Capitolul 8

Alte protocoale de securitate

8.1 Introducere

Creşterea volumului de comerţ on-line solicită utilizarea de protocoale care trebuie să asigure o multitudine de operaţii de securitate. Astfel, în desfăsurarea unei tranzacţii, cumpărătorul va trebui să ofere datele cardului său pentru a plăti un produs. De asemenea, vânzătorul trebuie să se asigure că are de-a face cu un cumpărător real căruia îi trimite produsul. Aceste informaţii transmise printr-o reţea nesigură cum este Internetul, generează multă reticenţă şi utilizatorii trebuie să fie siguri că sunt suficient de bine protejaţi.
De aceea, pentru securizarea datelor necesare unei tranzacţii (confidenţialitate, autentificare şi non-repudiere) s-au dezvoltat de-a lungul timpului o serie de protocoale.

Cele mai utilizate în perioada actuală sunt SSL (**S**ecure **S**ocket **L**ayer) construit de Netscape şi standardul Internet al SSL, cunoscut sub numele TLS (**T**ransport **L**ayer **S**ecurity). Ambele sunt implementate pe toate browserele Web, deşi sub forme diferite; deci companiile trebuie să dezvolte aplicaţii SSL distincte pentru Netscape şi Internet Explorer.

În acest moment există două versiuni SSL: SSL 2.0 care suportă doar autentificarea serverului şi SSL 3.1 care autentifcă atât serverul cât şi clientul. Variantele TLS 1.0 şi 1.1 realizează de asemenea ambele autentificări.

TSL şi SSL permit utilizatorilor să-şi definească nivelul de securitate pe care-l doresc. Ambele sunt standarde industriale şi sunt folosite în milioane de tranzacţii Internet. Utilizatorii pot selecta $RC4$, DES, $3DES$ sau AES pentru criptare, $RADIUS$ (username şi password), $RSA\ SecurID$ (username şi token+pin) pentru autentificare, sau pot folosi chiar certificate digitale $X.509$.

O comunicare sigură client-server solicită autentificarea ambelor părţi, un schimb de chei criptografice care conduc la o pre-cheie master agreată de ambii parteneri şi o criptare a datelor bazată pe chei generate din pre-cheia master. Când un client şi un server sunt

de acord să comunice pe baza unui protocol SSL sau TSL, ei trebuie să cadă de acord (prin protocoale de tip handshake[1]) şi asupra altor puncte:

1. Protocolul şi versiunea (TSL 1.0, TSL 1.1, $SSL2$, $SSL3$).
2. Sistemul de criptare.
3. Dacă vor autentificare sau nu.
4. Sistemul de criptare cu cheie publică folosit pentru generarea pre-cheii master.
5. Cum se vor genera cheile de sesiune pentru criptarea mesajelor.

Anvantaje al protocoalelor SSL şi TLS:

- Asigura o confidenţialitate şi o integritate a datelor convenabilă.
- Au abilitatea de a negocia chei de criptare unice, între parteneri care nu au comunicat anterior.
- Se pot aplica independent, toate deciziile privind detaliile de aplicare fiind luate in mod amiabil (handshacking).

8.2 TSL

Un protocol TLS este format din două etape, privite ca o stivă cu două straturi (layers): protocolul TLS de înregistrare şi cel de handshaking.

Protocolul TLS de înregistrare ia mesajele care trebuie transmise, fragmente de date din diverse blocuri interne, le arhivează (opţional), aplică o funcţie MAC, le criptează şi transmite rezultatul.

Protocolul handshaking permite serverului şi clientului să se autentifice reciproc şi să negocieze un algoritm de criptare şi o cheie de criptare, înainte de a se transmite/primi prinul bit de date.

În TLS este stabilită întâi o sesiune între client şi server, iar apoi se realizează o conexiune.

Definiţia 8.1. *O "sesiune" TLS este o asociere între un client şi un server, cu scopul de a defini un set de parametri de securitate pentru o perioadă mai lungă. Sesiunile sunt create prin protocoale de tip handshaking, scopul lor fiind de a evita negocieri lungi pentru stabilirea parametrilor de securitate la fiecare conexiune.*

[1]Un protocol handshake este un set de comenzi între doi parteneri, cu scopul de a controla fluxul de informaţii transmise. Este un protocol simplu de comunicare, care nu solicită autentificarea partenerilor.

Definiţia 8.2. *O "conexiune" este un transport care oferă anumite tipuri de servicii. Fiecare conexiune este asociată unei sesiuni de comunicări între doi parteneri.*

Parametrii de sesiune sunt:

- **Identificatorul de sesiune**: Un octet arbitrar ales de server pentru a identifica o stare de sesiune (activă sau resumabilă).
- **Certificat**: Un certificat $X.509v3$ asociat fiecărei părţi. Eventual, acest câmp poate fi nul.
- **Metoda de compresie**: Algoritmul care asigură compresia (înainte de criptare).
- **Detaliile de criptare**: Specifică algoritmul de criptare, algoritmul MAC ($MD5$ sau SHA) şi alte atribute criptografice legate de acestea.
- **Master secret**: O secvenţă secretă de 48 octeţi partajată de client şi server.
- **Is resumable**: Un bit care indică dacă sesiunea poate fi utilizată pentru a iniţia conexiuni noi.

Parametrii de conexiune sunt:

- **Nonce server şi client**: Secvenţă aleatoare de un octet generată de fiecare parte la fiecare conexiune.
- MAC **secret server**: Cheia secretă folosită în codul de autrentificare al datelor scrise de server.
- MAC **secret client**: Cheia secretă folosită în codul de autrentificare al datelor scrise de client.
- **Cheie server**: Cheia sistemului simetric folosită de server pentru criptarea datelor, şi de client pentru decriptare.
- **Cheie client**: Cheia sistemului simetric folosită de client pentru criptarea datelor, şi de server pentru decriptare.
- **Vectorii de iniţializare**: Dacă pentru criptare se foloseşte modul CBC, este necesar un vector de inîtlaizare (VI) pentru fiecare cheie. Acest câmp este completat prima oară de protocolul handshaking SSL; apoi ultimul bloc criptat din fiecare înregistrare este păstrat drept VI pentru înregistrarea următoare.
- **Numere de secvenţă**: Fiecare entitate menţine separat două numere de secvenţă pentru mesajele trimise şi primite la fiecare conexiune. Când este primit/trimis un mesaj de schimbare a specificaţiilor de criptare, aceste numere sunt resetate.
  Cele două numere de secvenţă ale fiecărei părţi sunt de tipul $unit64$ si nu depăşesc $2^{64} - 1$.

8.2.1 Protocolul de înregistrare TLS

Protocolul de înregistrare TLS oferă o conexiune securizată. Principalele sale proprietăţi sunt:

1. Conexiunea este privată. După un protocol iniţial handsake care stabileşte o cheie pre-master, se foloseşte pentru criptare un sistem simetric (AES, DES, $RC4$ etc).

2. Negocierea secretului master este sigură. Nici un intrus nu poate modifica comunicarea în timpul negocierii fără a fi detectat.

3. Identitatea celor două părţi poate fi autentificată folosind sisteme de criptare cu cheie publică (RSA, DSS etc).

4. Conexiunea este sigură. Transmiterea unui mesaj include un control al integrităţii folosind un $HMAC$ cu $MD5$ sau SHA ca funcţii de dispersie (vom nota aceste funcţii cu $HMAC - MD5(secret, date)$ respectiv $HMAC - MD5(secret, date)$). Se pot defini şi alte funcţii de dispersie adiţionale.

Structura unui protocol de înregistrare TLS este:

1. Mesajul clientului este împărţit în blocuri $M_1, \ldots M_n$ de lungimi egale, de cel mult 2^{14} octeţi fiecare.

2. Opţional, se aplică o funcţie de compresie.

3. Se calculează un cod de autentificare folosind $HMAC - MD5$ sau $HMAC - SHA$; acesta este adăugat mesajului (arhivat).

4. Se criptează folosind un sistem simetric de criptare (bloc sau fluid).
   Dacă se foloseşte un sistem bloc, mesajul (textul clar comprimat şi codul de autentificare) este completat astfel ca lungimea lui să fie multiplu al lungimii blocului de criptare.

5. Se adaugă un antet SSL şi rezultatul este trimis prin canal.

Pentru criptarea simetrică se foloseşte unul din sistemele de criptare: $RC4$ cu cheie pe 128 biţi, DES cu cheie de 56 biţi, $3DES$ cu cheie de 168 biţi, sau AES cu cheie de 128/256 biţi. Majoritatea implementărilor TLS negociază sistemul şi cheia, în funcţie de nivelul suportat de dotarea celui mai slab dintre parteneri.

Mărimea cheilor folosite de sistemul de criptare cu cheie publică depinde de autoritatea de certificare.

Exemplul 8.1. *VerySign utilizează chei de* 512 *sau* 1024 *biţi, în funcţie de softwareul serverului. Cheia privată VerySign folosită la semnarea certificatelor este de* 1024 *biţi, iar cheile de sesiune folosite în tranzacţiile TLS sunt cele mai puternice permise de legislaţia SUA (în general* 128 *sau* 256 *biţi).*

După protocolul de înregistrare, *TLS* cuprinde alte patru protocoale: handshake, alertă, specificaţii de schimbare a sistemului de criptare şi de prelucrare a datelor.

8.2.2 Protocolul handshake

Parametrii criptografici ai stării de sesiune sunt generaţi prin protocolul handshake, care constituie primul strat din *TLS*. Când un client *TLS* vrea săînceapă o comunicare cu serverul, ei cad de acord asupra unei versiuni a protocolului, selectează algoritmii criptografici, se autentifică reciproc (opţional) şi folosesc tehnici de criptare cu cheie publică pentru a genera secretele partajate.

Protocolul este împărţit în patru faze:

1. Mesaje de salut.
2. Autentificare şi schimb de chei client.
3. Autentificare şi schimb de chei server
4. Sfârşit.

Să detaliem fiecare din aceste faze.

Mesaje de salut

În specificaţii aceste mesaje sunt numite *Client_hello*, respectiv *Server_hello*. Scopul lor este de a stabili capacităţi sigure de legătură între client şi server; anume versiunea protocolului *TLS*, *ID*-ul sesiunii, sistemele de criptare şi compresie. În plus, se mai generează şi se transmit între parteneri două numere aleatoare: N_C şi N_S.

Când un client vrea să se conecteze la un server, i se cere să trimită un mesaj *Client_hello*. Acest mesaj conţine:

- Versiunea protocolului *TLS* sau *SSL* pe care doreşte clientul să îl folosească în timpul sesiunii. Aceasta trebuie să fie cea mai recentă versiune suportată de softul clientului.
- *Nonce_Client* (pentru evitarea atacurilor replay), format din:
  - Timpul curent şi data – în standard UNIX pe 32 biţi – conform cu ceasul intern al clientului.

  - 28 octeţi generaţi de un generator de numere pseudoaleatoare.

- O listă – ordonată descrescător după preferinţă – a sistemelor de criptare agreate de client. Fiecare propunere este formată din două secţiuni: un tip de schimb de chei şi informaţii despre un algoritm de criptare simetric (inclusiv lungimea cheii secrete) împreună cu un cod de autentificare MAC.
  Serverul va alege una din aceste propuneri, sau – dacă nici una nu este acceptabilă – returnează eroare handskhake şi închide conexiunea.

- O listă de algoritmi de compresie suportaţi de softul clientului, ordonată descrescător după preferinţă.

- Un ID de sesiune. Acest câmp poate fi gol dacă nu este accesibil nici un astfel de ID sau dacă clientul doreşte să genereze parametri de securitate noi.

Dacă poate găsi un set acceptabil de algoritmi în *Client_hello*, serverul trimite ca răspuns un mesaj *Server_hello*. Acesta conţine:

- Versiunea serverului: câmpul conţine cea mai mică versiune TLS sau SSL propusă în *Client_hello* şi cea mai mare versiune suportată de server.

- $Nonce\_Server$: 28 octeţi generaţi aleator de server, care trebuie să fie diferiţi de cei generaţi de client.

- O propunere (unică) de sistem de criptare, aleasă din lista propusă de client. Pentru sesiunile rezumabile, acest câmp este valoarea din câmpul de stare al sesiunii care se încheie.

- Compresie: Un algoritm de compresie selectat de server din lista propusă de client.

- ID de sesiune. Identitatea sesiunii care corespunde conexiunii.

Autentificare şi schimb de chei

În faza a doua, imediat după mesajele de salut, serverul trimite:

1. Certificatul său. Tipul acestuia trebuie să fie apropiat de algoritmii de schimb de chei propuşi; de obicei este un certificat $X.509v3$ (sau o variantă a sa).

2. Cheia de server.

3. Un mesaj care solicită certificatul clientului (opţional).

4. Un mesaj care confirmă că faza a doua este completă.

TLS suportă următorii algoritmi de schimb de chei:

- *RSA*: Cheia secretă este criptată cu cheia privată a serverului.
- **Diffie-Hellman de perioadă lungă**: Certificatul serverului conţine parametrii algoritmului Diffie-Hellman, semnaţi de o autoritate de certificare (CA).
- **Diffie-Hellman de perioadă scurtă**: Parametrii Diffie-Hellman sunt semnaţi folosind protocoalele RSA sau DSS ale serverului.
- **Diffie-Hellman anonim**: Parametrii Diffie-Hellman nu sunt semnaţi.

După cum se vede, parametrii cheii variază, depinzând de protocolul de schimb de chei propus de server. Ei sunt:

- *RSA*: n (modulul) şi a (exponentul public) – pentru o cheie temporară.
- **Diffie-Hellman**: p (modulul), g (generatorul grupului ciclic) şi y $(= g^x)$ (cheia publică a serverului).

Aceşti parametri propuşi de server sunt semnaţi prin crearea unei amprente (cu $MD5$ sau SHA) şi apoi criptarea cu cheia privată a serverului. Amprenta include de asemenea şi nonce-urile din mesajele de salut.
Deci, dacă h este funcţia de dispersie, acest mesaj are forma

$$h(Nonce\_Client \| Nonce\_Server \| Parametri\_server)$$

După trimiterea certificatului de autentificare, schimbul de chei şi (opţional) cererea de certificare, serverul trimite un mesaj de tip *server_hello* indicând că prima fază din protocolul handshake s-a terminat; în continuare, el aşteaptă răspunsul clientului.

În faza a treia, clientul:

1. Verifică validitatea certificatului trimis de server; în funcţie de aceasta depinde continuarea protocolului.

   Dacă serverul a trimis o cerere de certificare, clientul are două opţiuni: să trimită un mesaj de certificare, sau un mesaj de alertă prin care anunţă că nu are un astfel de certificat; acesta este doar o atenţionare, de care serverul poate să ţină cont (şi să încheie comunicarea) sau nu (şi să permită continuarea protocolului).

2. Trimite cheia pre-master. Aceasta se poate realiza

   - prin trimiterea ei criptată cu sistenul RSA, sau

   - transmiterea cheii publice Diffie-Hellman a clientului. Pe baza ei, cei doi vor cădea de acord ulterior asupra cheii comune pre-master.
   Dacă metoda aceasta foloseşte o semnătură RSA sau DSS, atunci este obligatorie cererea de certificare a clientului, iar clientul trebuie să prezinte un certificat ca răspuns.

Patrametrii pentru cheia pre-master sunt:

- RSA: O cheie pre-master de 48 octeţi, criptată cu cheia publică din certificatul serverului sau cu o cheie RSA temporară trimisă de server în faza a doua. Din cheia pre-master, partenerii vor obţine cheia secretă master.
- **Diffie-Hellman**: Valoarea publică ($g^x \ mod \ p$) a cheii clientului. Dacă şi serverul foloseşte tot protocolul Diffie-Hellman, cei doi deduc imediat cheia comună pre-master.

După ce s-a obţinut cheia pre-master, clientul şi serverul calculează cheia secretă master după formula:

$$SK_M = PRF(pre-master, \text{“}master\_secret\text{”}, Nonce\_Client + Nonce\_Server)$$

Valoarea “$master\_secret$” foloseşte o sursă de entropie pentru a genera valori aleatoare cu diverse scopuri: MAC-uri, chei secrete, valori de iniţializare (IV). Indiferent de varianta folosită pentru cheia pre-master, “$master\_secret$” are 48 octeţi.
PRF este un generator de numere pseudo-aleatoare (care va fi detaliat ulterior).

După calculul SK_M, cheia pre-master trebuie ştearsă din memorie.

Sfârşit

În această fază, clientul şi serverul actualizează specificaţiile sistemulor de criptare cu algoritmii de criptare, cheile şi funcţiile de dispersie agreate de ambele părţi.

Apoi, clientul trimite un mesaj de încheiere pentru a verifica aceste date. Acesta este primul mesaj protejat cu specificaţiile negociate. Forma sa este:

$$MD5(\text{“}master\_secret\text{”} \| pad_2 \| MD5(handshake\_mesaj \| Expeditor \| \text{“}master\_secret\text{”} \| pad_1));$$

sau

$$SHA(\text{“}master\_secret\text{”} \| pad_2 \| SHA(handshake\_mesaj \| Expeditor \| \text{“}master\_secret\text{”} \| pad_1));$$

unde

- pad_1 şi pad_2 sunt valorile definite de MAC,
- $handshake\_mesaj$ se referă la toate mesajele handshake schimbate până acum,
- $Expeditor$ se referă la un cod care identifică expeditorul: $0x434C4E54$ dacă este clientul, $0x53525652$ dacă este serverul.

După acest ultim mesaj, clientul şi serverul pot începe să comunice, transmiţând date confidenţiale.

8.2.3 Calculul cheii

Protocolul de înregistrare TLS solicită un algoritm care să genereze cheile şi componentele secrete din MAC ale parametrilor de securitate stabiliţi prin protocolul handshake. După cum am văzut, cheia master generată în timpul autentificării şi a schimbului de chei este folosită ca o sursă de entropie pentru generarea cheilor şi a MAC-lui.

Materialul cheii este generat astfel:

$$K_{bloc} = PRF(MS.PS, \text{“}cheie\ expandata\text{”}, N_S.PS \| N_C.PS)$$

unde:

- $MS.PS$: Parametrii de securitate ai secretului master. *Secretul master* este o secvenţă de 48 octeţi, partajată de cei doi parteneri conectaţi.
- “*cheie expandata*”: Reprezentarea ASCII pentru “*key expansion*”; este folosit ca marcă de identificare.
- N_S este un nonce de 32 octeţi generaţi de server. Parametrii săi de securitate sunt $N_S.PS$.
- Similar, N_C este un nonce de 32 octeţi generaţi de client. Parametrii săi de securitate sunt $N_C.PS$.
- PRF este o funcţie pseudoaleatoare care expandează secretele în blocuri de date. Ea ia la intrare trei parametri: un secret X, o sămânţă Y şi o marcă de identificare ”*label*”; ca rezultat, produce o secvenţă de lungime arbitrară.

K_{bloc} generează suficient de mult material pentru a construi – în ordine – patru date:

1. Componentă secretă din MAC client;
2. Componentă secretă din MAC server;
3. Cheie client;
4. Cheie server.

Din ele, clientul şi serverul generează MAC-urile (necesare autentificării) şi cheile pe care le folosesc – împreună cu IV – la criptarea mesajelor cu un sistem simetric.

Construirea funcţiei pseudoaleatoare PRF

PRF folosit în TLS construieşte PRF din doi generatori de numere pseudoaleatoare, sub o formă care garantează securitatea dacă cel puţin un generator este sigur.

Un PRF se generează în doi paşi:

1. Se construieşte o funcţie de expansiune $P_{hash}(secret, data)$ pentru a expanda un secret. Ea se bazează pe o funcţie de dispersie şi se defineşte astfel:

$$P_{hash}(X, Y) = HMAC_{hash}(X, A(1)\|Y)\|HMAC_{hash}(X, A(2)\|Y)\| \ldots$$

unde $A(\cdot)$ este definit

$$A(0) = Y, \;\; A(i) = HMAC_{hash}(X, A(i-1))$$

P_{hash} poate fi iterat cât este necesar pentru a genera o cantitate sufientă de date.

Exemplul 8.2. *Dacă se foloseşte ca funcţie de dispersie* $SHA-1$ *pentru a genera* 64 *octeţi de date, atunci vor fi necesare* 4 *iteraţii (până la* $A(4)$*).*

Ele vor genera 80 *octeţi, din care ultimii* 16 *se ignoră.*

2. Funcţia pseudoaleatoare PRF folosită de TLS se obţine astfel:

   (a) Se divide un secret în două părţi egale;

   (b) Una din jumătăţi este utilizată pentru a genera date cu P_{MD5}, iar cealaltă jumătate – pentru a genera date cu P_{SHA-1}.

   (c) Cele două ieşiri sunt XOR-ate.

Deci

$$PRF(X, label\|Y) = P_{MD5}(X_1, label\|Y) XOR P_{SHA-1}(X_2, label\|Y)$$

unde

- Secretul X este partajat în două părţi egale: $X = X_1\|X_2$.
- "*label*" este o secvenţăSCII, inclusă exact cum este scrisă (fără octet de lungime sau caracterul null).

Exemplul 8.3. *"plano tx" este procesată concatenând octeţii* 70 6C 61 6E 6F 20 74 78 *cu valoarea din* Y*.*

Construirea MAC-ului

MAC-ul folosit de TLS este construit astfel:

$$HMAC_{hash}(secret_{MAC}, num\_seq \| Tip \| Versiune \| Lungime \| Fragment))$$

unde

- *hash* este algoritmul de dispersie specificat de parametrii de securitate,
- *num_seq* este numărul de secvenţă pentru înregistrarea corespunzătoare,
- *Tip, Versiune* şi *Lungime* se referă la caracteristicile respective ale algoritmului de compresie specificat în TLS, iar *Fragment* este o porţiune compresată din datele asociate.

8.2.4 Protocolul de alertă

Când protocolul handshake detectează un mesaj de eroare, partea respectivă trimite partenerului un mesaj de alertă. TLS permite două tipuri de mesaje de alertă: totale şi avertizări.

Un mesaj de *alertă totală* indică o conexiune atât de rea încât necesită încheierea ei imediată. O *avertizare* indică existenţa anumitor probleme de conexiune.

Mesajele de alertă sunt criptate şi arhivate conform specificaţiilor date de starea curentă a conexiunii.

Să detaliem câteva mesaje de alertă totale:

- **Unexpected_message**: A apărut un mesaj având un tip care nu corespunde protocolului. Un astfel de mesaj nu trebuie să apară niciodată într-un protocol de comunicaţie implementat corect.
- **Bad_record_mac**: Alertă trimisă dacă s-a primit o înregistrare cu MAC incorect.
- **Decryption_failed**: Alertă trimisă dacă un text criptat TLS este decriptat într-un mod incorect: nu a fost multiplu par de lungimea blocului, valorile de control adăugate sunt incorecte etc.
- **Decrypt_error**: A eşuat o operaţie criptografică din protocolul handshake (nu se poate verifica corect o semnătură, decripta o cheie, valida terminarea unui mesaj etc).
- **Decompresion_failure**: Funcţia de dezarhivare primeşte o intrare incorectă (de exemplu datele au o lungime prea mare).
- **Handshake failure**: Expeditorul nu a putut negocia un set acceptabil de parametri de securitate din lista opţiunilor valabile.
- **Illegal_parameter**: Inconsistenţă între câmpul ataşat unui parametru şi alte câmpuri.

8.2.5 Protocolul de schimbare a specificaţiilor

În cadrul TLS, un protocol de schimbare a specificaţiilor (*Change Cipher Spec Protocol*) semnalează schimbarea strategiei de criptare. Protocolul constă dintr-un octet, criptat şi arhivat, în starea de conectare curentă. Când unul din parteneri trimite un astfel de anunţ, el notifică colegului său că tot ce urmează este protejat cu noile chei şi specificaţii de securitate negociate la început.

Un mesaj de schimbare a specificaţiilor poate fi trimis în timpul protocolului handshake, după acceptarea parametrilor de securitate, dar înainte de procedura de verificare a sfârşitului de mesaj.

8.2.6 Concluzii generale despre SSL/TSL

Câteva comparaţii între cele două aplicaţii prezentate – $IPsec$ şi SSL/TLS – care asigură securitatea comunicaţiilor Internet:

- $IPsec$ instalat pe reţele oferă un acces total la resurse, fiind indicate pentru conectarea unei game extrem de variate de utilizatori (birouri, reţele intranet, acces la cerere client, extranet controlat de client etc). Pentru orice resursă şi aplicaţie de reţea, $IPsec$ asigură un foarte buin acces securizat.
  Principalul său dezavantaj constă în complexitatea solicitării vis-a-vis de clinet: clientul trebuie să posede un software adecvat şi trebuie să folosească cunoştionţe avansate pentru a putea lucra cu acesta.

  Pe de-altă parte SSL – implementat prin standardul TLS – nu solicită nici un software pe partea client, aşa că aplicaţia poate fi accesată de pe orice calculator conectat la Internet. Dezavantajul constă ;intr-o securitate mai slabă. SSL este recomandat pentru e-Comerţ, portaluri de aplicaţii Web, reţele extranet cu parteneri multipli.

- Reţelele virtuale private $IPsec$ oferă canale tip *tunnel* de la parteneri externi spre reţelele interne. SSL asigură o modalitate simplă de conectare a clienţilor externi la aplicaţii ale serverelor interne.

- O implementare SSL oferă mai multă flexibilitate decât $IPsec$. Pentru că nu este necesar nici un software client, orice angajat poate avea acces la reţelele unei companii, folosind orice computer şi orice browser Internet.

- Deoarece SSL nu solicită reţelelor dificultatea configurăii, instalării şi funcţionării $IPsec$ pentru fiecare utilizator, în general, se consideră că o aplicaţie SSL este mai uşoară şi mai puţin costisitoare pentru client, decât o aplicaţie $IPsec$.

În general, o reţea VPN dotată cu SSL – implementat prin standardul TLS – are următoarele caracteristici:

- O reţea VPN utilizează SSL şi tehnologie proxy pentru a oferi utilizatorilor acces autorizat şi sigur la $HTTP$, aplicaţii server şi diverse resurse partajate. Ele asigură transportul datelor şi sesiuni sigure între orice browser şi orice server proxy dintr-un gateway VPN cu SSL implementat.

- SSL-ul unui VPN funcţionează ca un proxy pentru ambele părţi; nu există niciodată o conexiune directă spre o reţea privată. Accesul este permis numai spre aplicaţiile oferite de SSL.

- Într-o reţea VPN, SSL funcţionează ca un gateway: el stabileşte două conexiuni: una între browserul Web al clientului dintr-o reţea nesecurizată şi SSL proxy din VPN, şi o a doua conexiune între SSL proxy din VPN şi urilizatorul dintr-o reţea securizată; in acest mod este evitată orice legatură directă cu o reţea securizată. Deci SSL acţionează ca un server pentru client şi ca un client pentru server.

- SSL asigură că un utilizator autorizat are acces numai la anumite resurse, cele alocate de software-ul companiei prin aplicaţia SSL şi integrate de gestiunea traficului.

- Serverele proxy sparg conexiunea TCP/IP dintre client şi server, fără a mai forwarda şi adresa IP a pachetelor. Astfel, ele oferă o secruitate la trecerea prin reţele nesigure, ascunzând adresele IP ale utilizatorilor din reţelele securizate. Într-o reţea nesecurizată este vizibilă numai adresa IP publică a serverului proxy.

Dintre slăbiciunile unei reţele VPN dotate cu SSL, menţionăm:

- Computerele stochează în locaţii nesecurizate diverse informaţii importante.

- După închiderea conexiunii, parolele clienţilor rămân în zone publice.

- Parolele clienţilor sunt stocate de browser.

- Date importante (informaţiile cache, URL-urile, cookies, informaţii din history) create în timpul sesiunii pot rămâne pe calculatoare publice după închiderea completă a unei sesiuni.

- Fişiere downloadate rămân stocate în directoare temporare pe calculatoare publice.

- Clienţii uită adesea să dea logout (sau nu consideră necesar acest lucru).

- Clientul următor care foloseşte calculatorul public are acces la aplicaţii.

- Viruşi şi viermi pot fi transferaţi de pe calculatoare publice pe reţele interne.

8.3 Electronic Transaction Protocol (SET)

Protocolul SET a fost construit pentru tranzacţii financiare pe Internet, cu scopul de a oferi securitatea plăţilor cu card. El a rezultat prin unirea eforturilor de cercetare ale firmelor Visa şi MasterCard, ca o metodă de asigurare a tranzacţiilor electronice. La dezvoltarea specificaţiilor au cotnribuit de asemenea firme cunsocute ca IBM, $Microsoft$, $Netscape$, RSA, $VeriSign$.

La sfârşitul anilor $'90$ SET a fost aprobat drept un protocol de credit standard, dar nu a putut fi impus din cauza costurilor şi a problemelor ridicate de distribuţia certificatelor utilizatorilor. Cu toate acestea el este considerat în acest moment ca fiind un protocol ideal din punct de vedere al certificării, al semnăturilor digitale şi al algoritmilor de criptare care securizează cardurile de credit în tranzacţiile Internet.

SET utilizează criptografia pentru asigurarea următoarelor servicii de securitate:

1. Asigură autentificarea unui deţinător de card ca utilizator legitim al unui cont din care se pot efectua operaţii financiare. Ea se realizează folosind sistemul de criptare RSA şi certificate $X.509$.

2. Asigură autentificarea unei isntituţii de plată (bancă) de la care un vânzător primeşte valoarea bunurilor/serviciilor oferite unui client.

3. Asigură confidenţialitatea operaţiilor de plată, pe baza sistemului de criptare DES.

4. Asigură integritatea documentelor de plată, folosind funcţia de dispersie $SHA-1$.

8.3.1 Participanţii la un protocol SET

Pentru orice tranzacţie de plată cu card sunt implicate 5 entităţi:

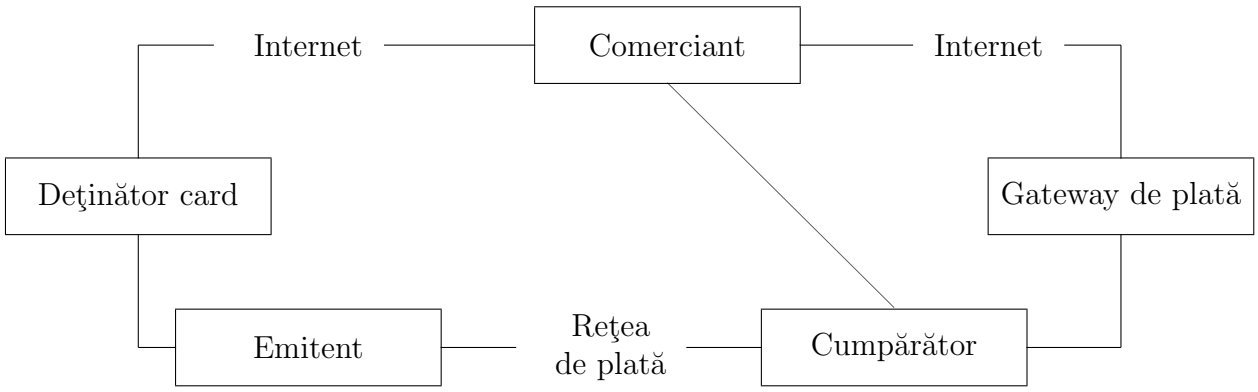
1. **Deţinător card**: Într-un context de comerţ electronic, consumatorii interacţionează cu comercianţii prin intermediul calculatoarelor personale. Un deţinător de card foloseşte un card de plată emis de societate numită "*Emitent*". Protocolul SET asigură că, în interacţiunea dintre deţinătorul de card şi comerciant, informaţia de pe card rămâne confidenţială.

2. **Emitent**: Este o instituţie financiară care stabileşte un cont pentru un utilizator şi îi eliberează acestuia un card de plăţi. Emitentul garantează onorarea – cu ajutorul cardului – a tranzacţiilor financiare autorizate.

3. **Comerciant**: Oferă spre vânzare bunuri sau servicii în schimbul unei plăti. Un comerciant care acceptă plata pe card trebuie să fie într-o relaţie de parteneriat cu un cumpărător.

4. **Cumpărător**: Este instituţia financiară care stabileşte un cont cu un comerciant şi procesează plătile autorizate de cardul asociat acestui cont.

5. **Gateway de plată**: Un sistem cu care operează cumpărătorul sau o componentă comună stabilită de comun acord, al cărui rol este de a procesa toate mesajele de plată (inclusiv instrucţiuni de plată venite de la deţinătorul cardului). El permite cumpărătorului să accepte tranzacţii SET prin Internet şi să le pregătească pentru a fi trimise la reţele comerciale de plată (cum sunt reţelele $MasterCard$ ale băncilor).



8.3.2 Tranzacţii SET

Paşii unei tranzacţii electronice SET sunt:

1. Cumparătorul (de fapt banca care gestionează contul clientului) solicită un portofel digital dela o bancă şi obţine un certificat digital, un card sau alt document autentificat de banca emitentă.

2. Comerciantul obţine un certificat digital de la banca cumpărător asociată. Dacă doreşte să facă vânzări on-line folosind SET, el va trebui să deţină două certificate valide: $CERT_{Com}$ – emis de cumpărătorul asociat şi $CERT_{Gateway}$ – de la gateway-ul de plată.

3. Clientul (posesorul cardului) caută – cu un browser – printr-un catalog on-line din pagina Web a comerciantului, selectează produsele pe care vrea să le cumpere şi completează fişa electronică de comandă. Plata o face folosind din portofelul digital primit de la bancă.

4. Clientul trimite comerciantului fişa electronică completată (FE), însoţită de instrucţiunile de plată (OP)– ambele semnate. Ôn protocolul SET comerciantul nu poate accesa numărul cărţii de credit a posesorului cardului – informaţie la care are acces numai banca emiţătoare. Posesorul cardului nu va trimite nici o informaţie sensibilă până ce nu îl autentifică complet pe comerciant.

5. Comerciantul primeşte FE şi OP. Nu are acces însă la OP deoarece aceasta este criptată cu cheia publică a gateway-ului de plată.

6. El trimite FE, OP şi datele sale de autentificare la banca asociată gateway-ului şi care va face plătile. Gateway-ul de plată translatează mesajul SET într-un protocol recunoscut şi folosit de reţeaua care a emis cardul. Gateway-ul de plată primeşte certificatele de la SET-ul asociat *Root* CA (un CA aflat în vârful ierarhiei de încredere).

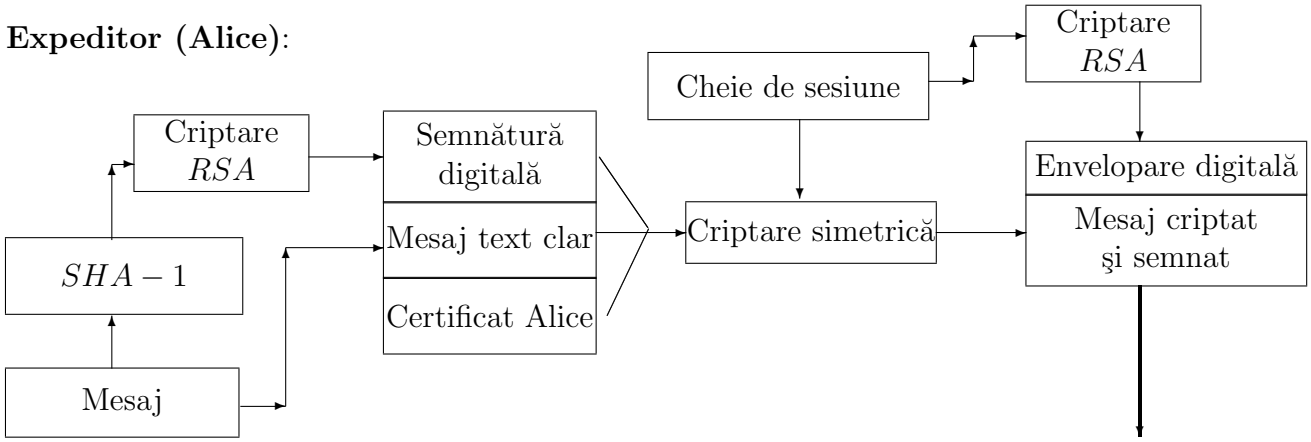
7. Gateway-ul autentifică pe cei doi parteneri, decriptează OP şi transmite informatîile băncii care efectuează plata. Banca trimite o cerere de autorizare pe care banca emitentă a cardului.

8. Banca emitentă primeşte cererea de autorizare ca pe orice solicitare de tranzacţie fizică. Cercetează contul posesorului de card şi – în funcţie de informaţiile pe care le deţine – aprobă sau respinge plata. Acest mesaj este trimis înapoi spre banca care efectuează plăti.

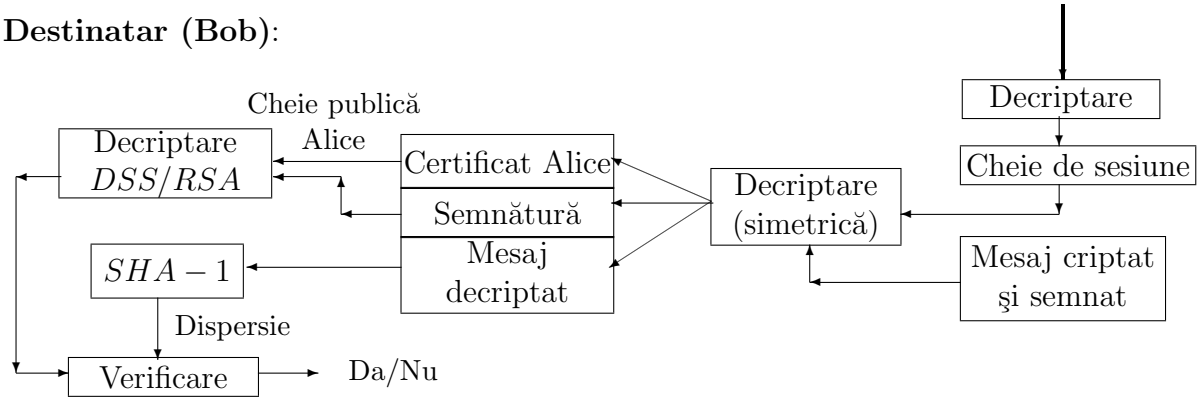
9. Mesajul trece din nou prin gateway-ul de plată şi este criptat înainte ca autorizaţia să fie trimisă comerciantului. Dacă acesta are confirmarea validităţii cardului de plată (inclusiv dacă suma este acoperită de creditul cardului), va livra solicitantului produsele cumpărate.

8.3.3 Autentificarea şi confidenţialitatea SET

Protocolul SET asigură servicii de confidenţialitate şi autentificare folosind primitive criptografice şi certificate digitale. La orice schimb de informaţii între deţinătorii de card, autorităţile de certificare, comercianţii şi băncile de emitere de card sau de plată – pentru obţinerea unui certificat, trimiterea unei instrucţiuni de plată sau o autorizare de plată – informaţia vehiculată este securizată prin semnături şi envelopări digitale, însoţite de criptări (simetrice sau cu cheie publică).



Destinatar (Bob):



Procesul de autentificare şi confidenţialitate *SET* cuprinde următoarele etape (descrise şi de figura de mai sus):

1. Expeditorul (*Alice*) generează o sesiune aleatoare. Cheia de sesiune este o cheie secretă one-time folosită pentru criptarea mesajului cu un sistem de criptare simetric.

2. Mesajul este arhivat cu funcţia de dispersie $SHA-1$ şi semnat cu RSA bazat pe cheia privată a lui *Alice*. Se formează astfel "*Semnătura digitală*".

3. Mesajul – text clar – este concatenat cu semnătura digitală şi cu certificatul lui *Alice*.

4. Ceea ce s-a obţinut se criptează cu un algoritm simetric (DES) folosind cheia secretă one-time de la Pasul 1.

5. Cheia de sesiune one-time este criptată cu RSA folosind cheia publică a lui *Alice*. Rezultatul este numit "*envelopare digitală*".

6. Se face o concatenare a cheii de sesiune criptate cu mesajul criptat şi semnat.

7. Pentru decriptarea şi autentificarea mesajului, *Bob* va parcurge paşii în ordine inversă.

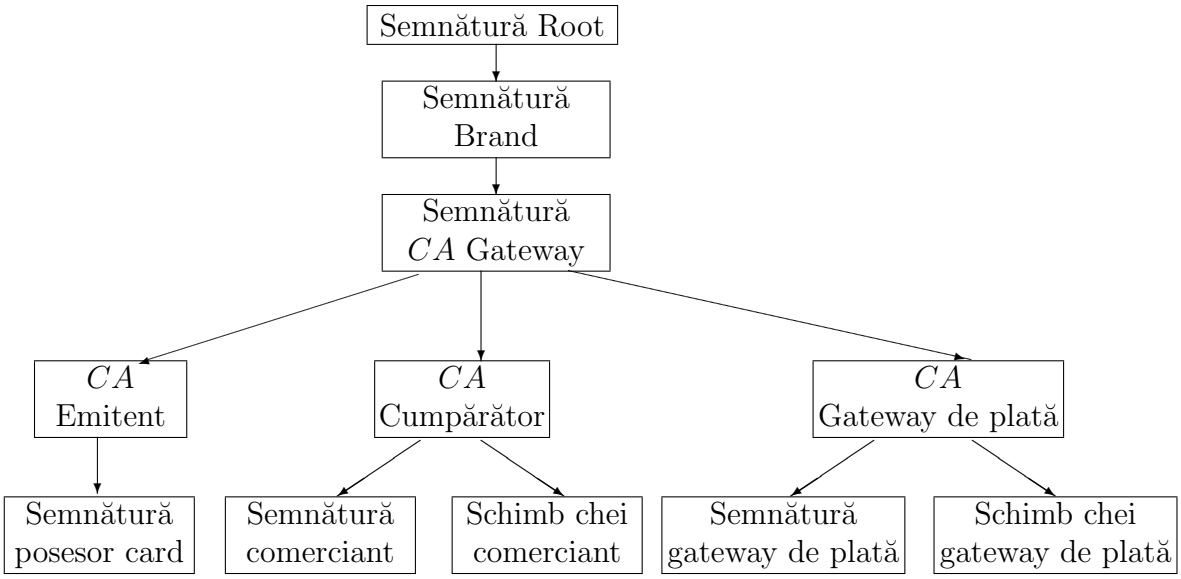
8.3.4 Ierarhia de încredere *SET*

În mediul *SET* există o ierarhie bine stabilită a autorităţilor de certificare. În vârf se află *SET Root CA* – deţinută şi gestionată de *Secure Electronic Transactions LLC*. Toate certificatele conţin cheia sa publică.

- **Certificatul proprietarului de card**:
  Un certificat al proprietarului de card funcţionează ca o reprezentare electronică a unui card de plăţi. El este aprobat de banca emiţătoare, nu conţine numărul contului şi nici data expirării. Acest certificat este transmis comercianţilor însoţit

de instrucţiuni de plată criptate. Când un comerciant primeşte un certificat al proprietarului de card, el este sigur – chiar dacă nu vede numărul de cont – că acest card de plăţi este validat de bancă.

- **Certificatul comerciantului**:
  Un comerciant primeşte două certificate de la instituţia financiară:
  - *Certificatul său propriu.* El îi asigură dreptul de a exista ca brand (entitate comercială);
  - *Certificatul gateway-ului de plată*, prin care îi sunt acceptate (şi plătite) de către cumpărător contravaloarea mărfurilor vândute.
- **Certificatul gateway-ului de plată**:
  Acest certificat est obţinut de cumpărător de la Autoritatea de Certificare a Brandului pentru gateway-ul său de plată. El include cheia publică şi cheia de criptare folosite de proprietarul de card pentru protejarea informaţiei legate de cont.
- **Certificatul cumpărătorului**:
  Banca cu rol de cumpărător trebuie certificată de emiţătorul de carduri de plată. Acesta îi dă astfel dreptul de a accepta sau respinge cererile de certificare venite direct de la comercianţi.
- **Certificatul emitentului**:
  Un emitent trebuie de asemenea certificat de emiţătorul de carduri de plată. Astfel, el poate opera ca o autoritate de certificare pentru cererile de certificare venite direct de la posesorii de carduri.



Graful arată structura arborescentă a ierarhiei *SET* de încredere.

8.3.5 Tranzacţii SET

Atunci când un client comandă un produs unui comerciant, protocolul SET desfăşoară trei tranzacţii:

1. Cererea de cumpărare;

2. Autorizarea de plată;

3. Efectuarea plăţii.

A. Cererea de cumpărare:
În această fază protocolul este format din 4 mesaje schimbate între client şi comerciant:

A1. Iniţierea cererii

Protocolul SET este invocat când posesorul de card este gata să lanseze un oridn de cumpărare: a selectat produsele pe care le doreşte, a completat forma de vânzare a comerciantului şi a fost de acord cu termenii şi condiţiile acestuia. În primul său mesaj, clientul iniţiază cererea de cumpărare solicitând comerciantului certificatul propriu şi certificatul emis de gateway-ul de plată afiliat.

A2. Iniţierea răspunsului

1. Software-ul comerciantului primeşte iniţierea cererii de cumpărare.

2. Generează un răspuns RI, îi aplică o funcţie de dispersie h (standardul este $SHA-1$) şi îl criptează (standardul este RSA) cu cheia secretă K a comerciantului. În acest fel produce o semnătură digitală $d_K(h(RI))$ a mesajului de răspuns RI.

3. Trimite clientului $(RI, d_K(h(RI)), CERT_{Com}, CERT_{Gateway})$.

Software-ul clientului verifică acest răspuns:

1. Decriptează semnătura folosind cheia publică a comerciantului (aflată în $CERT_{Com}$): $e_K(d_K(h(RI))) = h(RI)$.

2. Aplică funcţia de dispersie lui RI şi verifică dacă rezultatul $h(RI)$ este egal cu ce a obţinut anterior.

A3. Cererea de cumpărare

Dacă verificarea este pozitivă, clientul lansează un nou mesaj, conţinând instrucţiuni de cumpărare (CI) şi de plată (PI). De remarcat că CI conţine porţiuni din cererea de cumpărare, dar nu şi descrieri ale produselor comandate. Mai exact, software-ul clientului:

1. Pune în CI şi PI identificatorul de tranzacţie asignat de comerciant; acesta va fi folosit mai târziu – când comerciantul primeşte autorizaţia – de gateway-ul de plată pentru a asocia CI de PI.

2. Generează o semnătură dublă pentru CI şi PI: $\sigma_C = d_K{}'(h(h(PI)\|h(CI)))$, folosind cheia sa secretă K'.

3. Generează aleator o cheie de sesiune K_1 pentru un sistem simetric de criptare (Standard DES) cu care criptează $PI, h(CI)$ şi semnătura dublă. K_1 şi numărul de cont al cardului clientului sunt criptate cu cheia publică a gateway-ului de plată, formând un plic digital $PD = e_{Gateway}(CONT_{Client}, K_1)$.

4. Trimite comerciantului

$$(PD, e_{K_1}(PI\|\sigma_C), \sigma_C, h(PI), CI, CERT_{Client})$$

Primele două componente nu sunt accesibile comerciantului, ele fiind destinate gateway-ului de plată.

A4. Răspuns la cererea de cumpărare

Comerciantul:

1. Verifică $CERT_{Client}$ şi extrage din el cheia publică de criptare.

2. Calculează $e_{K'}(\sigma_C)$.

3. Calculează $h(CI)$, apoi $h(h(PI)\|h(CI))$ şi compară rezultatul cu ceea ce a obţinut la pasul anteior.

4. Dacă cele două valori sunt egale, deduce că CI este autentic şi procesează cererea de cumpărare (inclusiv trimiterea PI spre gateway pentru autorizare).

5. Generează un răspuns R la cererea de cumpărare şi îl semnează: $d_K(h(R))$.

6. Trimite clientului perechea $(d_K(h(R)), CERT_{Com})$.

B. Autorizarea de plată: Autorizaţia de plată constă din două mesaje:

- *Cererea de autorizare*: trimisă de comerciant spre gateway-ul de plată,

- *Autorizare de răspuns.*

Gateway-ul autentifică comerciantul, decriptează şi criptează informaţia de plată şi o transmite băncii cumpărătoare. Aceasta lansează şi ea o cerere de autorizare pe baza căreia banca emitentă poate aproba sau respinge tranzacţia posesorului de card. Banca cumpărătoare primeşte decizia şi o trimite gateway-ului de plată care emite comerciantului o autorizare de răspuns.

B1. Cererea de autorizare

Software-ul comerciantului:

1. Generează o cerere de autorizare AR care include suma solicitată, identificatorul tranzacţiei (din CI), şi alte informaţii specifice.
2. Semnează cererea: $\sigma = d_K(h(AR))$
3. Generează aleator o cheie de sesiune K_2 şi calculează
$$P = e_{K_2}(\sigma, AR),\ E = e_{K_{Gateway}}(K_2).$$
4. Trimite spre gateway-ul de plată mesajul
$$(P, E, PD, e_{K_1}(PI \| \sigma_C), CERT_{Com}, CERT_{Client})$$

B2. Autorizare de răspuns

Gateway-ul de plată:

1. Verifică $CERT_{Com}$, din care scoate cheia publică K a comerciantului.
2. Verifică cecerea de autorizare:
   (a) Află K_2, pe care o foloseşte pentru a determina AR şi σ.
   (b) Calculează $h(AR)$ şi compară rezultatul cu $e_K(\sigma)$.
3. Verifică $CERT_{Client}$, din care scoate cheia publică K' a clientului.
4. Obţine infoirmaţia de plată dată de client si verifică semnătura dual̆a.
   (a) Din PD obţine cheia K_1 pe care o foloseşte pentru a găsi PI şi σ_C.
   (b) Calculează $e_{K'}(\sigma_C) = h(h(PI) \| h(CI))$.
   (c) Calculează $h(PI)$, apoi $h(h(PI) \| h(CI))$, pe care îl compară cu rezultatul anterior.
   (d) Verifică consistenţa dintre cererea de autorizare a comerciantului şi PI trimis de client.

5. Trimite cererea de autorizare – folosind o reţea financiară normală – băncii care a eliberat cardul clientului.

6. Trimite comerciantului autorizarea de răspuns:

   (a) Generează o autorizare de răspuns AR, pe care o semnează: $\sigma_G = d_{Gateway}(h(AR))$.

   (b) Generează aleator o cheie de sesiune K_3 şi calculează
   $$Q = e_{K_3}(AR, \sigma_G),\ F = e_K(K_3).$$

   (c) Crează un fişier CT numit "*Capture token*" şi-l semnează:
   $$sign(CT) = d_{Gateway}(h(CT)).$$

   (d) Generează aleator o cheie de sesiuneK_4 si calculează $e_K(K_4),\ e_{K_4}(CT)$.

   (e) Trimite comerciantului
   $$(Q, F, e_K(K_4), e_{K_4}(CT), CERT_{Gateway})$$

B3. Verificarea de către comerciant a autorizării de răspuns

La primirea mesajului, software-ul comerciantului stochează autorizarea de răspuns şi CT – pentru a le folosi când va solicita plata. Completează apoi comanda clientului, trimiţându-i marfa şi serviciile specificate în CI.

Detaliat, comerciantul efectuează următorul protocol:

1. Verifică $CERT_{Gateway}$.

2. Găseşte cheia de sesiune K_3, cu ajurotul căreia calculează autorizaţia de răspuns AR.

3. Verifică semnătura a gateway-ului: calculează $e_{Gateway}(sign(h(AR))$ pe care o compară cu rezultatul obţinut prin aplicarea funcţiei de dispersie h lui AR obţinut anterior.

4. Stochează $Sign(CT)$ şi $e_{K_4}(CT)$. De observat că numai gateway-ul de plată poate afla tokenul de captură.

5. Finalizează procesarea cererii de cumpărare.

C. Efectuarea plăţii:

C1. Cererea de plată a comerciantului

După finalizarea comenzii de livrare a produsului cumpărat, comerciantul va solicita plata. Pentru aceasta:

1. Scrie o cerere de plată SP.

2. O semnează: $\sigma_{Com} = d_{Com}(h(SP))$.
3. Generează aleator o cheie de sesiune K_5.
4. Trimite spre gateway-ul de plată

$$(e_{Gateway}(K_5), e_{K_5}(SP), \sigma_{Com}, Sign(CT), e_{K_4}(CT), CERT_{Com})$$

C2. Răspunsul gateway-ului

Când un gatewayu de plată primeşte o solicitare de plată, el decriptează informaţia şi foloseşte CT pentru a genera o solicitare de plată sub formă de text clar, pe care o trimite băncii emiţătoare via un sistem de plată pe card. Apoi el generează un răspuns pe care îl transmite comerciantului.
Detaliat, gateway-ul de plată:

1. Verifică $CERT_{Com}$, din care scoate cheia publică a comerciantului.
2. Verifică cererea de plată a comerciantului.
   (a) Află K_5, cu ajutorul căreia calculează SP.
   (b) Verifică semnătura digitală a comerciantului, scoţând din ea $h(SP)$ şi comparând-o cu $h(SP)$ calculat din PR rezultat anterior.
   (c) Află K_4 folosind cheia sa privată şi calculează CT.
   (d) Verifică consistenţa dintre informaţiile cuprinse în CT şi SP.
3. Trimite SP – printr-o reţea financiară – spre banca care deţine contul clientului.
4. Generează un răspuns de plată RP şi îl semnează: $\sigma_{RP} = d_{Gateway}(h(RP))$.
5. Generează aleator o cheie de sesiune K_6.
6. Trimite comerciantului

$$(e_{Com}(K_6), e_{K_6}(PR, \sigma_{PR}), CERT_{Gateway})$$

C3. Verificarea răspunsului de plată

La primirea răspunsului la cererea sa de plată, software-ul comerciantului:

1. Verifică $CERT_{Gateway}$, din care scoate cheia publică de criptare.
2. Calculează cheia K_6, cu ajutorul căreia află răspunsul de plată RP.
3. Verifică semnătura digitală, decriptând σ_{RP} şi comparând rezultatul cu $h(RP)$ calculat pe baza informaţiilor de la pasul anterior.

În final, comerciantul stochează RP ca o confirmare că i se face plata.

Bibliografie

[1] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., Wright, T. (2006) - *Transport layer security (TLS) extensions* (RFC 4366).

`http://www.ietf.org/rfc/rfc4366.txt?number=4366`

[2] Dierks, T., Rescorla, E. (2006) - *The transport layer security (TLS) protocol version 1.1* (RFC 4346)

`http://www.ietf.org/rfc/rfc4346.txt?number=4346`

[3] Freier, A., Karlton, P., Kocher, P. (1996) – *The SSL protocol version 3.0*

`http://wp.netscape.com/eng/ssl3/3-SPEC.HTM#1`

[4] Santesson, S. (2006) - *TLS handshake message for supplemental data* (RFC 4680)

`http://www.ietf.org/rfc/rfc4680.txt?number=4680`

[5] Santesson, S., Ball, J., Medvinsky, A. (2006) - *TLS user mapping extension* (RFC 4681).

`http://www.ietf.org/rfc/rfc4681.txt?number=4681`

[6] *SET Secure Electronic Transaction Specification Book 1: Business Description* (1997)