (opţional) este certificatul lui Bob – trimis doar la cerere.
- $AUTH$ – câmp utilizat de Bob pentru a se autentifica faţă de $Alice$.
- SA_B2 completează negocierea pentru crearea $child-SA$, acceptând algoritmi propuşi de $Alice$ şi identificând protocolul negociat (AH sau ESP).
- TS_A, TS_B sunt selectorii de trafic. Dacă Bob este de acord cu selectorii propuşi de $Alice$, aceste valori sunt identice cu cele din mesajul anterior.

De remarcat că oricare din părţi poate iniţia acest al doilea schimb de mesaje (deci rolurile lui $Alice$ şi Bob pot fi inversate).

Chiar şi în cele mai simple scenarii de comunicare, aceste prime patru schimburi de mesaje (din $IKE-SA-INIT$ şi $IKE-SA-AUTH$) sunt destul de costisitoare ca resurse. În cele mai multe situaţii ele sunt însă preferabile, deoarece:

1. Deşi entităţi ca serverele $IPSec$ consumă timp pentru prelucrarea acestor mesaje, din informaţia obţinută se pot crea $child-SA$ multiple care vor putea fi folosite în contactele următoare, fără a mai trece prin acest start complet.

2. $IKE-INIT$ stabileşte un $IKE-SA$ care include informaţia secretă (partajată) utilizată în generarea asocierilor de securitate $child-SA$.

3. În $IKEv2$, primul $child-SA$ este creat pe baza schimbului de mesaje $IKE-SA-AUTH$. Celelalte asocieri de securitate $child-SA$ vor necesita un singur schimb de mesaje – extrem de simplu şi rapid – în $CREATE-child-SA$,

CREATE-child-SA

Noul schimb de mesaje dintre parteneri are ca scop generarea unor $child-SA$ noi şi modificarea cheilor asocierilor de securitate active. Toate mesajele sunt protejate criptografic folosind algoritmi de criptare şi chei negociate în $IKE-SA-INIT$ şi $IKE-SA-AUTH$. Totuşi, dacă se solicită garanţii suplimentare de securitate pentru aceste primitive, $CREATE-child-SA$ poate folosi informaţii Diffie-Hellman suplimentare pentru a genera chei noi.

- Protocolul începe cu $Alice$ care trimite mesajul

$$HDR\|SK\{[N^+], SA, N_A, [KE_A], TS_A, TS_B\}$$

unde

- HDR – antet care include SPI-urile celor doi parteneri, numărul de versiune IKE şi identificatorii de mesaj.
- $SK\{\ldots\}$ – criptarea datelor curente cu cheia SK_e şi asigurarea integrităţii cu cheia SK_a.

- $[N^+]$ – notificare (opţională) care conţine detalii suplimentare pentru $child - SA$.
- SA – asocierea de securitate propusă de *Alice*.
- N_A – un nonce.
- KE_A – o valoare nouă g^a pentru cheia Diffie-Hellman (opţional).
- TS_A, TS_B – selectori de trafic.

Tot mesajul este criptat şi integritatea sa protejată folosind cheile calculate din SK_d.

- *Bob* răspunde cu

$$HDR \| SK\{[N^+], SA, N_B, [KE_B], TS_A, TS_B\}$$

unde

- HDR – similar antetului din mesajul primit.
- $[N^+]$ – notificare (opţională) cu detalii suplimentare pentru $child - SA$.
- SA – algoritmii pe care îi agrează din asocierea de securitate primită.
- N_B – un nonce propriu.
- $[KE_B]$ – o nouă valoare Diffie-Hellman g^b (opţional).
- T_A, TS_B – selectorii de trafic agreaţi.

Şi acest mesaj este criptat şi integritatea sa protejată folosind cheile calculate din SK_d.

Într-o asociere de securitate, IKE, ESP şi AH folosesc cheile secrete doar o perioadă limitată de timp şi numai pentru o cantitate limitată de date. După expirarea unui SA este stabilită o nouă asociere de securitate, derivând alte chei din $IKE - SA$ sau $child - SA$.

1. Dacă $CREATE - child - SA$ este folosit în obţinerea noilor chei pentru $IKE - SA$, atunci are loc schimbul de mesaje
   (a) $Alice \longrightarrow Bob: \quad HDR \| SK\{SA, N_A, [Ke_A]\}$
   (b) $Bob \longrightarrow Alice: \quad HDR \| SK\{SA, N_B, [Ke_B]\}$
2. Dacă $CREATE - child - SA$ este folosit în obţinerea noilor chei pentru $child - SA$, atunci are loc schimbul de mesaje
   (a) $Alice \longrightarrow Bob: \quad HDR \| SK\{N(REKEY - SA), [N^+], SA, N_A, [Ke_A], TS_A, TS_B\}$
   (b) $Bob \longrightarrow Alice: \quad HDR \| SK\{[N^+], SA, N_B, [Ke_B], TS_A, TS_B\}$

De remarcat că *Alice* identifică $child - SA$ ale cărui chei trebuie schimbate notificând datele curente prin $N(REKEY - SA)$.

6.5.3 Schimbul de informaţii în IKE

După crearea unui $IKE{-}SA$, *Alice* şi *Bob* îşi pot transmite mesaje de control privind erori sau diverse notificări. Mesajele schimbate pot fi notificări (N), ştergeri (D) şi configurări de date (**C**onfiguration **P**ayloads - CP).
Schimbul de informaţii poate să nu conţină mesaje – să fie doar o verificare dacă partenerul mai este prezent (ceva de tipul "Mai eşti acolo ?").

Un schimb de informaţii este de forma:

$$\begin{array}{cccc} \multicolumn{2}{c}{Alice} & & Bob \\ HDR\|SK\{[N],[D],[CP],\ldots\} & \longrightarrow & & \\ & \longleftarrow & HDR\|SK\{[N],[D],[CP],\ldots\} & \end{array}$$

6.5.4 Generarea componentelor cheii în IKE

Într-un $IKE-SA$ sunt negociaţi patru algoritmi criptografici: algoritmi de criptare, de protecţia integrităţii, grupul Diffie-Hellman şi generatorul de numere pseudoaleatoare (prf). Componentele cheii pentru toţi algoritmii criptografici folosiţi în $IKE-SA$ şi $child-SA$ sunt obţinute totdeauna ca ieşiri ale unui algoritm prf.
$IKEv2$ foloseşte pentru schimbul de chei numai algoritmul Diffie-Hellman.

Un protocol Diffie-Hellman se bazează pe un modul p, un generator g al lui $GF(2^p)$ şi o cheie secretă a, respectiv b [2]. *Alice* şi *Bob* schimbă informaţia necesară în $IKE-INIT$ prin KE_A respectiv KE_B. Această informaţie cuprinde g^a, g^b şi nonce-urile N_A, N_B. Cheia comună de sesiune $SKEYSEED$ este calculată apoi de parteneri după formula

$$SKEYSEED = prf(N_A\|N_B, g^{ab})$$

Exemplul 6.3. *Să presupunem că Alice şi Bob au convenit prin schimbul de mesaje din* $IKE-INIT$ *la valorile comune* $p = 47,\ g = 12$.

Alice alege drept cheie secretă $a = 3$ *şi nonce* $N_A = 11$.
Ea va calcula $g^a = 12^3 = 36\ (mod\ 47)$ *şi trimite lui Bob* $KE_A = (36, 11)$.

Bob alege drept cheie secretă $b = 5$ *şi nonce* $N_B = 7$.
Va calcula $g^b = 12^5 = 14\ (mod\ 47)$ *şi trimite lui Alice* $KE_B = (14, 7)$.

La primirea lui KE_B, *Alice calculează cheia comună* $\left(g^b\right)^a = 14^3 = 18\ (mod\ 47)$.
Similar, Bob primeşte KE_A *şi calculează* $(g^a)^b = 36^5 = 18\ (mod\ 47)$.

Din aceste valori, ambii parteneri determină

$$SKEYSEED = prf(N_A\|N_B, g^{ab}) = prf(117, 18)$$

După ce *Alice* şi *Bob* calculează valoarea comună $SKEYSEED$, obţin din ea alte şapte chei:

[2] În terminologia $IKEv2$, a este înlocuit cu i (de la *Initiator*), iar b cu r (de la *Responder*).

- SK_d pentru derivarea cheilor noi folosite în $child - SA$.
- SK_{aA} şi SK_{aB} pentru protejarea integrităţii.
- $SK_{eA},\ SK_{eB}$ pentru criptare şi decriptare.
- $SK_{pA},\ SK_{pB}$ pentru autentificare.

Derivatele lui $SKEYSEED$ sunt calculate astfel:

$$\{SK_d \| SK_{aA} \| SK_{aB} \| SK_{eA} \| SK_{pA} \| SK_{pB}\} = prf^+(SKEYSEED, N_A \| N_B \| SPI_A \| SPI_B)$$

unde

- prf este o funcţie generatoare de numere pseudoaleatoare; de exemplu $PRF - AES - XCBC - PRF - 128$ (descrisă de standardul RFC 3664).
- $SPI_A,\ SPI_B$ sunt parametri de securitate indexaţi de *Alice* respectiv *Bob*.
- $prf^+(K, S) = T_1 \| T_2 \| \ldots \| T_n$, cu

$$T_1 = prf(SKEYSEED, N_A \| N_B \| SPI_A \| SPI_B \| 0x01),$$
$$T_i = prf(SKEYSEED, T_{i-1} \| N_A \| N_B \| SPI_A \| SPI_B \| 0x0i), \quad i > 1$$

  n este ales suficient de mare astfel ca să se acopere toţi biţii necesari definirii celor 7 chei noi.

Observaţia 6.1.

1. *Fiecare direcţie de trafic foloseşte chei diferite; deci SK_{eA} şi SK_{aA} vor proteja mesajele trimise de Alice, iar SK_{eB} şi SK_{aB} – pe cele trimise de Bob.*
2. *Deoarece $N_A,\ N_B$ sunt folosite drept chei în prf, aceste nonce-uri trebuie să constituie cel puţin jumătate din mărimea cheii din prf-ul negociat.*

6.5.5 Generarea componentelor cheii pentru $child - SA$

Dacă în $CREATE - child - SA$ se generează un nou $child - SA$, componentele cheii sunt generate astfel:

$$KEYCOMP = prf^+(SK_d, N_A \| N_B)$$

sau

$$KEYCOMP = prf^+(SK_d, g^{ab} \| N_A \| N_B)$$

În primul caz, N_A şi N_B sunt nonce-urile generate în comunicarea $CREATE - child - SA$. În al doilea caz, apare şi cheia partajată g^{ab} obţinută din acelaşi schimb de informaţii, componentă legată de Diffie-Hellman.

6.5.6 Integritatea şi autentificarea în IKE

Pentru autentificare, $IKEv2$ foloseşte algoritmi de semnătură digitală, partajare de secrete şi metode EAP (definite în RFC 3748).

Pentru autentifcarea datelor curente, $AUTH$ dispune de două informaţii: metoda de autentificare folosită şi datele de autentificat. $IKEv2$ foloseşte drept metode de autentificare: semnătura digitală RSA, semnătura digitală DSS şi codul de integritate a mesajelor bazat pe partajare.

Pentru a detalia, să revedem Paşii $1-4$ din $IKE-INIT$ şi $IKE-AUTH$.

1. La Pasul 3, datele de autentificat – pe care *Alice* le semnează în $AUTH$ – includ mesajul trimis la Pasul 1, completat cu N_B şi cu valoarea $prf(SK_{pB}, ID_{BB})$. Aceste ultime două valori nu au fost trimise explicit ci erau incluse în mesajul semnat $AUTH$. ID_{BB} este ID-ul datelor curente ale lui *Bob*, din care s-a eliminat antetul.

2. Similar, la Pasul 4, datele de autentificat – pe care *Bob* le semnează în $AUTH$ – includ mesajul trimis la Pasul 2, completat cu N_A şi cu valoarea $prf(SK_{pB}, ID_{AA})$. Aceste ultime două valori nu au fost trimise explicit ci erau incluse în mesajul semnat. ID_{AA} este ID-ul datelor curente ale lui *Alice*, din care s-a eliminat antetul.

Mesajele din $IKE-AUTH$ pot include un certificat sau o autoritate de certificare CA care legalizează semnăturile digitale folosite de parteneri. $IKEv2$ permite lui *Alice* să indice autorităţile de certificare şi tipurile de certificate pe care le foloseşte ($X.509$, $PKCS\ \#7$, PGP, $DNS\ SIG$ etc). După selectarea unui CA, protocolul acceptă autentificările acestuia asupra metodelor folosite.

Dacă *Alice* preferă o autentificare extinsă, ea nu va include la Pasul 3 datele $AUTH$. Atunci *Bob* va include – la Pasul 4 – un câmp EAP şi trimite SA_B2, TS_A, TS_B până când autentificarea este completă.

În această variantă, $IKE-INIT$ şi $IKE-AUTH$ vor avea forma următoare:

1. $Alice \longrightarrow Bob:\quad HDR\|SA_A1\|KE_A\|N_A.$
2. $Bob \longrightarrow Alice:\quad HDR\|SA_B1\|KE_B\|N_B\|[CERTREQ]$
3. $Alice \longrightarrow Bob:\quad HDR\|SK\{ID_A, [CERTREQ], [ID_B], SA_A2, TS_A, TS_B\}$
4. $Bob \longrightarrow Alice:\quad HDR\|SK\{ID_B, [CERT], AUTH, EAP\}$
5. $Alice \longrightarrow Bob:\quad HDR\|SK\{EAP\}$
6. $Bob \longrightarrow Alice:\quad HDR\|SK\{EAP\ (succes)\}$
7. $Alice \longrightarrow Bob:\quad HDR\|SK\{AUTH\}$
8. $Bob \longrightarrow Alice:\quad HDR\|SK\{AUTH, SA_B2, TS_A, TS_B\}$

6.5.7 Grupul de descriptori Diffie-Hellman

Schimbul de chei Diffie-Hellman este folosit în IKE pentru a genera componentele cheilor. După închiderea conexiunii, ambii parteneri "uită" nu numai cheile folosite, dar şi secretele care au stat la baza calculelor acestora.
IKE foloseşte trei reprezentări de grupuri: de exponenţiere modulară (numite $MODP$), grupuri pe curbe eliptice peste $GF(2^n)$ (numite $EC2N$) şi grupuri pe curbe eliptice peste $GP[P]$ (numite ECP). Pentru fiecare reprezentare sunt posibile diverse implementări, în funcţie de parametrii selectaţi.

Standardele pentru $IKEv1$ şi $IKEv2$ specifică următoarele grupuri:

- **Grup** 2: Grup de exponenţiere modulară cu un modul de 1024 biţi.

  Este definit de:

  - generatorul $g = 2$,

  - modulul $p = 2^{1536} - 2^{1472} - 1 + 2^{64} \cdot \{[2^{1406}\pi] + 741804\}$ (primalitatea lui a fost demonstrată riguros).

Valoarea în hexazecimal a acestui număr este:

$FFFFFFFF$	$FFFFFFFF$	$C90FDAA2$	$2168C234$	$C4C6628B$	$80DC1CD1$
$29024E08$	$8A67CC74$	$020BBEA6$	$3B139B22$	$514A0879$	$8E3404DD$
$EF9519B3$	$CD3A431B$	$302B0A6D$	$F25F1437$	$4FE1356D$	$6D51C245$
$E485B576$	$625E7EC6$	$F44C42E9$	$A637ED6B$	$0BFF5CB6$	$F406B7ED$
$EE386BFB$	$5A899FA5$	$AE9F2411$	$7C4B1FE6$	49286651	$ECE45B3D$
$C2007CB8$	$A163BF05$	$98DA4836$	$1C55D39A$	$69163FA8$	$FD24CF5F$
$83655D23$	$DCA3AD96$	$1C62F356$	$208552BB$	$9ED52907$	$7096966D$
$670C354E$	$4ABC9804$	$F1746C08$	$CA237327$	$FFFFFFFF$	$FFFFFFFF$

- **Grup** 14: Grup de exponenţiere modulară cu un modul de 2048 biţi.

- **Grup** 3: Grup pe o curbă eliptică peste $GF[2^{155}]$.

  Curba eliptică este bazată pe extensia Galois $GF[2^{155}]$ generată de polinomul $p(X) = X^{155} + X^{62} + 1$.

  Ecuaţia curbei este $y^2 + xy = x^3 + 471951$, iar generatorul grupului este punctul $P = (x, y) = (123, 456)$.

  Ordinul acestui grup este 4567192616659071619386556591434463519676923731 6 care se descompune în factorii primi

$$2^2 \cdot 3 \cdot 380599384721589301615546382619538626639743644 3$$

- **Grup** 4: Grup pe o curbă eliptică peste $GF[2^{185}]$.

Standardele ulterioare (RFC 3526) specifică grupuri Diffie-Hellman mai puternice, echivalente ca nivel de securitate cu sistemul de criptare simetric AES.

Exemplul 6.4. *O criptare AES pe* 128 *biţi este echivalentă cu un grup cu exponenţiere modulară pe* 3200 *biţi, iar AES pe* 192 *şi* 256 *biţi necesită – pentru securitate echivalentă – grupuri de înmulţire modulară pe* 8000 *respectiv* 15400 *biţi.*

Aceste standarde sunt:

- **Grup** 5: Grup de exponenţiere modulară cu un modul de 1536 biţi.
- **Grup** 15: Grup de exponenţiere modulară cu un modul de 3072 biţi.
- **Grup** 16: Grup de exponenţiere modulară cu un modul de 4096 biţi.
- **Grup** 17: Grup de exponenţiere modulară cu un modul de 6144 biţi.
- **Grup** 18: Grup de exponenţiere modulară cu un modul de 8192 biţi.

6.6 Schimbul de chei $IKEv1$

După cum s-a văzut, sistemul de gestiune al cheilor reprezintă o componentă fundamentală a $IPSec$-ului. Reamintim, pentru o sesiune de comunicări între *Alice* şi *Bob* sunt necesare patru chei.
În general există două tipuri de management al cheilor:

- *Manual*: un administrator de sistem configurează manual fiecare sistem cu fiecare cheie personală şi cu cheile altor sisteme de comunicare;
- *Automat*: un sistem automat permite – la cerere – crearea de chei pentru SA şi facilitează folosirea cheilor într-un sistem larg distribuit, cu o configuraţie evolutivă.

Managementul de distribuţie automată de chei folosit în $IKEv1$ (standard RFC 2408) este $ISAKMP/Oakley$ şi se referă numai la protocolul Diffie-Hellman. El constă din:

1. *Oakley Key Determination Protocol.*
2. *Internet Security Associations and Key Management Protocol* ($ISAKMP$):
   $ISAKMP$ furnizează un cadru pentru schimbul de chei prin intermediul reţelei Internet şi furnizează protocolul de suport specific, incluzând formatele, negocierea atributelor de securitate etc.

   $ISAKMP$ în sine nu alocă un anumit algoritm de generare a cheilor; mai degrabă, el constă dintr-un set de tipuri de mesaje care permite folosirea unui set de algoritmi de schimb de chei. *Oakley* este algoritmul de schimb de chei folosit de versiunea iniţială a $ISAKMP$.

6.6.1 Protocolul *Oakley* de determinare a cheilor

Oakley este un protocol de schimb de chei bazat pe algoritmul Diffie-Hellman.

Algoritmul Diffie-Hellman are două caracteristici atractive:

1. Cheile secrete sunt generate numai dacă şi când este necesar. Nu este nevoie să se păstreze cheile secrete o perioada lungă de timp, expunându-le la vulnerabilităţi ridicate;

2. Schimbul de chei nu necesită nici un fel de infrastructură pre-existentă, înafara unui acord asupra parametrilor globali.

În general se admite că există unele slăbiciuni ale protocolului Diffie-Hellman:

- Nu furnizează nici un fel de informaţie despre identităţile parţilor.

- Nu rezistă unui atac man-in-the-middle (lansat de *Oscar*) de tipul[3]:

  1. *Bob* trimite cheia publică e_B într-un mesaj adresat lui *Alice*;
  2. *Oscar* interceptează acest mesaj, salvează cheia publică şi trimite lui *Alice* un mesaj care conţine ID-ul lui *Bob* şi cheia publică e_O a lui *Oscar*. Mesajul este trimis în aşa fel încât apare ca şi cum ar fi fost trimis de *Bob*. *Alice* primeşte acest mesaj şi păstrează cheia publică a lui *Oscar* cu ID-ul lui *Bob*.
     Similar, *Oscar* reţine cheia publică e_A a lui *Alice* şi trimite lui *Bob* un mesaj cu cheia publică e_O şi ID-ul lui *Alice*.
  3. *Bob* calculează cheia secretă $K_1 = g^{e_B e_O}$, *Alice* calculează cheia secretă $K_2 = g^{e_A e_O}$. *Oscar* – dispunând de cheile publice e_A, e_B calculează ambele chei de sesiune K_1 şi K_2.
  4. De acum *Oscar* este capabil să transporte mesaje de la *Alice* la *Bob* şi invers, schimbându-le modul de criptare pe traseu, în aşa fel încât nici *Alice* şi nici *Bob* nu vor şti că informaţiile lor sunt citite de altcineva.

- Este un calcul intensiv. Prin urmare, este vulnerabil la un atac *prin colmatare (clogging)*, în care un oponent cere un număr foarte mare de chei. Victima consumă resurse de calcul considerabile (făcând exponenţieri modulare) în locul unei activităţi reale.

Prin structura sa, *Oakley* este proiectat să reţină avantajele lui Diffie-Hellman şi să combată slăbiciunile sale.

[3]Protocolul este o variantă a atacului man-in-the middle pentru chei publice, prezentat în [1], pag. 132.

Caracteristicile algoritmului *Oakley*

Algoritmul *Oakley* prezintă cinci proprietăţi importante:

1. Are un mecanism cunoscut sub numele de *cookie* pentru contracararea atacurilor prin colmatare (valorile N_A, N_B din $IKE - SA - INIT$).

2. Permite celor două părţi să negocieze o structură de grup, specificând parametrii globali ai algoritmului de schimb de chei Diffie-Hellman.

3. Foloseşte un nonce pentru a elimina atacurile prin replay.

4. Permite schimbul de valori de chei pentru Diffie-Hellman.

5. Autentifică schimbul de chei Diffie-Hellman pentru a elimina atacurile man-in-the-middle.

Exemplul 6.5. *Într-un atac prin colmatare, Oscar află adresa sursă a unui user legitim şi îi trimite cheia publică Diffie Hellman. Victima calculează o exponenţiere modulară pentru calcularea cheii secrete. Valorile N_A, N_B (cookie) din mesajele $IKE-SA-INIT$ asigură însă un singur calcul pentru $SKEYSEED$.*

Oakley foloseşte pentru schimbul de chei Diffie-Hellman următoarele grupuri:

- Exponenţiere modulară pe 768 biţi
  $p = 2^{768} - 2^{704} - 1 + 2^{64} \cdot \lfloor 2^{638} \cdot \pi \rfloor + 149686$
  $g = 2$.

- Exponenţiere modulară pe 1024 biţi
  $p = 2^{1024} - 2^{960} - 1 + 2^{64} \cdot \lfloor 2^{894} \cdot \pi \rfloor + 129093$
  $g = 2$

- Exponenţiere modulară pe 1536 biţi

- Grup aditiv pe curbe eliptice peste $GF(2^{155})$:
  - Generator (hexadecimal): $(X, Y) = (7B, 1C8)$
  - Curbă (hexadecimal): $y^2 = x^3 + 7338$.

- Grup aditiv pe curbe eliptice peste $GF(2^{185})$:
  - Generator (hexadecimal): $(X, Y) = (18, D)$
  - Curbă (hexadecimal): $y^2 = x^3 + 1EE9$.

Primele trei grupuri implementează algoritmul Diffie-Hellman folosind exponenţiere modulară, iar ultimele două grupuri aplică Diffie-Hellman pe curbe eliptice.

Oakley foloseşte un număr prim împotriva atacurilor prin replay. Fiecare număr este un pseudoaleator generat local. Numerele apar în răspunsuri şi sunt criptate de-a lungul anumitor porţiuni a schimbului de mesaje.

Pentru autentificare, *Oakley* foloseşte trei metode distincte (pentru fiecare existând diverse variante):

1. **Semnături digitale**: Schimbul de chei este autentificat prin semnarea unei amprente; fiecare entitate criptează amprenta cu cheia sa privată. Amprenta este generată peste parametri importanţi ai comunicării, cum ar fi ID User şi nonce-uri.

2. **Criptare cu cheie publică**: Schimbul este autentificat prin criptarea parametrilor (ID-uri, nonce-uri) cu cheia privată a lui *Alice*

3. **Criptare cu cheie simetrică**: O cheie generată de un mecanism extern poate fi folosită pentru autentificare schimbului de mesaje, prin criptarea simetrică a schimbului de parametri.

Exemplu de schimb de chei *Oakley*

Prezentăm schimbul de chei "în specificaţii" (varianta "agresivă") – numit astfel datorită celor trei mesaje care sunt schimbate între parteneri.

1. $Alice \longrightarrow Bob: \quad CK_A, OK-KEY, GRP, g^a, EHAO, NIDP, ID_A, ID_B, N_A, S_{K_A}(ID_A \| ID_B \| N_A \| GRP \| g^a \| EHAO)$

2. $Alice \longleftarrow Bob: \quad CK_B, CKY_A, OK-KEY, GRP, g^b, EHAS, NIDP, ID_B, ID_A, N_B, N_A, S_{K_B}\left(ID_B \| ID_A \| N_B \| N_A \| GRP \| g^b \| g^a \| EHAS\right)$

3. $Alice \longrightarrow Bob: \quad CK_A, CKY_B, OK-KEY, GRP, g^a, EHAS, NIDP, ID_A, ID_B, N_A, N_B, S_{K_B}\left(ID_A \| ID_B \| N_A \| N_B \| GRP \| g^a \| g^b \| EHAS\right)$

unde

- CK_A, CK_B – Cooky-urile celor doi parteneri (pachete de date care conţin informaţii despre SA şi schimbul curent de chei);
- CKY_A, CKY_B – Ecouri ale acestor cookie;
- $OK-KEY$ – Tipul mesajului de schimb de chei;
- GRP – Tipul grupului peste care este definit protocolul Diffie-Hellman;

- g^a, g^b – Cheile publice ale lui *Alice* respectiv *Bob*;
- $EHAO$, $EHAS$ – Pachete cu funcţii de criptare, dispersie şi autentificare propuse, respectiv selectate;
- $NIDP$ – Indică faptul că restul mesajului nu este criptat;
- ID_A, ID_B – ID-urile lui *Alice* şi *Bob*;
- N_A, N_B – nonce-uri generate aleator de *Alice* respectiv *Bob*;
- $S_{K_A}(\alpha), S_{K_B}(\alpha)$ – semnătura mesajului α de către *Alice* respectiv *Bob*, folosind cheia privată K_A (K_B).

Comentarii:

1. La primul pas *Alice* transmite un cookie, grupul care va fi folosit şi cheia sa publică Diffie-Hellman. *Alice* mai indică funcţia de dispersie şi algoritmii de autentificare folosiţi în acest schimb.

   Totodată ea include în mesaj şi identificatorii pentru *Alice* şi *Bob* şi nonce-ul ei pentru acest schimb.
   În final, *Alice* adaugă o semnătură folosind cheia sa privată, prin care semnează cei doi identificatori, nonce-ul, grupul cheia publică Diffie-Hellman şi algoritmii respectivi.

2. Când *Bob* primeşte mesajul, el verifică semnătura folosind cheia publică de semnătură a lui *Alice*. Dacă totul este în regulă, *Bob* răspunde cu ecoul unui cookie, identificatorul lui *Alice*, un nonce şi grupul agreat. *Bob* include de asemenea în mesaj propriul cookie şi identificator, cheia sa publică Diffie-Hellman, algoritmii selectaţi şi propriul nonce. În final, *Bob* adaugă o semnătură folosind cheia sa privată, care semnează cei doi identificatori, cele două nonce-uri, grupul, cheile publice Diffie-Hellman şi algoritmii selectaţi.

3. Când *Alice* primeşte al doilea mesaj, ea verifică semnătura folosind cheia publică a lui *Bob*.

   Valorile numerice din acest mesaj asigură că nu este un replay al unui mesaj mai vechi.

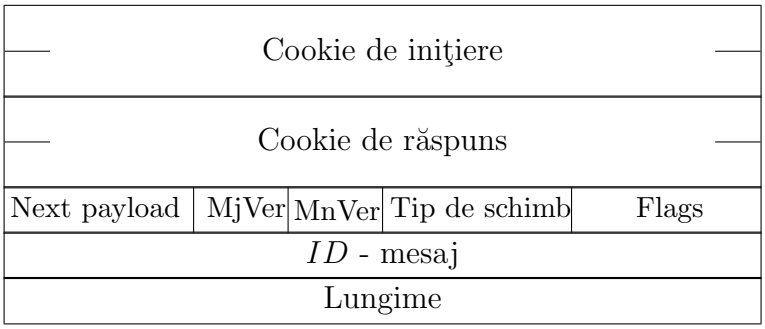
   Pentru ca schimbul să fie complet, *Alice* trebuie să trimită un mesaj lui *Bob*, pentru a certifica faptul că a primit cheia publică.

6.6.2 *ISAKMP*

ISAKMP defineşte proceduri şi formate de pachete de date pentru a stabili, negocia, modifica şi şterge asocieri de securitate.
Fiind o componentă a stabilirii unui *SA*, *ISAKMP* defineşte pachete de date curente pentru schimbul de chei generate şi de autentificare a datelor.

Formatul Antetului *ISAKMP*

Un mesaj *ISAKMP* este compus dintr-un antet *ISAKMP* urmat de unul sau mai multe date curente, toate încapsulate într-un protocol de transport.



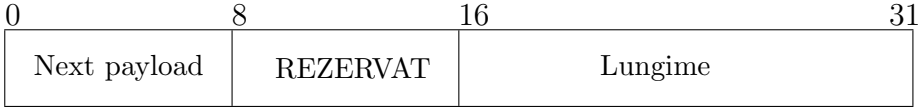
Câmpurile acestui antet *ISAKMP* sunt:

- *Cookie de iniţiere* (64 biţi): reprezintă cookie-ul lui *Alice* (care iniţiază, notifică sau elimină un *SA*).
- *Cookie de răspuns* (64 biţi): cookie-ul lui *Bob*, care răspunde; în primul mesaj trimis de *Alice*, el este "null".
- *Next Payload* (8 biţi): indică tipul primului pachet de date curente din mesaj.
- *MjVer* (4 biţi): indică versiunea curentă de *ISAKMP*.
- *MnVer* (4 biţi): indică versiunea secundară de folosire *ISAKMP*.
- *Tip de schimb* (8 biţi): indică tipul de schimb.
- *Flags* (8 biţi): indică opţiunile specifice acestui schimb de mesaje *ISAKMP*. Cel puţin doi biţi sunt deja definiţi: bitul de criptare – este setat dacă toate datele curente de după antete sunt criptate folosind unul din algoritmii de criptare asociaţi acestui *SA*, şi bitul de asociere – folosit pentru a asigura că mesajul criptat nu este primit înaintea stabilirii unui *SA*.
- *ID - mesaj* (32 biţi): un *ID* unic pentru acest mesaj.

- *Lungime* (32 biţi): lungimea totală a mesajului (antet plus toate datele curente asociate) în octeţi.

Tipuri de date curente

Toate datele curente din $ISAKMP$ încep cu acelaşi antet generic:



- *Next Payload*: are valoarea 0 dacă acesta este ultimul set de date curente din mesaj; altfel valoarea sa este tipul următorului set de date curente.
- *Lungime*: indică lungimea – în octeţi – a setului de date curente, inclusiv antetul generic.

Sunt posibile următoarele tipuri de date curente:

1. **Date** SA (SA): folosit pentru începerea negocierii unei asocieri de securitate. Drept parametri se folosesc: "Domeniu de interpretatare" (DOI) sub care are loc negocierea, şi *Situaţia* în care are loc negocierea.
2. **Date Propuse** (P): conţine informaţii folosite în timpul negocierii SA-ului. Setul de date curente indică protocolul pentru acest SA (ESP sau AH) pentru care sunt negociate serviciile şi mecanismele. Aici mai sunt incluse SPI (identitatea lui *Alice*) şi numărul de transformări. Fiecare transformare este conţinută într-un set de date curente de transformare.
3. **Date de transformare** (T): defineşte o transformare de securitate folosită pentru securizarea canalului de comunicaţii. Parametrii folosiţi sunt: "*Transform Number*" – care serveşte pentru identificarea acestui set de date, astfel ca *Bob* să îl poată folosi pentru a identifica acceptarea acestei transformari; "*Transform - ID*" şi "*SA - Attributes*" – identifică o transformare specifică (de exemplu $3DES$ pentru ESP, $HMAC - SHA - 1 - 96$ pentru AH) împreună cu atributele asociate (de exemplu lungimea amprentei).
4. **Date pentru schimb de chei** (KE): este utilizat pentru o paletă largă de tehnici de schimb de chei, inclusiv protocoalele *Oakley*, Diffie-Hellman şi RSA. Are un singur parametru ("*Key Exchange data*") care conţine datele necesare pentru generarea unei chei de sesiune şi este dependent de cheia algoritmului de schimb utilizat.
5. **Date de identificare** (ID): este folosit pentru a determina identitatea comunicării şi – eventual – pentru autentificarea informaţiei. Are doi parametri: "ID - *Type*" şi "ID - *Data*" (de obicei aici este o adresă $IPv4$ sau $IPv6$).

6. **Certificat** ($CERT$): specifică un certificat de cheie publică. Are doi parametri, "*Certificate Data*" şi "*Certificate Encoding*. Ultimul indică tipul certificatului şi conţine elemente din lista:
   - $PKCS\#7$ împachetat cu certificatul $X.509$;
   - Certificat PGP;
   - Cheie de semnătură DNS;
   - Certificat de semnătură $X.509$;
   - Certificat de schimb de chei $X.509$;
   - Token-uri Kerberos;
   - CRL (Lista certificatelor revocate);
   - ARL (Lista autorităţilor revocate);
   - Certificatul $SPKI$
7. **Cerere de certificare** (CR): Într-un schimb $ISAKMP$ *Alice* poate include în orice moment o solicitare a certificatului lui *Bob* sau al altei entităţi de comunicare.
8. **Date de dispersie** ($HASH$): conţine amprente generate de o funcţie de dispersie peste unele părţi ale mesajului şi/sau stări $ISAKMP$. Acest tip de date curente este folosit pentru verificarea integrităţii datelor din mesaj şi este util în special pentru serviciile de non-repudiere.
9. **Date de semnătură** (SIG): conţine rezultatul unei funcţii de semnătură digitală.
10. **Date de notificare** (N): conţine mesaje de eroare sau informaţii despre statusul asociat SA-ului respectiv sau SA-ului de negociere. $ISAKMP$ admite următoarele mesaje status: *Connected, Responder-Lifetime* (durata de viată a SA-ului, decisă de *Bob*), *Replay-status* (pentru confirmări pozitive din partea lui *Bob*), *Initial-Contact* (pentru informarea partenerului că acesta este primul SA stabilit cu sistemul).
11. **Date de şters** (D): indică SA-uri pe care *Alice* le-a şters din baza sa de date (şi deci nu mai sunt valabile).

Comunicaţii $ISAKMP$

$ISAKMP$ oferă un cadru pentru schimbul de mesaje, cu tipurile de date curente servind ca blocuri de construcţie. Conform specificaţiilor sunt posibile cinci tipuri de schimburi:

1. **Base Exchange**: permite schimbul de chei şi autentificarea datelor curente care vor fi transmise împreună. Acest lucru minimizează numărul schimburilor de mesaje.

(1)	*Alice*	$\longrightarrow$	*Bob* :	$SA; Nonce$	Începe negocierea $ISAKMP\ SA$.
(2)	*Bob*	$\longrightarrow$	*Alice* :	$SA; Nonce$	Se cade de acord asupra bazei SA.
(3)	*Alice*	$\longrightarrow$	*Bob* :	$KE; ID_A; AUTH$	Se trimite cheia generată.
					Bob verifică identitatea lui *Alice*.
(4)	*Bob*	$\longrightarrow$	*Alice* :	$KE; ID_B; AUTH$	*Alice* verifică identitatea lui *Bob*
					Se încheie stabilirea SA.

Primele două mesaje furnizează cookie-uri şi stabilesc tipul de protocol SA; ambele părţi folosesc un nonce pentru a se proteja împotriva atacurilor prin replay. Ultimele două mesaje stabilesc cheia de criptare şi verifică autenticitatea userilor, folosind un protocol de autentificare bazat pe ID-uri şi informaţiile din primele două mesaje.

2. **Identity Protection Exchange**: este o variantă extinsă a tipului anterior, pentru a întări protejarea identităţilor celor doi utilizatori [4].

(1)	*Alice*	$\longrightarrow$	*Bob* :	SA	Începe negocierea $ISAKMP\ SA$.
(2)	*Bob*	$\longrightarrow$	*Alice* :	SA	Se cade de acord asupra bazei SA.
(3)	*Alice*	$\longrightarrow$	*Bob* :	$KE; Nonce$	Se trimite cheia generată.
(4)	*Bob*	$\longrightarrow$	*Alice* :	$KE; Nonce$	Se trimite cheia generată.
(5)$*$	*Alice*	$\longrightarrow$	*Bob* :	$ID_A; Auth$	*Bob* verifică identitatea lui *Alice*.
(6)$*$	*Bob*	$\longrightarrow$	*Alice* :	$ID_B; Auth$	*Alice* verifică identitatea lui *Bob*.

Primele două mesaje stabilesc asocierea de secruitate. Următoarele două mesaje asigură schimbul de chei, folosind şi nonce-uri, pentru protecţie la atacul prin răspuns. După ce cheia de sesiune a fost calculată, cei doi parteneri schimbă mesaje criptate care conţin informaţii de autentificare (semnături digitale, certificate de validare a cheilor publice etc).

3. **Authentication Only Exchange**: este utilizat pentru a realiza o autentificare uniformă, fără schimburi de chei.

(1)	*Alice*	$\longrightarrow$	*Bob* :	$SA; Nonce$	Începe negocierea $ISAKMP\ SA$.
(2)	*Bob*	$\longrightarrow$	*Alice* :	$SA; Nonce; ID_B;$	Se cade de acord asupra bazei SA.
				$AUTH$	*Alice* verifică identitatea lui *Bob*
(3)	*Alice*	$\longrightarrow$	*Bob* :	$ID_A; AUTH$	Se încheie stabilirea SA.
					Bob verifică identitatea lui *Alice*.

Primele două mesaje stabilesc SA-ul. În plus, *Bob* foloseşte al doilea mesaj pentru a transmite ID-ul său şi un nonce poentru autentificare. *Alice* trimite un al treilea mesaj pentru a transmite prin ID propria sa autentficare.

4. **Aggressive exchange**: minimizează numărul schimburilor de mesaje, cu preţul nefurnizării protecţiei identităţii.

[4] Notaţia $*$ semnifică faptul că datele curente sunt criptate după antetul $ISAKMP$.

(1)	*Alice*	$\longrightarrow$	*Bob* :	$SA; KE; Nonce; ID_A$	Începe negocierea $ISAKMP\ SA$ şi schimbul de chei.
(2)	*Bob*	$\longrightarrow$	*Alice* :	$SA; KE; Nonce; ID_B; AUTH$	Se cade de acord asupra bazei SA. *Alice* verifică identitatea lui *Bob*
(3)$*$	*Alice*	$\longrightarrow$	*Bob* :	$AUTH$	Se încheie stabilirea SA. *Bob* verifică identitatea lui *Alice*.

În primul mesaj, *Alice* propune un SA împreună cu protocolul asociat şi opţiunile de transformare; în paralel ea lansează schimbul de chei şi furnizează propriul ID.

În al doilea mesaj, *Bob* indică acceptul SA-ului cu un anumit protocol şi transformare, completează schimbul de chei şi autentifică informaţia primită.

Cu al treilea mesaj, *Alice* transmite un rezultat autentificat care acoperă informaţiile anterioare; acest mesaj este criptat folosind cheia secretă comună de sesiune.

5. **Information Exchange**: conţine o transmitere one-way a informaţiei pentru managementul SA.

(1)$*$	*Alice*	$\longrightarrow$	*Bob*	N/D :	Anunţă o eroare, o notificare de status sau o ştergere.

6.7 Analiza $IPSec$

6.7.1 Performanţe $IPSec$

Atunci când se foloseşte $IPSec$, dimensiunea pachetului IP creşte din cauza adăugării antetelor specifice (ESP, AH, noul antet IP în cazul modului tunel). Acest lucru duce la creşterea raportului între dimensiunea antetelor şi cea a datelor, reducând banda efectivă de comunicare.

În plus, timpul necesar pentru construcţia antetelor şi aplicarea algoritmilor criptografici conduce la o întârziere suplimentară în transmisia pachetelor.

Acest dezavantaj devine supărător mai ales când capacităţile de procesare sunt mici.

6.7.2 Complexitate $IPSec$

- $IPSec$ prezintă dezavantajul unei complexităţi extrem de mari; este preţul plătit pentru numărul mare de opţiuni şi de flexibilitatea sa mult crescută.
  Complexitatea reprezintă un duşman al securităţii, din mai multe puncte de vedere. Algoritmii complecşi conduc la erori greu de remediat. În plus, un sistem complex este mult mai dificil şi mai costisitor de realizat şi întreţinut.

- De multe ori există mai multe posibilităţi de a implementa facilităţi identice sau similare.

- Definirea standardului actual $IPSec$ de către un comitet nu s-a dovedit a fi o alegere prea fericită. De aceea, pentru alegerea următoarelor standarde de securitate s-a ales varianta unui concurs, la care să participe diverse colective.

- În mod normal, un soft este realizat în urma unei metodologii de tip "încearcă-şi-repară". Detaliind, mici bucăţi de cod sunt implementate, testate şi – în cazul în care apar erori – fixate din nou. Aceste componente sunt apoi combinate în module mai mari, cărora la rândul lor li se aplică acelaşi tratament. Rezultatul constituie de multe ori un produs mai puţin funcţional decât cel aşteptat iniţial.
  Acest mod de construcţie are un efect devastator asupra sistemelor de securitate. Motivul principal este acela că securitatea unui sistem nu poate fi testată prea uşor, iar dacă părţile sistemului sunt considerate ca fiind independente, analiza este şi mai dificilă.

Singura modalitate de a testa securitatea unui sistem este aceea de a realiza analize care necesită mult timp şi efort. Dacă se consideră un sistem cu n opţiuni diferite, fiecare cu 2 posibilităţi de alegere, atunci sunt $n(n-1)/2 = \mathcal{O}(n^2)$ perechi diferite de opţiuni care pot interacţiona în moduri diferite şi 2^n configuraţii posibile. Fiecare dintre acestea este posibil să conducă la o falie de securitate. Se aşteaptă deci ca numărul de puncte slabe în securitate să crească foarte rapid odată cu complexitatea.

În concluzie, sistemele de securitate trebuie construite cât mai simplu cu putinţă. $IPSec$ este mult prea complex pentru a fi considerat cu adevărat sigur.

6.7.3 Documentaţie $IPSec$

Documentaţia $IPSec$ este stufoasă şi foarte dificil de înţeles. Ca o părere generală, înţelegerea $IPSec$ din documentaţie este practic imposibilă, părţi ale acesteia fiind foarte greu de citit.

O lipsă majoră a documentaţiei este nespecificarea clară a scopurilor $IPSec$. Deşi unele dintre acestea sunt destul de evidente, de multe ori administratorii sunt lăsaţi să le deducă singuri. Astfel, un designer care doreşte să utilizeze $IPSec$ şi nu cunoaşte exact funcţionalităţile sale, poate realiza un sistem care să nu atingă nivelul de securitate considerat.

Afirmaţia că $IPSec$ oferă securitate la nivel IP poate conduce la o folosire neadecvată a sa. Se poate considera – în mod eronat – că el este util ca sistem de securitate la nivel aplicaţie (de exemplu la autentificarea unui user pentru citirea e-mail-ului). $IPSec$ autentifică pachetele ca provenind de la cineva care cunoaşte o anumită cheie, chiar dacă se poate crede că autentifică o anumită adresă IP (cum poate fi de exemplu autentificarea prin trecerea printr-un firewall). Astfel de interpretări greşite ale $IPSec$ pot conduce la erori.

6.7.4 Eliminarea funcţionalităţilor duplicate

IPSec are două moduri de operare (*transport* şi *tunel*) şi două protocoale (*AH* şi *ESP*), fapt ce conduce la o complexitate ridicată. Două servere care doresc să comunice pot alege între 4 modalităţi diferite. Diferenţa dintre acestea este minoră, atât din punct de vedere al funcţionalităţii cât şi din punct de vedere al performanţei.

Există propunerea a renunţa la modul transport, fapt care ar conduce eliminarea diferenţelor dintre echipamentele de reţea în host-uri şi porţi de securitate (*security gateways*). Principala diferenţă dintre acestea este că porţile de securitate nu pot opera în modul transport.

Mulţi dintre cei care au analizat *IPSec* sunt de părere că protocolul *AH* poate fi eliminat. Funcţionalităţile oferite de *AH* şi *ESP* sunt similare. În modul transport, *AH* oferă o autentificare mai puternică întrucât autentifică şi câmpuri ale antetului *IP*. Dar – utilizat în mod tunel – *ESP* oferă acelaşi nivel de autentificare.

Protocolul *AH* autentifică indirect şi antetele de la nivelele inferioare, fapt care atrage după sine o multitudine de probleme, fiindcă există câmpuri care se pot modifica pe parcursul drumului de la sursă la destinaţie. Deci protocolul *AH* trebuie să fie conştient de ce se întâmplă la nivelele inferioare, pentru a evita utilizarea acestor câmpuri. Ca o consecinţă, construcţia obţinută este total neelegantă, cu probleme la extensiile viitoare ale protocolului *IP* – care pot adăuga câmpuri pe care *AH* le ignoră.

6.7.5 Utilizarea deficitară

Protocolul *ESP* permite ca atât autentificarea cât şi criptarea să fie opţionale.

Astfel, un administrator de sistem poate să configureze prost *IPSec*: el este conştient că are nevoie de criptare pentru asigurarea confidenţialităţii datelor, deci activează criptarea, însă lasă dezactivată opţiunea de autentificare.

Un alt exemplu de implementare deficitară este legată de alegerea greşită a algoritmilor criptografici. O bună implementare pentru *IPSec* impune ca toţi algoritmii criptografici utilizaţi să permită un nivel adecvat de securitate în toate situaţiile. Algoritmii criptografici buni sunt disponibili şi nu este nici un motiv să se utilizeze algoritmi slabi.

6.7.6 Ordinea operaţiilor de securitate

Atunci când sunt realizate atât autentificarea cât şi criptarea, *IPSec* realizează întâi criptarea şi apoi autentificarea textului criptat. Ordinea este exact inversă faţă de cea normală: mai întâi trebuie realizată autentificarea textului clar, şi apoi se criptează textul autentificat.

Autentificarea textului după criptare are totuşi un avantaj: receptorul şterge pachetele de date care nu se pot autentifica rapid, fără a fi nevoie de decriptarea lor. Acesta poate constitui un avantaj în cazul atacurilor de tip *DoS* (*Denial of Service*), când un număr

mare de pachete neautentificate ar putea ocupa resursele disponibile o perioadă lungă de timp.

6.7.7 Simetrizarea SA-urilor

Un SA (*Security Association*) poate fi folosit numai într-o direcţie; deci pentru o comunicare bidirecţională sunt necesare două SA-uri. Fiecare din ele implementează un singur mod de operare şi un singur protocol. Dacă se folosesc două protocoale (AH şi ESP) pentru un acelaşi pachet, atunci sunt necesare 2 SA-uri. Utilizarea a două protocoale şi a două moduri de operare conduce la creşterea complexităţii SA-urilor.

În practică există foarte puţine situaţii în care traficul este necesar să fie securizat într-o singură direcţie. De aceea, SA-urile sunt negociate în pereche. Pare mult mai logic ca SA-urile să devină bidirecţionale; astfel s-ar înjumătăţii numărul de SA-uri din sistem şi s-ar evita de asemenea configurări asimetrice, care pot fi uneori supărătoare.

6.7.8 SPD (Security Policy Database)

Politicile de securitate sunt stocate în SPD (*Security Police Database*). Pentru fiecare pachet de date transmis, se verifică cum trebuie procesat. SPD-ul poate specifica 3 acţiuni: poate arunca pachetul, poate lăsa pachetul să treacă fără să îi aplice $IPSec$, sau poate să îi aplice $IPSec$.
În ultimul caz se specifică şi ce SA se foloseşte.

Acesta este un mecanism extrem de flexibil, care permite un control la nivelul fiecărui pachet. Depinde de SPD dacă întreg traficul utilizează acelaşi SA sau dacă se generează SA-uri diferite pentru aplicaţii diferite sau pentru fiecare conexiune TCP diferită.
Există un surplus de informaţie, ceea ce oferă destul de multă informaţie unui eventual atacator.

De aceea, se consideră că rolul SPD-ului este strict acela de a determina dacă un pachet trebuie aruncat, dacă trebuie transmis fără $IPSec$, dacă se autentifică sau dacă se autentifică şi criptează. Nu este necesar să se ştie dacă pentru mai multe pachete se utilizează sau nu acelaşi SA.

6.7.9 Incompatibilităţi cu Internet Protocol

S. Kent şi R. Atkinson sugerează ([14]) că există anumite incompatibilităţi între $IPSec$ şi IP. În acest sens se consideră două echipamente situate la o mare distanţă unul de celălalt, care doresc să comunice în modul tunel. Între acestea există o mulţime de routere care nu cunosc existenţa SA-urilor şi care trebuie să transmită pachetele. Orice mesaj de tip $ICMP$ este neautentificat şi necriptat; deci orice astfel de mesaj va fi aruncat, fapt care conduce la pierderea funcţionalităţii IP.

Chiar dacă pe routere se implementează $IPSec$, tot nu se pot autentifica mesajele $ICMP$ decât dacă acestea îşi setează SA cu capetele tunelului, cu scopul trimiterii pachetului $ICMP$. Dar acesta trebuie să fie sigur că primeşte mesaje de la un router veritabil, ceea ce nu poate verifica decât eventual cu PKI, lucru inexistent la acest nivel.

6.7.10 Atacuri

Duplicarea mesajelor

În mod ideal, dacă se transmit mai multe copii ale aceluiaşi mesaj, atunci răspunsurile trebuie să fie identice. Mai mult, sistemul ar trebui să recunoască faptul că primeşte copii identice şi să replice primul răspuns transmis.

O implementare mai puţin atentă poate crea falii de securitate. De exemplu, se transmit mai multe mesaje cu zona de date modificată. Se presupune că implementarea Diffie - Hellman modulo p se realizează peste un subgrup de dimensiune q, unde $q|p-1$ şi $p-1$ are mulţi divizori mici. *Oscar* poate determina cheia secretă prin repetarea datelor KE care conţin elemente de ordin mic şi verificând care din setul de chei posibile este utilizat la criptare. Aceasta îi relevă ordinul elementului trimis ca date KE.
Combinând mai multe rezultate, se determină cheia secretă Diffie - Hellman.

Atacul de tip DoS

Utilizarea elementelor de tip "*cookie*" elimină o parte din atacurile de tip DoS.

Măsura implică creşterea în dificultate a realizării unui astfel de atac, dar nu-l elimină în totalitate. Impactul introducerii acestor mijloace de apărare are ca revers creşterea complexităţii.

6.8 Concluzii

Cu toate criticile care i se aduc, $IPSec$ constituie cel mai bun protocol de securitate existent în momentul de faţă.

Comparat cu alte protocoale de securitate, $IPSec$ oferă multe avantaje din punct de vedere arhitectural şi o foarte mare flexibilitate. Detaliile securităţii reţelei sunt de obicei ascunse aplicaţiilor. În plus, $IPSec$ poate fi transparent şi utilizatorilor finali, eliminând necesitatea instruirii persoanelor în vederea utilizării mecanismelor de securitate.
Un canal sigur poate fi configurat de la un cap la altul (protejând traficul dintre două entităţi), route-to-route (protejând traficul care trece peste o mulţime particulară de legături) etc.

Deci $IPSec$ poate realiza securizarea utilizatorilor individuali (dacă acest lucru este necesar).

Bibliografie

[1] Atanasiu, A. – *Securitatea Informaţiei, vol. 1 (Criptografie)*, ed. InfoData, Cluj 2007

[2] Craig A. Shue, Minaxi G. – *IPSec: Performance Analysis and Enhancements*, *http://www.csiir.ornl.gov/shue/research/icc07.pdf*

[3] Stallings, W. – *Network Security Essentials, Applications and Standards*, Third Edition,Pearson Prentice Hall, 2007

[4] Stallings, W. – *Cryptography and Network Security Principles and Practices*, Fourth Edition, Prentice Hall, 2005

[5] Ferguson, N., Schneier, B. – *A cryptographic Evaluation of IPSec*, $http://www.schneier.com/paper-IPSec.pdf$

[6] Frankel, S., Herbert, H. (2003) – *The AES-XCBC-MAC-96 algorithm and its use with IPSec* (RFC 3566), $http://www.ietf.org/rfc/rfc3566.txt?number=3566$

[7] Gleeson, B., Lin A., Heinanen, J., Armitage, G., Malis, A. (2000) – *A framework for IP based virtual private networks* (RFC 2764), $http://www.ietf.org/rfc/rfc2764.txt?number=2764$

[8] Hoffman, P. (2004) – *The AES-XCBC-PRF-128 algorithm fo the internet key exchange protocol (IKE)* (RFC 3664), $http://www.ietf.org/rfc/rfc3664.txt?number=3664$

[9] Hoffman, P. (2005) – *Cryptographic suites for IPSec* (RFC 4308), $http://www.ietf.org/rfc/rfc4308.txt?number=4308$

[10] Kaufman, C. (Ed.). (2005) – *Internet key exchange (IKEv2)* (RFC 4306), $http://www.ietf.org/rfc/rfc4306.txt?number=4306$

[11] Kent, S. (2005a) – *IP Authentication header* (RFC 4302), $http://www.ietf.org/rfc/rfc4302.txt?number=4302$

[12] Kent, S. (2005b) – *IP Encapsulating security payload (ESP)* (RFC 4303), $http://www.ietf.org/rfc/rfc4303.txt?number=4303$

[13] Kent, S., Seo, K. (2005) – *Security architecture for the Internet protocol* (RFC 4301), $http://www.ietf.org/rfc/rfc4301.txt?number = 4301$

[14] Kent, S., Atkinson, R. – *Security Architecture for Internet Protocol.*

[15] Kivinen, T., Kojo, M. (2003) – *More modular exponential (MODP) Diffie-Hellman Groups for IKE (RFC 3526), http://www.ietf.org/rfc/rfc3526.txt?number=3526*

[16] Perlman, R., Kaufman, C. – *Analysis of the IPSec Key Exchange Standard, http://warlord.nologin.org/papers/IPSec-analysis.pdf*

[17] Orman, H. (1998) – *The OAKLEY key determination protocol (RFC 2412), http://www.ietf.org/rfc/rfc2412.txt?number=2412*

[18] Schiller, J. (2005) – *Cryptographic algorithms for use in the Internet key exchange version 2 (IKEv2) (RFC 4307), http://www.ietf.org/rfc/rfc4307.txt?number=4307*

[19] *An Introduction to IP Security Encryption (IPSec Negotiation/IKE Protocols) Cisco Systems, http://www.cisco.com/en/US/tech/tk583/tk372/technologies_tech_note 09186a0080094203.shtml*

[20] *Red ISAKMP and Oakley Information, http://www.cisco.com/en/US/tech/tk583/tk372/ technologies_tech_note09186a0080093c2b.shtml*

[21] *Group Domain of Interpretation (GDOI), http://en.wikipedia.org/wiki/GDOI*

[22] $ISAKMP$, *http://en.wikipedia.org/wiki/ISAKMP*

[23] *Securing Data in Transit with IPSec, http://www.windowsecurity.com/articles/Securing_Data_in_Transit_with_IPSec.html*

[24] *H3C ItoIP Solutions Expert, http://www.h3c.com/portal/Products_Solutions/ Technology/Security_and_VPN/Technology_Introduction/200701/195610_57_0.htm*

[25] *IPSec, http://cis.poly.edu/ ross/networksecurity/IPSec.ppt*

[26] *An Illustrated Guide To IPSec, http://unixwiz.net/techtips/iguide-IPSec.html*