TEMA 8: FUNCȚII HASH CRIPTOGRAFICE ȘI SEMNĂTURI DIGITALE

Obiectivele temei:

- Introducere în funcții hash criptografice și analiza standardelor de hashing.
- Introducere în semnături digitale și analiza celor mai utilizate scheme de semnare.

Cuvinte-cheie:

funcție hash,
message-digest,
coliziuni,
MD5,
familia SHA,
semnătură digitală,
schema de semnătură RSA,
schema de semnătură ElGamal,
certificate digitale,
centru de certificare.

Criptografia cu cheie secretă și cea cu cheie publică ajută la asigurarea confidențialității informației. Al treilea tip de algoritmi criptografici îl constituie *funcțiile hash criptografice,* care participă la asigurarea integrității și autenticității informației.

8.1. Funcții hash criptografice

Funcțiile hash criptografice joacă un rol fundamental în criptografia modernă. Cu toate că sunt înrudite cu funcțiile hash convenționale, utilizate în mod obișnuit în aplicațiile informatice non-criptografice, ele diferă în mai multe aspecte importante. Aici ne vom limita doar la funcțiile hash criptografice (în continuare, pur și simplu funcții hash) și, în special, la utilizarea lor pentru asigurarea integrității datelor și pentru autentificarea mesajelor.

8.1.1. Noțiuni generale și proprietăți ale funcțiilor hash

O funcție hash efectuează transformări asupra unei intrări de o lungime arbitrară generând o ieşire de lungime fixă care este o imagine a intrării. Valoarea imaginii returnate se numeşte *valoare hash*. Aceste funcţii se mai numesc şi funcţii „*message digest*”, adică „*esența mesajului*”, deoarece din valoarea hash este imposibil de reconstruit intrarea din care s-a făcut imaginea. Există multe funcţii care pot transforma intrarea de lungime arbitrară în o ieşire de lungime fixă, însă funcţiile care prezintă interes din punct de vedere criptografic sunt funcţiile hash de sens unic (*one–way hash functions*). Ecuaţia generală care descrie funcţiile hash este

$$h = H(m),$$

unde m este intrarea, H este funcţia şi h – valoarea hash.

Caracteristicile speciale ale funcţiilor de sens unic sunt următoarele:

- pentru orice intrare m este uşor de calculat h;
- având h este dificil de calculat m astfel încât se fie satisfăcută relaţia $h = H(m)$;

- pentru un m dat este dificil de găsit m_1 astfel încât $H(m) = H(m_1)$; în acest caz m și m_1 sunt numite coliziuni.

Aceste proprietăţi ale funcţiilor hash de sens unic folosite în criptografie sunt extrem de importante, deoarece dacă un atacator ar putea găsi cu uşurinţă o valoare de intrare m_1 care produce acelaşi rezultat cu m în cazul în care cineva semnează m, atacatorul ar putea afirma că a fost semnat m_1. O cerinţă adiţională pentru aceste funcţii este că pentru ele să fie dificil de găsit două mesaje aleatoare care să aibă acelaşi imagine, altfel spus:

- este dificil de găsit mesajele aleatoare m şi m_1 astfel încât $H(m) = H(m_1)$.

Un posibil atac, dacă ultima cerinţă nu este îndeplinită, ar fi atacul numit „atacul zilei de naştere" (birthday atack). Acest tip de atac şi-a primit numele de la paradoxul zilelor de naştere. În mod surprinzător (este dovedit matematic) probabilitatea ca în orice grup de 23 de persoane doi sau mai mulţi indivizi să aibă acelaşi zi de naştere este mai mare decât $\frac{1}{2}$.

Atacul, descris de *Bruce Schneier*, presupune folosirea unei funcţii hash care produce un rezultat de 64 biţi. Paşii care trebuie parcurşi pentru ca Bob să obţină o semnătură de la Alice pe un document care n-ar fi semnat, în condiţii normale, niciodată de Alice sunt:

1. Bob pregăteşte două versiuni ale documentului, una care este în ordine şi va fi semnat de Alice, şi una care conţine informaţii false care nu ar fi semnate de Alice;
2. Bob face schimbări subtile pe fiecare document (de ex. inserare de spaţii etc.), obţinând 2^{23} variante ale documentelor;
3. Bob calculează valorile Hash a fiecărei variante ale celor două documente, şi găseşte câte una din fiecare grup care produc aceleaşi valori hash;
4. Bob trimite varianta modificată a documentului pe care Alice este dispusă să semneze, folosind un protocol unde se semnează numai valoarea hash;
5. Acuma Bob poate susţine că documentul pereche cu informaţiile false a fost semnată de Alice.

Nu este uşor de găsit o funcţie care satisface toate cerinţele de mai sus, totuşi există destul de multe, mai mult sau mai puţin sigure, funcţii hash folosite în criptografie. Mai jos urmează descrierea succintă a unora dintre cele mai cunoscute astfel de funcţii.

8.1.2. Algoritmul MD5

MD5 (MD provine din *Message Digest*) este o funcţie hash de sens unic, proiectat de Ron Rivest și face parte din seria MD de funcții hash: MD2, MD4, MD5, MD6. Algoritmul produce un hash, sau altfel zis imagine, de 128 biţi a mesajului de intrare.

Principiile după care s-a realizat algoritmul sunt următoarele:

- *Securitate*. Este imposibil de găsit două mesaje care au aceiaşi imagine, presupunând că nu există altă metodă de criptanaliză decât cea a forţei brute.
- *Securitate directă*. Securitatea nu se bazează pe presupuneri, cum ar fi de exemplu dificultatea de a factoriza numere mari în cazul RSA.
- *Viteză*. Algoritmul trebuie să fie potrivit pentru implementări rapide de software, bazându-se pe manipulaţii de bit cu operanzi de 32 biţi.
- *Simplicitate şi compactitate*. Algoritmul trebuie să fie cât se poate de simplu, fără structuri mari de date sau un program complicat.
- *Favorizare de arhitecturi Little-Endian.* Algoritmul este optimizat pentru arhitecturi de microprocesoare (mai ales Intel).

Variantă precedentă a algoritmului a fost MD4, dar acesta după ce a fost introdus a fost criptanalizat cu succes, ce l-a îndemnat pe autor să-si îmbunătăţească codul. Astfel a fost conceput MD5, ca variantă MD4 îmbunătăţită.

MD-5 este proiectat în baza construcției (funcției) hash Merkle–Damgard, o metodă de elaborare a funcțiilor hash rezistente la coliziuni (figura 8.1). Astfel, după nişte procesări iniţiale MD5 procesează textul de intrare în blocuri de 512 de biţi, care sunt mai departe separaţi în 16 sub-blocuri de 32 biţi fiecare. Algoritmul produce un set de 4 blocuri de 32 biţi, care concatenate dau ieşirea de 128 bit.

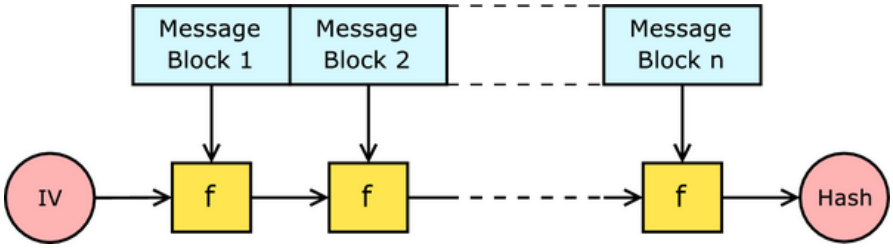

Figura 8.1. *Shema construcţiei Merkle–Damgard*

Mai detaliat, inițial mesajul este completat, iar lungimea finală a lui va fi un multiplul al lui 512. Acest procedeu se realizează adăugând un bit de 1 la sfârşitul mesajului şi atâtea zerouri câţi sunt necesari pentru ca mesajul original să aibă o lungime cu 64 de biţi mai scurtă decât un multiplu a lui 512. Biții rămași fiind completați cu 64 de biţi care reprezintă lungimea originală a mesajului modulo 2^{64}. Acest procedeu asigură că completarea va arăta diferit pentru mesaje diferite.

Pentru calcule, sunt inițializate 4 variabile cu dimensiunea de 32 de biți, iar valorile inițiale sunt date în cifre hexazecimale:

A = 0x01234567,
B = 0x89abcdef,
C = 0xfedcba98,
D = 0x76543210.

Starea inițială ABCD se numește vectorul de inițializare.

Cele patru variabile sunt copiate în alte variabile: A în a, B în b, C în c iar D în d.

De asemenea, sunt utilizate 4 funcții neliniare pentru patru runde și tabelul constantelor $T[1\ldots64]$:

$$F(X,Y,Z) = (X \,\&\, Y) \mid ((!X) \,\&\, Z),$$
$$G(X,Y,Z) = (X \,\&\, Z) \mid (Y \,(!Z)),$$
$$H(X,Y,Z) = X \text{ xor } Y \text{ xor } Z,$$
$$I(X,Y,Z) = Y \text{ xor } (X \mid (!Z)),$$
$$T[n]=\text{int}(2^{32} \cdot \text{abs}|\sin n|),$$

unde '*xor*', '&', '|' și '*!*' sunt operatorii logici pe biți *XOR, AND, OR* și *NOT*. Valiabilele *X*, Y și *Z* sunt numite cuvinte, la fel și valorile fiecărei funcții.

Fiecare bloc de 512 biți trece prin patru cicluri de calcul a câte 16 runde fiecare. Pentru aceasta blocul este reprezentat ca un vector *X* cu 16 cuvinte de 32 de biți fiecare. Toate rundele au aceeași structură de forma [*abcd k s i*], definită în felul următor

$$a = b+((a + f(b, c, d)+X[k]+ T[i]) <<< s),$$

unde *f* este funcția neliniară utilizată la runda respectivă, *k* este indicele cuvântului de 32 de biți din blocul curent de 512 de biți al mesajului, iar <<< *s* reprezintă deplasarea ciclică la stânga cu *s* biți a argumentului de 32 de biți obținut. Numărul *s* este specificat separat pentru fiecare rundă.

În final rezultatul este copiat în una dintre variabilele inițiale *A, B, C* sau *D* (figura 8.2), iar la după parcurgerea tuturor blocurilor vom obține valoarea hash-ului stocat în variabile *ABCD* în formă hexazecimală.

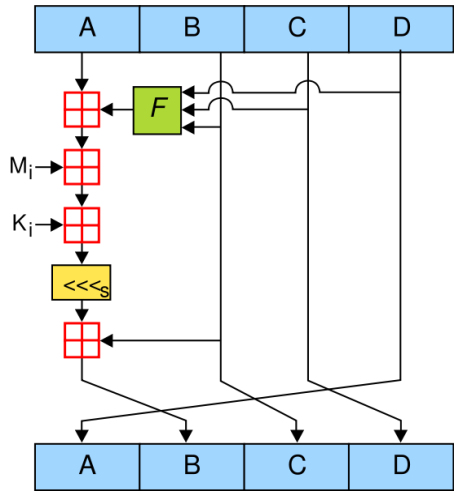

Figura 8.2. *Schema ciclului principal al MD5*

Exemplu. Fie *m*='*Funcţii HASH*'.

În acest caz *MD*5(*m*) =*MD*5(*Funcţii HASH*) = 8e31fb99ef4072b4d3d93014651eb13a, iar cei 128 de biți sunt reprezentați de cele 32 de cifre hexazecimale (32x4=128).

8.1.3. Familia SHA de funcții hash

SHA sau Secure Hash Algorithm este o familie de funcții hash criptografice publicate de Institutul Național de Standarde și Tehnologie (NIST) ca standarde federale de procesare a informației (FIPS), care include:

SHA-0: Un retronim aplicat versiunii originale a funcției hash pe 160 de biți publicată în 1993 sub numele „SHA". Aceasta a fost retrasă la scurt timp după publicare din cauza unui „defect semnificativ" nedezvăluit și înlocuită cu versiunea ușor revizuită SHA-1.

SHA-1: O funcție de hash pe 160 de biți, care seamănă cu algoritmul MD5. Acesta a fost proiectat de Agenția Națională de Securitate (NSA) să facă parte din DSA[1] (Digital Signature Algorithm). Pe parcurs au fost descoperite slăbiciuni criptografice ale *SHA*-1, iar standardul nu a mai fost aprobat pentru majoritatea utilizărilor criptografice după 2010.

SHA-2: O familie de două funcții hash similare, cu dimensiuni diferite de blocuri, cunoscute sub numele de SHA-256 și SHA-512, care diferă în dimensiunea variabilelor (cuvintelor); SHA-256 utilizează cuvinte pe 32 de biți, iar SHA-512 - pe 64 biți. Există, de asemenea și versiuni trunchiate ale fiecărui standard, cunoscute sub numele de SHA-224, SHA-384, SHA-512/224 și SHA-512/256, de asemenea proiectate de către ANS.

SHA-3: O funcție hash numită anterior *Keccak*, selectată în 2012 după o competiție publică între dezvoltatori non-ANS. Acesta susține aceleași lungimi de hash ca și SHA-2, iar structura sa internă diferă semnificativ de restul familiei SHA.

Standardele corespunzătoare acestei familii de funcții hash sunt: FIPS PUB 180 (SHA original), FIPS PUB 180-1 (SHA-1), FIPS PUB 180-2 (SHA-1, SHA-256, SHA-384, SHA-512). NIST are o versiune actualizată Draft FIPS 202, SHA-3, separată de standardul Secure Hash Standard (SHS).

8.1.4. Algoritmul SHA-2

SHA-2 este utilizat la momentul de față în calitate de standard securizat pentru funcțiile hash criptografice în SUA. El a fost aprobat în anul 2002, înlocuind vechiul algoritm SHA-1. Pe parcurs a suferit câteva redactări, fiind adăugate și funcții hash noi. Ultima redactare a fost aprobată în anul 2012.

[1] *DSA* (*Digital Signature Algorithm*) - Algoritmul pentru semnături digitale, este un standard al guvernului Statelor Unite ale Americii pentru semnăturile digitale. A fost propus de National Institute of Standards and Technology (NIST) în august 1991 pentru utilizare în cadrul standardului Digital Signature Standard (DSS), specificat în FIPS 186 și adoptat în 1993. O revizie minoră a fost emisă în 1996 sub numele de FIPS 186-1 [1]. Standardul a fost extins în 2000 ca FIPS 186-2 [2] și în 2009 ca FIPS 186-3 [3].

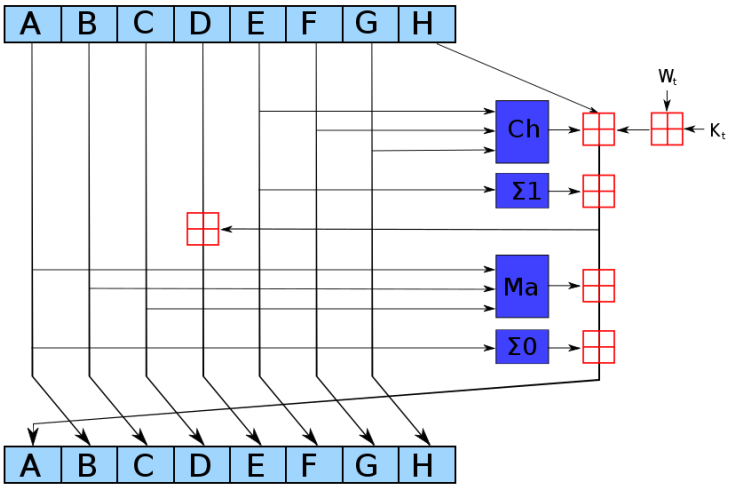

Figura 8.3. *Schema ciclului unei iterații SHA*-2

Funcțiile hash ale familiei SHA-2 sunt construite pe baza construcției Merkle-Damgard (la fel ca MD5 sau SHA-1), iar schema generală de funcționare este următoarea (figura 8.3):

- mesajul inițial se completează, apoi este împărțit în blocuri a câte 16 cuvinte;
- algoritmul parcurge fiecare bloc al mesajului completat printr-un ciclu cu 64 sau 80 de iterații (runde);
- la fiecare iterație sunt transformate 2 dintre cuvinte, restul participând la calcularea funcției de transformare;
- rezultatele procesării fiecărui bloc sunt sumate, rezultatul fiind valoarea funcției hash.

În procesul de calculare a hash-ului algoritmul aplică următorii operanzi pe biți: *concatenarea*, *adunarea*, *AND*, *OR*, *XOR*, *SHR* și *ROOTR*, unde *SHR* este deplasarea logică la dreapta, iar *ROOTR* – deplasarea ciclică la dreapta.

8.1.5. SHA-3 - prezentare generală

În anul 2004 criptograful Xiaoyun Wang a descoperit un defect la SHA-1, care a redus în mod drastic timpul necesar pentru a găsi o coliziune, făcând nesigure sistemele care utilizează acest algoritm. La acel moment NIST aprobase deja succesorul lui SHA-1, familia de algoritmi SHA-2, creând motive de îngrijorare că defectul s-ar putea extinde şi la acesta.

În anul 2007, după ce au început să apară temeri că algoritmii hash existenţi la acea vreme, SHA-1 și SHA-2, ar putea fi defectuoşi, NIST a dat startul unui concurs pentru algoritmul care va sta la baza noului standard SHA-3.

În timp ce cercetătorii de la NIST analizau cerinţele pentru competitorul SHA-3, aceştia au constatat că algoritmul SHA-2 nu a fost viciat în modul presupus mai sus. În loc să întrerupă proiectul de cercetare, NIST a decis că SHA-3 ar trebui să fie o opţiune complementară. Algoritmul ideal ar avea o structură diferită criptografic, făcând mai puţin probabil ca un atac asupra SHA-2 să

afecteze de asemenea SHA-3. Mai mult chiar, acesta ar fi mai potrivit pentru o gamă mare de dispozitive de calcul.

Dintr-un total de 64 de înregistrări care au avut loc inițial, NIST a identificat cinci finaliști în 2010. Câștigătorul, algoritmul Keccak, se bazează pe o „construcție hash de tip burete" (*sponge construction*), numită astfel deoarece funcția sa este flexibilă, asemeni unui burete fizic. Datele de intrare într-o funcție hash sunt „absorbite" în faza de hashing și apoi „stoarse" pentru a produce hash. Într-o construcție de tip burete, stoarcerea mai rapidă produce un hash mai rapid, dar mai puțin sigur și vice-versa, oferind flexibilitate.

Algoritmul nou este apreciat de unii specialiști ca fiind în general mai rapid decât predecesorul său și prin urmare consumă mai puțină energie, pe când alți experți în securitate sunt mai puțin siguri de utilitatea unui nou algoritm. Având în vedere faptul că nu există nici o vulnerabilitate evidentă a SHA-2, sistemele pot decide să utilizeze în continuare acest algoritm, mai degrabă decât să îl schimbe. SHA-2 este încă un algoritm bun, dar poate SHA-3 ar fi mai eficient în unele situații. NIST nu încurajează pe nimeni să abandoneze SHA-2 în favoarea SHA-3, deoarece cei doi algoritmi sunt foarte buni, iar SHA-3 nu este mai bun decât SHA-2 pe toate planurile.

8.1.5. Aplicații ale funcțiilor hash

Funcțiile hash criptografice sunt în prezent pe larg utilizate ca identificator al fișierelor, pentru verificarea parolelor, a integrității fișierelor sau a mesajelor, pentru, realizarea protocoalelor proof-of-work, etc.

Verificarea integrității este una dintre aplicațiile practice importante ale funcțiilor hash, care poate fi realiză prin compararea hash-ului mesajului (sau fișierului) calculat înainte și după transmitere (sau după orice alt eveniment). Din acest motiv, majoritatea algoritmilor de semnătură digitală doar confirmă autenticitatea hash-ului mesajului care urmează să fie „semnat", verificarea autenticității acestui hash fiind și dovada autenticității mesajului.

Hash-urile făcute cu MD5, SHA1 sau SHA2 sunt uneori afișate împreună cu fișierele respective site-uri sau forumuri pentru a permite verificarea integrității. Această practică stabilește un lanț de încredere atâta timp cât hash-urile sunt afișate pe un site autentificat de HTTPS.

Verificarea parolei este o aplicație înrudită cu verificarea integrității. Stocarea tuturor parolelor utilizatorilor sub formă de text clar poate duce la o încălcare majoră a securității în cazul în care fișierul cu parole este compromis. O modalitate de a reduce acest pericol este de a stoca doar hash-ul fiecărei parole. Pentru a autentifica un utilizator, hash-ul parolei prezentată de utilizator este comparat cu hash-ul stocat (această abordare însă împiedică recuperarea parolelor originale dacă sunt uitate sau pierdute și trebuie înlocuite cu altele noi).

Înainte de aplicarea funcției hash parola este adesea concatenată cu o valoare aleatorie, non-secretă, numită *sare* (salt), care este stocată împreună cu hash-ul. Sarea este, de fapt, o secvență de

caractere pe care o adăugăm parolei, astfel încât hash-ul să fie diferit. Astfel, chiar dacă doi utilizatori ar alege aceeași parolă (fie ea și simplă), hash-urile lor ar fi diferite.

Aceste hash-uri sunt însă calculate foarte rapid și ar fi posibil un atac asupra unor parole individuale. Pentru a împiedica acest lucru sunt utilizate funcții hash mai lente, cum sunt de exemplu *Bcrypt, Scrypt* sau *PBKDF2*.

În 2013, a fost anunțat un concurs pentru a alege un nou algoritm standard pentru hash-ul parolelor, iar câștigătorul, selectat în iulie 2015, a fost un nou algoritm de extindere a cheii, *argon2*. În iunie 2017, NIST a emis o nouă revizuire a regulilor privind autentificarea digitală, NIST SP 800-63B-3, în care se spune că parolele trebuiesc memorate într-o formă rezistentă la atacurile offline, cu adăugare de sare și cu aplicarea unei funcții hash potrivite.

Protocoale proof-of-work (ce ar putea fi tradus ca dovada muncii) reprezintă o măsură economică pentru a descuraja atacurile de tip „denial-of-service" și alte abuzuri legate de servicii, cum ar fi spamul într-o rețea, care necesită o anumită activitate de la solicitantul serviciului, care se exprimă de obicei prin consumarea timpului de procesare a uni calculator. O caracteristică cheie a acestor scheme este asimetria acestora: activitatea trebuie să fie moderat dificilă (dar fezabilă) pentru solicitant, și în același timp ușor de verificat pentru furnizorul de servicii. Proof-of-work se folosește în rețeaua Bitcoin pentru a securiza diverse aspecte prin a face (artificial) costisitoare validarea tranzacțiilor pentru utilizatorii din rețea utilizând funcțiile hash.

Identificarea fișierelor de asemenea poate fi realizată cu ajutorul funcțiilor hash într-un mod fiabil. Mai multe sisteme de gestionare a codurilor sursă, inclusiv Git, Mercurial și Monotone, utilizează hash-ul SHA-1 a diferitor tipuri de conținut (conținutul fișierelor, ierarhia de directoare, informațiile despre origine etc.) pentru a le identifica în mod unic. Hash-urile sunt folosite pentru a identifica fișiere pe rețelele de partajare a fișierelor peer-to-peer. De exemplu, într-o conexiune *ed2k*, hash-ul generat de algortimul MD-4 este combinat cu dimensiunea fișierului, furnizând suficiente informații pentru localizarea surselor fișierelor, descărcarea fișierului și verificarea conținutului său.

Una dintre principalele aplicații ale unei funcții hash obișnuite este de a permite căutarea rapidă a datelor în tabelele de dispersie[2]. Funcțiile hash criptografice pot realiza foarte bine acest lucru.

Cu toate acestea, în comparație cu funcțiile hash standard, funcțiile hash criptografice tind să fie mult mai scumpe din punct de vedere computațional. Din acest motiv ele tind să fie folosite în contexte în care este necesar ca utilizatorii să se protejeze împotriva posibilității de falsificare

[2] Un *tabel de dispersie* sau *tabel hash* este o structură de date care implementează interfața unui tablou asociativ, și anume: permite stocarea perechi (cheie, valoare) și efectuarea a trei operații: adăugarea unei perechi noi, căutarea și ștergerea perechii după cheia cunoscută.

(crearea de date cu același hash ca și datele așteptate) de către participanții potențialii rău intenționați.

Funcțiile hash pot fi de asemenea utilizate pentru generarea de biți pseudoaletori sau pentru a obține chei noi sau parole dintr-o singură cheie sau parolă securizată.

8.2. Semnături digitale

Sistemele de criptare cu cheie secretă sau publică ne asigură nivelul cerut de confidențialitate a informației, verificarea integrității poate fi realizată cu utilizarea funcțiilor hash, în timp ce autenticitatea poate fi asigurată cu o primitivă criptografică, numită semnătură digitală.

8.2.1. Noțiuni generale și proprietăți ale semnăturii digitale

Semnătura digitală (sau electronică) este o schemă matematică pentru demonstrarea autenticităţii unui mesaj sau document electronic. O semnătură digitală validă este un motiv de încredere pentru destinatarul mesajului că acest mesaj a fost creat de către un expeditor cunoscut, astfel încât el nu va putea nega faptul trimiterii mesajului (*autentificarea* şi *non-repudierea*) și că mesajul nu a fost modificat în drum (*integritatea*). Semnăturile digitale sunt în general utilizate pentru distribuţia de software, în tranzacţiile financiare, precum și în alte cazuri în care este importantă detectarea eventualei falsificări sau manipulări.

Semnătura electronica nu este o semnătură scanată, o pictogramă, o poză sau o hologramă şi nici nu este un smart-card.

O schemă de semnătură digitală se bazează pe trei algoritmi:

- algoritmul de *selectare aleatoare a unei chei private* care se va asocia unei chei publice;
- algoritmul de *semnare* care, aplicat unei chei private şi unui document digital, generează semnătura digitală;
- algoritmul de *verificare* a semnăturii digitale, care aplicat cheii publice şi semnăturii digitale, acceptă sau respinge mesajul de conformitate.

Semnăturile digitale reprezintă echivalentul electronic al semnăturilor de mână, acest concept fiind introdus ca funcţionalitate adiţională a sistemelor de criptare cu cheie publică de către Diffie şi Hellman în 1976, în absenţa unei scheme criptografice pentru acest scop. Obiectivul principal de securitate pe care îl asigură semnăturile digitale îl reprezintă *non-repudierea*, şi anume faptul că o entitate odată ce a semnat o informaţie nu poate nega că a emis acea informaţie şi orice altă entitate neutră poate verifica acest lucru. Semnăturile digitale reprezintă deci o valoare numerică care leagă conţinutul unui mesaj de identitatea unei entităţi. În principiu, orice algoritm asimetric poate fi utilizat pentru crearea unei semnături digitale prin *inversarea rolului cheii publice cu cea privată*, iar primele propuneri de semnături digitale se găsesc în lucrările lui Rivest, Rabin şi

ElGamal. Trebuie de subliniat că şi algoritmi simetrici pot fi folosiţi pentru a crea semnături digitale de tip *one-time* dar acestea sunt rar utilizate în practică.

Pentru o semnătură digitală sunt necesare două proprietăţi-cerințe de bază:

- în primul rând, o semnătură generată dintr-un mesaj fix și o cheie fixă privată ar trebui să verifice autenticitatea acestui mesaj utilizând cheia publică corespunzătoare;
- în al doilea rând, ar trebui să fie imposibil de generat o semnătură validă pentru o entitate care nu posedă cheia privată.

Alte câteva proprietăţi ale semnăturilor digitale sunt:

- trebuie să fie uşor de calculat doar de către cel care semnează mesajul (funcţia de semnare trebuie să fie uşor de calculat);
- trebuie să fie uşor de verificat de către oricine (funcţia de verificare trebuie să fie uşor de calculat);
- trebuie să deţină o durată de viaţă corespunzătoare, adică semnătura să nu poată fi falsificată până când nu mai este necesară scopului în care a fost creată.

Să precizăm încă o dată cele trei motive principale pentru care se recomanda folosirea semnăturilor digitale.

*Autenticitatea***.** Deşi documentele digitale pot să includă informaţii despre identitatea celui care le-a emis, aceste informaţii pot să nu fie corecte. Semnătura digitală poate fi folosită pentru autentificarea sursei documentului. Atunci când dreptul de proprietate asupra unei semnături digitale aparţine unei anumite persoane, semnătura digitală arată că documentul a fost eliberat de către acea persoană. Importanţa acestui aspect apare în dovedirea autenticităţii documentelor digitale în context financiar-contabil. De exemplu actele contabile ale unei firme sunt trimise corect de către firma de contabilitate către administratorul firmei. Daca acesta va modifica aceste acte încercând să schimbe informaţiile financiare sau orice fel de informaţii transmise ca fişier PDF/A[3] semnat electronic, pentru a putea obţine un credit de la o bancă, în momentul în care banca deschide acele documente, semnătura electronică va fi invalidată, deci firma de contabilitate nu va putea fi considerată responsabilă de conţinutul documentului, persoana care a transmis documentul modificat nu va putea fi urmărită în justiţie pentru fals şi uz de fals, iar banca va fi asigurată că nu cade în capcana acordării unui credit pe baza unor documente false.

*Integritatea***.** Semnătura digitală aplicată unui document electronic reprezintă o garanţie a integrităţii documentului atunci când este validată de o autoritate publică. Cheia aleatoare se creează

[3] *PDF/A* este o versiune standardizată ISO a documentului portabil (PDF) specializat pentru utilizarea în arhivarea și păstrarea pe termen lung a documentelor electronice. PDF/A diferă de PDF prin interzicerea unor caracteristici necorespunzătoare pentru arhivarea pe termen lung, cum ar fi legarea fontului (spre deosebire de încorporarea fonturilor) și criptarea.

pe baza unor criterii multiple care includ printre altele şi detalii despre conţinutul documentului. Orice modificare a conţinutului unui document digital înseamnă o cheie aleatoare nouă diferită de cheia aleatoare folosită pentru aplicarea semnăturii digitale. În clipa în care se solicită validarea documentului cele două chei aleatoare vor fi diferite, iar documentul se va putea considera ca având conţinutul alterat.

Imposibilitatea repudierii (*non-repudiere*). Odată ce este emis un document digital semnat electronic, atâta vreme cât semnătura electronică este validă pe documentul digital, autorul documentului nu îşi poate declina răspunderea pentru conţinutul documentului cu semnătura electronică validă. În plus, nu poate nega faptul că documentul a fost semnat personal, deoarece legislaţia în vigoare prevede faptul că deţinătorul unei semnături digitale nu are dreptul să înstrăineze sau împrumute *token*-ul (dispozitivul criptografic) pe care există certificatul digital calificat pe care îl deţine. Din acest punct de vedere, exemplul de mai sus în care firma de contabilitate transmite documentele financiare firmei pentru care le întocmeşte, dacă a comis greşeli în întocmirea acestor acte, rămâne responsabilă pentru datele prezentate, câta vreme documentul digital transmis poartă o semnătură validă.

Pentru o semnătură digitală a mesajului m de către entitatea A putem folosi notaţia S, $S_A(m)$ sau $Sig_A(m)$.

Există două categorii distincte de semnături digitale: *semnături digitale cu recuperarea mesajului* și *semnături digitale cu anexă*

La utilizarea semnăturilor digitale din prima categorie mesajul poate fi recuperat direct din semnătura digitală. Cel mai simplu exemplu de construcţie este prin inversarea rolului cheii publice şi celei private în cazul schemei RSA. Pentru a evita fraudarea lor este obligatorie aplicarea unei funcţii de redundanţă[4] asupra mesajului după care semnătura propriu-zisă se aplică asupra mesajului redundant.

Semnături digitale cu anexă (sau cu apendice, figura 8.4) sunt semnături digitale din care mesajul nu poate fi recuperat, și este trimis adiţional ca anexă la semnătura digitală. Acestea se pot construi uşor prin aplicarea unei funcţii hash asupra mesajului şi criptarea hash-ului obţinut. Datorită eficienţei computaţionale în semnarea mesajelor de dimensiuni mari (deoarece se semnează efectiv doar hash-ul mesajului care are o lungime fixă pentru fiecare algoritm hash indiferent de lungimea mesajului), aceste semnături sunt cele mai utilizate în practică. Totodată orice semnătură digitală cu recuperarea mesajului poate fi uşor convertită în semnătură cu anexă.

[4] În informatică, *redundanța* reprezintă un excedent de semnale pentru transmiterea fidelă a unei cantități de informație.

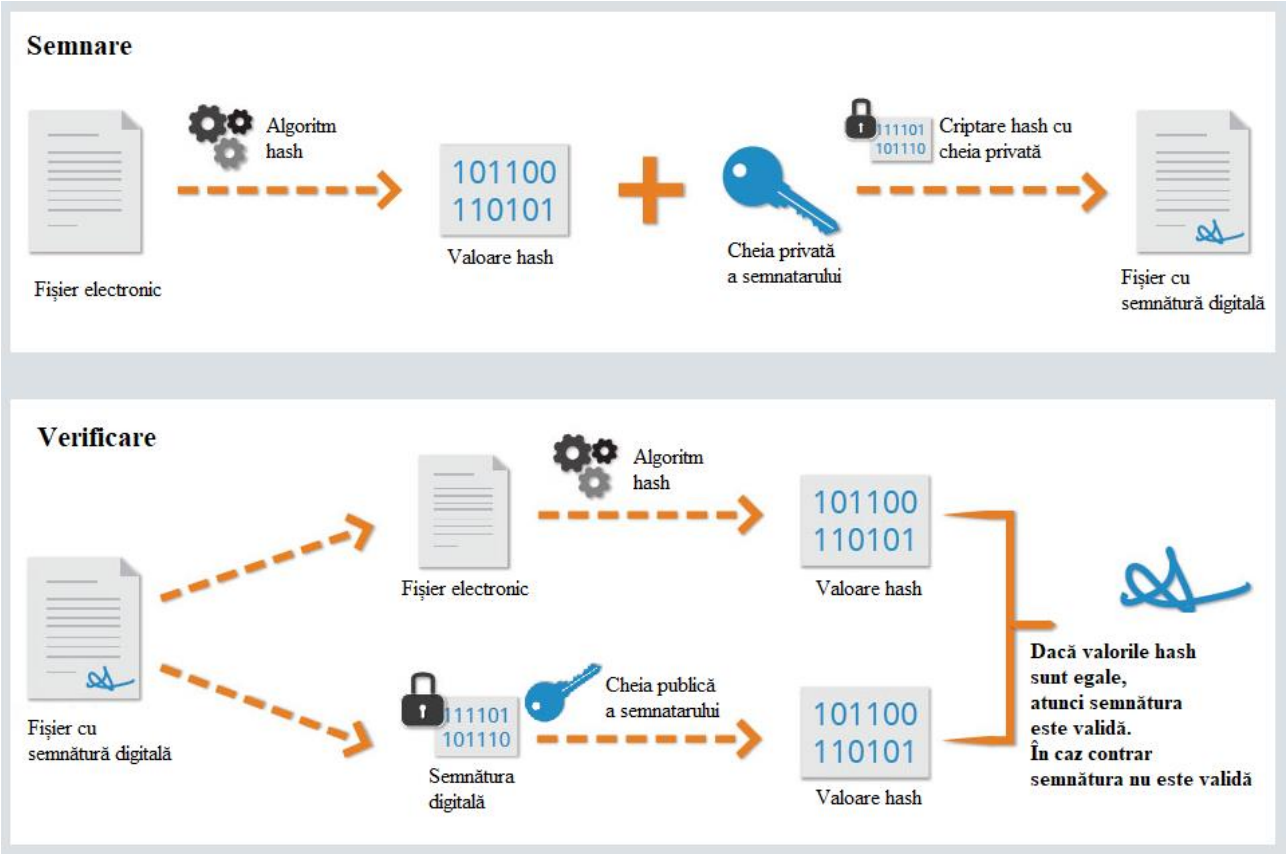

Figura 8.4. *Schema unei semnături digitale fără criptarea mesajului*

În cazul în care este importantă și confidențialitatea documentului electronic semnat, atunci schema pentru semnătura cu apendice se completează, după aplicarea algoritmului hash, cu criptarea fișierului electronic prin utilizarea de obicei a unui algoritm simetric, pentru care ambii participanți dispun de o cheie comună. La verificare destinatarul va decripta mai întâi fișierul cu același algoritm de criptare simetrică, apoi va urma aceiași pași ca și în versiunea precedentă.

Semnăturile digitale de asemenea pot fi clasificate în *semnături deterministe* respectiv *randomizate* după cum algoritmul de semnare foloseşte valori aleatoare şi nu returnează aceeaşi semnătură pentru acelaşi mesaj de fiecare dată.

O altă clasificare a algoritmilor de semnătură mai poate fi în *algoritmi de unică folosinţă* (one-time) sau pentru *folosire multiplă* (multiple-time).

Să analizăm în continuare câteva dintre schemele de semnătură digitală aplicate în prezent.

8.2.2. Schema de semnătură RSA

Algoritmul de generare a semnăturii digitale RSA este același ca și algoritmul obișnuit de criptare RSA, cu unele particularități. Fie Bob – posesorul perechii de chei publice PK_B și private SK_B pentru RSA.

a) pentru a utiliza aceste chei la *criptare* se procedează astfel:

- oricine dorește folosește PK_B pentru a cripta și a-i transmite lui Bob informația criptată;

- doar Bob, adică posesorul cheii privare, poate decripta informația primită utilizând SK_B.

b) pentru *semnătura digitală* procedeul e următorul:

- pentru a semna un document, Bob îi calculează mai întâi valoarea hash, apoi criptează această valoare cu cheia lui privată SK_B, rezultând semnătura digitală a acestui document; apoi Bob transmite destinatarului documentul cu semnătura obținută;
- pentru a verifica semnătura ea trebuie decriptată cu cheia publică a lui Bob PK_B, obținând valoarea hash a documentului.

Schema semnăturii digitale pentru o entitate *A* constă din următoarele etape:

1. *Generarea cheilor*. Această etapă nu se deosebește de etapa similară a algoritmului de criptare RSA. În rezultatul ei se obține perechea de chei a entităţii *A*: cheia publică este perechea (*e*, *n*) și cheia privată este *d* sau (*d*, *n*).
2. *Generarea semnăturii*. Entitatea *A* execută următoarele:
   - calculează $S = [H(m)]^d \bmod n$, unde *H* este o funcţie hash;
   - semnătura mesajului *m* este *S*.
3. *Verificarea semnăturii*. Pentru a verifica semnătura *S* a mesajului *m*, entitatea *B* execută următoarele:
   - obţine cheia publică autentică (*e*, *n*) a entităţii *A*;
   - calculează $H_1 = S^e \bmod n$ şi $H_2 = H(m) \bmod n$, utilizând aceeași funcție hash; în cazul în care $H_1 = H_2$ semnătura e verificată, în caz contrar înseamnă că s-a întâmplat ceva în canalul de comunicaţii sau *A* vrea să-l înşele pe *B*.

Exemplu. De generat cheile și de semnat mesajul *m* aplicând schema RSA, dacă se cunoaște valoarea zecimală a funcției hash: $H(m)=130$.

1. *Generarea cheilor*:
   - fie $p = 53$ şi $q = 61$, două numere prime secrete ale lui *A*;
   - se calculează $n = 53 \cdot 61 = 3233$ şi $\varphi(n) = (p-1) \cdot (q-1) = 52 \cdot 60 = 3120$;
   - se alege o cheie secreta $d = 71$ ($cmmdc(71, 3120) = 1$);
   - se calculează cheia publică $e = 791$ ($71 \cdot d = 1 \bmod 3120$);
2. *Generarea semnăturii*:
   - deoarece se cunoaște $H(m)=130$, putem semna documentul:
     $S = [H(m)]^d = (130^{71}) \bmod 3233 = 1883$;
   - semnătura electronică obţinută este $S = 1883$;
   - Se transmite *S* obţinut la pasul anterior şi *m* (textul clar – fără secretizare).

3. *Verificarea semnăturii*: *B* primeşte pachetul *S* şi *m*, după care calculează *H*(*m*) în două moduri şi le va compara:
    - va calcula *H*(*m*) cu cheia publică a lui *A*:
      $H_1 = S^e = (1883^{71}) \bmod 3233 = 130$;
    - va calcula *H*(*m*) prin aplicarea aceleași funcții hash ca și *A* și va obține
      $H_2 = H(m) = 130$;
    - $H_1 = H_2$, deci semnătura este validă.

Aici mai trebuie de menționat că nu vom avea situații precum la criptarea RSA, care să ne impună să divizăm valoarea hash în segmente cu valori mai mici ca *n*, deoarece în conformitate cu standardele în vigoare valoarea lui *n* este mai mare decât valoarea funcției hash.

8.2.3. Schema de semnătură ElGamal

Această schemă a fost propusă de ElGamal împreună cu schema de criptare cu chei publice, diferențele aici sunt puțin mai mari.

1. *Generarea cheilor*. Fiecare entitate generează cheia publică şi cheia privată corespunzătoare. Entitatea *A* execută următoarele:
    - generează un număr prim mare *p* şi un generator α al grupului multiplicativ $\boldsymbol{Z_p}^*$;
    - selectează aleator un număr întreg *a* astfel încât $1 \le a \le p-2$;
    - calculează $y = \alpha^a \bmod p$.
    - cheia publică a entităţii *A* este (p, α, y); cheia privată este *a*.
2. *Generarea semnăturii*. Entitatea *A* semnează un mesaj *m* de o lungime arbitrară.
   Pentru aceasta, entitatea *A* execută următoarele:
    1. selectează aleator un număr întreg secret *k*, $1 \le k \le p-2$, astfel încât $cmmdc(k, p-1) = 1$;
    2. calculează $r = \alpha^k \bmod p$ şi $k^{-1} \bmod (p-1)$;
    3. calculează $s = k^{-1}(H(m) - a \cdot r) \bmod (p-1)$, unde *H* este o funcţie hash;
    4. semnătura mesajului *m* este perechea (r, s).
3. *Verificarea semnăturii.* Pentru a verifica semnătura (r, s) a mesajului *m*, entitatea *B* execută următoarele:
    1. obţine cheia publică autentică (p, α, y) a entităţii *A* ;
    2. verifică dacă $1 \le r \le p-1$ (dacă această inegalitate nu are loc, semnătura (r, s) nu e validă);
    3. calculează $v_1 = y^r r^s \bmod p$;
    4. calculează $H(m)$ şi $v_2 = \alpha^{H(m)} \bmod p$, unde *H* este funcţia hash aplicată la semnare;
    5. semnătura (r, s) este acceptată dacă şi numai dacă $v_1 = v_2$.

Exemplu. De generat cheile și de semnat mesajul *m* aplicând schema *ElGamal*, dacă se cunoaște valoarea zecimală a funcției hash: $H(m)$ =1463.

1. *Generarea cheilor*:
   - *A* generează numărul prim p=2357 şi generatorul α=2 al grupului $\boldsymbol{Z}^*_{2357}$;
   - *A* alege cheia privată a = 1751 şi calculează
     $y = \alpha^a \bmod p = 21751 \bmod 2357 = 1185$.
   - cheia publică a lui *A* este (p = 2357, α = 2, y = 1185),
     iar cheia sa privată este a = 1751.
2. *Generarea semnăturii:*
   - pentru a semna mesajul *m* cu $H(m)$ =1463, *A* selectează aleator un întreg k =1529, calculează
     $r = \alpha^k \bmod p = 2^{1529} \bmod 2357 = 1490$ şi
     $k^{-1} \bmod (p-1) = 245$;
   - *apoi A* calculează
     S=245·(1463–1751·1490) mod 2356=1777.

   Semnătura mesajului m = 1463 este perechea (r =1490, s =1777);
3. *Verificarea semnăturii*:
   - pentru verificarea semnăturii, *B* calculează
     $v_1 = 1185^{1490} \cdot 1490^{1777} \bmod 2357 = 1072$,
     $H(m)$ =1463 (utilizând aceeași funcție hash ca și *A*),
     $v_2 = \alpha^{H(m)} \bmod p = 2^{1463} \bmod 2357 = 1072$;
   - Entitatea *B* acceptă semnătura deoarece $v_1 = v_2$.

8.2.4. Certificate digitale

Aplicarea semnăturii digitale în viața reală nu se face prin efectuarea acestor calcule de către cel care semnează sau verifică, calculele fiind făcute automat prin intermediul instrumentelor software sau hardware respective, iar semnătura digitală este asigurată de un certificat digital.

În criptografie, un *certificat digital*, cunoscut și sub denumirea de certificat al cheii publice sau certificat de identitate, este un document electronic folosit pentru a demonstra dreptul de proprietate asupra unei chei publice. Certificatul include informații despre cheie, informații despre identitatea proprietarului său (numit subiect sau deţinător) și semnătura digitală a unei entități care a verificat conținutul certificatului (numit emitent). Dacă semnătura este validă și software-ul care examinează certificatul are încredere în emitent, atunci poate utiliza acea cheie pentru a comunica în siguranță cu deţinătorul certificatului.

În criptarea pe e-mail, la semnarea codului sau în sistemele de semnături electronice, deţinătorul unui certificat este de obicei o persoană sau o organizație. Cu toate acestea în TLS[5] subiect al certificatului este de obicei un computer sau un alt dispozitiv, deși certificatele TLS pot identifica organizații sau persoane fizice în plus față de rolul principal în identificarea dispozitivelor.

Similar conceptului de carte de identitate sau paşaport, unde dreptul de a emite un astfel de document revine unei singure autorităţi la nivel naţional în baza unor documente prezentate de solicitant, un certificat digital este emis de anumite entităţi denumite autorităţi de certificare. O astfel de autoritate, pentru a fi acreditată, trebuie să îndeplinească reguli stricte, astfel încât un certificat emis de ea să reprezinte un grad ridicat de încredere.

Certificatele digitale sunt emise conform standardului ITU X.509 versiunea 3 descris prin RFC 2459, „Internet X.509 Public Key Infrastructure Certificate and CRL Profile". Un certificat digital conţine în principiu următoarele informaţii (figura 8.5):

- versiunea, seria, identificatorul certificatului;
- numele şi adresa de email a deţinătorului (persoană fizică sau juridică);
- adresa URL a server-ului care utilizează certificatul respectiv;
- cheia publică a deţinătorului și algoritmul cheii publice;
- numele autorităţii de certificare ce a emis certificatul digital;
- seria certificatului digital;
- valabilitatea certificatului digital (data de început şi data de sfârşit);
- unele informații opționale: identificatorul unic al emitentului, identificatorul unic al subiectului, extensii.

[5] *Transport Layer Security* (TLS) și Secure Sockets Layer (SSL), predecesorul său, sunt protocoale criptografice care permit comunicații sigure pe Internet. Între SSL 3.0 și TLS 1.0 există anumite diferențe, dar protocolul rămâne aproximativ același. Termenul „SSL" folosit aici se poate referi la ambele protocoale, excepție făcând cazurile specificate explicit în context. Utilizarea tehnologiei SSL oferă un grad mai mare de confidențialitate și de siguranță decât o conexiune web criptată. Aceasta reduce riscul ca informațiile să fie interceptate și utilizate incorect.

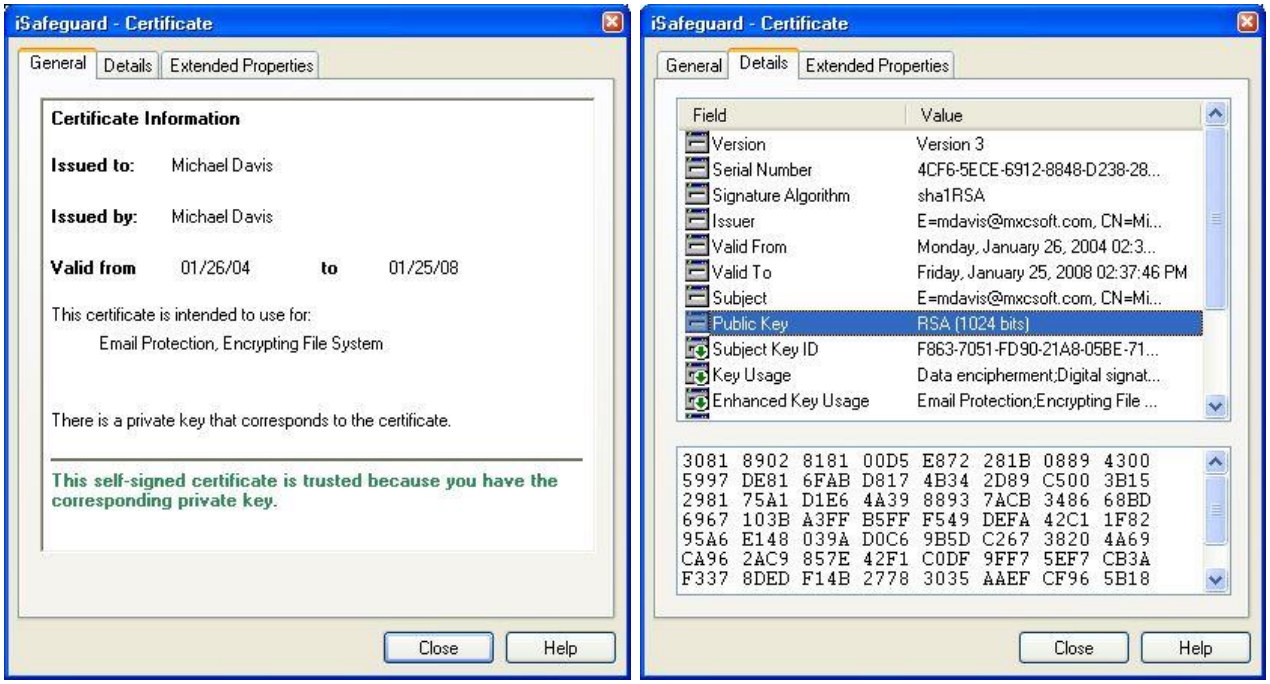

Figura 8.5. *Informații generale și detaliate ale uni certificat digital*

Certificatele digitale sunt în general utilizate pentru:

- autentificarea şi criptarea informaţiilor între servere şi browsere web;
- autentificarea şi criptarea conexiunilor într-o reţea locală – LAN;
- autentificarea şi criptarea mesajelor trimise prin email în cadrul unei reţele locale sau între terţi;
- autentificarea şi criptarea conexiunilor între servere;

Certificatele digitale pot fi de mai multe tipuri:

- *de uz intern* – sunt utilizate în cadrul reţelelor locale aparţinând unor companii şi nu pot fi utilizate în relaţii cu terţii deoarece nu sunt emise de autorităţi de certificare acreditate;
- *certificatele cu autentificare pe domeniu* – la emiterea unui certificat Autoritatea de Certificare verifică numai faptul că cel ce a depus cererea de emitere deţine domeniul respectiv;
- *certificate cu autentificare totală* – sunt acordate de o autoritate de certificare numai după verificări amănunţite cu privire la existenţa afacerii respective şi a deţinătorului domeniului respectiv;
- *certificate pentru semnarea aplicaţiilor software* – sunt utilizate de companiile producătoare de software la semnarea digitală a codurilor software pentru a preveni compromiterea acestora cu aplicaţii malware atunci când sunt downloadate de pe Internet;

- *certificate cu validare extinsă* – reprezintă certificatele digitale cu cel mai înalt nivel de încredere posibil.

Cel mai comun format pentru certificatele de chei publice este definit de X.509. Deoarece X.509 este foarte general, formatul este constrâns în continuare de profiluri definite pentru anumite cazuri de utilizare, cum ar fi Infrastructura cheilor publice (X.509) așa cum este definită în RFC 5280.

În Republica Moldova certificarea cheilor publice este realizată de *centrele de certificare*. În conformitate cu legislația în vigoare centrele de certificare se creează după principiul ierarhic, după cum urmează:

- centrul de certificare de nivel superior - primul nivel;
- centrele de certificare care prestează servicii de certificare a cheilor publice terţelor persoane - nivelul al doilea;
- centrele de certificare create în scopuri corporative, care nu prestează servicii de certificare a cheilor publice terţelor persoane - nivelul al treilea.

Numărul centrelor de certificare de nivelul doi şi trei nu se limitează.

Centrul de certificare a cheilor publice de nivel superior a fost creat în cadrul Serviciului de Informaţii şi Securitate al Republicii Moldova în conformitate cu Legea nr. 246 din 15.07.2004 cu privire la documentul electronic şi semnătura digitală cu respectarea prevederilor Directivei Uniunii Europene UE/EC/99/93 din 29 iunie 2006. El este pilonul de bază a infrastructurii cheilor publice în Republica Moldova.

Având ca destinație principală certificarea cheilor publice ale prestatorului de servicii de certificare în domeniul aplicării semnăturii electronice avansate calificate, Centrul de certificare a cheilor publice de nivel superior îndeplinește următoarele funcţii:

- certifică cheile publice ale prestatorilor de servicii de certificare în domeniul aplicării semnăturii electronice avansate calificate;
- suspendă valabilitatea, repune în funcţiune şi revocă certificatele cheilor publice emise de acest prestator;
- creează şi ţine registrul certificatelor cheilor publice ale prestatorului de servicii de certificare în domeniul aplicării semnăturii electronice avansate calificate de nivelul al doilea;
- confirmă autenticitatea şi valabilitatea certificatelor cheilor publice ale prestatorilor de servicii de certificare în domeniul aplicării semnăturii electronice avansate calificate.

Pornind de la funcţiile sale, Centrul de certificare a cheilor publice de nivel superior nu oferă servicii în domeniul semnăturii electronice beneficiarilor finali. În acest sens, Centrul este

garant de legalitate şi de credibilitate în raporturile dintre centrele de certificare de nivelul doi şi clienţii nemijlociţi ai acestora (persoane fizice şi juridice).

Centrele de certificare a cheilor publice de nivel doi sunt *Centrul de Telecomunicaţii Speciale* (prin intermediul căruia se realizează și semnătura mobilă Moldcell sau Orange) și *Centrul de Certificare al Întreprinderii de Stat* "*Fiscservinform*".

Întrebări și subiecte pentru aprofundarea cunoștințelor și lucrul individual

1. Care este deosebirea fundamentală dintre funcțiile hash obișnuite de cele criptografice?
2. Enunțați și comentați trei aplicații ale funcțiilor hash criptografice.
3. Explicați principiul general de funcționare a semnăturii digitale.
4. Enunțați și comentați trei principii ale securității informației care sunt susținute de semnătura digitală.
5. Modelați semnarea mesajului *M*='*Semnătură digitală*' aplicând algoritmii RSA și ElGamal. Pentru funcția hash utilizați algoritmul CRC32.
6. Care sunt beneficiile utilizării certificatelor digitale și cum se realizează verificarea acestor certificate?