Capitolul 3

Securitatea poştei electronice

În mediile electronice de transmitere a informaţiei, poşta electronică ocupă o poziţie centrală. Ea permite utilizatorilor să comunice prin mesaje, folosind facilităţile oferite de reţelele de calculatoare existente. În general serviciul de e-mail este cea mai utilizată aplicaţie, dar – în acelaşi timp – este şi cea mai puţin sigură.

Astfel, pentru a trimite sau primi un mesaj, *Alice* trebuie să fie conectată la un server de e-mail. Serverul stochează mesajul, după care îl trimite spre alt server – care face acelaşi lucru. În acest fel, e-mailurile pot traversa mai multe servere până ajung la destinaţie, fiecare din acestea păstrând câte o copie a mesajului. *Alice* (sau *Bob*) nu poate şterge mesajul din aceste servere intermediare; ele rămân aici până când administratorul severului decide acest lucru[1].

Un mod de a proteja mesajele este criptarea lor folosind diverse produse, cum ar fi Privacy Enhanced Mail (PEM), MIME Object Security Services ($MOSS$), $X.400$, PGP sau $S/MIME$. Ultimele două vor fi prezentate în acest capitol. PGP este o specificaţie, iar $S/MIME$ este un protocol; ambele sunt compatibile cu serviciile Internet actuale.

3.1 PGP (Pretty Good Privacy)

Este un protocol[2] creat de Phil Zimmenmann ([8]) în 1991 pentru poştă electronică şi pentru aplicaţii legate de stocarea fişierelor. În acest moment este cel mai utilizat sistem de securitate oferit de serviciile de poştă electronică. O parte din cauzele care au condus la această supremaţie sunt:

1. Este accesibil free în versiuni care funcţionează pe o gamă largă de platforme (inclusiv WINDOWS, UNIX, MacIntosh).

[1]Unele companii s-au orientat spre această posibilă piaţă şi dezvoltă softuri capabile să şteargă e-mailurile stocate în serverele intermediare. Aceste servicii necesită însă costuri suplimentare.

[2]Numele "Pretty Good Privacy" a fost inspirat de numele unui magazin, "Ralphś Pretty Good Grocery", aflat într-un oraş imaginar, Lake Wobegon, dintr-un foileton radiofonic scris de Garrison Keillor.

2. Este bazat pe algoritmi criptogafici consideraţi siguri. În particular sunt folosiţi RSA, DSS şi *ElGamal* (pentru criptarea cu cheie publică), $CAST-128, IDEA$, $3DES$ (pentru criptarea simetrică) şi $SHA-1$ (ca funcţie de dispersie).

3. Are o arie largă de aplicabilitate, de la corporaţii care doresc să aleagă şi să întărească o schemă standard pentru criparea fişierelor şi mesajelor, până la persoane individuale (sau reţele) care doresc să comunice în siguranţă pe Internet.

4. Nu a fost creat, dezvoltat sau controlat de nici un organism guvernamental sau organizaţie de standarde. Acest argument îl face atractiv celor care se tem de un posibil control instituţional.

În prezent, protocoalele PGP asigură cinci servicii: *autentificare, confidenţialitate, compresie, compatibilitate e-mail* şi *segmentare*.

3.1.1 Autentificare

Este un serviciu de semnătură digitală cu appendix, oferit de PGP. Protocolul este următorul:

Intrare: Mesajul m este trimis de *Alice* către *Bob*.

- *Alice*, în calitate de expeditor, efectuează următorii paşi:
  1. Generează o amprentă $h(m)$ a mesajului, folosind o funcţie de dispersie criptografică.
  2. Criptează cu cheia secretă: $d_{Alice}(h(m))$.
  3. Aplică o funcţie de compresie Z perechii $(m, d_{Alice}(h(m)))$.
- La recepţia mesajului $Z((m, d_{Alice}(h(m)))$, *Bob* efectuează următorii paşi:
  1. Decomprimă (cu Z^{-1}) şi află mesajul m precum şi $d_{Alice}(h(m))$.
  2. Folosind cheia publică a lui *Alice*, determină $h(m) = e_{Alice}(d_{Alice}(h(m)))$.
  3. Aplică funcţia de dispersie h lui m şi compară rezultatul cu $h(m)$. Dacă cele două secvenţe coincid, mesajul m este acceptat ca autentic.

Pentru implementări se folosesc: $SHA-1$ drept funcţie de dispersie, RSA sau DSA pentru sistemul de criptare/semnare folosit de *Alice*, şi ZIP pentru funcţia de compresie. De asemenea, un cuplu (α, β) se utilizează sub formă concatenată $\alpha \| \beta$.

Securitatea protocolului se bazează pe securitatea sistemelor folosite (SHA/RSA sau SHA/DSA).

De obicei semnăturile sunt ataşate mesajului (sau fişierului) pe care-l semnează. Sunt posibile totuşi şi semnături detaşate. Acestea sunt păstrate de expeditor pentru a fi folosite în diverse scopuri. De exemplu, dacă mai multe părţi semnează un contract, fiecare semnătură este independentă şi se aplică doar documentului iniţial (altfel – într-un lanţ de semnături dependente – fiecare semnătură ar conduce la apariţia unui nou mesaj, diferit de cel anterior).
Sau – ca un alt exemplu – semnătura unui program executabil poate detecta o posibilă virusare ulterioară.

3.1.2 Confidenţialitate

Alt serviciu de bază asigurat de *PGP* este *confidenţialitatea*, oferită de criptarea mesajelor transmise sau stocate local (ca fişiere). În ambele situaţii sistemul de criptare preferat este $CAST-128$ (ca alternative sunt folosite şi $IDEA$ sau $3DES$), iar modul de criptare este CFB pe 64 biţi.

În implementări se foloseşte RSA sau *ElGamal* pentru criptarea cu cheie publică.

Observaţia 3.1. *Timpul de criptare se optimizează combinând un sistem de criptare cu cheie publică şi unul simetric (se ştie că un sistem de criptare simetric – de exemplu $CAST-128$ – este mult mai rapid decât unul cu cheie publică: RSA sau ElGamal).*

Intrare: Mesajul m este trimis de *Alice* către *Bob*.

- Expeditorul *Alice* efectuează următorii paşi:
  1. Arhivează m cu un protocol de arhivare Z; se obţine $Z(m)$.
  2. Generează aleator o cheie de sesiune K de 128 biţi.
  3. Efectuează criptarea $e_K(Z(m))$.
  4. Criptează cheia de sesiune folosind cheia publică a lui *Bob*: $e_{Bob}(K)$.
  5. Trimite cuplul $\alpha = (e_{Bob}(K), e_K(Z(m)))$.
- La primirea mesajului α, *Bob* efectuează următorii paşi:
  1. Află cheia de sesiune $K = d_{Bob}(e_{Bob}(K))$.
  2. Decriptează partea a doua a mesajului primit: $d_K(e_K(Z(m))) = Z(m)$.
  3. Dezarhivează cu Z^{-1} şi află mesajul m.

Observaţia 3.2. *Ideea de cheie de sesiune se referă de fapt la o cheie one-time. Protocolul de distribuţie a cheii de sesiune nu este necesar, deoarece nu se solicită o confirmare a cheii de către Bob. Modul de funcţionare al poştei electronice face ca protocolul prin care*

ambele părţi să folosească o cheie de sesiune comună nu este util; fiecare mesaj este însoţit de propria sa cheie, securizată printr-o criptare cu o cheie publică.

Generarea unei chei de sesiune se realizează în modul următor: Alice introduce o parolă, folosind tastatura sau mouse-ul. PGP foloseşte parola şi timpii de scriere la tastatură (respectiv de mişcare al mouse-ului) pentru a genera o cheie aleatoare care va fi folosită de un sistem simetric de criptare.

Cele două utilizări ale PGP (autenticitate şi confidenţialitate) pot fi asociate pentru acelaşi mesaj. În prima fază se generează o semnătură a textului clar, care se adaugă mesajului. Apoi perechea *(text clar, semnătură)* este criptată folosind un sistem simetric ($CAST-128, IDEA$ sau $3DES$), iar cheia de sesiune se criptează folosind un sistem cu cheie publică (RSA sau $ElGamal$).
Principalele motive pentru care este preferată această ordine (şi nu invers: criptarea mesajului, urmată de semnarea textului criptat) sunt:

1. În general este mai convenabil să stocăm semnătura unui text clar şi nu a unui text criptat (a cărui cheie ar trebui reţinută de asemenea).

2. Dacă se cere şi verificarea de către o terţă parte, de obicei aceasta nu trebuie implicată în procesul de decriptare, ci doar în cel de verificare a semnăturii.

Deci, dacă se cer ambele servicii PGP, *Alice* va semna întâi mesajul cu cheia sa secretă, apoi îl va cripta cu cheia de sesiune, iar pe aceasta o va cripta cu componenta publică a cheii lui *Bob*.

3.1.3 Compresie

În general, PGP face o compresie a mesajului după semnarea lui, dar înaintea criptării. Aceasta duce la o optimizare a spaţiului folosit atât în mesajele e-mail cât şi în stocarea fişierelor.

Algoritmul de compresie folosit de PGP este ZIP, descris în Anexa 1.

Observaţia 3.3. *Semnătura este generată înaintea compresiei deoarece:*

1. *Este preferabil să semnăm un mesaj necomprimat, deoarece pentru verificări va fi oferit textul clar. În caz contrar, sau se va păstra în memorie doar o versiune comprimată a mesajului, sau acesta va fi comprimat ori de câte ori se va solicita verificarea semnăturii.*

2. *Chiar dacă este uşor de recomprimat mesajul ori de câte ori este necesară verificarea, apare o dificultate datorită faptului că ZIP este un algoritm nedeterminist: diverse implementări duc la rate de compresie diferite şi deci la forme diferite. Aceste versiuni sunt totuşi inter-operabile: orice versiune poate decompresa corect*

ieşirea din oricare altă versiune. Aplicând funcţia de dispersie şi semnătura după compresie, vom obliga ca toate implementările PGP să conducă la aceeaşi compresie.

Observaţia 3.4. *Criptarea mesajului după compresie asigură o securitate sporită, datorită redondanţei mult mai reduse a textului compresat (comparativ cu textul clar).*

Indiferent de serviciile solicitate (autentificare şi/sau confidenţialitate), în final transmisia va conţine blocuri de octeţi arbitrari. Multe sisteme de e-mail permit însă transmiterea doar a blocurilor ASCII. Pentru a fi compatibil cu această restricţie, *PGP* efectuează şi o conversie a octeţilor la secvenţe de caractere ASCII printabile. Conversia folosită este $radix - 64$ (Anexa 2): fiecare grup de 3 octeţi binari este transformat în patru caractere ASCII.

Folosirea sistemului $radix - 64$ măreşte mesajul cu 33%, extensie compensată de rata de compresie (ZIP are de obicei o rată de compresie de 50%).
Astfel, dacă un fişier are lungimea n, după compresie şi conversie, lungimea lui va fi de aproximativ $1,33 \times 0,5 \times n = 0,665 \times n$.

Observaţia 3.5. *Algoritmul $radix - 64$ converteşte tot şirul de intrare, considerat ca o secvenţă binară de octeţi. Deci, dacă un mesaj este doar semnat (nu şi criptat), conversia se aplică deşi mare parte din text este deja ASCII. Aceasta îl face necitibil unui intrus ocazional – deci oferă o anumită confidenţialitate mesajului.*
Există opţiunea ca PGP să convertească (cu $radix-64$) numai componenta de semnătură a mesajului; în acest fel Bob îl va putea citi direct, fără a folosi PGP. PGP va fi utilizat numai pentru a verifica semnătura.

3.1.4 Segmentare şi reasamblare

Există adesea o restricţie referitoare la lungimea mesajelor transmise prin e-mail; de exemplu, multe facilităţi accesibile prin Internet impun o lungime maximă de 50.000 sau 65.000 octeţi. Orice mesaj mai lung trebuie spart în blocuri mai mici şi trimise prin mesaje e-mail separate.

PGP segmentează automat un mesaj mai lung în mesaje care sunt trimise separat prin e-mail. Blocurile sunt puse în ordine în fişiere cu extensia $.as1$, $.as2$ etc. Segmentarea este realizată după încheierea celorlaltor operaţii (inclusiv conversia). Deci cheia de sesiune şi semnătura vor apare o singură dată, la începutul primului segment. După recepţia tuturor segmentelor, *PGP* face reasamblarea lor în ordinea indicată de extensii, eliminând headerele şi alte detalii de e-mail.

3.1.5 Chei criptografice şi servere de chei

PGP-ul foloseşte patru tipuri de chei: chei de sesiune (simetrice, one-time), chei publice, chei private şi chei (simetrice) bazate pe parolă.

Există trei cerinţe referitoare la modul de gestiune al acestor chei:

1. O procedură sigură de a genera chei de sesiune aleatoare.

2. Fiecare utilizator trebuie să deţină mai multe perechi de chei publice/private care schimbate din când în când. Dacă are loc o astfel de modificare, destinatarul *Bob* va şti doar vechea cheie până la o eventuală actualizare. În plus, utilizatorul ar putea opta la un moment dat pentru mai multe perechi de chei – fie pentru a interacţiona simultan cu mai mulţi destinatari, fie doar pentru a îmbunătăţi securitatea, limitând utilizarea unei anumite chei la doar o porţiune din text. Dificultatea în aceste cazuri este determinată de faptul că nu există o modalitate de relaţionare între utilizatori şi cheile lor publice.
   Sunt necesare, aşadar, anumite proceduri de identificare a unor chei particulare.

3. Fiecare utilizator *PGP* trebuie să menţină un fişier cu propriile sale perechi de chei publice/private, precum şi un fişier cu cheile publice corespunzătoare.

Generarea cheilor de sesiune

Fiecare cheie de sesiune este asociată unui singur mesaj şi este folosită exclusiv pentru criptarea şi decriptarea acestuia. Cele două operaţii se realizează – aşa cum s-a văzut – cu ajutorul unor algoritmi simetrici: $CAST-128$ şi $IDEA$ care folosesc chei de 128 biţi sau $3DES$ care foloseşte chei de 168 biţi.

Exemplul 3.1. *Să luăm în considerare utilizarea algoritmului* $CAST-128$.

Numerele aleatoare pe 128 *de biţi sunt generate folosind chiar algoritmul de criptare. Intrarea generatorului de numere pseudo-aleatoare constă dintr-o cheie de* 128 *de biţi şi două blocuri de* 64 *de biţi tratate ca texte clare ce urmează să fie criptate.* $CAST-128$ *generează două blocuri criptate de* 64 *biţi fiecare, care sunt apoi concatenate pentru a forma cheia de sesiune de* 128 *de biţi.*

Cele două texte clare de câte 64 *biţi aflate la intrarea în generatorul de numere pseudo-aleatoare provin – la rândul lor – dintr-o secvenţă de* 128 *biţi generaţi aleator. Aceste numere folosesc ca sursă tastele apăsate de utilizator sau mişcările de mouse.*

PGP-ul menţine un buffer de 256 *octeţi aleatori. De fiecare dată când PGP-ul aşteaptă apăsarea unei taste, înregistrează timpul – în format de* 32 *biţi – de la care începe aşteptarea. Când o tastă este apăsată, PGP-ul înregistrează momentul la care a avut loc acest eveniment, precum şi valoarea pe* 8 *biţi a caracterului tastat. Datele referitoare la timp şi caracter sunt utilizate pentru a genera o cheie care – la rândul ei – este folosită pentru a cripta valoarea curentă a buffer-ului de octeţi aleatori.*
Numărul aleator astfel generat este combinat cu cheia de sesiune rezultată din algoritmul $CAST-128$ *pentru a forma noua intrare a generatorului de numere pseudo-aleatoare. Datorită algoritmului* $CAST-128$*, rezultatul va fi un fişier de chei de sesiune cât mai apropiate de un model aleator.*

Identificatori de chei

Cheia de sesiune este criptată cu cheia publică a lui *Bob*, astfel încât numai acesta este capabil să recupereze cheia de sesiune şi – mai departe – să decripteze mesajul. Dacă fiecare parte implicată foloseşte o singură pereche de chei publice/private, atunci *Bob* va şti ce cheie să folosească pentru a recupera cheia de sesiune: cheia sa privată (unică).

Dacă însă fiecare utilizator poate dispune de mai multe perechi de chei publice/private problema nu mai este la fel de simplă.

O soluţie ar fi transmiterea cheii publice odată cu mesajul. *Bob* ar putea verifica la primire dacă într-adevăr cheia trimisă se regăseşte printre cheile sale publice şi – în caz afirmativ – să treacă la decriptarea mesajului.
Această schemă poate fi implementată uşor, dar consumă foarte mult spaţiu, deoarece o cheie criptată cu RSA poate avea câteva sute de cifre.

A doua rezolvare constă în asocierea unui identificator fiecărei chei publice, care să fie unic pentru raporturile cu un anumit utilizator. În acest mod, o combinaţie între ID-ul utilizatorului şi ID-ul cheii ar fi suficientă pentru a identifica în mod unic o cheie.

Această soluţie ridică, totuşi o altă problemă: ID-ul cheii trebuie alocat şi stocat astfel încât atât *Alice* cât şi *Bob* să poată realiza transferul între ID-ul cheii şi cheia publică.

Soluţia adoptată de PGP constă în alocarea unui ID fiecărei chei publice care să fie – cu o probabilitate mare – unică în raport cu un anumit destinatar. ID-ul fiecărei chei publice constă din cei mai puţin semnificativi 64 biţi ai acesteia. Astfel, ID-ul cheii publice KP_a este $KP_a \ (mod\ 2^{64})$. Aceasta este o dimensiune suficientă pentru ca probabilitatea existenţei duplicatelor să fie redusă.

Servere (Inele) de chei

Cheile trebuie să fie stocate şi organizate într-un mod care să permită o utilizare efectivă şi sistematică de către toate părţile. PGP-ul furnizează o pereche de structuri de date fiecărui utilizator: una pentru a stoca perechile de chei publice/private deţinute de utilizatorul respectiv şi una pentru a stoca perechile de chei publice corespunzătoare celorlaltor utilizatori.
Aceste structuri de date sunt cunoscute sub numele de "*serverul de chei private*" şi respectiv "*serverul de chei publice*"[3].

Timestamp	ID cheie	Cheie publică	Cheie privată	ID utilizator
⋮	⋮	⋮	⋮	⋮
T_i	$PU_i \ (mod\ 2^{64})$	PU_i	$e_{h(P_i)}(PR_i)$	Utilizator i
⋮	⋮	⋮	⋮	⋮

În tabelul de mai sus este prezentată structura serverului de chei private. Fiecare linie reprezintă o pereche de chei publice/private deţinute de un utilizator; ea este formată din:

[3]Unele referinţe folosesc termenul "Inele de chei"private/publice).

- **Timestamp**: data/ora la care a fost generată perechea de chei.
- ID **cheie**: cei mai puţin semnificativi 64 biţi pentru această linie (reprezentată în binar).
- **Cheia publică**: componenta publică a perechii de chei.
- **Cheia privată**: componenta privată a perechii de chei; acest câmp este criptat.
- ID **utilizator**: de obicei, acest câmp este reprezentat de adresa e-mail a utilizatorului.

Serverul de chei private poate fi indexat fie după ID-ul utilizatorului, fie după ID-ul cheii. Deşi cheia privată va fi stocată şi accesibilă doar pe calculatorul utilizatorului care a creato şi care deţine perechile de chei, valoarea ei trebuie păstrată cât mai în siguranţă.
Din acest motiv, cheia privată este criptată (cu $CAST-128$, $IDEA$ sau $3DES$) şi abia apoi înregistrată pe server. Procedura este următoarea:

1. *Alice* alege o parolă ce va fi utilizată în criptarea cheilor private.
2. De fiecare dată când sistemul generează o nouă pereche de chei publice/private folosind RSA, *Alice* este solicitată să introducă parola. Utilizând $SHA-1$, este generată o amprentă pe 160 de biţi, iar parola se şterge.
3. Sistemul criptează cheia privată folosind $CAST-128$ cu amprenta – trunchiată pe 128 biţi – drept cheie. Amprenta este apoi ştearsă şi cheia privată criptată este stocată pe serverul de chei private.

Serverul de chei publice, folosit pentru stocarea cheilor publice ale celorlaltor utilizatori conectaţi cu posesorul serverului de chei, are o structură similară. Astfel, el conţine toate câmpurile serverului de chei private: Timestamp, ID cheie, Cheia publică, ID - utilizator (cu menţiunea că un utilizator poate avea mai multe ID-uri asociate unei singure chei).

În linii mari, cele două servere sunt construite în conformitate cu o autoritate centrală care eliberează certificate (detalii referitoare la PKI vor fi prezentate în capitolul următor).
Fiecare intrare în serverul cheilor publice reprezintă o cheie publică certificată. Fiercărei linii din tabel îi este asociat un câmp de **Legitimare Cheie**; el indică în ce măsură PGP-ul va considera că aceasta reprezintă o cheie publică validă pentru utilizator, şi cât de ridicat este nivelul ei de încredere. Acest câmp este calculat de PGP.

De asemenea, fiecărei linii i se asociază una sau mai multe semnături (procurate de posesorul serverului) care semnează certificatul. La rândul ei, fiecare semnătură are asociat un câmp **Signature Trust** care indică în ce măsură are încredere utilizatorul că

semnătura respectivă certifică o cheie publică. Un alt câmp – **Legitimare Cheie** – provine din colecţia de câmpuri *Signature Trust* pentru intrarea respectivă.
În sfârşit, fiecare intrare defineşte o cheie publică asociată cu un anumit proprietar şi un câmp **Owner trust** este inclus pentru a indica în ce măsură această cheie publică este de încredere în semnarea altor certificate de chei publice; acest nivel de încredere este stabilit de utilizator. Deci câmpurile *Signature Trust* sunt de fapt copii ale câmpului *Owner* din altă intrare din tabel. Modalitatea detaliată de calcul a valorilor acestor câmpuri precum şi lista valorilor pe care le poate lua fiecare intrare sunt discutate în [5] şi [7].

Modul de transmitere a mesajelor şi cheilor

Presupunem că *Alice* doreşte să transmită un mesaj semnat şi criptat. Ea va parcurge următorii paşi:

1. Semnarea mesajului:
   (a) *PGP*-ul alege cheia privată a lui *Alice* din serverul cheilor private, folosind ID-ul ei de utilizator. Dacă acest ID nu este dat ca parametru al comenzii, se trimite prima cheie privată din server.
   (b) *PGP*-ul cere lui *Alice* parola, pentru a recupera cheia privată necriptată.
   (c) Este generată componenta mesajului care conţine semnătura.
2. Criptarea mesajului:
   (a) *PGP*-ul generează o cheie de sesiune şi criptează mesajul.
   (b) *PGP*-ul primeşte cheia publică a lui *Bob* din serverul de chei publice, folosind drept index ID-ul acestuia.
   (c) Este generată componenta mesajului care conţine cheia de sesiune.

La primirea mesajului, *Bob* va efectua următorii paşi:

1. Decriptarea mesajului:
   (a) *PGP*-ul primeşte cheia privată a lui *Bob* din serverul de chei private.
   (b) *PGP*-ul cere lui *Bob* parola, pentru a recupera cheia privată necriptată.
   (c) *PGP*-ul recuperează cheia de sesiune şi decriptează mesajul.
2. Autentificarea mesajului:
   (a) *PGP*-ul primeşte cheia publică a lui *Alice* din serverl de chei publice, folosind ca index ID-ul cheii din semnătura mesajului.
   (b) *PGP*-ul recuperează conţinutul necriptat al mesajului.
   (c) *PGP*-ul criptează acest conţinut necriptat şi-l compară cu mesajul primit. Dacă cele două sunt identice, mesajul este autentic; în caz contrar, mesajul iniţial a fost alterat de un posibil intrus.

3.1.6 Structura de pachete a mesajelor securizate cu *PGP*

Un fişier *PGP* este alcătuit din:

- Un **pachet mesaj**. Conţine datele care au importanţă pentru utilizator şi care vor fi trimise sau stocate (componenta mesaj), precum şi un antet (header) care conţine informaţii generate de *PGP*: numele fişierului şi un timestamp (care indică data creerii mesajului sau fişierului).
- Un **pachet semnătură**. Conţine o combinaţie între informaţii legate de cheia publică a lui *Alice* şi o amprentă a mesajului (obţinută prin aplicarea unei funcţii de dispersie criptografică asupra componentei mesaj).

  Semnătura este formată din:

  1. Timestamp: data la care a fost creată semnătura.
  2. Amprenta mesajului: reprezentată de un cod de 160 biţi generaţi cu $SHA-1$ şi criptat cu cheia privată a lui *Alice*. Funcţia de dispersie este calculată pentru timestamp-ul semnăturii, concatenat cu o porţiune de date din componenta mesaj.
     Includerea în amprenta mesajului a timpului la care a fost generată semnătura protejează împotriva unui atac man-in-the-middle. Excluderea numelui fişierului şi a timestamp-ului pentru componenta mesaj asigură faptul că semnăturile detaşate sunt identice cu semnăturile ataşate, cu excepţia faptului că nu prefixează mesajul.

Semnăturile detaşate sunt calculate într-un fişier separat care nu conţine nici unul dintre câmpurile din antetul componentei mesaj.

3. Cei mai semnificativi octeţi ai amprentei mesajului: aceştia îl ajută pe *Bob* să determine dacă s-a folosit cheia publică corectă pentru decriptarea amprentei.
4. ID-ul cheii publice a lui *Alice*: serveşte la identificarea cheii publice care ar trebui folosită pentru a decripta amprenta; identifică de asemenea şi cheia privată utilizată pentru criptarea amprentei.

Pachetelor *mesaj* şi *semnătură* li se poate aplica o compresie folosind ZIP şi pot fi criptate cu ajutorul unei chei de sesiune.

- Un **pachet cheie de sesiune**. Pachetele cheie de sesiune includ cheia de sesiune şi identificatorul cheii publice a lui *Bob* – care a fost utilizată de *Alice* pentru a cripta cheia de sesiune.
  În interiorul pachetului principal se află un pachet cu cheia publică de sesiune criptată. Pachetele de date (criptate identic) sunt precedate de un pachet ce conţine cheia publică de sesiune criptată – pentru fiecare cheie utilizată în criptarea mesajului. Mesajul este criptat cu cheia de sesiune, care este la rândul ei criptată şi stocată în pachetul cheii de sesiune.

  Corpul pachetului cheii de sesiune este alcătuit din:

  1. Un octet reprezentând cifra 3.
  2. 8 octeţi ai ID-ului cheii publice cu care este criptată cheia de sesiune.
  3. 8 octeţi care indică algoritmul folosit pentru cheia publică.
  4. Un şir de octeţi reprezentând cheia publică criptată. Conţinutul acestui şir este dependent de algoritmul utilizat pentru cheia publică:

     - pentru RSA: un MPI (întreg în precizie multiplă reprezentând valoarea criptată a lui $m^e \ (mod\ n)$;

     - pentru $ElGamal$: două MPI – unul pentru valoarea $g^k \ (mod\ p)$ şi altul pentru $m \cdot y^k \ (mod\ p)$.

  Dacă se foloseşte compresia, atunci criptarea se aplică după compresia pachetului format din pachetul mesaj şi pachetul semnătură.

3.2 *OpenPGP*

Una din problemele pe care *PGP* (prin compania *PGP Inc.*) a trebuit să le rezolve a fost aceea de a combina caracterul de produs free cu posibilitatea de a utiliza cele mai bune sisteme de criptare, funcţii de dispersie şi de compresie. Cum algoritmi ca RSA sau

ElGamal nu pot fi utilizaţi fără licenţă, iar standardul intern al companiei menţionează explicit: "nu se va folosi nici un algoritm cu dificultăţi de licenţă", au apărut mai multe idei:

- Deoarece criptarea este una din componentele de bază din *PGP*, ar fi bine să se scrie propriul soft de securitate, care să fie inclus în produs.
- Să se construiască o variantă sub forma unui standard de tip open.

În final, Zimmermann a font convins de avantajele celei de a doua variante şi în iulie 1997, *PGP Inc.* a propus *IETF* (*Internet Engineering Task Force* – societatea care dezvoltă şi promovează standardele Internet) un standard nou numit *OpenPGP*. *IETF* obţine dreptul de a folosi numele *OpenPGP* pentru a descrie acest standard şi protocoalelele suportate de acesta.

OpenPGP este o variantă "legalizată" a *PGP* şi se dezvoltă permanent. Specificaţia actuală este *RFC* 4880 (apărută în noiembrie 2007).

Există şi variante de *OpenPGP* dezvoltate de diverse societăţi. Cea mai cunoscută în acest moment este *GNU Privacy Guard* (*GnuPG* sau *GPG*) propusă de *Free Software Foundation*.
GnuPG este free împrună cu codul sursă, sub licenţă *GPL* (*GNU* General Public License).

3.3 Comentarii finale

Luând în considerare informaţiile făcute publice, nu există o metodă cunoscută care să permită spargerea mesajelor *PGP* cu ajutorul atacului prin criptanaliză sau prin metode computaţionale. Pentru primele versiuni ale *PGP*-ului ([6]) s-au descoperit unele falii teoretice de securitate, care au fost rezolvate ulterior.

Securitatea *PGP*-ului se bazează pe ipoteza că algoritmii utilizaţi nu pot fi atacaţi prin criptanaliză directă cu echipamentele şi tehnologiile actuale. De exemplu, în versiunea originală, algoritmul *RSA* era utilizat pentru criptarea cheilor de sesiune; puterea criptografică a *RSA*-ului se bazează pe dificultatea problemei factorizării numerelor întregi. Începând însă cu a doua versiune, pentru criptarea cheilor a început să fie utilizat sistemul *IDEA*, pentru care nu sunt cunoscute vulnerabilităţi. Cum versiunile ulterioare au adăgat noi şi noi algoritmi, gradul de insecuritate al acestora variază odată cu algoritmii utilizaţi.

Periodic sunt lansate versiuni noi de *PGP*; dacă apar unele falii de securitate, acestea sunt rezolvate pe parcurs.

Exemplul 3.2. *Numeroase situaţii arată că autorităţile se găsesc adesea în imposibilitate de a decripta prin atacuri tradiţionale mesajele interceptate. De exemplu, în 2006, guvernul SUA – găsind aproape imposibilă accesarea fişierelor criptate cu PGP ale unui*

inculpat – i-a cerut acestuia să le furnizeze parola; acest lucru însă contravine amendamentului 5, care dă dreptul unui acuzat să nu se incrimineze singur.

Având în vedere utilizarea îndelungată a *PGP*-ului, îmbunătăţirile sistematice care i-au fost aduse, precum şi rezistenţa confirmată la atacuri prin criptanaliză, sistemul îşi dovedeşte viabilitatea.

3.4 *MIME*

MIME (Multipurpose Internet Mail Extension) ([2]) este un standard care are ca scop redefinirea formatelor mesajelor e-mail, permiţând:

- Folosirea în mesaje de caractere dinafara setului *ASCII*;
- Un set extins de formate pentru mesaje (înafara modului text);
- Mesaje compuse (Multipart message bodies);
- Headere cu informaţii care folosesc caractere dinafara setului *ASCII*.

S/MIME (*Secure MIME*) este o suplimentare (bazată pe tehnologia *RSA* Data Security) a securităţii formatului standard *MIME* destinat creşterii securităţii mesajelor poştei electronice. În general specificatţia[4] *S/MIME* este utilizată în standardul pentru uzul comercial şi instituţional, pe când *PGP* este folosit pentru utilizatorii obişnuiţi ai e-mailului.

Standardul *MIME* a apărut ca urmare a necesităţii transmiterii – prin intermediul poştei electronice – de imagini, înregistrări video sau mesaje text scrise în alte limbi decât cea engleză. Înaintea acestuia, transferul informaţiilor multimedia între parteneri putea fi realizat doar în anumite circumstanţe, cu programe de translatare speciale, efort şi costuri, iar cea mai mare parte a traficului poştei electronice era limitat la folosirea caracterelor ASCII. Mai mult, fiecare client de mail avea posibilitatea de a trimite şi de a recepţiona imagini, dar nu într-un format recognoscibil de către celelalte sisteme. *MIME* extinde poşta electronică la caractere *UNICODE* într-o manieră simplă, compatibilă cu versiunile anterioare şi totodată deschisă posibilităţii de extindere în continuare ([2]).

La scurt timp de la apariţie, formatul *MIME* şi-a găsit aplicaţii şi înafara domeniului poştei electronice. Când fondatorii World Wide Web au creat funcţionalitatea *hypertext*, s-a considerat uşor de întrebuinţat framework-ul *MIME* pentru definirea noului tip de date hypertext şi pentru specificarea scripturilor *HTML*. În momentul în care s-a trecut la definirea regulilor pentru *HTML*, încorporarea imaginilor în paginile web a devenit o simplă formalitate, deoarece pentru *MIME* se realizase deja definirea centralizată a formatelor tipurilor de imagini.

[4] *S/MIME* este o specificaţie, în timp ce *PGP* este un produs.

Formatul $MIME$ este derivat dintr-un alt format "tradiţional": RFC 822 (folosit şi astăzi).

3.4.1 Formatul RFC 822

Este formatul standard pentru mesajele text trimise prin poşta electronică. În contextul RFC 822 mesajele sunt privite similar scrisorilor obişnuite: un text pus într-un plic. Plicul conţine informaţia necesară pentru realizarea unei transmisii şi receptîonări corecte. Conţinutul este obiectul care va fi livrat destinatarului. Standardul RFC 822 se aplică doar conţinutului.

Conţinutul include o mulţime de câmpuri de antet care vor fi utilizate de sistem pentru crearea plicului, iar standardul este astfel creat încât să faciliteze accesarea automată a acestor informaţii de către diverse programe. Standardul RFC a fost descris pentru prima oară în 1982 de către Davin Crocker în "*Standard for the Format of ARPA Internet Text Messages*".

Un mesaj în format RFC 822 are două părţi: un antet (*header*) – utilizat de agentul care asigură serviciul de e-mail, şi un text ASCII (*corp*).

O linie de antet are forma:

$$< cuvant\ cheie >:< atribute >$$

Formatul permite ca o linie mai mare să fie scrisă pe mai multe linii. Cele mai utilizate cuvinte cheie sunt: *From, To, Subject* şi *Date*.

În funcţie de mesaj, pot apare şi alte linii de antet, cum ar fi *CC, List Info* etc.

Exemplul 3.3.

From: Cipher Editor < cipher − editor@ieee − security.org >
Subject: [Cipher Newsletter] Electronic CIPHER, Issue 60, May 18, 2004
Date: Tue, 18 May 2004 13:20:49 -0600
To: < cipher@mailman.xmission.com >
CC: < adrian@galaxyng.com >
List Info: "subscriptions for the IEEE online newsletter, Cipher"
Show details...

Alt câmp care apare frecvent în antetele formatului RFC 822 este *Message ID*. Această linie conţine un identificator unic asociat mesajului.

3.4.2 Structura formatului $MIME$

SMTP

SMTP (*Simple Mail Transfer Protocol*) este unul din primele protocoale (apare la începutul anilor 80) folosite pentru transmiterea mesajelor în format electronic pe Internet. Avantajul său – comparativ cu alte protocoale contemporane (de exemplu $UUCP$) – era

robusteţea sa în cazul când partenerii de dialog erau legaţi la reţea tot timpul. După 2001 existau cel puţin 50 implementări $SMTP$ (atât servere cât şi clienţi), cele mai cunoscute fiind *Postfix, qmail, Novell GroupWise, Exim, Novell NetMail* şi *Microsoft Exchange Server*.

În $SMTP$, comunicarea între client şi server se realizează numai în format $ASCII$ pe 7 biţi, numit NVT (*Network Virtual Terminal*). Cel mai răspândit format NVT este $NVT\ ASCII$. Acesta operează cu un set de caractere pe 8 biţi, în care cei mai puţin semnificativi 7 biţi sunt identici cu cei din $ASCII$, iar bitul cel mai semnificativ este 0.

Protocolul de comunicare între *Alice* şi server se desfăşoară astfel:

1. Iniţial *Alice* lansează protocolul de conexiune cu serverul şi aşteaptă ca acesta să-i răspundă cu mesajul *220 Service Ready*.

2. După primirea mesajului cu codul 220, *Alice* trimite comanda *HELO* prin care se identifică.

3. Odată ce comunicarea a fost stabilită, *Alice* poate trimite unul sau mai multe mesaje, poate încheia conexiunea sau poate folosi diverse servicii (de exemplu verificarea adreselor de e-mail).

   Serverul trebuie să răspundă după fiecare comandă, indicând dacă aceasta a fost acceptată, dacă se aşteaptă şi alte comenzi sau dacă există erori de scriere.

4. Pentru a trimite un mesaj se foloseste întâi comanda $MAIL$ prin care se specifică adresa lui *Alice*. Dacă această comandă este corectă serverul va răspunde cu *250 OK*.

5. *Alice* trimite apoi o serie de comenzi $RCPT$ prin care specifică destinatarii mesajului. Serverul va răspunde cu *550 No such user here*, sau *250 OK*, în funcţie de corectitudinea comenzii primite.

6. După ce destinatarii sunt identificaţi, *Alice* trimite comanda $DATA$, prin care anunţă serverul că va începe să scrie conţinutul mesajului.

   Serverul poate răspunde cu mesajul "*503 Command out of sequence*" sau "*554 No valid recipients*" dacă nu a primit comenzile $MAIL$ sau $RCPT$ sau aceste comenzi nu au fost acceptate.
   Dacă serverul răspunde cu *354 Start mail input*, *Alice* poate introduce textul mesajului.

7. Sfârşitul comunicării este marcat cu $< CR >< LF > . < CR >< LF >$.

Rezumând, un server $SMTP$ trebuie să cunoască cel puţin următoarele comenzi:

HELO	-	specificare *Alice* ca expeditor;
EHLO	-	specificare *Alice* cu cerere de mod extins;
MAIL FROM	-	date de identificare *Alice*;
RCPT TO	-	specificare *Bob* ca destinatar;
DATA	-	conţinutul mesajului;
RSET		Reset;
QUIT	-	încheierea sesiunii;
HELP	-	ajutor pentru comenzi;
VRFY	-	verificarea unei adrese;
EXPN	-	expandarea unei adrese;
VERB	-	informaţii detaliate.

Trecerea de la $SMTP$ la $MIME$

$MIME$ este o extensie a formatului RFC 822, construit cu scopul de a elimina o serie de restricţii ale protocolului $SMTP$. Astfel:

1. $SMTP$ nu poate transmite fişiere executabile sau alte obiecte binare.
   Deşi există diverse scheme de conversie a fişierelor binare în fişiere text care sunt folosite de $SMTP$, niciunul din ele nu este standard (nici măcar de facto).

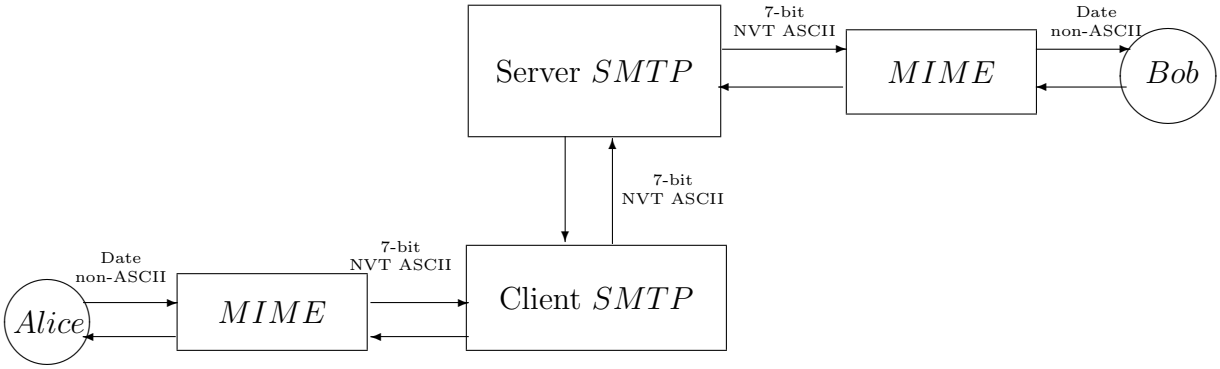
2. $SMTP$ nu poate transmite texte care conţin caractere specifice diverselor limbi (reprezentate prin coduri pe cel puţin 8 biţi) deoarece este limitat la $ASCII$ pe 7 biţi.

3. Serverele $SMTP$ acceptă doar mesaje limitate ca mărime.

4. Unele implementări $SMTP$ nu sunt pe deplin conforme chiar şi cu standardele $SMTP$ definite în RFC 822. Cele mai frecvente probleme apărute se referă la:

   - Ştergerea, adăugarea sau reordonarea comenzilor $< CR >$ (*Carriage Return*)şi $< LF >$ (*Line Feed*).
   - Trunchierea sau ştergerea liniilor mai lungi de 76 caractere;
   - Eliminarea spaţiilor albe (*Tab*-ul sau caracterul spaţiu);
   - Aranjarea liniilor din mesaj la o lungime standard;
   - Conversia caracterului Tab într-un multiplu de caractere spaţiu.

În figura de mai jos sunt reprezentate o serie de funcţii care permit transformarea caracterelor $nonASCII$ în caractere $ASCII$ şi invers.
Specificaţiile sunt cuprinse în standardele RFC 2045, ... RFC 2049.



O specificaţie $MIME$ conţine următoarele elemente:

1. Cinci linii noi în antet, care oferă informaţii despre corpul mesajului.

2. Reprezentări standardizate care suportă elemente multimedia pentru poşta electronică.

3. Codificări de transfer care permit conversia oricărui format într-un format standard, protejat la modificări efectuate de sistemul de mail.

Cele cinci linii noi de antet definite în $MIME$ sunt:

- $MIME - Version$: este însoţit de parametrul 1.0. El indică faptul că mesajul este codificat conform regulilor RFC 2045 şi RFC 2046.

- $Content - Type$: Descrie datele din mesaj, cu suficiente detalii pentru ca serverul de la destinaţie să poată selecta un mecanism adecvat care să reprezinte utilizatorului datele respective într-o manieră adecvată.

- $Content - Transfer - Encoding$: Indică tipul de transformare folosit pentru reprezentarea mesajului într-o manieră acceptabilă pentru transfer.

- $Content - ID$: Este folosit pentru a identifica în mod unic entităţile $MIME$, indiferent de contextul în care se află.

- $Content - Description$: Conţine o descriere a obiectului care formează mesajul; indicaţia este utilă atunci când mesajul nu este de tip text (de exemplu date audio).

Ultimele două cuvinte cheie sunt opţionale şi pot fi ignorate eventual în implementare.

Tipurile de date descrise prin specificaţia $Content - Type$

Tipurile pe care le poate avea conţinutul unui mesaj (conform cu RFC 2046) sunt sumarizate în tabelul următor:

Tip	**Subtip**	**Descriere**
Text	Plain	Un text neformatat (de exemplu $ASCII$)
	Enriched	Oferă o flexibilitate sporită a formatului.
Multipart	Mixed	Diferitele componente ale mesajului sunt independente dar sunt transmise împreună. Ele sunt prezentate destinatarului în ordinea în care apar în mesaj.
	Parallel	Diferă de *Mixed* prin faptul că ordinea componentelor în mesaj este aleatoare.
	Alternative	Componentele sunt versiuni alternative pentru aceeaşi informaţie. Ele sunt ordonate crescător după similitudinea cu originalul, iar sistemul de mail al destinatarului va lista varianta "cea mai bună".
	Digest	Diferă de *Mixed* prin faptul că tipul principal al fiecărei părţi este $message/rfc822$.
Message	rfc822	Corpul este un mesaj conform cu RFC 822.
	Partial	Permite fragmentarea unui mesaj mare într-un mod accesibil sistemului de mail al destinatarului
	External-body	Conţine un pointer la un obiect care există în altă parte.
Image	jpeg	Imaginea este în format $JPEG$, codificată $JFIF$.
	gif	Imaginea este în format GIF.
Video	mpeg	Format $MPEG$.
Audio	Basic	Se foloseşte un canal $ISDN$ pe 8 biţi, codificat la o rată standard de 8 kHz.
Application	PostScript	Postscript Adobe
	octet-stream	Date reprezentate în binar pe 8 biţi.

Sunt şapte tipuri generale de date şi 15 subtipuri (care specifică un anumit format pentru tipul de date respectiv).

MIME: *Content - Transfer - Encoding*

Altă componentă majoră a specificaţiei $MIME$ este o definire a codificării pentru corpurile mesajului. Atributele admise de această specificaţie sunt listate în tabelul următor.

Trei din aceste atribute ($7bit, 8bit, binary$) indică faptul că nu se aplică nici o codificare, oferind doar anumite informaţii despre natura datelor.
Pentru transferul $SMTP$ forma $7bit$ este suficient de sigură.

$7bit$	Datele sunt reprezentate prin linii scurte de caractere $ASCII$.
$8bit$	Liniile sunt scurte, dar pot fi caractere non-$ASCII$.
$binary$	Liniile nu sunt obligatoriu suficient de scurte pentru transferul $SMTP$.
$quoted-printable$	Dacă datele conţin suficient de mult text $ASCII$, acesta rămâne accesibil unei citiri directe.
$base64$	Fiecare bloc de 6 biţi se codifică într-un bloc de 8 biţi – caracter $ASCII$ printabil.
$x-token$	Codificare nestandard.

$MIME$ utilizează două metode de codificare a datelor: $quoted-printable$ şi $base64$[5]. Prima permite un transfer uşor de citit, iar a doua oferă o securitate sporită tuturor tipurilor de date.

3.4.3 Securitatea $MIME$ bazată pe $OPENPGP$

Încercarea de a combina protocoalele PGP şi $MIME$ este naturală, dar a întâmpinat mai multe dificultăţi, cea mai semnificativă fiind dată de incapacitatea de a recupera conţinutul mesajelor semnate fără parsarea structurilor de date specifice PGP-ului.

Standardul RFC 1847 defineşte formate de securitate multipart pentru $MIME$. Acestea separă conţinutul mesajului semnat de semnătură. PGP-ul poate genera în urma criptării mesajelor fie caractere $ASCII$, fie şiruri arbitrare de octeţi, generând şi o semnătură sau extrăgând datele cheii publice. Protocolul de transmitere a datelor solicită ca acestea să fie în format $ASCII$. Dacă mesajul trebuie transmis în mai multe părţi, trebuie folosit mecanismul $MIME$ de transmitere parţială, în locul formatului $OpenPGP\ ASCII$ multipart.

Agenţii de mail tratează şi interpretează mesajele multipart în mod opac, ceea ce înseamnă că datele nu vor suferi nici o alterate. Totuşi, unele servere pot detecta faptul că următorul gateway[6] nu suportă formatul $MIME$ sau datele pe 8 biţi şi va realiza o conversie la $base64$.
Aceasta reprezintă o problemă serioasă pentru mesajele semnate, semnătura lor fiind invalidată la efectuarea unei astfel de operaţii. Din acest motiv, toate mesajele semnate folosind acest protocol trebuie restrânse la o reprezentare pe 7 biţi.

Înainte de criptarea cu $OpenPGP$, datele trebuie trecute în forma canonică $MIME$ (descrisă mai sus). Criptarea cu $OpenPGP$ este anunţată prin antetul *content type multipart/encrypted* şi trebuie să aibă valoarea parametrului de protocol *application/pgp-encrypted*.

Corpul mesajului criptat cu $MIME$ este compus din două părţi:

[5]Codificarea "$base64$" este identică cu "$radix-64$" folosită de PGP. Singura diferenţă constă în adăugarea – la $base64$ – a unei sume de control pe 24 biţi. Suma este calculată la intrarea datelor, înainte de codificare, iar apoi este codificată cu algoritmul $radix64$, fiind separată (în transmisie) de restul datelor prin simbolul "=".

[6]Un "gateway" este un computer/server care asigură transferul de informaţie între diverse reţele de comunicaţie.

- Prima parte are antetul de conţinut *application/pgp-encrypted* şi deţine informaţia de control.
- A doua parte trebuie să conţină mesajul criptat; ea este etichetată cu antetul *application/octet-stream*.

Pentru generarea semnăturii digitale cu *OpenPGP*:

- Mesajul care urmează să fie semnat trebuie adus la forma canonică specifică antetului *content type*.
- Se aplică o criptare de tipul *Content Transfer Encoding*. În particular, capetele de linie ale mesajului trebuie să folosească secvenţe $< CR >< LF > (LF$.
- Se adaugă antetele de conţinut $MIME$.
- Din mesajul semnat trebuie înlăturate spaţiile.
- Semnătura digitală se calculează atât pentru datele care urmează să fie semnate cât şi pentru mulţimea antetelor de conţinut.
- Semnătura trebuie să fie detaşată de mesajul pe care-l însoţeşte, pentru a se evita modificarea mesajului (cu o anumită procedură).

Se observă că *OpenPGP*-ul respectă convenţia ca mesajul semnat să se termine cu o secvenţă $< CR >< LF >$.

La primirea unui mesaj semnat, un client de mail trebuie:

- Să aducă capetele de linie $< LC >< LF >$ la forma canonică, înainte de verificarea semnăturii.
- Să trimită serviciului de verificare a semnăturii mesajul semnat şi antetele de conţinut, împreună cu semnătura *OpenPGP*.

Uneori se doreşte atât semnarea digitală, cât şi criptarea mesajului care urmează să fie trimis. Există două modalităţi pentru realizarea acestui lucru:

- Mesajul este întâi semnat conform antetului *multipart/signature* şi apoi criptat conform antetului *multipart/encrypted*, pentru a forma corpul final al mesajului.
  Acesta este cel mai folosit standard pentru trimiterea mesajelor.
- Pachetul *OpenPGP* descrie o modalitate pentru semnarea şi criptarea datelor într-un singur mesaj *OpenPGP*.
  Această metodă favorizează reducerea supraprocesării şi creşte compatibilitatea cu implementările non-$MIME$ ale *OpenPGP*-ului. Mesajele criptate şi semnate în

această manieră combinată trebuie să urmeze aceleaşi reguli canonice ca un obiect *multipart/signed*. Este permis în mod explicit unui agent de mail să decripteze un mesaj combinat şi să-l rescrie ca un obiect *multipart/signed* folosind semnătura încorporată în versiunea criptată.

3.4.4 Controverse legate de $MIME$

Pe parcursul definirii standardului $MIME$ au apărut diverse controverse şi dileme. Câteva caracteristici discutate, dar care nu au fost incluse în standardul final, sunt:

- **Criptarea imbricată**: Deşi prezentă în primele versiuni ale formatului, în final s-a renunţat la aceasta deoarece structura de ansamblu a mesajului nu putea fi vizibilă fără o decodificare prealabilă. În al doilea rând, pot exista criptări imbricate în care aceeaşi informaţie este codificată de mai multe ori. O consecinţă a acestei decizii constă în creşterea complexităţii pentru gateway.
- **Uuencode**: A fost luat în considerare la început pentru a înlocui *base*64. A fost respins deoarece:
  - nu există o definire riguros formalizată;
  - rezultatul în urma aplicării acestuia nu era robust pentru mai multe noduri gateway.
  - uneori fişierele Uuencode ajung într-o formă care nu poate fi decodificată.
- **Compresia**: Deşi iniţial s-a considerat necesar un algoritm de compresie, acesta nu a fost adoptat datorită situaţiei legale incerte privind algoritmii de compresie şi lipsei unei experienţe îndelungate a autorilor $MIME$ în acest domeniu.
- **Numărul atributelor Content-type**: S-au discutat propuneri legate de un număr nelimitat de atribute, dar în acelaşi timp s-a pus problema uşurinţei recunoaşterii acestora în cadrul antetului. În final, mecanismul sub-tipurilor a fost adoptat ca un compromis.

3.5 $S/MIME$

3.5.1 Comparaţii cu PGP şi $OpenPGP$

$S/MIME$ are o utilizare limitată doar la poşta electronică, în timp ce PGP-ul are şi alte aplicaţii cum ar fi în VPN (*Virtual Private Network*), criptarea volumelor hard-disk-ului, arhivarea şi criptarea fişierelor, batch processing etc.

Lungimea cheilor are o importanţă deosebită pentru PGP şi $S/MIME$; din acest punct de vedere securitatea oferită de $S/MIME$ este mult mai scăzută: chei de până la

1024 biţi în timp ce specificaţia $ANSI$ recomandă chei de minim 1024 biţi pentru RSA şi Diffie-Hellman (DH).

Una din specificaţiile $S/MIME$ ([3]), descrie o facilitate numită PoP (*Proof of Possesion of Private Key*) care permite transmiterea şi stocarea cheilor private de către autoritatea de certificare, în momentul în care utilizatorul solicită un certificat (a se vedea capitolul dedicat PKI). PGP-ul nu are o astfel de problemă de securitate legată de recuperarea cheilor, dar are un mecanism care a stârnit multe controverse, numit *Additional Key Recovery* (ADK). Această facilitate nu este inclusă însă în specificaţia $OpenPGP$.

Se pot construi versiuni ale protocolului $S/MIME$ care sunt exemple clare ale noţiunii de *cleptografie*[7] – prin care se implementează rutine de generare a cheilor DH sau RSA care produc chei aparent normale dar care conţin o trapă secretă prin care un terţ poate recupera cheia privată. Acest proces reprezintă o extindere interesantă a conceptului de canale subliminale. Astfel de canale nu pot exista însă în cazul PGP-ului: orice inspectare a codului sursă ar depista rapid orice problemă de acest tip.

Utilizatorii $S/MIME$ dinafara SUA trebuie să utilizeze chei simetrice $RC2/40$ cu o lungime de 40 biţi şi chei publice mai mici de 582 biţi.

Deci, din perspectiva securităţii PGP-ul poate fi considerat de departe mai sigur decât $S/MIME$, cel puţin din două motive:

- $S/MIME$ este limitat la chei publice de 1024 biţi; PGP-ul poate utiliza chei până la 4096 biţi.
- Specificaţia PGP-ului este publică, fiind aşadar supusă unei permanente revizuiri de către toţi specialiştii, în timp ce utilizatorii $S/MIME$ pot doar să aibă încredere în eficienţa implementării.

3.5.2 Funcţionalitatea protocolului $S/MIME$

Din acest punct de vedere, $S/MIME$ şi PGP seamănă foarte mult. Amândouă oferă posibilitatea de a semna şi/sau cripta mesaje.

Să detaliem aceste funcţii pentru $S/MIME$:

- Poate cripta orice tip de conţinut şi orice cheie de criptare, pentru unul sau mai mulţi destinatari (*enveloped data*).
- Poate semna date (*signed data*). O semnătură digitală este generată folosind amprenta corpului mesajului care va fi criptat cu cheia secretă a expeditorului. Perechea formată din corpul mesajului şi semnătură sunt apoi codificate cu *base*64.

  Această funcţie poate fi prelucrată numai dacă destinatarul este dotat cu facilităţi $S/MIME$.

[7] Cleptografia reprezintă studiul sustragerii de informaţii în mod securizat şi subliminal, constituind o extensie naturală a teoriei canalelor subliminale.

- Poate semna date în clar (*clear-signed data*). Ca şi în cazul anterior, se generează o semnătură a corpului mesajului, după care semnătura (şi numai aceasta) se codifică cu *base*64.

  În acest caz orice destinatar (chiar şi fără $S/MIME$) poate citi mesajul, deşi nu poate verifica semnătura.

- Poate semna şi cripta date (*signed and enveloped data*). Corpul mesajului (criptat sau nu) se semnează, apoi semnătura (singură sau împreună cu corpul mesajului – criptat sau nu) se criptează din nou.

3.5.3 Algoritmii criptografici folosiţi de $S/MIME$

Principalii algoritmi criptografici folosiţi de $S/MIME$ sunt:

- **Pentru crearea unei amprente**: Funcţiile de dispersie folosite sunt $SHA-1$ şi $MD5$.

- **Pentru criptarea amprentei** (şi generarea semnăturii):
  - Ambele părţi folosesc DSS.
  - Expeditorul foloseşte RSA pentru criptare.
  - Destinatarul trebuie să poată verifica semnături RSA cu chei până la 1024 biţi.

- **Pentru criptarea cheilor de sesiune**:
  - Ambele părţi folosesc *ElGamal*.
  - Expeditorul mai poate folosi pentru criptare RSA cu chei până la 1024 biţi.
  - Destinatarul trebuie să poată efectua decriptări RSA.

- **Pentru criptarea mesajelor cu chei de sesiune one-time**:
  - Expeditorul poate folosi criptarea cu $3DES$ sau $RC2/40$.
  - Expeditorul trebuie să poată decripta $3DES$ (sau eventual $RC2/40$).

De remarcat similitudinea cu algoritmii de criptare folosiţi de PGP. Varianta utilizării $RC2$ pe 40 biţi este recomandată doar pentru implementările sistemelor americane care mai menţin în unele servicii acest sistem de criptare (destul de slab).

Specificaţiile $S/MIME$ includ o discuţie asupra procedurii de selecţie a algoritmului de criptare.

De obicei, *Alice* trebuie să facă două alegeri. Întâi, ea trebuie să determine dacă *Bob* este capabil să decripteze un mesaj criptat cu un sistem de criptare dat. Apoi, dacă *Bob*

acceptă numai texte criptate cu sisteme slabe, *Alice* trebuie să decidă dacă poate trimite mesajul folosind o criptare slabă. Pentru aceasta, toţi partenerii trebuie să anunţe în prealabil capacităţile solicitate pentru decriptarea mesajelor pe care le trimit, în ordinea descrescătoare a preferinţelor.
Fiecare destinatar poate stoca aceste informaţii pentru a le folosi ulterior.

Pentru a trimite un mesaj, *Alice* trebuie să verifice următoarele variante (în ordine):

1. Dacă ea are o listă de capacităţi de decriptare ale lui *Bob*, va alege primul sistem de criptare pe care este capabilă să-l folosească.

2. Dacă nu are lista lui *Bob*, dar deţine cel puţin un mesaj de la el, atunci mesajul trimis de *Alice* va folosi acelaşi algoritm de criptare folosit în mesajele respective.

3. Dacă nu deţine nici o informaţie despre capacităţile de decriptare ale lui *Bob*, dar este dispusă să rişte ca mesajul să nu poată fi citit, va trimite un mesaj criptat cu $3DES$.

4. Dacă nu deţine nici o informaţie despre capacităţile de decriptare ale lui *Bob*, şi nu vrea să rişte ca mesajul să nu poată fi citit de acesta, atunci *Alice* va trimite un mesaj criptat cu $RC2/40$.

În plus, mesajele trimise prin poşta electronică (în special pentru $S/MIME$) beneficiază şi de o serie de protecţii PKI (certificări, semnături ale unei autorităţi centrale etc) recomandate de laboratoarele RSA.

3.5.4 Servicii de securitate îmbunătăţite pentru $S/MIME$

Definiţia 3.1. *Un mesaj triplu împachetat reprezintă un mesaj care a fost semnat, criptat şi apoi semnat din nou.*

Expeditorii care semnează mesajul interior şi pe cel exterior pot fi una sau aceeaşi persoană, sau entităţi diferite. Specificaţia $S/MIME$ nu limitează numărul de împachetări încuibărite, deci se pot realiza mai mult de trei împachetări ale aceluiaşi mesaj.

Semnătura interioară este utilizată pentru verificarea integrităţii conţinutului, a nonrepudierii originii acestuia, precum şi adăugarea la mesaj a anumitor atribute. Aceste atribute sunt transmise de la expeditorul iniţial până la destinatarul final, indiferent de numărul de entităţi intermediare.

Atributele semnate pot fi folosite pentru controlarea accesului la corpul mesajului. Conţi-nutul criptat al mesajului asigură confidenţialitatea acestuia, inclusiv confidenţialitatea atributelor incluse în semnătura interioară.

Semnătura exterioară asigură autenticitatea şi integritatea informaţiei procesate "salt-cu-salt"[8], fiecare *salt* reprezentând o entitate intermediară – un agent de mail de exemplu.

Semnătura exterioară asociază atribute corpului criptat. Aceste atribute pot fi folosite pentru controlul accesului şi luarea deciziilor de rutare.

Mesaje triplu împachetate

Etapele urmate pentru crearea unui mesaj triplu împachetat sunt:

1. Se porneşte de la conţinutul original (corpul mesajului).

2. Mesajul original este încapsulat cu antetele $MIME$ *content-type* corespunzătoare.

3. Se semnează antetul $MIME$ şi conţinutul original rezultate din pasul 2.

4. Se adaugă o structură $MIME$ corespunzătoare mesajului rezultat la pasul 3. Mesajul rezultat se numeşte *semnătură interioară*.

5. Rezultatul se criptează ca un bloc unitar, într-o structură $MIME$ sau obiect *pkcs7 - mime*.

6. Se adaugă antetele $MIME$ corespunzătoare: *content-type*-ul pentru structura $MIME$ sau obiectul *pkcs7 - mime* cu parametri, precum şi antete $MIME$ opţionale, cum ar fi *Content-Transfer-Encoding* sau *Content-Disposition*.

7. Rezultatul de la 6 (antetele $MIME$ şi corpul criptat) se semnează ca un bloc unitar.

8. Utilizând o logică asemănătoare cu cea de la pasul 4, se adaugă o structură $MIME$ corespunzătoare mesajului semnat de la pasul 7. Mesajul rezultat poartă numele de *semnătură exterioară* şi reprezintă – de fapt – mesajul triplu împachetat.

Un mesaj triplu împachetat are mai multe nivele (layere) de încapsulare. Structura diferă în funcţie de formatul ales pentru semnarea porţiunilor mesajului. Datorită modului în care $MIME$ structurează datele, acestea nu apar într-o anumită ordine, ceea ce face ca noţiunea de *layer* să devină oarecum ambiguă.

[8] *Salt-cu-salt (hop-by-hop)* reprezintă un tip de rutare; operaţia de rutare constă în determinarea nodului intermediar următor (adiacent), care la rândul lui poate redirecţiona mesajele către destinaţia finală şi, în general, nu permite determinarea întregii secvenţe de noduri intermediare.

3.6 *PEM*

3.7 Anexa 1: Algoritmul de compresie *ZIP*

PGP foloseşte algoritmul de compresie *ZIP* scris de Jean-Lup Gailly, Mark Adler şi Richard Wales şi folosit ca utilitar pe UNIX. Funcţional, el este echivalent cu PKZIP, sistemul creat de PKWARE şi folosit de Windows.
Algoritmul de zipare este – se pare – cel mai utilizat algoritm de compresie; este free şi a fost implementat pentru toate sistemele actuale de calcul.

Bazele teoretice ale *ZIP*-ului (şi a tuturor variantelor sale) au fost puse în 1977 de Jacob Ziv şi Abraham Lempel, care au construit algoritmul *LZ*77 de spargere a informaţiei conţinute intr-un buffer.
Motivaţia este că multe cuvinte dintr-un text (patternuri de imagini în cazul unui GIF etc) se repetă. Această secvenţă care se repetă poate fi înlocuită cu un cod scurt.

Exemplul 3.4. *([7], pag.* 392*) Să considerăm următorul text (fără semnificaţie semantică):*

the brown fox jumped over the brown foxy jumping frog

Lungimea lui este de 53 *octeţi* = 424 *biţi. Algoritmul LZ*77 *parcurge texul de la stânga spre dreapta.*
Iniţial, fiecare caracter este codificat într-o secvenţă de 9 *biţi, unde primul bit este* 1*, iar următorii opt biţi conţin codul ASCII al caracterului.*
Algoritmul caută secvenţele de lungime maximă care se repetă. În cazul nostru, aceasta este **the brown fox**. *Prima apariţie a secvenţei este păstrată, iar celelalte sunt înlocuite de un pointer la această apariţie şi de un număr care dă lungimea secvenţei. În cazul nostru, a doua apariţie se înlocuieşte cu perechea* $(26, 13)$ *(secvenţa de aici repetă pe cea care a început* 26 *caractere mai devreme, pe o lungime de* 13 *caractere).*

Să presupunem că sunt două opţiuni de codificare a perechii: un pointer pe 8 *biţi şi un număr pe* 4 *biţi (opţiunea* 00*) sau un pointer pe* 12 *biţi şi un număr pe* 6 *biţi (opţiunea* 01*). Opţiunea este dată la începutul perechii.*

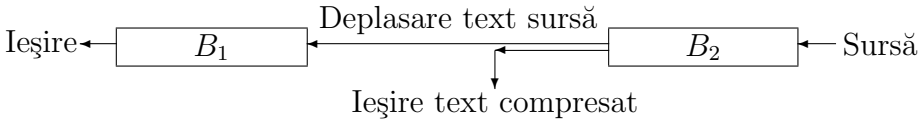
Deci a doua apariţie a secvenţei **the brown fox** *este codificată* $< 00_b >< 26_d >< 13_d >$ *(b - binar, d - zecimal), sau* 00 00011010 1101.

În continuare, din mesaj a rămas litera **y***, secvenţa* $< 00_b >< 27_d >< 5_d >$ *(care înlocuieşte un spaţiu, plus textul* **jump***) şi secvenţa de caractere* **ing frog**. *A rămas*

the brown fox jumped over $0_b26_d13_d$ **y** $0_b27_d5_d$ **ing frog**

Textul compresat are 35 *caractere de câte* 9 *biţi şi două coduri; în total* $35 \times 9 + 2 \times 14 = 343$ *biţi; o rată de compresie de* $1,24$.

Practic, algoritmul *LZ*77 foloseşte două buffere (în figură, B_1 şi B_2).



B_1 conţine ultimele n caractere ale sursei care au fost prelucrate, iar B_2 conţine următoarele p caractere care urmează să fie prelucrate.

Algoritmul caută dacă un prefix α ($|\alpha| = s \geq 2$) din B_2 se află ca subşir în B_1.
Dacă nu, primul caracter din B_2 iese codificat pe 9 biţi în secvenţa compresată şi – simultan – este deplasat în B_1. Dacă α este în B_1, se continuă căutarea pentru a găsi cel mai lung subşir comun, de lungime k. Acesta este scos ca text compresat sub forma unui triplet (indicator, pointer, k). Cele mai din stânga k caractere din B_1 sunt eliminate, iar cele k caractere din topul lui B_2 trec în B_1.

3.8 Anexa 2: Conversia $radix - 64$

Informaţiile transferate prin comunicaţiile digitale sunt secvenţe de biţi, fără nici o formatare. Algoritmii de recepţie formatează biţii în blocuri de 64, 32 sau 8 biţi.
Deoarece sistemele de e-mail permit transmiterea şi recepţia doar a codurilor $ASCII$ pe 8 biţi, protocoalele de e-mail convertesc textul criptat alocând fiecărui grup de 6 biţi, un caracter $ASCII$ printabil. Algoritmul de codificare folosit de ambele protocoale (PGP, $S/MIME$) este numit $radix - 64$ ([1]).

Principalele sale caracteristici sunt:

- Domeniul de definiţie este format din mulţimea caracterelor reprezentabile binar (atenţie ! nu o anumită codificare binară, ci orice caracter reprezentabil în binar).

- Mulţimea valorilor este formată din 65 caractere printabile: unul este pentru legătură (pad), iar celelalte $2^6 = 64$ sunt reprezentate pe 6 biţi.

- Nu există caractere de control; deci un mesaj codificat în $radix - 64$ va trece fără probleme prin sisteme de poştă care scanează secvenţele pentru depistarea caracterelor de control.

- Caracterul $-$ (cu semnificaţie în multe sisteme, inclusiv RFC 822) nu este folosit; deci el trebuie eliminat anterior.

Codificarea celor 64 caractere din sistemul $radix - 64$ este dată de tabelul următor. Sunt folosite cele 52 litere (mari şi mici), cele zece cifre şi caracterele $+, /$.

Valoare 6-biţi	Cod Caracter	Valoare 6-biţi	Cod Caracter	Valoare 6-biţi	Cod Caracter	Valoare 6-biţi	Cod Caracter
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						pad	=

Procedura de codificare este simplă: fiecare secvenţă de 3 octeţi (24 cifre binare) este împărţită în patru secvenţe de 6 biţi fiecare. Apoi fiecare grup de 6 biţi este codificat prin caracterul din tabel.

Deci o intrare de 24 biţi este expandată la ieşire pe 32 biţi.

Exemplul 3.5. *Fie secvenţa de* 24 *biţi*

$$00100011\ \ 01011100\ \ 10010001$$

($235C91$ *în hexazecimal). Aranjată în grupuri de câte* 6 *biţi, ea este*

$$001000\ \ 110101\ \ 110010\ \ 010001.$$

Valorile zecimale ale acestor patru secvenţe binare sunt $8, 53, 50$ *şi* 17*.*
Folosind acum tabela $radix-64$ *se obţine* $I1yR$*.*

Trecută în format $ASCII$ *(*8 *biţi, cu* 0 *pe bitul de paritate), avem*

$$01001001\ \ 00110001\ \ 01111001\ \ 01010010$$

Sau – în hexazecimal – 49317952*.*

Altfel spus, codificarea $radix-64$ *transformă* $235C91$ *în* 49317952*.*

3.9 Exerciţii

1. *PGP* foloseşte pentru $CAST-128$ modul de criptare CFB, în timp ce majoritatea sistemelor de criptare preferă modul CBC. Reamintim:

În modul CBC : $C_i = e_K(C_{i-1} \oplus P_i), \quad P_i = C_{i-1} \oplus d_K(C_i)$

În modul CFB : $C_i = e_K(C_{i-1}) \oplus P_i, \quad P_i = C_i \oplus e_K(C_{i-1})$

Ambele moduri oferă acelaşi grad de securitate. Sugeraţi un motiv pentru care PGP preferă folosirea modului CFB.

2. Comprimaţi folosind codul $LZ77$ textul

A fost odată ca-n poveşti, A fost ca niciodată.

3. Formalizaţi în pseudocod algoritmul $LZ77$. Scrieţi un program pentru implementarea lui.

4. Scrieţi programe pentru codificarea şi decodificarea mesajelor folosind $radix - 64$.

Bibliografie

[1] D. Atkins, W. Stallings, P. Zimmermann – *PGP mesage exchange formats* (RFC 1001), http://www.ietf.org/rfc/rfc1991.txt?number=1991

[2] N. S. Borenstein – *MIME: A Portable and Robust Multimedia Format for Internet Mail*, Springer-Verlag, 1993.

[3] M. Myers, X. Liu, B. Fox, H. Prafullchandra, J. Weinstein – *Certificate Request Syntax*, (1997) http://tools.ietf.org/html/draft-ietf-smime-crs-00

[4] M. Mogollon – *Cryptography and Security Services: Mechanisms and Applications*, Cybertech Publishing, 2007.

[5] M. Y. Rhee – *Internet Security - Cryptographic Principles, Algorithms and Protocols*, Wiley Interscience, 2003.

[6] B. Schneier – *Applied Cryptography*, ed. II-a, Wiley Interscience, 1996

[7] W. Stallings – *Cryptography and Network Security: Principles and Practice*, Prentice Hall, Second Editoon, 1999.

[8] P. Zimmermann – *An introduction to Cryptography*, Network Associates, 2(2003) http://www.pgpi.org/doc/guide/6.5/en/intro/