

# Alte protocoale de securitate

Prof. Dr. Adrian Atanasiu

Universitatea Bucureşti

April 10, 2014

## 1 Introducere

## 2 *TLS ca implementare pentru SSL*

## 3 Protocolul de înregistrare *TLS*

- Protocolul de alertă
- Concluzii

## 4 Electronic Transaction Protocol (*SET*)

- Tranzacții *SET*
- Protocolele de tranzacții *SET*

Creșterea volumului de comerț on-line solicită utilizarea de protocoale care trebuie să asigure o multitudine de operații de securitate.

Acstea informații transmise prin rețea nesigură cum este Internetul, generează multă reticență și utilizatorii trebuie să fie siguri că sunt suficient de bine protejați.

Creșterea volumului de comerț on-line solicită utilizarea de protocoale care trebuie să asigure o multitudine de operații de securitate.

Aceste informații transmise printr-o rețea nesigură cum este Internetul, generează multă reticență și utilizatorii trebuie să fie siguri că sunt suficient de bine protejați.

De aceea, pentru securizarea datelor necesare unei tranzacții (confidențialitate, autentificare și non-repudiere) s-au dezvoltat de-a lungul timpului o serie de protocoale.

Cele mai utilizate sunt *SSL* ([Secure Socket Layer](#)) construit de Netscape și standardul Internet al *SSL*, cunoscut sub numele *TLS* ([Transport Layer Security](#)).

Creșterea volumului de comerț on-line solicită utilizarea de protocole care trebuie să asigure o multitudine de operații de securitate.

Aceste informații transmise printr-o rețea nesigură cum este Internetul, generează multă reticență și utilizatorii trebuie să fie siguri că sunt suficient de bine protejați.

De aceea, pentru securizarea datelor necesare unei tranzacții (confidențialitate, autentificare și non-repudiere) s-au dezvoltat de-a lungul timpului o serie de protocole.

Cele mai utilizate sunt **SSL** ([Secure Socket Layer](#)) construit de Netscape și standardul Internet al **SSL**, cunoscut sub numele **TLS** ([Transport Layer Security](#)).

Ambele sunt implementate pe toate browserele Web, deși sub forme diferite; deci companiile trebuie să dezvolte aplicații **SSL** distincte pentru Netscape și Internet Explorer.

*TLS și SSL permit utilizatorilor să-și definească nivelul de securitate pe care-l doresc.*

*TLS* și *SSL* permit utilizatorilor să-și definească nivelul de securitate pe care-l doresc.

Când un client și un server sunt de acord să comunice pe baza unui protocol *SSL* sau *TLS*, ei trebuie să cadă de acord (prin protocole de tip handshake: protocol simplu de comunicare, care nu solicită autentificarea partenerilor) și asupra altor puncte:

*TLS* și *SSL* permit utilizatorilor să-și definească nivelul de securitate pe care-l doresc.

Când un client și un server sunt de acord să comunice pe baza unui protocol *SSL* sau *TLS*, ei trebuie să cadă de acord (prin protocole de tip handshake: protocol simplu de comunicare, care nu solicită autentificarea partenerilor) și asupra altor puncte:

- 1 Protocolul și versiunea (*TLS 1.0*, *TLS 1.1*, *SSL2*, *SSL3*).

*TLS* și *SSL* permit utilizatorilor să-și definească nivelul de securitate pe care-l doresc.

Când un client și un server sunt de acord să comunice pe baza unui protocol *SSL* sau *TLS*, ei trebuie să cadă de acord (prin protocole de tip handshake: protocol simplu de comunicare, care nu solicită autentificarea partenerilor) și asupra altor puncte:

- 1 Protocolul și versiunea (*TLS 1.0*, *TLS 1.1*, *SSL2*, *SSL3*).
- 2 Sistemul de criptare.

*TLS* și *SSL* permit utilizatorilor să-și definească nivelul de securitate pe care-l doresc.

Când un client și un server sunt de acord să comunice pe baza unui protocol *SSL* sau *TLS*, ei trebuie să cadă de acord (prin protocole de tip handshake: protocol simplu de comunicare, care nu solicită autentificarea partenerilor) și asupra altor puncte:

- 1 Protocolul și versiunea (*TLS 1.0*, *TLS 1.1*, *SSL2*, *SSL3*).
- 2 Sistemul de criptare.
- 3 Dacă vor autentificare sau nu.

*TLS* și *SSL* permit utilizatorilor să-și definească nivelul de securitate pe care-l doresc.

Când un client și un server sunt de acord să comunice pe baza unui protocol *SSL* sau *TLS*, ei trebuie să cadă de acord (prin protocole de tip handshake: protocol simplu de comunicare, care nu solicită autentificarea partenerilor) și asupra altor puncte:

- 1 Protocolul și versiunea (*TLS 1.0*, *TLS 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.

*TLS* și *SSL* permit utilizatorilor să-și definească nivelul de securitate pe care-l doresc.

Când un client și un server sunt de acord să comunice pe baza unui protocol *SSL* sau *TLS*, ei trebuie să cadă de acord (prin protocole de tip handshake: protocol simplu de comunicare, care nu solicită autentificarea partenerilor) și asupra altor puncte:

- 1 Protocolul și versiunea (*TLS 1.0*, *TLS 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.

Un protocol *TLS* este format din două etape (layers):

Un protocol *TLS* este format din două etape (layers):

- **Protocolul *TLS* de înregistrare:** ia mesajele care trebuie transmise, fragmente de date din diverse blocuri interne, le arhivează (optional), aplică o funcție *MAC*, le cripteză și transmite rezultatul.

Un protocol *TLS* este format din două etape (layers):

- **Protocolul *TLS* de înregistrare:** ia mesajele care trebuie transmise, fragmente de date din diverse blocuri interne, le arhivează (optional), aplică o funcție *MAC*, le cripteză și transmite rezultatul.
- **Protocolul de handshaking:** permite serverului și clientului să se autentifice reciproc și să negocieze un algoritm de criptare și o cheie de criptare, înainte de a se transmite/primi primul bit de date.

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

## Definiție

*O “sesiune” **TLS** este o asociere între un client și un server, cu scopul de a defini un set de parametri de securitate pentru o perioadă mai lungă.*

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

## Definiție

*O "sesiune" TLS este o asociere între un client și 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.*

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

### Definiție

*O “sesiune” TLS este o asociere între un client și 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.*

### Definiție

*O “conexiune” este un transport care oferă anumite tipuri de servicii.*

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

## Definiție

O "sesiune" TLS este o asociere între un client și 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.

## Definiție

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:

- **Identifierul de sesiune:** Un octet arbitrar ales de server pentru a identifica o stare de sesiune (activă sau resumabilă).

## Parametrii de sesiune:

- **Identifierul 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 părți. Eventual, acest câmp poate fi nul.

## Parametrii de sesiune:

- **Identifierul 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 părți. Eventual, acest câmp poate fi nul.
  - **Metoda de compresie:** Algoritmul care asigură compresia (înainte de criptare).

## Parametrii de sesiune:

- **Identifierul 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 părți. 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*) și alte atrbute criptografice legate de acestea.

## Parametrii de sesiune:

- **Identifierul 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 părți. 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*) și alte atrbute criptografice legate de acestea.
- **Master secret**: O secvență secretă de 48 octeți partajată de client și server.

## Parametrii de sesiune:

- **Identifierul 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 părți. 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*) și alte atrbute criptografice legate de acestea.
  - **Master secret**: O secvență secretă de 48 octeți partajată de client și server.
  - **Is resumable**: Un bit care indică dacă sesiunea poate fi utilizată pentru a iniția conexiuni noi.

## Parametrii de conexiune:

- **Nonce server și client:** Secvență aleatoare de un octet generată de fiecare parte la fiecare conexiune.

## Parametrii de conexiune:

- **Nonce server și client:** Secvență aleatoare de un octet generată de fiecare parte la fiecare conexiune.
- **MAC server:** Cheia secretă folosită în codul de autentificare al datelor scrise de server.

## Parametrii de conexiune:

- **Nonce server și client:** Secvență aleatoare de un octet generată de fiecare parte la fiecare conexiune.
- **MAC server:** Cheia secretă folosită în codul de autentificare al datelor scrise de server.
- **MAC client:** Cheia secretă folosită în codul de autentificare al datelor scrise de client.

## Parametrii de conexiune:

- **Nonce server și client:** Secvență aleatoare de un octet generată de fiecare parte la fiecare conexiune.
- **MAC server:** Cheia secretă folosită în codul de autentificare al datelor scrise de server.
- **MAC client:** Cheia secretă folosită în codul de autentificare al datelor scrise de client.
- **Cheie server:** Cheia sistemului simetric – folosită de server pentru criptarea datelor și de client pentru decriptare.

## Parametrii de conexiune:

- **Nonce server și client**: Secvență aleatoare de un octet generată de fiecare parte la fiecare conexiune.
  - **MAC server**: Cheia secretă folosită în codul de autentificare al datelor scrise de server.
  - **MAC client**: Cheia secretă folosită în codul de autentificare al datelor scrise de client.
  - **Cheie server**: Cheia sistemului simetric – folosită de server pentru criptarea datelor și de client pentru decriptare.
  - **Cheie client**: Cheia sistemului simetric – folosită de client pentru criptarea datelor și de server pentru decriptare.

## Parametrii de conexiune:

- **Vectorii de inițializare:** Dacă pentru criptare se folosesc modul *CBC*, este necesar un vector de inițializare (*VI*) pentru fiecare cheie.

## Parametrii de conexiune:

- **Vectorii de inițializare:** Dacă pentru criptare se folosesc modul *CBC*, este necesar un vector de inițializare (*VI*) pentru fiecare cheie.  
Acest câmp este completat inițial de protocolul handshaking *SSL*; apoi ultimul bloc criptat din fiecare înregistrare este păstrat drept *VI* pentru înregistrarea următoare.

## Parametrii de conexiune:

- **Vectorii de inițializare:** Dacă pentru criptare se folosește modul *CBC*, este necesar un vector de inițializare (*VI*) pentru fiecare cheie.  
Acest câmp este completat inițial de protocolul handshaking *SSL*; apoi ultimul bloc criptat din fiecare înregistrare este păstrat drept *VI* pentru înregistrarea următoare.
- **Numere de secvență:** Fiecare entitate menține separat două numere de secvență pentru mesajele trimise și primite la fiecare conexiune.  
Când este primit/trimis un mesaj de schimbare a specificațiilor de criptare, aceste numere sunt resetate.

## Parametrii de conexiune:

- **Vectorii de inițializare:** Dacă pentru criptare se folosește modul *CBC*, este necesar un vector de inițializare (*VI*) pentru fiecare cheie.  
Acest câmp este completat inițial de protocolul handshaking *SSL*; apoi ultimul bloc criptat din fiecare înregistrare este păstrat drept *VI* pentru înregistrarea următoare.
- **Numere de secvență:** Fiecare entitate menține separat două numere de secvență pentru mesajele trimise și primite la fiecare conexiune.  
Când este primit/trimis un mesaj de schimbare a specificațiilor de criptare, aceste numere sunt resetate.  
Cele două numere de secvență ale fiecărei părți sunt de tipul *unit64* și nu depășesc  $2^{64} - 1$ .

## Protocolul de înregistrare *TLS*

Oferă o conexiune securizată. Principalele proprietăți:

- 1 Conexiunea este privată. După un protocol inițial handshake care stabilește o cheie pre-master, se folosește pentru criptare un sistem simetric (*AES*, *DES*, *RC4* etc).

## Protocolul de înregistrare *TLS*

Oferă o conexiune securizată. Principalele proprietăți:

- 1 Conexiunea este privată. După un protocol inițial handshake care stabilește o cheie pre-master, se folosește pentru criptare un sistem simetric (*AES*, *DES*, *RC4* etc).
  - 2 Negocierea secretului master este sigură.



# Protocolul de înregistrare *TLS*

Oferă o conexiune securizată. Principalele proprietăți:

- 1 Conexiunea este privată. După un protocol inițial handshake care stabilește o cheie pre-master, se folosesc pentru criptare un sistem simetric (*AES*, *DES*, *RC4* etc).
- 2 Negocierea secretului master este sigură.
- 3 Identitatea celor două părți poate fi autentificată folosind sisteme de criptare cu cheie publică (*RSA*, *DSS* etc).



# Protocolul de înregistrare *TLS*

Oferă o conexiune securizată. Principalele proprietăți:

- 1 Conexiunea este privată. După un protocol inițial handshake care stabilește o cheie pre-master, se folosesc pentru criptare un sistem simetric (*AES*, *DES*, *RC4* etc).
- 2 Negocierea secretului master este sigură.
- 3 Identitatea celor două părți poate fi autentificată folosind sisteme de criptare cu cheie publică (*RSA*, *DSS* etc).
- 4 Conexiunea este sigură. Transmiterea unui mesaj include un control al integrității folosind un *HMAC* cu *MD5* sau *SHA* ca funcții de dispersie; vom nota aceste funcții cu  
*HMAC – MD5(secret, date)* respectiv  
*HMAC – SHA(secret, date)*.

# Protocolul de înregistrare *TLS*

Oferă o conexiune securizată. Principalele proprietăți:

- 1 Conexiunea este privată. După un protocol inițial handshake care stabilește o cheie pre-master, se folosesc pentru criptare un sistem simetric (*AES*, *DES*, *RC4* etc).
- 2 Negocierea secretului master este sigură.
- 3 Identitatea celor două părți poate fi autentificată folosind sisteme de criptare cu cheie publică (*RSA*, *DSS* etc).
- 4 Conexiunea este sigură. Transmiterea unui mesaj include un control al integrității folosind un *HMAC* cu *MD5* sau *SHA* ca funcții de dispersie; vom nota aceste funcții cu  
*HMAC* –  $MD5(secret, date)$  respectiv  
*HMAC* –  $SHA(secret, date)$ .  
Se pot defini și alte funcții de dispersie adiționale.

Structura unui protocol de înregistrare *TLS*

- 1** Mesajul clientului este împărțit în blocuri  $M_1, \dots, M_n$  de lungimi egale, de cel mult  $2^{14}$  octeți fiecare.

Structura unui protocol de înregistrare *TLS*

- 1 Mesajul clientului este împărțit în blocuri  $M_1, \dots, M_n$  de lungimi egale, de cel mult  $2^{14}$  octeți fiecare.
  - 2 Se aplică o funcție de compresie (optional).

### Structura unui protocol de înregistrare TLS

- 1 Mesajul clientului este împărțit în blocuri  $M_1, \dots M_n$  de lungimi egale, de cel mult  $2^{14}$  octeți fiecare.
  - 2 Se aplică o funcție de compresie (optional).
  - 3 Se calculează un cod de autentificare folosind  $HMAC - MD5$  sau  $HMAC - SHA$ ; acesta este adăugat mesajului (arhivat).

## Structura unui protocol de înregistrare *TLS*

- 1 Mesajul clientului este împărțit în blocuri  $M_1, \dots, M_n$  de lungimi egale, de cel mult  $2^{14}$  octeți fiecare.
- 2 Se aplică o funcție de compresie (optional).
- 3 Se calculează un cod de autentificare folosind *HMAC – MD5* sau *HMAC – SHA*; acesta este adăugat mesajului (arhivat).
- 4 Se cripteză folosind un sistem simetric de criptare (bloc sau fluid).  
Dacă se folosește un sistem bloc, mesajul (textul clar comprimat și codul de autentificare) este completat astfel ca lungimea lui să fie multiplu al lungimii blocului de criptare.

## Structura unui protocol de înregistrare *TLS*

- 1 Mesajul clientului este împărțit în blocuri  $M_1, \dots, M_n$  de lungimi egale, de cel mult  $2^{14}$  octeți fiecare.
- 2 Se aplică o funcție de compresie (optional).
- 3 Se calculează un cod de autentificare folosind *HMAC – MD5* sau *HMAC – SHA*; acesta este adăugat mesajului (arhivat).
- 4 Se cripteză folosind un sistem simetric de criptare (bloc sau fluid).  
Dacă se folosește un sistem bloc, mesajul (textul clar comprimat și codul de autentificare) este completat astfel ca lungimea lui să fie multiplu al lungimii blocului de criptare.
- 5 Se adaugă un antet *SSL* și rezultatul este trimis prin canal.

Pentru criptarea simetrică se folosesc: *RC4*, *DES*, *3DES* sau *AES*.

Majoritatea implementărilor *TLS* negociază sistemul și cheia, în funcție de nivelul suportat de dotarea celui mai slab dintre parteneri.

Pentru criptarea simetrică se folosește: *RC4*, *DES*, *3DES* sau *AES*.

Majoritatea implementărilor *TLS* negociază sistemul și cheia, în funcție 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.



Pentru criptarea simetrică se folosește: *RC4*, *DES*, *3DES* sau *AES*.

Majoritatea implementărilor *TLS* negociază sistemul și cheia, în funcție 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.

## Exemplu

*VerySign utilizează chei de 512 sau 1024 biți, în funcție de software-ul serverului. Cheia privată VerySign folosită la semnarea certificatelor este de 1024 biți, iar cheile de sesiune folosite în tranzacțiile TLS sunt cele mai puternice permise de legislația SUA (în general 128 sau 256 biți).*

După protocolul de înregistrare, *TLS* cuprinde alte patru protocoale:

După protocolul de înregistrare, *TLS* cuprinde alte patru protocoale:

## 1 Handshake,

După protocolul de înregistrare, *TLS* cuprinde alte patru protocoale:

- 1 Handshake,
  - 2 Alertă,

După protocolul de înregistrare, *TLS* cuprinde alte patru protocoale:

- 1 Handshake,
- 2 Alertă,
- 3 Specificații de schimbare a sistemului de criptare,

După protocolul de înregistrare, *TLS* cuprinde alte patru protocoale:

- 1 Handshake,
- 2 Alertă,
- 3 Specificații de schimbare a sistemului de criptare,
- 4 Specificații de prelucrare a datelor.

## Protocolul handshake

Generează parametrii criptografici ai stării de sesiune.

## Protocolul handshake

Generează parametrii criptografici ai stării de sesiune.

Când un client *TLS* vrea să înceapă o comunicare cu serverul, cei doi:

- Cad de acord asupra unei versiuni a protocolului,

## Protocolul handshake

Generează parametrii criptografici ai stării de sesiune.

Când un client *TLS* vrea să înceapă o comunicare cu serverul, cei doi:

- Cad de acord asupra unei versiuni a protocolului,
- Selectează algoritmii criptografici,

# Protocolul handshake

Generează parametrii criptografici ai stării de sesiune.

Când un client *TLS* vrea să înceapă o comunicare cu serverul, cei doi:

- Cad de acord asupra unei versiuni a protocolului,
- Selectează algoritmii criptografici,
- Se autentifică reciproc (optional),

## Protocolul handshake

Generează parametrii criptografici ai stării de sesiune.

Când un client *TLS* vrea să înceapă o comunicare cu serverul, cei doi:

- Cad de acord asupra unei versiuni a protocolului,
  - Selectează algoritmii criptografici,
  - Se autentifică reciproc (optional),
  - Folosesc tehnici de criptare cu cheie publică pentru a genera secretele partajate.

## Fazele protocolului handshake

## 1 Mesaje de salut.

## Fazele protocolului handshake

- 1 Mesaje de salut.
  - 2 Autentificare și schimb de chei client.

## Fazele protocolului handshake

- 1 Mesaje de salut.
  - 2 Autentificare și schimb de chei client.
  - 3 Autentificare și schimb de chei server.

## Fazele protocolului handshake

- 1 Mesaje de salut.
  - 2 Autentificare și schimb de chei client.
  - 3 Autentificare și schimb de chei server.
  - 4 Sfârșit.

## Mesaje de salut

Sunt numite *Client\_hello*, respectiv *Server\_hello*.

## Mesaje de salut

Sunt numite *Client\_hello*, respectiv *Server\_hello*.

Când un client vrea să se conecteze la un server, îl se cere să trimită un mesaj *Client\_hello*.



## Mesaje de salut

Sunt numite *Client\_hello*, respectiv *Server\_hello*.

Când un client vrea să se conecteze la un server, îl se cere să trimită un mesaj *Client\_hello*.

Acest mesaj conține:

- Versiunea protocolului *TLS* sau *SSL* pe care dorește clientul să îl folosească în timpul sesiunii.  
Aceasta trebuie să fie cea mai recentă versiune suportată de softul clientului.



## Mesaje de salut

Sunt numite *Client\_hello*, respectiv *Server\_hello*.

Când un client vrea să se conecteze la un server, îl se cere să trimită un mesaj *Client\_hello*.

Acest mesaj conține:

- Versiunea protocolului *TLS* sau *SSL* pe care dorește 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).

## Mesaje de salut

Sunt numite *Client\_hello*, respectiv *Server\_hello*.

Când un client vrea să se conecteze la un server, își cere să trimită un mesaj *Client\_hello*.

Acetă mesaj conține:

- Versiunea protocolului *TLS* sau *SSL* pe care dorește 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). Este format din:
    - Timpul curent și data – în standard UNIX pe 32 biți – conform cu ceasul intern al clientului.



## Mesaje de salut

Sunt numite *Client\_hello*, respectiv *Server\_hello*.

Când un client vrea să se conecteze la un server, îl se cere să trimită un mesaj *Client\_hello*.

Acest mesaj conține:

- Versiunea protocolului *TLS* sau *SSL* pe care dorește 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). Este format din:
  - Timpul curent și data – în standard UNIX pe 32 biți – conform cu ceasul intern al clientului.
  - 28 octeți generați de un generator de numere pseudoaleatoare.

- O listă – ordonată descrescător după preferință – a sistemelor de criptare agreate de client.

- O listă – ordonată descrescător după preferință – a sistemelor de criptare agreate de client.

Fiecare propunere este formată din două secțiuni: un tip de schimb de chei și informații despre un algoritm de criptare simetric împreună cu un cod de autentificare *MAC*.

Serverul va alege una din aceste propuner, sau – dacă nici una nu este acceptabilă – returnează eroare handshake și închide conexiunea.

- O listă – ordonată descrescător după preferință – a sistemelor de criptare agreate de client.  
Fiecare propunere este formată din două secțiuni: un tip de schimb de chei și informații despre un algoritm de criptare simetric împreună cu un cod de autentificare *MAC*.  
Serverul va alege una din aceste propuner, sau – dacă nici una nu este acceptabilă – returnează eroare handshake și închide conexiunea.
- O listă de algoritmi de compresie suportați de softul clientului, ordonată descrescător după preferință.

- O listă – ordonată descrescător după preferință – a sistemelor de criptare agreate de client.  
Fiecare propunere este formată din două secțiuni: un tip de schimb de chei și informații despre un algoritm de criptare simetric împreună cu un cod de autentificare *MAC*.  
Serverul va alege una din aceste propunerি, sau – dacă nici una nu este acceptabilă – returnează eroare handshake și închide conexiunea.
  - O listă de algoritmi de compresie suportați de softul clientului, ordonată descrescător după preferință.
  - Un *ID* de sesiune. Acest câmp poate fi gol dacă nu este accesibil nici un astfel de *ID* sau dacă clientul dorește 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 conține:

Dacă poate găsi un set acceptabil de algoritmi în *Client\_hello*, serverul trimite ca răspuns un mesaj *Server\_hello*.

Acesta conține:

- Versiunea serverului: cea mai mică versiune *TLS* sau *SSL* propusă în *Client\_hello* și cea mai mare versiune suportată de server.

Dacă poate găsi un set acceptabil de algoritmi în *Client\_hello*, serverul trimite ca răspuns un mesaj *Server\_hello*.

Acesta conține:

- Versiunea serverului: cea mai mică versiune *TLS* sau *SSL* propusă în *Client\_hello* și cea mai mare versiune suportată de server.
  - *Nonce\_Server*: 28 octeți generați aleator de server, diferiți de cei generați de client.

Dacă poate găsi un set acceptabil de algoritmi în *Client\_hello*, serverul trimite ca răspuns un mesaj *Server\_hello*.

Acesta conține:

- Versiunea serverului: cea mai mică versiune *TLS* sau *SSL* propusă în *Client\_hello* și cea mai mare versiune suportată de server.
  - *Nonce\_Server*: 28 octeți generați aleator de server, diferiți de cei generați de client.
  - O propunere de sistem de criptare, aleasă din lista propusă de client.

Dacă poate găsi un set acceptabil de algoritmi în *Client\_hello*, serverul trimite ca răspuns un mesaj *Server\_hello*.

Acesta conține:

- Versiunea serverului: cea mai mică versiune *TLS* sau *SSL* propusă în *Client\_hello* și cea mai mare versiune suportată de server.
- *Nonce\_Server*: 28 octeți generați aleator de server, diferiți de cei generați de client.
- O propunere de sistem de criptare, aleasă din lista propusă de client.
- Compresie: Un algoritm de compresie – selectat de server din lista propusă de client.

Dacă poate găsi un set acceptabil de algoritmi în *Client\_hello*, serverul trimite ca răspuns un mesaj *Server\_hello*.

Acesta conține:

- Versiunea serverului: cea mai mică versiune *TLS* sau *SSL* propusă în *Client\_hello* și cea mai mare versiune suportată de server.
- *Nonce\_Server*: 28 octeți generați aleator de server, diferiți de cei generați de client.
- O propunere de sistem de criptare, aleasă din lista propusă de client.
- Compresie: Un algoritm de compresie – selectat de server din lista propusă de client.
- *ID* de sesiune: identitatea sesiunii care corespunde conexiunii.

## Autentificare și schimb de chei

Imediat după mesajele de salut, serverul trimite:

## Autentificare și schimb de chei

Imediat după mesajele de salut, serverul trimite:

- 1** Certificatul său. De obicei este un certificat X.509v3 (sau o variantă a sa).

## Autentificare și schimb de chei

Imediat după mesajele de salut, serverul trimite:

- 1 Certificatul său. De obicei este un certificat X.509v3 (sau o variantă a sa).
  - 2 Cheia de server.

## Autentificare și schimb de chei

Imediat după mesajele de salut, serverul trimite:

- 1 Certificatul său. De obicei este un certificat X.509v3 (sau o variantă a sa).
  - 2 Cheia de server.
  - 3 Un mesaj care solicită certificatul clientului (optional).

## Autentificare și schimb de chei

Imediat după mesajele de salut, serverul trimite:

- 1 Certificatul său. De obicei este un certificat X.509v3 (sau o variantă a sa).
  - 2 Cheia de server.
  - 3 Un mesaj care solicită certificatul clientului (optional).
  - 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.

*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 conține parametrii algoritmului Diffie-Hellman, semnați de o autoritate de certificare (*CA*).



*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 conține parametrii algoritmului Diffie-Hellman, semnați de o autoritate de certificare (CA).
- **Diffie-Hellman de perioadă scurtă:** Parametrii Diffie-Hellman sunt semnați folosind protocoalele *RSA* sau *DSS* ale serverului.



*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 conține parametrii algoritmului Diffie-Hellman, semnați de o autoritate de certificare (CA).
- **Diffie-Hellman de perioadă scurtă:** Parametrii Diffie-Hellman sunt semnați folosind protocoalele *RSA* sau *DSS* ale serverului.
- **Diffie-Hellman anonim:** Parametrii Diffie-Hellman nu sunt semnați.

Parametrii cheii variază, în funcție de protocolul de schimb de chei propus de server. Ei sunt:

Parametrii cheii variază, în funcție de protocolul de schimb de chei propus de server. Ei sunt:

- RSA:  $n$  (modulul) și  $a$  (exponentul public) – pentru o cheie temporară.



Parametrii cheii variază, în funcție de protocolul de schimb de chei propus de server. Ei sunt:

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



Parametrii cheii variază, în funcție de protocolul de schimb de chei propus de server. Ei sunt:

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

Acești parametri propuși de server sunt semnați prin crearea unei amprente (cu *MD5* sau *SHA*) și apoi criptarea cu cheia privată a serverului.

Amprenta include de asemenea și nonce-urile din mesajele de salut.

Deci, dacă  $h$  este funcția de dispersie, mesajul de salut are forma

$$h(\text{Nonce\_Client} \parallel \text{Nonce\_Server} \parallel \text{Parametri\_server})$$

Deci, dacă  $h$  este funcția de dispersie, mesajul de salut are forma

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

După trimiterea certificatului de autentificare, schimbul de chei și (optional) cererea de certificare, serverul trimit un mesaj de tip *server\_hello* indicând că prima fază din protocolul handshake s-a terminat.

Deci, dacă  $h$  este funcția de dispersie, mesajul de salut are forma

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

După trimiterea certificatului de autentificare, schimbul de chei și (optional) cererea de certificare, serverul trimit un mesaj de tip *server\_hello* indicând că prima fază din protocolul handshake s-a terminat.

În continuare, el așteaptă răspunsul clientului.

## Clientul:

- #### **1 Verifică validitatea certificatului trimis de server.**

## Clientul:

- 1** Verifică validitatea certificatului trimis de server.

Dacă serverul a trimis o cerere de certificare, clientul poate trimite:

- un mesaj de certificare.

## Clientul:

- 1 Verifică validitatea certificatului trimis de server.

Dacă serverul a trimis o cerere de certificare, clientul poate trimite:

- un mesaj de certificare,
- un mesaj de alertă prin care anunță că nu are un astfel de certificat.

Acesta este doar o atenționare, de care serverul poate să țină cont sau nu.

## Clientul:

### 1 Verifică validitatea certificatului trimis de server.

Dacă serverul a trimis o cerere de certificare, clientul poate trimite:

- un mesaj de certificare,
- un mesaj de alertă prin care anunță că nu are un astfel de certificat.

Acesta este doar o atenționare, de care serverul poate să țină cont sau nu.

### 2 Trimit cheia pre-master.

## Clientul:

- 1 Verifică validitatea certificatului trimis de server.

Dacă serverul a trimis o cerere de certificare, clientul poate trimite:

- un mesaj de certificare,
- un mesaj de alertă prin care anunță că nu are un astfel de certificat.

Acesta este doar o atenționare, de care serverul poate să țină cont sau nu.

- 2 Trimit cheia pre-master. Aceasta se poate realiza
  - prin trimiterea ei criptată cu sistemul *RSA*,



## Clientul:

- 1 Verifică validitatea certificatului trimis de server.

Dacă serverul a trimis o cerere de certificare, clientul poate trimite:

- un mesaj de certificare,
- un mesaj de alertă prin care anunță că nu are un astfel de certificat.

Acesta este doar o atenționare, de care serverul poate să țină cont sau nu.

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

- prin trimiterea ei criptată cu sistemul *RSA*, sau
- transmiterea cheii publice Diffie-Hellman a clientului.

## Clientul:

- 1 Verifică validitatea certificatului trimis de server.

Dacă serverul a trimis o cerere de certificare, clientul poate trimite:

- un mesaj de certificare,
- un mesaj de alertă prin care anunță că nu are un astfel de certificat.

Acesta este doar o atenționare, de care serverul poate să țină cont sau nu.

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

- prin trimiterea ei criptată cu sistemul *RSA*, sau
- transmiterea cheii publice Diffie-Hellman a clientului.

Dacă se folosește o semnătură *RSA* sau *DSS*, atunci este obligatorie cererea de certificare a clientului.

## Parametrii cheii pre-master

- **RSA**: O cheie pre-master de 48 octeți, 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 obține cheia secretă master.

## Parametrii cheii pre-master

- **RSA:** O cheie pre-master de 48 octeți, 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 obține cheia secretă master.
  - **Diffie-Hellman:** Valoarea publică ( $g^x \bmod p$ ) a cheii clientului.

## Parametrii cheii pre-master

- **RSA:** O cheie pre-master de 48 octeți, 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 obține cheia secretă master.
  - **Diffie-Hellman:** Valoarea publică ( $g^x \bmod p$ ) a cheii clientului.

Dacă și serverul folosește tot protocolul Diffie-Hellman, cei doi deduc imediat cheia comună pre-master.

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

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

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

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

Valoarea “*master\\_secret*” folosește o sursă de entropie pentru a genera valori aleatoare cu diverse scopuri: *MAC*-uri, chei secrete, valori de inițializare (*IV*).

Indiferent de varianta folosită pentru cheia pre-master, “*master\_secret*” are 48 octeți.

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

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

Valoarea “*master\\_secret*” folosește o sursă de entropie pentru a genera valori aleatoare cu diverse scopuri: *MAC*-uri, chei secrete, valori de inițializare (*IV*).

Indiferent de varianta folosită pentru cheia pre-master, “*master\_secret*” are 48 octeți.

*PRF* este un generator de numere pseudo-aleatoare.

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

$$SK_M = PRF(\text{pre}-\text{master}, \text{"master\_secret"}, \text{Nonce}_\text{Client} + \text{Nonce}_\text{Server})$$

Valoarea “*master\_secret*” folosește o sursă de entropie pentru a genera valori aleatoare cu diverse scopuri: *MAC*-uri, chei secrete, valori de inițializare (*IV*).

Indiferent de varianta folosită pentru cheia pre-master, “*master\_secret*” are 48 octeți.

*PRF* este un generator de numere pseudo-aleatoare.

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

## Sfârșit

Clientul și serverul actualizează specificațiile sistemelor de criptare cu algoritmii de criptare, cheile și funcțiile de dispersie agreate de ambele părți.

## Sfârșit

Clientul și serverul actualizează specificațiile sistemelor de criptare cu algoritmii de criptare, cheile și funcțiile de dispersie agreate de ambele părți.

Apoi, clientul trimite un mesaj de încheiere pentru a verifica aceste date.



## Sfârșit

Clientul și serverul actualizează specificațiile sistemelor de criptare cu algoritmii de criptare, cheile și funcțiile de dispersie agreate de ambele părți.

Apoi, clientul trimite un mesaj de încheiere pentru a verifica aceste date.

Acesta este primul mesaj protejat cu specificațiile negociate, de forma:

$$MD5("master\_secret" \parallel pad_2 \parallel MD5(handshake\_mesaj \parallel Expeditor \parallel "master\_secret" \parallel pad_1));$$

## Sfârșit

Clientul și serverul actualizează specificațiile sistemelor de criptare cu algoritmii de criptare, cheile și funcțiile de dispersie agreate de ambele părți.

Apoi, clientul trimite un mesaj de încheiere pentru a verifica aceste date.

Acesta este primul mesaj protejat cu specificațiile negociate, de forma:

$MD5("master\_secret" \parallel pad_2 \parallel MD5(handshake\_mesg) \parallel Expeditor \parallel "master\_secret" \parallel pad_1))$

sau

$SHA("master\_secret" \parallel pad_2 \parallel SHA(handshake\_mesai \parallel Expeditor \parallel "master\_secret" \parallel pad_1))$

- $pad_1$  și  $pad_2$  sunt valorile definite de  $MAC$ ,

- $pad_1$  și  $pad_2$  sunt valorile definite de *MAC*,
  - *handshake\_mesaj* se referă la toate mesajele handshake schimbate până acum,

- $pad_1$  și  $pad_2$  sunt valorile definite de *MAC*,
  - *handshake\_mesaj* se referă la toate mesajele handshake schimilate până acum,
  - *Expeditor* se referă la un cod care identifică expeditorul: 434C4E54 dacă este clientul, 53525652 dacă este serverul.



- $pad_1$  și  $pad_2$  sunt valorile definite de *MAC*,
- *handshake\_mesaj* se referă la toate mesajele handshake schimbată până acum,
- *Expeditor* se referă la un cod care identifică expeditorul: 434C4E54 dacă este clientul, 53525652 dacă este serverul.

După acest ultim mesaj, clientul și serverul pot începe să comunice, transmițând date confidențiale.

## Calculul cheii

Materialul cheii este generat astfel:

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

## Calculul cheii

Materialul cheii este generat astfel:

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

unde:

- **MS.PS:** Parametrii de securitate ai secretului master.  
*Secretul master* este o secvență de 48 octeți, partajată de cei doi parteneri conectați.

## Calculul cheii

Materialul cheii este generat astfel:

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

unde-

- *MS.PS*: Parametrii de securitate ai secretului master.  
*Secretul master* este o secvență de 48 octeți, partajată de cei doi parteneri conectați.
  - “*cheie expandată*”: Reprezentarea ASCII pentru “*key expansion*”; este folosit ca marcă de identificare.

## Calculul cheii

Materialul cheii este generat astfel:

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

under-

- *MS.PS*: Parametrii de securitate ai secretului master.  
*Secretul master* este o secvență de 48 octeți, partajată de cele doi parteneri conectați.
  - “*cheie expandata*”: Reprezentarea ASCII pentru “*key expansion*”; este folosit ca marcă de identificare.
  - $N_S$  este un nonce de 32 octeți generați de server. Parametrii săi de securitate sunt  $N_S.PS$ .



## Calculul cheii

Materialul cheii este generat astfel:

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

unde:

- $MS.PS$ : Parametrii de securitate ai secretului master.  
*Secretul master* este o secvență de 48 octeți, partajată de cei doi parteneri conectați.
- “*cheie expandata*”: Reprezentarea ASCII pentru “*key expansion*”; este folosit ca marcă de identificare.
- $N_S$  este un nonce de 32 octeți generați de server. Parametrii săi de securitate sunt  $N_S.PS$ .
- Similar,  $N_C$  este un nonce de 32 octeți generați de client. Parametrii săi de securitate sunt  $N_C.PS$ .

## Calculul cheii

Materialul cheii este generat astfel:

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

unde:

- *MS.PS*: Parametrii de securitate ai secretului master.  
*Secretul master* este o secvență de 48 octeți, partajată de cei doi parteneri conectați.
- “*cheie expandata*”: Reprezentarea ASCII pentru “*key expansion*”; este folosit ca marcă de identificare.
- *N<sub>S</sub>* este un nonce de 32 octeți generați de server. Parametrii săi de securitate sunt *N<sub>S</sub>.PS*.
- Similar, *N<sub>C</sub>* este un nonce de 32 octeți generați de client. Parametrii săi de securitate sunt *N<sub>C</sub>.PS*.
- *PRF* este o funcție pseudoaleatoare care amprentează secretele.

$K_{bloc}$  generează suficient de mult material pentru a construi:

$K_{bloc}$  generează suficient de mult material pentru a construi:

- ## **1 Componentă secretă din MAC client;**

$K_{bloc}$  generează suficient de mult material pentru a construi:

- 1 Componentă secretă din *MAC* client;
  - 2 Componentă secretă din *MAC* server;

$K_{bloc}$  generează suficient de mult material pentru a construi:

- 1 Componență secretă din *MAC* client;
- 2 Componență secretă din *MAC* server;
- 3 Cheia client;

$K_{bloc}$  generează suficient de mult material pentru a construi:

- 1 Componenta secretă din *MAC* client;
  - 2 Componenta secretă din *MAC* server;
  - 3 Cheia client;
  - 4 Cheia server.

$K_{bloc}$  generează suficient de mult material pentru a construi:

- 1 Componență secretă din *MAC* client;
- 2 Componență secretă din *MAC* server;
- 3 Cheia client;
- 4 Cheia server.

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

## Construirea funcției pseudoaleatoare *PRF*

Un PRF se generează în doi pași:

- 1 Se construiește o funcție de expansiune  $P_{hash}(secret, data)$  pentru a expanda un secret.

## Construirea funcției pseudoaleatoare *PRF*

Un PRF se generează în doi pași:

- 1 Se construiește o funcție de expansiune  $P_{hash}(secret, data)$  pentru a expanda un secret.  
Ea se bazează pe o funcție de dispersie și se definește

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

## Construirea funcției pseudoaleatoare *PRF*

Un PRF se generează în doi pași:

- 1 Se construiește o funcție de expansiune  $P_{hash}(secret, data)$  pentru a expanda un secret.  
Ea se bazează pe o funcție de dispersie și se definește

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

unde  $A(\cdot)$  este definit

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

## Construirea funcției pseudoaleatoare *PRF*

Un PRF se generează în doi pași:

- 1 Se construiește o funcție de expansiune  $P_{hash}(secret, data)$  pentru a expanda un secret.  
Ea se bazează pe o funcție de dispersie și se definește

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

unde  $A(\cdot)$  este definit

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

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

# Construirea *PRF*

2 Funcția pseudoaleatoare *PRF* folosită de *TLS* se obține astfel:

## Construirea *PRF*

- 2 Funcția pseudoaleatoare *PRF* folosită de *TLS* se obține astfel:

  - 1 Se divide un secret în două părți egale;

## Construirea *PRF*

- 2 Funcția pseudoaleatoare *PRF* folosită de *TLS* se obține astfel:

  - 1 Se divide un secret în două părți egale;
  - 2 Una din jumătăți este utilizată pentru a genera date cu  $P_{MD5}$ , iar cealaltă jumătate – pentru a genera date cu  $P_{SHA-1}$ .

## Construirea *PRF*

- 2 Funcția pseudoaleatoare  $PRF$  folosită de  $TLS$  se obține astfel:

  - 1 Se divide un secret în două părți egale;
  - 2 Una din jumătăți este utilizată pentru a genera date cu  $P_{MD5}$ , iar cealaltă jumătate – pentru a genera date cu  $P_{SHA-1}$ .
  - 3 Cele două ieșiri sunt  $XOR$ -ate.

Deci

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

Deci

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

unde

- Secretul  $X$  este partajat în două părți egale:  $X = X_1 \parallel X_2$ .

Deci

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

unde

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

Deci

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

unde

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

### Exemplu

"plano tx" este procesată concatenând octetii  
70 6C 61 6E 6F 20 74 78 cu valoarea din  $Y$ .

## Construirea MAC-ului

MAC-ul folosit de *TLS*:

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

## Construirea MAC-ului

MAC-ul folosit de *TLS*:

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

unde

- *hash* este algoritmul de dispersie specificat de parametrii de securitate,

## Construirea MAC-ului

MAC-ul folosit de *TLS*:

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

unde

- *hash* este algoritmul de dispersie specificat de parametrii de securitate,
  - *num\_seq* este numărul de secvență pentru înregistrarea corespunzătoare,

## Construirea *MAC*-ului

*MAC*-ul folosit de *TLS*:

$$\text{HMAC}_{\text{hash}}(\text{secret}_{\text{MAC}}, \text{num\_seq} \parallel \text{Tip} \parallel \text{Versiune} \parallel \text{Lungime} \parallel \text{Fragment})$$

unde

- *hash* este algoritmul de dispersie specificat de parametrii de securitate,
- *num\_seq* este numărul de secvență pentru înregistrarea corespunzătoare,
- *Tip*, *Versiune* și *Lungime* se referă la caracteristicile respective ale algoritmului de compresie specificat în *TLS*, iar *Fragment* este o porțiune compresată din datele asociate.



## 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** și **avertizări**.



## 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** și **avertizări**.

Un mesaj de *alertă totală* indică o conexiune atât de rea încât necesită încheierea ei imediată.



## 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** și **avertizări**.

Un mesaj de **alertă totală** indică o conexiune atât de rea încât necesită încheierea ei imediată.

O **avertizare** indică existența anumitor probleme de conexiune.



## 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** și **avertizări**.

Un mesaj de **alertă totală** indică o conexiune atât de rea încât necesită încheierea ei imediată.

O **avertizare** indică existența anumitor probleme de conexiune.

Mesajele de alertă sunt criptate și arhivate conform specificațiilor date de starea curentă a conexiunii.



## Câteva mesaje de alertă totale

- **Unexpected\_message:** A apărut un mesaj având un tip care nu corespunde protocolului.

## Câteva mesaje de alertă totale

- **Unexpected\_message**: A apărut un mesaj având un tip care nu corespunde protocolului.
  - **Bad\_record\_mac**: S-a primit o înregistrare cu *MAC* incorrect.



## Câteva mesaje de alertă totale

- **Unexpected\_message**: A apărut un mesaj având un tip care nu corespunde protocolului.
- **Bad\_record\_mac**: S-a primit o înregistrare cu *MAC* incorrect.
- **Decryption\_failed**: Un text criptat *TLS* este decriptat într-un mod incorrect.

## Câteva mesaje de alertă totale

- **Unexpected\_message**: A apărut un mesaj având un tip care nu corespunde protocolului.
  - **Bad\_record\_mac**: S-a primit o înregistrare cu *MAC* incorrect.
  - **Decryption\_failed**: Un text criptat *TLS* este decriptat într-un mod incorrect.
  - **Decrypt\_error**: A eşuat o operaţie criptografică din protocolul handshake.



## Câteva mesaje de alertă totale

- **Unexpected\_message**: A apărut un mesaj având un tip care nu corespunde protocolului.
- **Bad\_record\_mac**: S-a primit o înregistrare cu *MAC* incorect.
- **Decryption\_failed**: Un text criptat *TLS* este decriptat într-un mod incorect.
- **Decrypt\_error**: A eșuat o operație criptografică din protocolul handshake.
- **Decompression\_failure**: Funcția de dezarchivare primește o intrare incorectă.



## Câteva mesaje de alertă totale

- **Unexpected\_message:** A apărut un mesaj având un tip care nu corespunde protocolului.
- **Bad\_record\_mac:** S-a primit o înregistrare cu *MAC* incorect.
- **Decryption\_failed:** Un text criptat *TLS* este decriptat într-un mod incorect.
- **Decrypt\_error:** A eșuat o operație criptografică din protocolul handshake.
- **Decompression\_failure:** Funcția de dezarchivare primește o intrare incorectă.
- **Handshake\_failure:** Expeditorul nu a putut negocia un set acceptabil de parametri de securitate.



## Câteva mesaje de alertă totale

- **Unexpected\_message:** A apărut un mesaj având un tip care nu corespunde protocolului.
- **Bad\_record\_mac:** S-a primit o înregistrare cu *MAC* incorect.
- **Decryption\_failed:** Un text criptat *TLS* este decriptat într-un mod incorect.
- **Decrypt\_error:** A eșuat o operație criptografică din protocolul handshake.
- **Decompression\_failure:** Funcția de dezarchivare primește o intrare incorectă.
- **Handshake\_failure:** Expeditorul nu a putut negocia un set acceptabil de parametri de securitate.
- **Illegal\_parameter:** Inconsistență între câmpul atașat unui parametru și alte câmpuri.



## Protocolul de schimbare a specificațiilor

Un protocol de schimbare a specificațiilor (*Change Cipher Spec Protocol*) semnalează schimbarea strategiei de criptare.

## Protocolul de schimbare a specificațiilor

Un protocol de schimbare a specificațiilor (*Change Cipher Spec Protocol*) semnalează schimbarea strategiei de criptare.

Protocolul constă dintr-un octet – criptat și arhivat – în starea de conectare curentă.



## Protocolul de schimbare a specificațiilor

Un protocol de schimbare a specificațiilor (*Change Cipher Spec Protocol*) semnalează schimbarea strategiei de criptare.

Protocolul constă dintr-un octet – criptat și arhivat – în starea de conectare curentă.

Când unul din parteneri trimite un astfel de anunț, el notifică colegului său că tot ce urmează este protejat cu noile chei și specificații de securitate negociate la început.



## Protocolul de schimbare a specificațiilor

Un protocol de schimbare a specificațiilor (*Change Cipher Spec Protocol*) semnalează schimbarea strategiei de criptare.

Protocolul constă dintr-un octet – criptat și arhivat – în starea de conectare curentă.

Când unul din parteneri trimite un astfel de anunț, el notifică colegului său că tot ce urmează este protejat cu noile chei și specificații de securitate negociate la început.

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



## Concluzii

O rețea *VPN* dotată cu *SSL* – implementat prin standardul *TLS* – are următoarele caracteristici:



## Concluzii

O rețea VPN dotată cu *SSL* – implementat prin standardul *TLS* – are următoarele caracteristici:

- Utilizează *SSL* și tehnologie proxy pentru a oferi utilizatorilor acces autorizat și sigur la *HTTP*, aplicații server și diverse resurse partajate.  
Asigură transportul datelor și sesiuni sigure între orice browser și orice server proxy dintr-un gateway *VPN* cu *SSL* implementat.



## Concluzii

O rețea VPN dotată cu *SSL* – implementat prin standardul *TLS* – are următoarele caracteristici:

- Utilizează *SSL* și tehnologie proxy pentru a oferi utilizatorilor acces autorizat și sigur la *HTTP*, aplicații server și diverse resurse partajate.  
Asigură transportul datelor și sesiuni sigure între orice browser și orice server proxy dintr-un gateway VPN cu *SSL* implementat.
- *SSL*-ul unui VPN funcționează ca un proxy pentru ambele părți; nu există niciodată o conexiune directă spre o rețea privată.  
Accesul este permis numai spre aplicațiile oferite de *SSL*.



## Concluzii

O rețea VPN dotată cu *SSL* – implementat prin standardul *TLS* – are următoarele caracteristici:

- Utilizează *SSL* și tehnologie proxy pentru a oferi utilizatorilor acces autorizat și sigur la *HTTP*, aplicații server și diverse resurse partajate.  
Asigură transportul datelor și sesiuni sigure între orice browser și orice server proxy dintr-un gateway VPN cu *SSL* implementat.
- *SSL*-ul unui VPN funcționează ca un proxy pentru ambele părți; nu există niciodată o conexiune directă spre o rețea privată.  
Accesul este permis numai spre aplicațiile oferite de *SSL*.
- *SSL* funcționează ca un gateway: stabilește o conexiune între browserul Web al clientului dintr-o rețea nesecurizată și *SSL* proxy din VPN, și altă conexiune între *SSL* proxy din VPN și utilizatorul dintr-o rețea securizată.

## Concluzii

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



## Concluzii

- *SSL* asigură că un utilizator autorizat are acces numai la anumite resurse, cele alocate de software-ul companiei prin aplicația *SSL* și integrate de gestiunea traficului.
- Serverele proxy sparg conexiunea *TCP/IP* dintre client și server, fără a mai forwarda și adresa *IP* a pachetelor. Astfel, ele oferă o securitate la trecerea prin rețele nesigure, ascunzând adresele *IP* ale utilizatorilor din rețelele securizate.



## Concluzii

- *SSL* asigură că un utilizator autorizat are acces numai la anumite resurse, cele alocate de software-ul companiei prin aplicația *SSL* și integrate de gestiunea traficului.
- Serverele proxy sparg conexiunea *TCP/IP* dintre client și server, fără a mai forwarda și adresa *IP* a pachetelor. Astfel, ele oferă o securitate la trecerea prin rețele nesigure, ascunzând adresele *IP* ale utilizatorilor din rețelele securizate.

Într-o rețea nesecurizată este vizibilă numai adresa *IP* publică a serverului proxy.

#### Slăbiciuni ale unei rețele VPN dotate cu SSL

- Computerele stochează în locații nesecurizate diverse informații importante.

Slăbiciuni ale unei rețele VPN dotate cu *SSL*

- Computerele stochează în locații nesecurizate diverse informații importante.
  - După închiderea conexiunii, parolele rămân în zone publice.



Concluzii

## Slăbiciuni ale unei rețele VPN dotate cu *SSL*

- Computerele stochează în locații nesecurizate diverse informații importante.
- După închiderea conexiunii, parolele rămân în zone publice.
- Parolele clienților sunt stocate de browser.



## Concluzii

## Slăbiciuni ale unei rețele VPN dotate cu SSL

- Computerele stochează în locații nesecurizate diverse informații importante.
- După închiderea conexiunii, parolele rămân în zone publice.
- Parolele clienților sunt stocate de browser.
- Date importante (informațiile cache, *URL*-urile, cookies, informații din history) create în timpul sesiunii pot rămâne pe calculatoare publice după închiderea completă a unei sesiuni.



## Concluzii

## Slăbiciuni ale unei rețele VPN dotate cu SSL

- Computerele stochează în locații nesecurizate diverse informații importante.
- După închiderea conexiunii, parolele rămân în zone publice.
- Parolele clienților sunt stocate de browser.
- Date importante (informațiile cache, *URL*-urile, cookies, informații din history) create în timpul sesiunii pot rămâne pe calculatoare publice după închiderea completă a unei sesiuni.
- Fișiere downloadate rămân stocate în directoare temporare pe calculatoare publice.



## Concluzii

## Slăbiciuni ale unei rețele VPN dotate cu SSL

- Computerele stochează în locații nesecurizate diverse informații importante.
- După închiderea conexiunii, parolele rămân în zone publice.
- Parolele clienților sunt stocate de browser.
- Date importante (informațiile cache, *URL*-urile, cookies, informații din history) create în timpul sesiunii pot rămâne pe calculatoare publice după închiderea completă a unei sesiuni.
- Fișiere downloadate rămân stocate în directoare temporare pe calculatoare publice.
- Clienții uită (sau nu consideră necesar) să dea logout.



## Concluzii

## Slăbiciuni ale unei rețele VPN dotate cu SSL

- Computerele stochează în locații nesecurizate diverse informații importante.
- După închiderea conexiunii, parolele rămân în zone publice.
- Parolele clienților sunt stocate de browser.
- Date importante (informațiile cache, *URL*-urile, cookies, informații din history) create în timpul sesiunii pot rămâne pe calculatoare publice după închiderea completă a unei sesiuni.
- Fișiere downloadate rămân stocate în directoare temporare pe calculatoare publice.
- Clienții uită (sau nu consideră necesar) să dea logout.
- Clientul următor care folosește calculatorul public are acces la aplicații.



## Concluzii

## Slăbiciuni ale unei rețele VPN dotate cu SSL

- Computerele stochează în locații nesecurizate diverse informații importante.
- După închiderea conexiunii, parolele rămân în zone publice.
- Parolele clienților sunt stocate de browser.
- Date importante (informațiile cache, *URL*-urile, cookies, informații din history) create în timpul sesiunii pot rămâne pe calculatoare publice după închiderea completă a unei sesiuni.
- Fișiere downloadate rămân stocate în directoare temporare pe calculatoare publice.
- Clienții uită (sau nu consideră necesar) să dea logout.
- Clientul următor care folosește calculatorul public are acces la aplicații.
- Pot fi transferați viruși și viermi de pe calculatoare publice pe retele interne.



# Protocolul *SET*

Construit pentru tranzacții financiare pe Internet, pentru a oferi securitatea plășilor cu card.



# Protocolul *SET*

Construit pentru tranzacții financiare pe Internet, pentru a oferi securitatea plășilor cu card.

El a rezultat prin unirea eforturilor de cercetare ale firmelor Visa și MasterCard, ca o metodă de asigurare a tranzacțiiilor electronice.



## Protocolul *SET*

Construit pentru tranzacții financiare pe Internet, pentru a oferi securitatea plășilor cu card.

El a rezultat prin unirea eforturilor de cercetare ale firmelor Visa și MasterCard, ca o metodă de asigurare a tranzacțiiilor electronice.

La sfârșitul anilor "90 *SET* a fost aprobat drept un protocol de credit standard, dar nu a putut fi impus din cauza costurilor și a problemelor ridicate de distribuția certificatelor utilizatorilor.



## Protocolul *SET*

Construit pentru tranzacții financiare pe Internet, pentru a oferi securitatea plășilor cu card.

El a rezultat prin unirea eforturilor de cercetare ale firmelor Visa și MasterCard, ca o metodă de asigurare a tranzacțiiilor electronice.

La sfârșitul anilor "90 *SET* a fost aprobat drept un protocol de credit standard, dar nu a putut fi impus din cauza costurilor și a problemelor ridicate de distribuția certificatelor utilizatorilor.

Este considerat ca fiind un protocol ideal din punct de vedere al certificării, al semnăturilor digitale și al algoritmilor de criptare care securizează cardurile de credit în tranzacțiiile Internet.



*SET* utilizează criptografia pentru asigurarea următoarelor servicii de securitate:

- 1 Autentificarea unui deținător de card ca utilizator legitim al unui cont din care se pot efectua operații financiare.  
Se realizează folosind sistemul de criptare *RSA* și certificate *X.509*.



*SET* utilizează criptografia pentru asigurarea următoarelor servicii de securitate:

- 1 Autentificarea unui deținător de card ca utilizator legitim al unui cont din care se pot efectua operații financiare.  
Se realizează folosind sistemul de criptare *RSA* și certificate *X.509*.
- 2 Autentificarea unei instituții de plată (bancă).



*SET* utilizează criptografia pentru asigurarea următoarelor servicii de securitate:

- 1 Autentificarea unui deținător de card ca utilizator legitim al unui cont din care se pot efectua operații financiare.  
Se realizează folosind sistemul de criptare *RSA* și certificate *X.509*.
- 2 Autentificarea unei instituții de plată (bancă).
- 3 Confidențialitatea operațiilor de plată, pe baza sistemului de criptare *DES*.



*SET* utilizează criptografia pentru asigurarea următoarelor servicii de securitate:

- 1 Autentificarea unui deținător de card ca utilizator legitim al unui cont din care se pot efectua operații financiare.  
Se realizează folosind sistemul de criptare *RSA* și certificate *X.509*.
- 2 Autentificarea unei instituții de plată (bancă).
- 3 Confidențialitatea operațiilor de plată, pe baza sistemului de criptare *DES*.
- 4 Integritatea documentelor de plată, folosind funcția de dispersie *SHA – 1*.

## Participanții unei tranzacție de plată cu card

- 1 **Deținător card:** Folosește un card de plată emis de societate numită “*Emitent*”.

## Participanții unei tranzacție de plată cu card

- 1 **Deținător card**: Folosește un card de plată emis de societate numită “*Emisor*”.
- 2 **Emisor**: Instituție financiară care stabilește un cont pentru un utilizator și îi eliberează acestuia un card de plăți. Garantează onorarea – cu ajutorul cardului – a tranzacțiilor financiare autorizate.

## Participanții unei tranzacție de plată cu card

- 1 **Deținător card**: Folosește un card de plată emis de societate numită “*Emisor*”.
- 2 **Emisor**: Instituție financiară care stabilește un cont pentru un utilizator și îi eliberează acestuia un card de plată. Garantează onorarea – cu ajutorul cardului – a tranzacțiilor financiare autorizate.
- 3 **Comerçiant**: Oferă spre vânzare bunuri sau servicii în schimbul unei plăți.

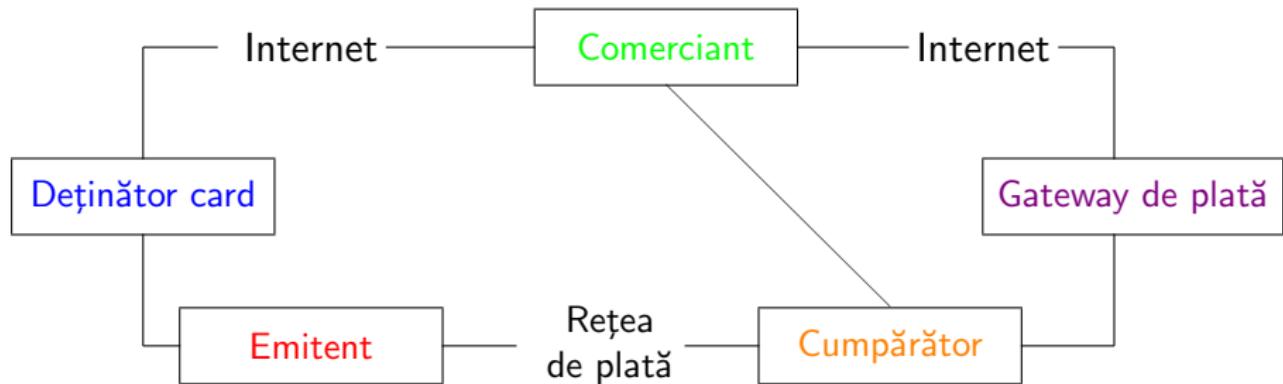


## Participanții unei tranzacție de plată cu card

- 1 **Deținător card**: Folosește un card de plată emis de societate numită “*Emisor*”.
- 2 **Emisor**: Instituție financiară care stabilește un cont pentru un utilizator și îi eliberează acestuia un card de plată. Garantează onorarea – cu ajutorul cardului – a tranzacțiilor financiare autorizate.
- 3 **Comerçant**: Oferă spre vânzare bunuri sau servicii în schimbul unei plăti.
- 4 **Cumpărător**: Instituția financiară care stabilește un cont cu un comerciant și procesează plătile autorizate de cardul asociat acestui cont.

## Participanții unei tranzacție de plată cu card

- 1 **Deținător card**: Folosește un card de plată emis de societate numită “*Emisor*”.
- 2 **Emisor**: Instituție financiară care stabilește un cont pentru un utilizator și îi eliberează acestuia un card de plată. Garantează onorarea – cu ajutorul cardului – a tranzacțiilor financiare autorizate.
- 3 **Comerçant**: Oferă spre vânzare bunuri sau servicii în schimbul unei plăți.
- 4 **Cumpărător**: Instituția financiară care stabilește un cont cu un comerciant și procesează plățile autorizate de cardul asociat acestui cont.
- 5 **Gateway de plată**: Sistem cu care operează cumpărătorul sau o componentă comună stabilită de comun acord, al cărui rol este de a procesa toate mesajele de plată.



Tranzacții *SET*Pașii unei tranzacții *SET*:

- 1 Cumpărătorul (banca cumpărător) solicită un portofel digital dela o bancă și obține un certificat digital, un card sau alt document autentificat de banca emitentă.

## Pașii unei tranzacții *SET*:

- 1 Cumpărătorul (banca cumpărător) solicită un portofel digital dela o bancă și obține un certificat digital, un card sau alt document autentificat de banca emitentă.
- 2 Comerçantul obține un certificat digital de la banca cumpărător asociată.

## Pașii unei tranzacții *SET*:

- 1 Cumpărătorul (banca cumpărător) solicită un portofel digital dela o bancă și obține un certificat digital, un card sau alt document autentificat de banca emitentă.
- 2 Comerçantul obține un certificat digital de la banca cumpărător asociată.  
Dacă dorește să facă vânzări on-line folosind *SET*, el va trebui să dețină două certificate valide:  $CERT_{Com}$  – emis de cumpărătorul asociat și  $CERT_{Gateway}$  – de la gateway-ul de plată.

Tranzacții *SET*Pașii unei tranzacții *SET*:

- 1 Cumpărătorul (banca cumpărător) solicită un portofel digital dela o bancă și obține un certificat digital, un card sau alt document autentificat de banca emitentă.
- 2 Comerçantul obține un certificat digital de la banca cumpărător asociată.  
Dacă dorește să facă vânzări on-line folosind *SET*, el va trebui să dețină două certificate valide:  $CERT_{Com}$  – emis de cumpărătorul asociat și  $CERT_{Gateway}$  – 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 și completează fișa electronică de comandă.

Plata se face folosind din portofelul digital primit de la băncă.



## Tranzacții *SET*

- 4 Clientul trimite comerciantului fișă electronică completată (*FE*), însăcăpată de instrucțiunile de plată (*OP*) – ambele semnate.

Tranzacții *SET*

- 4 Clientul trimite comerciantului fișă electronică completată (*FE*), însorită de instrucțiunile de plată (*OP*) – ambele semnate.
- 5 Comerciantul primește *FE* și *OP*.  
Nu are acces însă la *OP* deoarece aceasta este criptată cu cheia publică a gateway-ului de plată.

Tranzacții *SET*

- 4 Clientul trimite comerciantului fișă electronică completată (*FE*), însătoare de instrucțiunile de plată (*OP*) – ambele semnate.
- 5 Comerciantul primește *FE* și *OP*.  
Nu are acces însă la *OP* deoarece aceasta este criptată cu cheia publică a gateway-ului de plată.
- 6 El trimite *FE*, *OP* și datele sale de autentificare la banca asociată gateway-ului și care va face plătile.

Tranzacții *SET*

- 4 Clientul trimite comerciantului fișă electronică completată (*FE*), însăși de instrucțiunile de plată (*OP*) – ambele semnate.
- 5 Comerciantul primește *FE* și *OP*.  
Nu are acces însă la *OP* deoarece aceasta este criptată cu cheia publică a gateway-ului de plată.
- 6 El trimite *FE*, *OP* și datele sale de autentificare la banca asociată gateway-ului și care va face plătile.  
Gateway-ul de plată translatează mesajul *SET* într-un protocol recunoscut și folosit de rețeaua care a emis cardul. El primește certificatele de la *SET*-ul asociat *Root CA* (un *CA* aflat în vârful ierarhiei de încredere).

Tranzacții *SET*

- 4 Clientul trimite comerciantului fișă electronică completată (*FE*), însătoare de instrucțiunile de plată (*OP*) – ambele semnate.
- 5 Comerciantul primește *FE* și *OP*.  
Nu are acces însă la *OP* deoarece aceasta este criptată cu cheia publică a gateway-ului de plată.
- 6 El trimite *FE*, *OP* și datele sale de autentificare la banca asociată gateway-ului și care va face plătile.  
Gateway-ul de plată translatează mesajul *SET* într-un protocol recunoscut și folosit de rețeaua care a emis cardul. El primește 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* și transmite informațiile băncii care efectuează plata.  
Banca trimite o cerere de autorizare pe care banca emitentă a cardului.

- 8 Banca emitentă primește cererea de autorizare ca pe orice solicitare de tranzacție fizică.

Tranzacții *SET*

- 8 Banca emitentă primește cererea de autorizare ca pe orice solicitare de tranzacție fizică.

Cercetează contul posesorului de card și – în funcție de informațiile pe care le deține – aproba sau respinge plata. Acest mesaj este trimis înapoi spre banca care efectuează plăți.

Tranzacții *SET*

- 8 Banca emitentă primește cererea de autorizare ca pe orice solicitare de tranzacție fizică.  
Cercetează contul posesorului de card și – în funcție de informațiile pe care le deține – aproba sau respinge plata. Acest mesaj este trimis înapoi spre banca care efectuează plăți.
- 9 Mesajul trece din nou prin gateway-ul de plată și este criptat înainte ca autorizația să fie trimisă comerciantului.

Tranzacții *SET*

- 8 Banca emitentă primește cererea de autorizare ca pe orice solicitare de tranzacție fizică.  
Cercetează contul posesorului de card și – în funcție de informațiile pe care le deține – aproba sau respinge plata. Acest mesaj este trimis înapoi spre banca care efectuează plăți.
- 9 Mesajul trece din nou prin gateway-ul de plată și este criptat înainte ca autorizația să fie trimisă comerciantului.  
Dacă acesta are confirmarea validității cardului de plată (inclusiv dacă suma este acoperită de creditul cardului), va livra solicitantului produsele cumpărate.

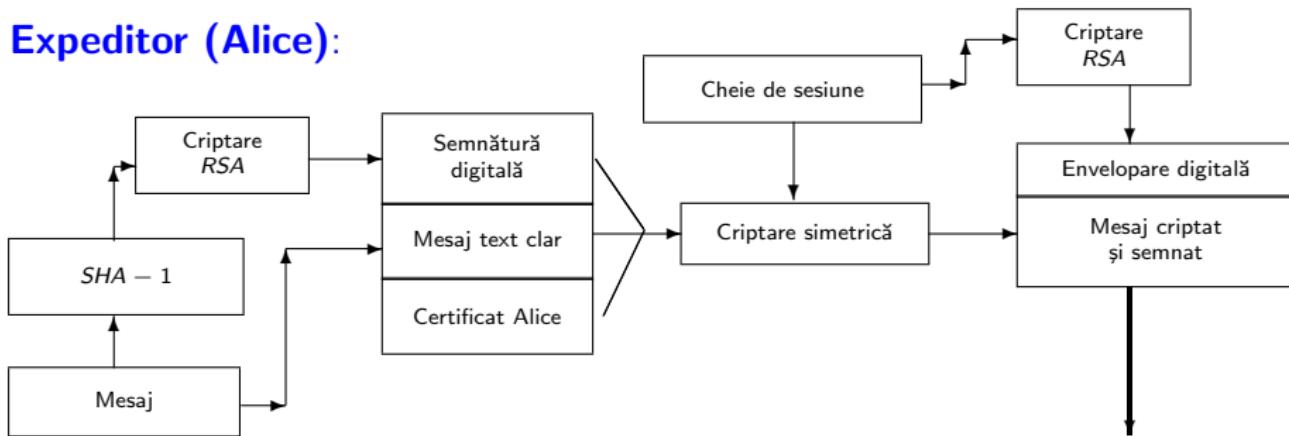


Tranzacții SET

# Autentificarea și confidențialitatea SET

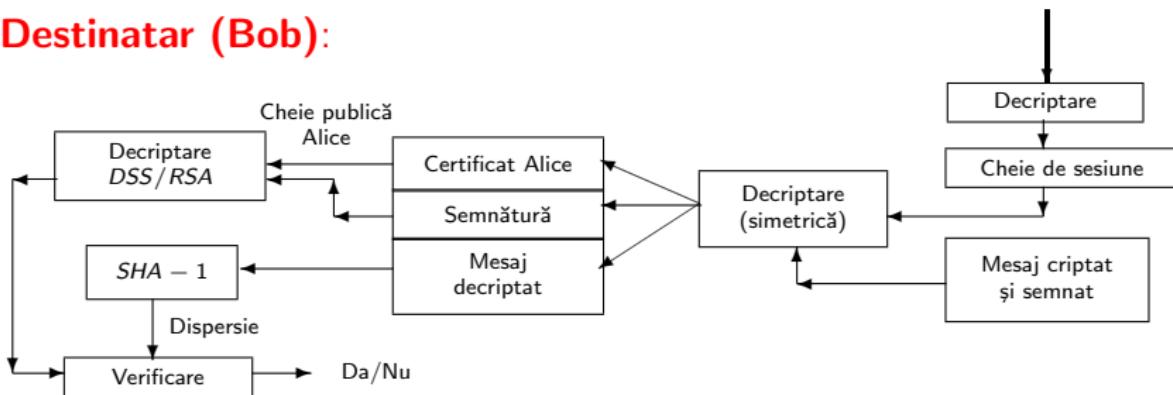
Cuprinde următoarele etape:

## Expeditor (Alice):





## Tranzacții SET

**Destinatar (Bob):**

Tranzacții *SET*

- 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.

Tranzacții *SET*

- 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 funcția de dispersie *SHA – 1* și semnat cu *RSA* bazat pe cheia privată a lui *Alice*. Se obține “*Semnătura digitală*”.

Tranzacții *SET*

- 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 funcția de dispersie *SHA – 1* și semnat cu *RSA* bazat pe cheia privată a lui *Alice*. Se obține “*Semnătura digitală*”.
- 3 Mesajul – text clar – este concatenat cu semnătura digitală și cu certificatul lui *Alice*.

Tranzacții *SET*

- 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 funcția de dispersie *SHA – 1* și semnat cu *RSA* bazat pe cheia privată a lui *Alice*. Se obține “*Semnătura digitală*”.
- 3 Mesajul – text clar – este concatenat cu semnătura digitală și cu certificatul lui *Alice*.
- 4 Ceea ce s-a obținut se cripteză cu un algoritm simetric (*DES*) folosind cheia secretă one-time de la Pasul 1.

- 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 funcția de dispersie *SHA – 1* și semnat cu *RSA* bazat pe cheia privată a lui *Alice*. Se obține “*Semnătura digitală*”.
- 3 Mesajul – text clar – este concatenat cu semnătura digitală și cu certificatul lui *Alice*.
- 4 Ceea ce s-a obținut se cripteză 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 “*enveloparea digitală*”.

- 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 funcția de dispersie *SHA – 1* și semnat cu *RSA* bazat pe cheia privată a lui *Alice*. Se obține “*Semnătura digitală*”.
- 3 Mesajul – text clar – este concatenat cu semnătura digitală și cu certificatul lui *Alice*.
- 4 Ceea ce s-a obținut se cripteză 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 “*enveloparea digitală*”.
- 6 Se face o concatenare a cheii de sesiune criptate cu mesajul criptat și semnat.

- 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 funcția de dispersie *SHA – 1* și semnat cu *RSA* bazat pe cheia privată a lui *Alice*. Se obține “*Semnătura digitală*”.
- 3 Mesajul – text clar – este concatenat cu semnătura digitală și cu certificatul lui *Alice*.
- 4 Ceea ce s-a obținut se cripteză 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 “*enveloparea digitală*”.
- 6 Se face o concatenare a cheii de sesiune criptate cu mesajul criptat și semnat.
- 7 Pentru decriptare și autentificare, *Bob* va parcurge pașii în ordine inversă.

Tranzacții *SET*Ierarhia de încredere *SET*

În mediul *SET* există o ierarhie a autoritaților de certificare.

## Ierarhia de încredere *SET*

În mediul *SET* există o ierarhie a autoritaților de certificare.

În vârf se află *SET Root CA* – deținută și gestionată de *Secure Electronic Transactions LLC*.

Toate certificatele conțin cheia sa publică.

## Certificatul proprietarului de card

Funcționează ca o reprezentare electronică a unui card de plăți.  
El este aprobat de banca emițătoare, nu conține numărul contului și nici data expirării.  
Acum certificat este transmis comercianților însorit de instrucțiuni de plată criptate.



## Certificatul proprietarului de card

Funcționează ca o reprezentare electronică a unui card de plată.

El este aprobat de banca emițătoare, nu conține numărul contului și nici data expirării.

Acest certificat este transmis comercianților însotit de instrucțiuni de plată criptate.

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

Tranzacții *SET*

## Certificatul comerciantului

Un comerciant primește două certificate de la instituția financiară:

## Certificatul comerciantului

Un comerciant primește două certificate de la instituția financiară:

- *Certificatul său propriu*, care îi asigură dreptul de a exista ca brand (entitate comercială);

## Certificatul comerciantului

Un comerciant primește două certificate de la instituția financiară:

- *Certificatul său propriu*, care îi asigură dreptul de a exista ca brand (entitate comercială);
- *Certificatul gateway-ului de plată*, prin care îi sunt acceptate (și plătite) de către cumpărător contravalorarea mărfurilor vândute.

## Certificatul gateway-ului de plată

Este obținut de cumpărător de la Autoritatea de Certificare a Brandului pentru gateway-ul său de plată.

## Certificatul gateway-ului de plată

Este obținut de cumpărător de la Autoritatea de Certificare a Brandului pentru gateway-ul său de plată.

El include cheia publică și cheia de criptare folosite de proprietarul de card pentru protejarea informației legate de cont.

Tranzacții *SET*

## Certificatelor cumpărătorului/emitentului

Banca cu rol de cumpărător trebuie certificată de emițătorul de carduri de plată.

Acesta îi dă astfel dreptul de a accepta sau respinge cererile de certificare venite direct de la comercianți.

Tranzacții *SET*

## Certificatele cumpărătorului/emitentului

Banca cu rol de cumpărător trebuie certificată de emițătorul de carduri de plată.

Acesta îi dă astfel dreptul de a accepta sau respinge cererile de certificare venite direct de la comercianți.

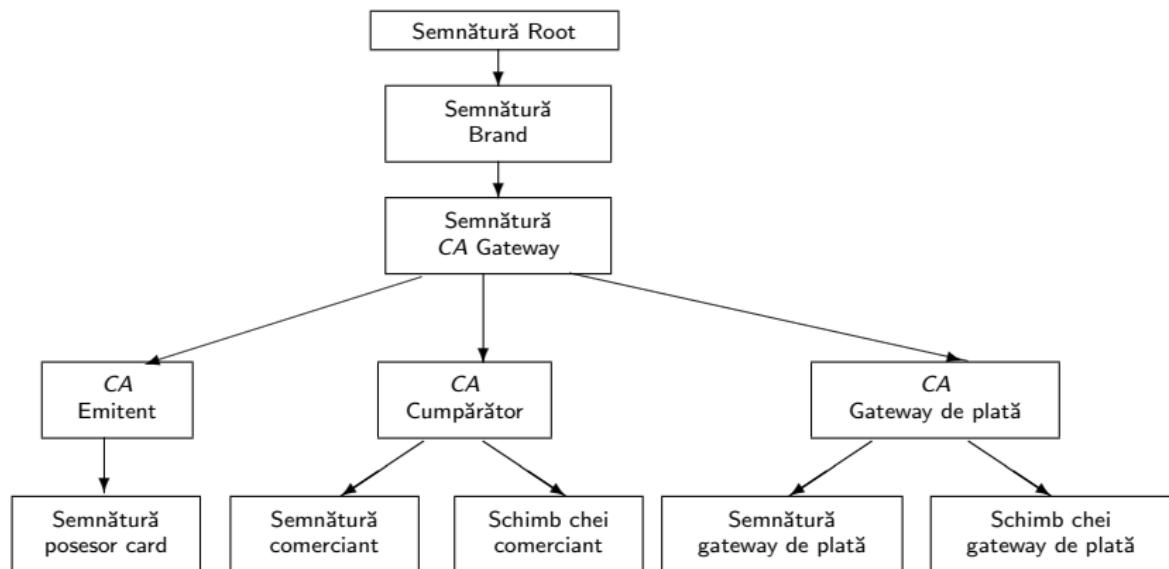
Un emitent trebuie să dețină de asemenea certificat de emițătorul de carduri de plată.

Astfel, el poate opera ca o autoritate de certificare pentru cererile de certificare venite direct de la posesorii de carduri.



Tranzacții SET

# Structura arborescentă a ierarhiei SET de încredere



Când un client comandă un produs unui comerciant, protocolul *SET* desfășoară trei tranzacții:

## Protocolele de tranzacții *SET*

Când un client comandă un produs unui comerciant, protocolul *SET* desfășoară trei tranzacții:

- A. Cererea de cumpărare;

Când un client comandă un produs unui comerciant, protocolul *SET* desfășoară trei tranzacții:

- A. Cererea de cumpărare;
  - B. Autorizarea de plată;

Protocolele de tranzacții *SET*

Când un client comandă un produs unui comerciant, protocolul *SET* desfășoară trei tranzacții:

- A. Cererea de cumpărare;
- B. Autorizarea de plată;
- C. Efectuarea plății.

## A. Cererea de cumpărare:

Protocol format din 4 mesaje schimbată între client și comerciant:



## A. Cererea de cumpărare:

Protocol format din 4 mesaje schimbată între client și comerciant:

- 1 Inițierea cererii,

## A. Cererea de cumpărare:

Protocol format din 4 mesaje schimbată între client și comerciant:

- 1 Inițierea cererii,
- 2 Inițierea răspunsului,



## A. Cererea de cumpărare:

Protocol format din 4 mesaje schimbată între client și comerciant:

- 1 Inițierea cererii,
- 2 Inițierea răspunsului,
- 3 Cererea de cumpărare,

## A. Cererea de cumpărare:

Protocol format din 4 mesaje schimbată între client și comerciant:

- 1 Inițierea cererii,
- 2 Inițierea răspunsului,
- 3 Cererea de cumpărare,
- 4 Răspuns la cererea de cumpărare.

Protocolele de tranzacții *SET*

## Inițierea cererii

Protocolul *SET* este invocat când posesorul de card este gata să lanseze un ordin de cumpărare:

## Inițierea cererii

Protocolul *SET* este invocat când posesorul de card este gata să lanseze un ordin de cumpărare:

- a selectat produsele pe care le dorește,

## Inițierea cererii

Protocolul *SET* este invocat când posesorul de card este gata să lanseze un ordin de cumpărare:

- a selectat produsele pe care le dorește,
- a completat forma de vânzare a comerciantului,

## Inițierea cererii

Protocolul *SET* este invocat când posesorul de card este gata să lanseze un ordin de cumpărare:

- a selectat produsele pe care le dorește,
- a completat forma de vânzare a comerciantului,
- a fost de acord cu termenii și condițiile acestuia.

Protocoloale de tranzacții *SET*

## Inițierea cererii

Protocolul *SET* este invocat când posesorul de card este gata să lanseze un ordin de cumpărare:

- a selectat produsele pe care le dorește,
- a completat forma de vânzare a comerciantului,
- a fost de acord cu termenii și condițiile acestuia.

În primul său mesaj, clientul inițiază cererea de cumpărare solicitând comerciantului certificatul propriu și certificatul emis de gateway-ul de plată afiliat.

# Inițierea răspunsului

- 1 Software-ul comerciantului primește inițierea cererii de cumpărare.

# Inițierea răspunsului

- 1 Software-ul comerciantului primește inițierea cererii de cumpărare.
- 2 Generează un răspuns *RI*, îi aplică o funcție de dispersie *h* (standardul este *SHA – 1*) și îl cripteză (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*.



# Inițierea răspunsului

- 1 Software-ul comerciantului primește inițierea cererii de cumpărare.
- 2 Generează un răspuns  $RI$ , îi aplică o funcție de dispersie  $h$  (standardul este  $SHA - 1$ ) și îl cripteză (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 Trimit clientului

$$(RI, d_K(h(RI)), CERT_{Com}, CERT_{Gateway})$$

Protocoalele de tranzacții *SET*

Software-ul clientului verifică acest răspuns:

Protocoalele de tranzacții *SET*

Software-ul clientului verifică acest răspuns:

- 1 Decriptează semnătura folosind cheia publică a comerciantului (aflată în *CERT<sub>Com</sub>*):

$$e_K(d_K(h(RI))) = h(RI)$$

Protocoalele de tranzacții *SET*

Software-ul clientului verifică acest răspuns:

- 1 Decriptează semnătura folosind cheia publică a comerciantului (aflată în *CERT<sub>Com</sub>*):

$$e_K(d_K(h(RI))) = h(RI)$$

- 2 Aplică funcția de dispersie lui *RI* și verifică dacă rezultatul  $h(RI)$  este egal cu ce a obținut anterior.

## Cererea de cumpărare

Clientul lansează un nou mesaj, conținând instrucțiuni de cumpărare (*CI*) și de plată (*PI*).

## Cererea de cumpărare

Clientul lansează un nou mesaj, conținând instrucțiuni de cumpărare (*CI*) și de plată (*PI*).

Anume, software-ul clientului:

- 1 Pune în *CI* și *PI* identificatorul de tranzacție asignat de comerciant.

## Cererea de cumpărare

Clientul lansează un nou mesaj, conținând instrucțiuni de cumpărare (*CI*) și de plată (*PI*).

Anume, software-ul clientului:

- 1 Pune în *CI* și *PI* identificatorul de tranzacție asignat de comerciant.
- 2 Generează o semnătură duală pentru *CI* și *PI*:  
$$\sigma_C = d'_K(h(h(PI) \parallel h(CI))),$$
 folosind cheia sa secretă *K'*.



## Protocoalele de tranzacții SET

## Cererea de cumpărare

Clientul lansează un nou mesaj, conținând instrucțiuni de cumpărare (*CI*) și de plată (*PI*).

Anume, software-ul clientului:

- 1 Pune în *CI* și *PI* identificatorul de tranzacție asignat de comerciant.
- 2 Generează o semnătură duală pentru *CI* și *PI*:  
 $\sigma_C = d'_K(h(h(PI) \parallel h(CI)))$ , folosind cheia sa secretă  $K'$ .
- 3 Generează aleator o cheie de sesiune  $K_1$  pentru un sistem simetric de criptare (Standard *DES*) cu care cripteză *PI*,  $h(CI)$  și semnătura duală.  $K_1$  și numărul de cont al cardului clientului sunt criptate cu cheia publică a gateway-ului de plată, formând un plic digital

$$PD = e_{Gateway}(CONT_{Client}, K_1)$$



## Protocolele de tranzacții *SET*

### 4 Trimite comerciantului

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

## 4 Trimitere către comerciant

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

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

# Răspuns la cererea de cumpărare

Comerciantul:

- 1 Verifică *CERT<sub>Client</sub>* și extrage din el cheia publică de criptare.

# Răspuns la cererea de cumpărare

Comerciantul:

- 1 Verifică  $CERT_{Client}$  și extrage din el cheia publică de criptare.
- 2 Calculează  $e_{K'}(\sigma_C)$ .

# Răspuns la cererea de cumpărare

Comerciantul:

- 1 Verifică  $CERT_{Client}$  și extrage din el cheia publică de criptare.
- 2 Calculează  $e_{K'}(\sigma_C)$ .
- 3 Calculează  $h(Cl)$ , apoi  $h(h(PI) \parallel h(Cl))$  și compară rezultatul cu ceea ce a obținut la pasul anterior.

# Răspuns la cererea de cumpărare

Comerciantul:

- 1 Verifică  $CERT_{Client}$  și extrage din el cheia publică de criptare.
- 2 Calculează  $e_{K'}(\sigma_C)$ .
- 3 Calculează  $h(Cl)$ , apoi  $h(h(PI) \parallel h(Cl))$  și compară rezultatul cu ceea ce a obținut la pasul anterior.
- 4 Dacă cele două valori sunt egale, deduce că  $Cl$  este autentic și procesează cererea de cumpărare (inclusiv trimiterea  $PI$  spre gateway pentru autorizare).

# Răspuns la cererea de cumpărare

Comerciantul:

- 1 Verifică  $CERT_{Client}$  și extrage din el cheia publică de criptare.
- 2 Calculează  $e_{K'}(\sigma_C)$ .
- 3 Calculează  $h(Cl)$ , apoi  $h(h(PI) \parallel h(Cl))$  și compară rezultatul cu ceea ce a obținut la pasul anterior.
- 4 Dacă cele două valori sunt egale, deduce că  $Cl$  este autentic și procesează cererea de cumpărare (inclusiv trimitera  $PI$  spre gateway pentru autorizare).
- 5 Generează un răspuns  $R$  la cererea de cumpărare și îl semnează:  $d_K(h(R))$ .

# Răspuns la cererea de cumpărare

Comerciantul:

- 1 Verifică  $CERT_{Client}$  și extrage din el cheia publică de criptare.
- 2 Calculează  $e_K'(\sigma_C)$ .
- 3 Calculează  $h(Cl)$ , apoi  $h(h(PI) \parallel h(Cl))$  și compară rezultatul cu ceea ce a obținut la pasul anterior.
- 4 Dacă cele două valori sunt egale, deduce că  $Cl$  este autentic și procesează cererea de cumpărare (inclusiv trimiterea  $PI$  spre gateway pentru autorizare).
- 5 Generează un răspuns  $R$  la cererea de cumpărare și îl semnează:  $d_K(h(R))$ .
- 6 Trimitе clientului perechea

$$(d_K(h(R)), CERT_{Com})$$

## B. Autorizarea de plată

Constă din două mesaje:

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

## B. Autorizarea de plată

Constă din două mesaje:

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

## B. Autorizarea 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 și cripteză informația de plată și o transmite băncii cumpărătoare.

## B. Autorizarea 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 și criptează informația de plată și o transmite băncii cumpărătoare.

Aceasta lansează și ea o cerere de autorizare pe baza căreia banca emitentă poate aproba sau respinge tranzacția posesorului de card.

## B. Autorizarea 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 și criptează informația de plată și o transmite băncii cumpărătoare.

Aceasta lansează și ea o cerere de autorizare pe baza căreia banca emitentă poate aproba sau respinge tranzacția posesorului de card. Banca cumpărătoare primește decizia și o trimitе gateway-ului de plată care emite comerciantului o autorizare de răspuns.

# Cererea de autorizare

Software-ul comerciantului:

- 1 Generează o cerere de autorizare *AR* care include suma solicitată, identificatorul tranzacției (din *CI*), și alte informații specifice.

# Cererea de autorizare

Software-ul comerciantului:

- 1 Generează o cerere de autorizare *AR* care include suma solicitată, identificatorul tranzacției (din *CI*), și alte informații specifice.
- 2 Semnează cererea:  $\sigma = d_K(h(AR))$

# Cererea de autorizare

Software-ul comerciantului:

- 1 Generează o cerere de autorizare *AR* care include suma solicitată, identificatorul tranzacției (din *CI*), și alte informații specifice.
- 2 Semnează cererea:  $\sigma = d_K(h(AR))$
- 3 Generează aleator o cheie de sesiune  $K_2$  și calculează

$$P = e_{K_2}(\sigma, AR), E = e_{K_{Gateway}}(K_2)$$

# Cererea de autorizare

Software-ul comerciantului:

- 1 Generează o cerere de autorizare *AR* care include suma solicitată, identificatorul tranzacției (din *CI*), și alte informații specifice.
- 2 Semnează cererea:  $\sigma = d_K(h(AR))$
- 3 Generează aleator o cheie de sesiune  $K_2$  și calculează

$$P = e_{K_2}(\sigma, AR), E = e_{K_{Gateway}}(K_2)$$

- 4 Trimit spre gateway-ul de plată mesajul

$$(P, E, PD, e_{K_1}(PI \parallel \sigma_C), CERT_{Com}, CERT_{Client})$$



# Autorizare de răspuns

Gateway-ul de plată:

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică  $K$  a comerciantului.

# Autorizare de răspuns

Gateway-ul de plată:

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică  $K$  a comerciantului.
- 2 Verifică cererea de autorizare:

# Autorizare de răspuns

Gateway-ul de plată:

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică  $K$  a comerciantului.
- 2 Verifică cererea de autorizare:
  - 1 Află  $K_2$ , pe care o folosește pentru a determina  $AR$  și  $\sigma$ .

# Autorizare de răspuns

Gateway-ul de plată:

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică  $K$  a comerciantului.
- 2 Verifică cererea de autorizare:
  - 1 Află  $K_2$ , pe care o folosește pentru a determina  $AR$  și  $\sigma$ .
  - 2 Calculează  $h(AR)$  și compară rezultatul cu  $e_K(\sigma)$ .

# Autorizare de răspuns

Gateway-ul de plată:

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică  $K$  a comerciantului.
- 2 Verifică cererea de autorizare:
  - 1 Află  $K_2$ , pe care o folosește pentru a determina  $AR$  și  $\sigma$ .
  - 2 Calculează  $h(AR)$  și compară rezultatul cu  $e_K(\sigma)$ .
- 3 Verifică  $CERT_{Client}$ , din care scoate cheia publică  $K'$  a clientului.

Protocoalele de tranzacții *SET*

- 4 Obține informația de plată dată de client și verifică semnătura duală.

- 4 Obține informația de plată dată de client și verifică semnătura duală.
  - 1 Din *PD* obține cheia  $K_1$  pe care o folosește pentru a găsi *PI* și  $\sigma_C$ .



4 Obține informația de plată dată de client și verifică semnătura duală.

- 1 Din *PD* obține cheia  $K_1$  pe care o folosește pentru a găsi *PI* și  $\sigma_C$ .
- 2 Calculează

$$e_{K'}(\sigma_C) = h(h(PI) \parallel h(CL))$$

4 Obține informația de plată dată de client și verifică semnătura duală.

- 1 Din *PD* obține cheia  $K_1$  pe care o folosește pentru a găsi *PI* și  $\sigma_C$ .
- 2 Calculează

$$e_{K'}(\sigma_C) = h(h(PI) \parallel h(CL))$$

- 3 Calculează  $h(PI)$ , apoi  $h(h(PI) \parallel h(CL))$ , pe care îl compară cu rezultatul anterior.

4 Obține informația de plată dată de client și verifică semnătura duală.

- 1 Din *PD* obține cheia  $K_1$  pe care o folosește pentru a găsi *PI* și  $\sigma_C$ .
- 2 Calculează

$$e_{K'}(\sigma_C) = h(h(PI) \parallel h(CI))$$

- 3 Calculează  $h(PI)$ , apoi  $h(h(PI) \parallel h(CI))$ , pe care îl compară cu rezultatul anterior.
- 4 Verifică consistența dintre cererea de autorizare a comerciantului și *PI* trimis de client.

Protocolele de tranzacții *SET*

- 4 Obține informația de plată dată de client și verifică semnătura duală.
  - 1 Din *PD* obține cheia  $K_1$  pe care o folosește pentru a găsi *PI* și  $\sigma_C$ .
  - 2 Calculează
$$e_{K'}(\sigma_C) = h(h(PI)\|h(CI))$$
  - 3 Calculează  $h(PI)$ , apoi  $h(h(PI)\|h(CI))$ , pe care îl compară cu rezultatul anterior.
  - 4 Verifică consistența dintre cererea de autorizare a comerciantului și *PI* trimis de client.
- 5 Trimit cererea de autorizare – folosind o rețea financiară normală – băncii care a eliberat cardul clientului.

## 6 Trimitte comerciantului autorizarea de răspuns:

Protocoalele de tranzacții *SET*

## 6 Trimitre comerciantului autorizarea de răspuns:

- 1 Generează o autorizare de răspuns *AR*, pe care o semnează:

$$\sigma_G = d_{\text{Gateway}}(h(\textit{AR})).$$

## 6 Trimitte comerciantului autorizarea de răspuns:

- 1 Generează o autorizare de răspuns  $AR$ , pe care o semnează:

$$\sigma_G = d_{\text{Gateway}}(h(AR)).$$

- 2 Generează aleator o cheie de sesiune  $K_3$  și calculează

$$Q = e_{K_3}(AR, \sigma_G), F = e_K(K_3)$$

## 6 Trimitre comerciantului autorizarea de răspuns:

- 1 Generează o autorizare de răspuns  $AR$ , pe care o semnează:

$$\sigma_G = d_{Gateway}(h(AR)).$$

- 2 Generează aleator o cheie de sesiune  $K_3$  și calculează

$$Q = e_{K_3}(AR, \sigma_G), F = e_K(K_3)$$

- 3 Crează un fișier  $CT$  numit "Capture token" și-l semnează:

$$sign(CT) = d_{Gateway}(h(CT))$$

Protocoalele de tranzacții *SET*

## 6 Trimitre comerciantului autorizarea de răspuns:

- 1 Generează o autorizare de răspuns  $AR$ , pe care o semnează:

$$\sigma_G = d_{Gateway}(h(AR)).$$

- 2 Generează aleator o cheie de sesiune  $K_3$  și calculează

$$Q = e_{K_3}(AR, \sigma_G), F = e_K(K_3)$$

- 3 Crează un fișier  $CT$  numit "Capture token" și-l semnează:

$$sign(CT) = d_{Gateway}(h(CT))$$

- 4 Generează aleator o cheie de sesiune  $K_4$  și calculează  
 $e_K(K_4), e_{K_4}(CT)$ .

## Protocolele de tranzacții SET

## 6 Trimitre comerciantului autorizarea de răspuns:

- 1 Generează o autorizare de răspuns  $AR$ , pe care o semnează:

$$\sigma_G = d_{Gateway}(h(AR)).$$

- 2 Generează aleator o cheie de sesiune  $K_3$  și calculează

$$Q = e_{K_3}(AR, \sigma_G), F = e_K(K_3)$$

- 3 Crează un fișier  $CT$  numit "Capture token" și-l semnează:

$$sign(CT) = d_{Gateway}(h(CT))$$

- 4 Generează aleator o cheie de sesiune  $K_4$  și calculează  
 $e_K(K_4), e_{K_4}(CT)$ .

- 5 Trimitre comerciantului

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

# Verificarea de către comerciant a autorizării de răspuns

La primirea mesajului, software-ul comerciantului stochează autorizarea de răspuns și *CT* – pentru a le folosi când va solicita plata.

# Verificarea de către comerciant a autorizării de răspuns

La primirea mesajului, software-ul comerciantului stochează autorizarea de răspuns și *CT* – pentru a le folosi când va solicita plata.

Detaliat:

- 1 Verifică *CERT<sub>Gateway</sub>*.



## Verificarea de către comerciant a autorizării de răspuns

La primirea mesajului, software-ul comerciantului stochează autorizarea de răspuns și *CT* – pentru a le folosi când va solicita plata.

Detaliat:

- 1 Verifică *CERT<sub>Gateway</sub>*.
- 2 Găsește cheia de sesiune  $K_3$ , cu ajutorul căreia calculează autorizația de răspuns *AR*.



## Verificarea de către comerciant a autorizării de răspuns

La primirea mesajului, software-ul comerciantului stochează autorizarea de răspuns și *CT* – pentru a le folosi când va solicita plata.

Detaliat:

- 1 Verifică  $CERT_{Gateway}$ .
- 2 Găsește cheia de sesiune  $K_3$ , cu ajutorul căreia calculează autorizația de răspuns *AR*.
- 3 Verifică semnatura a gateway-ului: calculează  $e_{Gateway}(sign(h(AR)))$  pe care o compară cu rezultatul obținut prin aplicarea funcției de dispersie  $h$  lui *AR* obținut anterior.



## Verificarea de către comerciant a autorizării de răspuns

La primirea mesajului, software-ul comerciantului stochează autorizarea de răspuns și  $CT$  – pentru a le folosi când va solicita plată.

Detaliat:

- 1 Verifică  $CERT_{Gateway}$ .
- 2 Găsește cheia de sesiune  $K_3$ , cu ajutorul căreia calculează autorizația de răspuns  $AR$ .
- 3 Verifică semnatura a gateway-ului: calculează  $e_{Gateway}(sign(h(AR)))$  pe care o compară cu rezultatul obținut prin aplicarea funcției de dispersie  $h$  lui  $AR$  obținut anterior.
- 4 Stochează  $Sign(CT)$  și  $e_{K_4}(CT)$ . De observat că numai gateway-ul de plată poate afla tokenul de captură.



## Verificarea de către comerciant a autorizării de răspuns

La primirea mesajului, software-ul comerciantului stochează autorizarea de răspuns și  $CT$  – pentru a le folosi când va solicita plată.

Detaliat:

- 1 Verifică  $CERT_{Gateway}$ .
- 2 Găsește cheia de sesiune  $K_3$ , cu ajutorul căreia calculează autorizația de răspuns  $AR$ .
- 3 Verifică semnatura a gateway-ului: calculează  $e_{Gateway}(sign(h(AR)))$  pe care o compară cu rezultatul obținut prin aplicarea funcției de dispersie  $h$  lui  $AR$  obținut anterior.
- 4 Stochează  $Sign(CT)$  și  $e_{K_4}(CT)$ . De observat că numai gateway-ul de plată poate afla tokenul de captură.
- 5 Finalizează procesarea cererii de cumpărare.



## C. Efectuarea plășii:

După finalizarea comenzi de livrare a produsului cumpărat, comerciantul va solicita plata.

## C. Efectuarea plășii:

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

- 1 Scrie o cerere de plată *SP*.

## C. Efectuarea plășii:

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

- 1 Scrie o cerere de plată *SP*.
- 2 O semnează:  $\sigma_{Com} = d_{Com}(h(SP))$ .

## C. Efectuarea plășii:

După finalizarea comenzi 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  $K_5$ .

## C. Efectuarea plășii:

După finalizarea comenzi 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 Trimitte spre gateway-ul de plată

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

Protocolele de tranzacții *SET*

# Răspunsul gateway-ului

Când un gateway de plată primește o solicitare de plată

# Răspunsul gateway-ului

Când un gateway de plată primește o solicitare de plată

- 1 Verifică *CERT<sub>Com</sub>*, din care scoate cheia publică a comerciantului.

# Răspunsul gateway-ului

Când un gateway de plată primește o solicitare de plată

- 1 Verifică *CERT<sub>Com</sub>*, din care scoate cheia publică a comerciantului.
- 2 Verifică cererea de plată a comerciantului.

# Răspunsul gateway-ului

Când un gateway de plată primește o solicitare de plată

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică a comerciantului.
- 2 Verifică cererea de plată a comerciantului.
  - 1 Află  $K_5$ , cu ajutorul căreia calculează  $SP$ .

# Răspunsul gateway-ului

Când un gateway de plată primește o solicitare de plată

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică a comerciantului.
- 2 Verifică cererea de plată a comerciantului.
  - 1 Află  $K_5$ , cu ajutorul căreia calculează  $SP$ .
  - 2 Verifică semnătura digitală a comerciantului, scoțând din ea  $h(SP)$  și comparând-o cu  $h(SP)$  calculat din  $PR$  rezultat anterior.

# Răspunsul gateway-ului

Când un gateway de plată primește o solicitare de plată

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică a comerciantului.
- 2 Verifică cererea de plată a comerciantului.
  - 1 Află  $K_5$ , cu ajutorul căreia calculează  $SP$ .
  - 2 Verifică semnătura digitală a comerciantului, scoțând din ea  $h(SP)$  și comparând-o cu  $h(SP)$  calculat din  $PR$  rezultat anterior.
  - 3 Află  $K_4$  folosind cheia sa privată și calculează  $CT$ .

# Răspunsul gateway-ului

Când un gateway de plată primește o solicitare de plată

- 1 Verifică  $CERT_{Com}$ , din care scoate cheia publică a comerciantului.
- 2 Verifică cererea de plată a comerciantului.
  - 1 Află  $K_5$ , cu ajutorul căreia calculează  $SP$ .
  - 2 Verifică semnătura digitală a comerciantului, scoțând din ea  $h(SP)$  și comparând-o cu  $h(SP)$  calculat din  $PR$  rezultat anterior.
  - 3 Află  $K_4$  folosind cheia sa privată și calculează  $CT$ .
  - 4 Verifică consistența dintre informațiile cuprinse în  $CT$  și  $SP$ .

Protocoalele de tranzacții *SET*

- 3 Trimit *SP* – printr-o rețea financiară – spre banca care deține contul clientului.

Protocolele de tranzacții *SET*

- 3 Trimit *SP* – printr-o rețea financiară – spre banca care deține contul clientului.
- 4 Generează un răspuns de plată *RP* și îl semnează:

$$\sigma_{RP} = d_{Gateway}(h(RP))$$

Protocolele de tranzacții *SET*

- 3 Trimit *SP* – printr-o rețea financiară – spre banca care deține contul clientului.
- 4 Generează un răspuns de plată *RP* și îl semnează:

$$\sigma_{RP} = d_{Gateway}(h(RP))$$

- 5 Generează aleator o cheie de sesiune  $K_6$ .

Protocolele de tranzacții *SET*

- 3 Trimit *SP* – printr-o rețea financiară – spre banca care deține contul clientului.
- 4 Generează un răspuns de plată *RP* și îl semnează:

$$\sigma_{RP} = d_{Gateway}(h(RP))$$

- 5 Generează aleator o cheie de sesiune  $K_6$ .
- 6 Trimit comerciantului

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

# Verificarea răspunsului de plată

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

## Verificarea răspunsului de plată

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

- 1 Verifică *CERT<sub>Gateway</sub>*, din care scoate cheia publică de criptare.

## Verificarea răspunsului de plată

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

- 1 Verifică *CERT<sub>Gateway</sub>*, din care scoate cheia publică de criptare.
- 2 Calculează cheia  $K_6$ , cu ajutorul căreia află răspunsul de plată *RP*.

## Verificarea răspunsului de plată

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

- 1 Verifică *CERT<sub>Gateway</sub>*, 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  $\sigma_{RP}$  și comparând rezultatul cu  $h(RP)$  calculat pe baza informațiilor de la pasul anterior.

## Verificarea răspunsului de plată

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

- 1 Verifică *CERT<sub>Gateway</sub>*, 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  $\sigma_{RP}$  și comparând rezultatul cu  $h(RP)$  calculat pe baza informațiilor de la pasul anterior.

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



## Protocolele de tranzacții *SET*

# Sfârșit