Capitolul 5

Protocoale de gestiune a cheilor

5.1 Proprietăţi generale

Am văzut că sistemele bazate pe chei publice nu necesită un canal sigur pentru transmiterea unei chei private. Aceste avantaj este redus însă de faptul că un canal cu cheie publică este mult mai lent decât unul cu cheie simetrică, el necesitând protocoale de transmitere, scheme de autentificare etc.

În acest capitol vom aborda această problemă, de stabilire a unor modalităţi de a schimba mesaje între mai mulţi utilizatori, folosind protocoale rapide, fiecare utilizator având propria sa cheie privată.

Protocoalele de gestiune a cheilor acoperă diverse variante. Astfel există

1. Protocoalele de **distribuire a cheilor**: o autoritate generează o cheie K pe care o transmite mai departe altor entităţi (utilizatori).

2. Protocoalele de **punere de acord** (key agreement): toţi participanţii stabilesc o cheie K folosind scheme de partajare a secretelor peste un canal public.

Protocoalele de punere de acord pot fi:

- **implicite**: *Alice* este asigurată că nimeni înafară de *Bob* nu poate determina cheia K. De remarcat că acest lucru nu înseamnă că *Alice* este sigură că *Bob* are efectiv cheia K. Un astfel de protocol se numeşte *punere de acord autentificată* (AK).

- **explicite**: *Alice* este sigură că *Bob* a calculat efectiv cheia K.
  De obicei, aceste protocoale se obţin prin completarea punerilor de acord implicite cu protocoale de confirmare a cheii (de către *Bob*). Astfel de protocoale sunt numite AKC (protocoale de *punere de acord explicită cu confirmare*).

Baza de comunicare este o reţea (nesigură) care leagă n utilizatori. În general se presupune că există o autoritate T numită *arbitru*, considerată apriori ca fiind onestă; ea

verifică identităţile utilizatorilor, alegerea şi transmiterea cheilor iniţiale, precum şi alte probleme curente.

Deoarece reţeaua nu este sigură, participanţii trebuie să prevadă unele atacuri din partea lui *Oscar*. Acesta poate fi pasiv (doar interceptează mesaje) sau activ.

Un adversar activ este capabil:

- Să modifice un mesaj în cursul transmisiei;
- Să transmită mesaje vechi;
- Să încerce să se prezinte drept unul din utilizatorii reţelei.

Obiectivul lui *Oscar* este:

- Să îi facă pe *Alice* şi *Bob* să accepte o cheie *invalidă*;
- Să îi facă pe *Alice* şi *Bob* să creadă că au schimbat chei între ei.

Scopul unui protocol de *distribuţie a cheilor* sau de *punere de acord* este ca în final cei doi participanţi să deţină aceeaşi cheie K, necunoscută de ceilalţi utilizatori (exceptând eventual T).

Alte proprietăţi de securitate care pot fi avute în vedere (considerând că *Alice* şi *Bob* sunt oneşti):

- **Securitatea cheii**: Fiecare execuţie a protocolului de generare a cheii trebuie să genereze o cheie secretă unică – numită *cheie de sesiune*. Este important ca aceste chei să ofere lui *Oscar* doar o cantitate limitată de informaţii şi să limiteze cât mai mult posibil – în caz de compromitere – expunerea altor chei.
  Protocolul trebuie să realizeze acest deziderat chiar dacă *Oscar* a acumulat informaţii deduse din cheile de sesiune generate anterior.
- **Secretul cheilor ulterioare**: Compromiterea cheii secrete a unuia sau mai multor utilizatori să nu afecteze secretul cheilor de sesiune anterioare stabilite de utilizatorii oneşti. Adesea se face distincţie între situaţia când numai cheia secretă a lui *Alice* este compromisă, şi cea când cheile lui *Alice* şi *Bob* sunt compromise simultan.
- **Impersonalizarea cheii compromise**: Dacă *Oscar* a descoperit cheia secretă a lui *Alice*, el îi poate lua locul în protocoalele de gestiune a cheii (o "impersonalizează" pe *Alice*). Acest lucru nu trebuie însă să-l impersonalizeze pe *Bob* faţă de *Alice*.
- **Anonimitatea componentelor cheii**: *Bob* nu poate fi obligat să împartă o cheie cu o entitate pe care nu o cunoaşte; de exemplu, *Bob* poate crede (eronat) că împarte o cheie cu *Charlie* în timp ce alt utilizator – *Alice* – crede (corect) că cheia este partajată cu *Bob*.

Alte performanţe care pot fi urmărite la protocoalele de gestiune a cheilor:

1. Număr minim de mesaje interschimbate;
2. Număr cât mai mic de biţi transmişi;
3. Număr cât mai mic de calcule;
4. Existenţa unei faze de precalcul.

Pentru uşurinţa formalizării, în acest capitol vom nota utilizatorii (*Alice, Bob, Charlie*, ...) cu $A, B, C, \ldots$, iar autoritatea (arbitrul, serverul) cu T.

5.2 Protocoale de distribuire a cheilor

Schema generală pentru această clasă de protocoale de gestiune a cheilor este: pentru orice pereche de utilizatori (A, B), arbitrul T generează aleator o cheie $K_{A,B} = K_{B,A}$ pe care o transmite lui A şi B printr-un canal securizat (neutilizat pentru comunicaţiile uzuale). Această cheie va fi folosită ulterior de cei doi utilizatori pentru a inter-schimba mesaje.

Deci, în această fază, T generează C_n^2 chei pe care le distribuie celor n utilizatori. Fiecare utilizator trebuie să păstreze cheia sa, precum şi cele $n-1$ chei de comunicare cu ceilalţi utilizatori.

Această etapă este sigură, dar ridică o serie de probleme cum ar fi:

- Existenţa de canale securizate între T şi fiecare din cei n participanţi. De exemplu, pentru o reţea cu 1000 utilizatori sunt necesare $C_{1000}^2 = 499.500$ canale sigure de comunicaţie. Mai mult, la venirea unui utilizator nou trebuie securizate alte 1000 canale.
- Obligaţia pentru fiecare participant să stocheze $n-1$ chei şi să participe la C_n^2 transmisii de chei solicitate de server.
  Chiar pentru o reţea mică, acest lucru devine destul de dificil.

Scopul protocoalelor prezentate este de a micşora (fără a periclita siguranţa comunicării) cantitatea de informaţie care trebuie transmisă sau stocată.

5.2.1 Protocolul Blom

Fie n ($n \geq 3$) utilizatori şi p ($p \geq n$) un număr prim. Fiecare cheie este un element din Z_p^*. Se alege un număr întreg $k \in Z_{n-1}^*$.
Valoarea lui k este numărul maxim de intruşi împotriva contra cărora poate fi asigurată protecţia.

În procedeul Blom – definit iniţial în 1984 pe baza codurilor separabile cu distanţă maximă (MDS) şi simplificat în 1993 – T transmite fiecărui utilizator, prin canale securizate, $k+1$ elemente din Z_p (în loc de $n-1$ chei în varianta generală).

Fiecare pereche de utilizatori (A, B) poate calcula o cheie de sesiune $K_{A,B} = K_{B,A}$.

Condiţia de securitate este: orice coaliţie de k utilizatori – diferiţi de A şi B – nu poate obţine informaţii despre $K_{A,B}$.

Pentru cazul $k = 1$ protocolul Blom este:

1. T face publice: un număr prim p şi – pentru fiecare utilizator A – un număr $r_A \in Z_p$. Toate numerele r_A sunt distincte.
2. T generează aleator trei numere $a, b, c \in Z_p$ şi formează polinomul
$$f(x, y) = a + b(x + y) + cxy \pmod p$$
3. Pentru fiecare utilizator A, T determină polinomul
$$g_A(x) = f(x, r_A) \pmod p$$
şi transmite $g_A(x)$ lui U printr-un canal securizat.
4. Dacă A şi B doresc să comunice între ei, cheia lor secretă este
$$K_{A,B} = K_{B,A} = f(r_A, r_B)$$

Observaţia 5.1.

- *Aplicaţia $g_A(x)$ de la pasul 3 este o transformare afină, de forma*
$$g_A(x) = a_A + b_A x$$
unde
$$a_A = a + br_A \pmod p, \qquad b_A = b + cr_A \pmod p.$$
- *La pasul 4, utilizatorul A poate calcula cheia secretă $K_{A,B} = f(r_A, r_B) = g_A(r_B)$, iar B – în mod similar – $K_{B,A} = f(r_A, r_B) = g_B(r_A)$.*

Exemplul 5.1. *Să presupunem că $p = 17$ şi sunt 3 utilizatori A, B, C – având cheile publice $r_A = 12$, $r_B = 7$ şi respectiv $r_C = 1$.*
Presupunem că arbitrul alege $a = 8$, $b = 7$, $c = 2$. Atunci
$$f(x, y) = 8 + 7(x + y) + 2xy$$

Polinoamele g sunt

$$g_A(x) = 7 + 14x, \qquad g_B(x) = 6 + 4x, \qquad g_C(x) = 15 + 9x.$$

Cele trei chei private sunt

$$K_{A,B} = 3, \quad K_{A,C} = 4, \quad K_{B,C} = 10.$$

A poate calcula $K_{A,B}$ prin

$$g_A(r_B) = 7 + 14 \cdot 7 = 3 \pmod{17};$$

B poate calcula $K_{B,A}$ prin

$$g_B(r_A) = 6 + 4 \cdot 12 = 3 \pmod{17}.$$

Să arătăm că nici un utilizator nu poate obţine singur informaţii asupra cheilor private ale celorlaltor participanţi.

Teorema 5.1. *Protocolul Blom pentru $k = 1$ este necondiţionat sigur*[1] *contra oricărui atac individual.*

Demonstraţie. Să presupunem că utilizatorul C doreşte să calculeze cheia

$$K_{A,B} = a + b(r_A + r_B) + cr_Ar_B \pmod{p}$$

Valorile r_A şi r_B sunt publice, dar a, b, c sunt secrete. C cunoaşte propriile sale valori

$$a_C = a + br_C \pmod{p} \qquad b_C = b + cr_C \pmod{p}$$

care sunt coeficienţii propriului său polinom $g_C(x)$.

Vom arăta că informaţia deţinută de C este consistentă cu orice valoare posibilă $m \in Z_p$ a cheii $K_{A,B}$. Deci C nu poate obţine nici o informaţie asupra cheii $K_{A,B}$.
Să considerăm ecuaţia matricială în Z_p:

$$\begin{pmatrix} 1 & r_A + r_B & r_Ar_B \\ 1 & r_C & 0 \\ 0 & 1 & r_C \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} m \\ a_C \\ b_C \end{pmatrix}$$

Prima ecuaţie este de fapt $K_{A,B} = m$, iar celelalte două dau informaţia cunoscută de C despre a, b, c.

Determinantul matricii este

$$r_C^2 + r_Ar_B - (r_A + r_B)r_C = (r_C - r_A)(r_C - r_B) \pmod{p}$$

Cum r_C este distinct de r_A şi r_B, acest determinant este nenul, deci sistemul admite o soluţie unică (a, b, c).
Cu alte cuvinte, orice valoare $m \in Z_p$ este compatibilă cu informaţia deja deţinută de C. □

[1] Un protocol este *necondiţionat sigur* dacă securitatea sa nu depinde de ordinul de complexitate al calculelor necesare pentru spargerea sa.

În schimb, orice coaliţie între doi participanţi C, D poate conduce la aflarea cheii $K_{A,B}$. Aceştia dispun de informaţiile:

$$\begin{cases} a_C = a + br_C, & b_C = b + cr_C, \\ a_D = a + br_D, & b_D = b + cr_D \end{cases}$$

Avem un sistem de 4 ecuaţii cu 3 necunoscute, din care se poate afla imediat soluţia unică (a, b, c). După ce s-au aflat aceste trei valori, se poate construi polinomul $f(x, y)$, din care se poate determina mai departe restul informaţiei.

Protocolul Blom poate fi generalizat pentru a rezista la atacul unei alianţe de k utilizatori. Singura modificare se face la pasul 2, unde arbitrul foloseşte polinomul

$$f(x, y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{i,j} x^i y^j \quad (mod\ p)$$

unde $a_{i,j} \in Z_p$ $(0 \leq i \leq k,\ 0 \leq j \leq k)$ şi $a_{i,j} = a_{j,i}$ pentru orice i şi j.

5.3 Protocolul Needham - Schroeder

Un protocol de distribuţie a cheilor, celebru prin implicaţiile sale în stabilirea de chei de comunicaţie este definit în 1978 de Roger Needham şi Michael Schroder ([62]).

Protocolul Needham - Schroeder (NS) a fost propus în două variante:

- Protocolul NS cu chei simetrice. El a fost destinat stabilirii unei chei de sesiune între doi utilizatori conectaţi pe o reţea şi a stat ulterior la baza construirii protocolului Kerberos.
- Protocolul NS cu chei publice. Intenţa sa a fost de a oferi o autentificare reciprocă a celor doi utilizatori care comunică prin reţea

5.3.1 Protocolul NS cu chei simetrice

În acest protocol, A iniţiază un protocol de comunicaţie cu B, iar T este un server – considerat de ambii utilizatori ca fiind "de încredere". Se mai folosesc:

- $K_{A,T}$ – o cheie simetrică, cunoscută de A şi T;
- $K_{B,T}$ – o cheie simetrică, cunoscută de B şi T;
- N_A, N_B – numere (nonces) generate aleator de A respectiv B.
- $K_{A,B}$ – cheia (simetrică) de sesiune între A şi B.

Observaţia 5.2. *În protocoalele de securitate, termenul "nonce" reprezintă "un număr folosit o singură dată". În mod uzual, el este un număr generat aleator sau pseudo-aleator prin algoritmi care asigură un grad sporit de securitate. Ideea este ca un astfel de număr să nu mai poată fi generat a doua oară sub nici o formă. Scopul este evitarea reuşirii unor atacuri de tip reluare sau dicţionar.*
Este utilizat frecvent în protocoalele de autentificare, de partajare a secretelor, de stabilirea de chei de sesiune sau de construcţie de amprente.

În acest protocol vom nota cu $\{\alpha\}_K$ criptarea mesajului α folosind cheia K.
Varianta de bază a protocolului NS este:

1. A trimite lui T tripletul (A, B, N_A).
2. T trimite lui A mesajul: $\{N_A, K_{A,B}, B, \{K_{A,B}, A\}_{K_{B,T}}\}_{K_{A,T}}$.
3. A decriptează mesajul şi transmite mai departe lui B: $\{K_{A,B}, A\}_{K_{B,T}}$.
4. B decriptează şi trimite spre A mesajul $\{N_B\}_{K_{A,B}}$.
5. A răspunde lui B cu $\{N_B - 1\}_{K_{A,B}}$.

Comentarii asupra paşilor efectuaţi de protocolul NS:

1. A anunţă serverul că doreşte stabilirea unui canal de comunicaţie cu B.
2. Serverul generează cheia $K_{A,B}$ pe care o retrimite lui A în două copii: una pentru uzul lui A, iar alta – criptată cu $K_{B,T}$ – pe care A o va forwarda lui B.

   Deoarece A poate solicita chei pentru comunicarea cu mai mulţi utilizatori, nonce-ul asigură faptul că serverul a răspuns la acest mesaj; de asemenea, includerea lui B îi spune lui A cu cine va partaja această cheie de sesiune.
3. A forwardează cheia lui B; acesta o poate decripta folosind cheia de comunicare cu serverul; în acest fel se asigură şi o autentificare a datelor.
4. Prin acest mesaj, B îi arată lui A că deţine cheia de sesiune $K_{A,B}$.
5. A arată prin acest mesaj că este conectat în continuare şi că deţine cheia de sesiune.

Protocolul este vulnerabil la un atac prin reluare (identificat de Denning şi Sacco). Dacă *Oscar* deţine o valoare compromisă (mai veche) pentru $K_{A,B}$, el poate returna lui B mesajul $\{K_{A,B}, A\}_{K_{B,T}}$. Acesta îl va accepta, neputând să distingă dacă ce a primit este o cheie nouă sau nu.
Această slăbiciune va fi corectată de protocolul Kerberos (secţiunea următoare) prin introducerea unei ştampile de timp sau de nonce-uri.

5.3.2 Protocolul NS cu chei publice

A şi B folosesc un server T care distribuie – la cerere – chei publice. Aceste chei sunt:

- (K_{PA}, K_{SA}) – componenta publică şi secretă a cheii lui A.
- (K_{PB}, K_{SB}) – componenta publică şi secretă a cheii lui B.
- (K_{PT}, K_{ST}) – componenta publică şi secretă a cheii serverului T, cu menţiunea că K_{ST} este folosită pentru criptare, iar K_{PT} – pentru decriptare (deci T foloseşte cheia sa pentru autentificare).

Protocolul este următorul:

1. A trimite lui T perechea (A, B) prin care solicită cheia publică a lui B.
2. T trimite lui A elementul $\{K_{PB}, B\}_{K_{ST}}$.
3. A generează nonce-ul N_A şi-l trimite lui B: $\{N_A, A\}_{K_{PB}}$.
4. B trimite lui T perechea (B, A) solicitând cheia publică a lui A.
5. T răspunde cu $\{K_{PA}, A\}_{K_{ST}}$.
6. B generează nonce-ul N_B şi-l trimite lui A: $\{N_A, N_B\}_{K_{PA}}$.
   Trimiterea lui N_A arată lui A că B deţine de cheia de decriptare K_{SB}.
7. A confirmă că deţine cheia de decriptare, trimiţând lui B elementul $\{N_B\}_{K_{PB}}$.

La sfârşitul protocolului, A şi B cunosc fiecare identităţile celuilalt şi nonce-urile N_A, N_B.

Securitatea protocolului

Varianta cu cheie publică a protocolului Needham - Schroeder nu rezistă atacului *man-in-the-middle*. Dacă *Oscar* (notat cu O) îl convinge pe A să iniţieze o sesiune cu el, poate retrimite mesajele lui B şi să-l convingă că este în contact cu A.
Ignorând traficul de informaţii (din protocol) cu serverul T, atacul se desfă-şoară astfel:

1. A trimite lui O mesajul $\{N_A, A\}_{K_{PO}}$.
2. O decriptează cu K_{SO}, după care trimite lui B mesajul $\{N_A, A\}_{K_{PB}}$.

3. B răspunde lui O (crezând că este A) cu mesajul $\{N_A, N_B\}_{K_{PA}}$.

4. O trimite acest mesaj mai departe, spre A.

5. A răspunde lui O cu $\{N_B\}_{K_{PO}}$.

6. O extrage de aici N_B şi-l trimite lui B, criptat cu cheia publică a acestuia: $\{N_B\}_{K_{PB}}$.

În final, B este convins că comunică cu A şi că N_A, N_B sunt cunoscute doar de el şi de A.

Atacul a fost prezentat prima oară în 1995 de Gavin Lowe. Acesta construieşte o variantă a protocolului, numită ulterior protocolul *Needham - Schroeder - Lowe*. Modificarea se referă la pasul 6 din protocol, unde mesajul $\{N_A, N_B\}_{K_{PA}}$ – trimis de B către A – este înlocuit cu $\{N_A, N_B, B\}_{K_{PA}}$.

5.4 Protocolul Kerberos

Kerberos este unul din cele mai răspândite sisteme de gestiune a cheilor – în special versiunile 4 şi 5 (ultimul, adoptat ca standard Internet *RFC* 1510). Protocolul se bazează parţial pe protocolul Needham - Schroeder, aducând unele îmbunătăţiri legate de securitate.

În acest protocol, fiecare utilizator A împarte cu T o cheie DES[2] notată K_A.
De asemenea, lui A i se asociază (în general sub controlul lui T) o secvenţă de informaţii care-l identifică în mod unic (nume, adresă, CNP, număr telefon etc). Această secvenţă se notează cu $ID(A)$.

La solicitarea lui A de a comunica cu B, arbitrul T efectuează următoarele operaţii:

- Generează o cheie K;
- Înregistrează ora H a cererii;
- Stabileşte o durată L de validitate a lui K; deci cheia de sesiune este validă în intervalul de timp $[T, T + L]$.

Tichetul cu informaţiile (K, H, L) sunt transmise de T lui A, apoi lui B. Protocolul Kerberos este prezentat pe pagina următoare.

În acest protocol, fiecare din cele patru mesaje m_i transmise are rolul său bine determinat.
Astfel, m_1 şi m_2 servesc la transmiterea confidenţială a cheii K. La rândul lor, m_3 şi m_4 asigură o confirmare a cheii; după primirea ei, A şi B sunt siguri că dispun de aceeaşi cheie de sesiune K.

Rolul lui H şi L este acela de protejare contra unui atac activ constând din înregistrarea unor mesaje vechi şi – ulterior – retransmiterea lor.

[2]Ultimele versiuni folosesc modul CBC de implementare a sistemului DES.

1. T generează K, H şi L.
2. T calculează mesajele
$$m_1 = \{K, ID(B), H, L\}_{K_A}, \quad m_2 = \{K, ID(A), H, L\}_{K_B}$$
pe care le trimite lui A.
3. A decriptează m_1 şi află $K, H, L, ID(B)$. Pe urmă calculează
$$m_3 = \{ID(A), H\}_K$$
şi trimite lui B mesajele m_2 şi m_3.
4. B află $K, H, L, ID(A)$ din decriptarea lui m_2. Cu ajutorul lui K decriptează m_3 şi află $H, ID(A)$.
Verifică dacă cele două valori pentru H şi $ID(A)$ sunt identice.
5. B calculează
$$m_4 = \{H + 1\}_K$$
pe care îl trimite lui A.
6. A decriptează m_4 şi verifică dacă mesajul obţinut este $H + 1$.

Există diverse opţiuni de implementare a tichetului (K, H, L), cum ar fi de exemplu:

- *Tichet eliberat pe termen lung.* Există riscul de a permite atacuri asupra cheii.
- *Tichet postdatat.* Riscul este ca *Oscar* să afle tichetul înainte de utilizarea sa; de aceea multe servere au permisiunea de a le putea respinge.
- *Tichet proxy.* Un client dă permisiune serverului de a iniţia anumite comunicaţii în numele său.

Una din slăbiciunile sistemului *Kerberos* constă în imposibilitatea unei bune sincronizări a ceasurilor utilizatorilor. În practică se admit anumite decalaje, stabilite de comun acord.

În plus, spargerea sistemului DES a condus – în unele standarde – la înlocuirea sistemului *Kerberos*.

5.5 Schimbul de chei Diffie - Hellman

Dacă nu se acceptă un furnizor universal de chei, atunci va trebui stabilit un protocol de punere de acord.
Primul şi cel mai cunoscut astfel de protocol este *protocolul de schimb de chei Diffie - Hellman.*
Definit în 1976 ([27]) şi folosit în diverse variante, el se bazează pe problema logaritmului discret.

5.5.1 Variante ale protocolului Diffie - Hellman

Pentru iniţializare, să considerăm:

- p – număr prim (de 1024 biţi – în protocoalele standard);
- q – divizor prim al lui $p-1$ (de 160 biţi);
- $\alpha \in Z_p^*$, element de ordin q.
- h – funcţie de dispersie criptografică (de exemplu $SHA1$);
- (sig_T, ver_T) – algoritm de semnătură a arbitrului T.

Utilizatorul A dispune de cheia secretă $a \in Z_q^*$ şi cheia publică $Y_A = \alpha^a$.

Pe baza componentelor publice (stocate în $ID(A)$) şi la solicitarea lui A, arbitrul T eliberează un certificat

$$Cert(A) = (ID(A), Y_A, sig_T(ID(A), Y_A))$$

care este de asemenea public.
Orice utilizator B poate verifica autenticitatea certificatului $Cert(A)$ şi poate obţine de aici – printre altele – cheia publică Y_A.

Aceleaşi elemente (b, $Y_B = \alpha^b$, $ID(B)$, $Cert(B)$) sunt generate pentru utilizatorul B.

Observaţia 5.3.

1. *Arbitrul nu trebuie să cunoască cheia secretă "a" pentru a produce certificatul. Când A intră în reţea, se generează un astfel de certificat, care poate fi păstrat în baza de date sau poate fi comunicat chiar de A la fiecare utilizare. Semnătura lui T permite oricui să verifice autenticitatea informaţiei pe care o conţine.*
2. *Înainte de a elibera $Cert(A)$, T se convinge că A posedă cheia secretă "a" corespunzătoare cheii publice Y_A (un astfel de protocol, numit "provocare - răspuns", este prezentat în Capitolul 7).*
   În acest mod se evită un atac prin care Oscar înregistrează Y_A drept cheia sa publică, îngreunând astfel posibilitatea ca un alt utilizator B să intre în contact cu A.

3. De asemenea, T face o "validare" a cheii publice Y_A, verificând relaţiile

$$1 < Y_A < p, \quad Y_A^q \equiv 1 \ (mod\ p)$$

Sunt două versiuni ale protocolului de bază Diffie - Hellman, în funcţie de durata valabilităţii cheii de sesiune generate:

Protocolul Diffie - Hellman de perioadă scurtă:

1. A generează aleator $x \in Z_q^*$ şi trimite lui B valoarea $R_A = \alpha^x \ \ (mod\ p)$;
2. B generează aleator $y \in Z_q^*$ şi trimite lui A valoarea $R_B = \alpha^y \ \ (mod\ p)$;
3. A calculează $K_{A,B} = R_B^x = \alpha^{xy}$;
4. B calculează $K_{B,A} = R_A^y = \alpha^{xy}$.

Acest protocol[3] asigură o autentificare implicită a cheii de sesiune. Avantajul său este acela că fiecare execuţie a protocolului generează o cheie de sesiune nouă.
Dezavantajul este că nu rezistă la un atac *man-in-the-middle*, neexistând nici o autentificare a utilizatorilor implicaţi.

Protocolul Diffie - Hellman de perioadă lungă:

1. A trimite lui B certificatul $Cert(A)$;
2. B trimite lui A certificatul $Cert(B)$;
3. A extrage Y_B din $Cert(B)$ şi calculează $K_{A,B} = Y_B^a = \alpha^{ab}$;
4. B extrage Y_A din $Cert(A)$ şi calculează $K_{B,A} = Y_A^b = \alpha^{ab}$.

Avantajul acestei scheme constă în autentificarea reciprocă a partenerilor. Dezavantajul este că la fiecare execuţie a protocolului se obţine aceeaşi cheie de sesiune; cheile se schimbă doar odată cu schimbarea certificatelor – după o perioadă de timp relativ lungă.

Exemplul 5.2. *Să presupunem $p = 25307$ şi $\alpha = 2$. Dacă luăm $a = 3578$, vom avea $Y_A = 2^{3578} = 6113 \ \ (mod\ 25307)$, valoare pusă în certificatul lui A.*

Să presupunem că B alege $b = 19956$; atunci $Y_B = 2^{19956} = 7984 \ \ (mod\ 25307)$.

A poate calcula cheia comună

$$K_{A,B} = 7984^{3578} = 3694 \quad (mod\ 25307)$$

[3]Schimbul de chei din sistemul *SSH* (Secure SHell) – care asigură comunicarea pentru sistemele bazate pe UNIX – are implementat protocolul Diffie - Hellman de perioadă scurtă.

Aceiaşi cheie este calculată şi de B:

$$K_{U,V} = 6113^{19956} = 3694 \quad (mod\ 25307)$$

Matsumoto, Takashima şi Imai ([54]) combină cele două protocoale într-unul singur:

Protocolul $MTI/C0$:

1. A generează aleator $x \in Z_q^*$ şi trimite lui B valoarea $T_A = Y_B^x \ \ (mod\ p)$;
2. B generează aleator $y \in Z_q^*$ şi trimite lui A valoarea $T_B = Y_A^y \ \ (mod\ p)$;
3. A calculează $K_{A,B} = (T_B)^{a^{-1}x} = \alpha^{xy}$;
4. B calculează $K_{B,A} = (T_A)^{a^{-1}y} = \alpha^{xy}$.

Este interesant că acest protocol – spre deosebire de variantele din care provine – nu asigură o autentificare implicită a cheii. Astfel, *Oscar* poate – printr-un atac de tip *man-in-the-middle* – să înlocuiască T_A şi T_B cu numărul 1.
Atunci A şi B vor obţine cheia 1, cheie cunoscută şi de *Oscar*.

5.5.2 Securitatea protocolului Diffie - Hellman

Să studiem securitatea acestui protocol contra unui atac (activ sau pasiv).

Semnătura lui T pe certificate împiedică producerea de informaţii publice false. Deci vor rămâne în discuţie numai atacurile pasive, care se reduc la problema:

"*C poate determina* $K_{A,B}$ *dacă nu este A sau B ?*"

Altfel spus: "Fiind date α^a şi α^b, ambele $(mod\ p)$, se poate calcula α^{ab} $(mod\ p)$?"

Aceasta este cunoscută şi sub numele de *problema Diffie - Hellman* (variante ale acestei probleme sunt prezentate în Anexa 2):

Fie $I = (p, \alpha, \beta, \gamma)$ unde p este număr prim, $\alpha \in Z_p$ este primitiv, iar $\beta, \gamma \in Z_p^*$.
Se poate determina $\beta^{log_\alpha \gamma}$ $(mod\ p)$? (sau – echivalent, $\gamma^{log_\alpha \beta}$ $(mod\ p)$)

Evident, securitatea protocolului Diffie - Hellman (de gestiune a cheilor) faţă de atacurile pasive este echivalentă cu dificultatea problemei Diffie - Hellman.

Dacă C poate calcula a (sau b) plecând de la Y_A (respectiv Y_B), el poate deduce $K_{A,B}$ aşa cum face A (respectiv B).
Aceasta conduce la rezolvarea unei probleme de logaritm discret. Deci, dacă problema logaritmului discret în Z_p este dificilă, atunci protocolul de punere de acord a cheii Diffie - Hellman nu poate fi atacat.

Conjectura este aceea că problema Diffie - Hellman este echivalentă cu problema logaritmului discret (aşa cum s-a conjecturat că spargerea sistemului RSA este echivalentă cu factorizarea unui număr).

Teorema 5.2. *A sparge sistemul de criptare El Gamal este echivalent cu a rezolva problema Diffie - Hellman.*

Demonstraţie. Să reamintim sistemul de criptare *El Gamal*:
O cheie este $K = (p, \alpha, a, \beta)$ unde $\beta = \alpha^a \ (mod\ p)$; a este secret, iar p, α, β sunt publice.
Criptarea unui mesaj $x \in Z_p$ se face alegând aleator un număr $k \in Z_{p-1}^*$; apoi

$$e_K(x, p) = (y_1, y_2)$$

unde

$$y_1 = \alpha^k \pmod p, \qquad y_2 = x\beta^k \pmod p.$$

Pentru $y_1, y_2 \in Z_p^*$, decriptarea este definită prin

$$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \pmod p.$$

Să presupunem că dispunem de un algoritm $\mathcal{A}$ care rezolvă problema Diffie - Hellman şi încercăm un atac asupra mesajului criptat (y_1, y_2).
Aplicând $\mathcal{A}$ asupra intrărilor p, α, y_1 şi β avem:

$$\mathcal{A}(p, \alpha, y_1, \beta) = \mathcal{A}(p, \alpha, \alpha^k, \alpha^a) = \alpha^{ka} = \beta^k \pmod p$$

De aici rezultă $x = y_2(\beta^k)^{-1} \ (mod\ p)$, deci se poate decripta mesajul (y_1, y_2).

Invers, să presupunem că dispunem de un algoritm $\mathcal{B}$ de decriptare pentru sistemul *El Gamal*. Deci $\mathcal{B}$ admite la intrare $(p, \alpha, \beta, y_1, y_2)$ şi calculează

$$x = y_2 \left(y_1^{log_\alpha \beta}\right)^{-1} \pmod p$$

Fiind dată o apariţie $(p, \alpha, \beta, \gamma)$ a problemei Diffie - Hellman, se poate calcula uşor

$$\mathcal{B}(p, \alpha, \beta, \gamma, 1)^{-1} = 1 \left(\left(\gamma^{log_\alpha \beta}\right)^{-1}\right)^{-1} = \gamma^{log_\alpha \beta} \pmod p$$

□

Totuşi, protocolul de bază Diffie - Hellman are o vulnerabilitate majoră: nu rezistă unui atac *man-in-the-middle*.
Schematic, protocolul Diffie - Hellman se efectuează după schema următoare:

$$A \xrightarrow{\alpha^a} \xleftarrow{\alpha^b} B$$

Să presupunem că într-un atac activ, *Oscar* (O) se interpune între A şi B în modul următor:

$$A \xrightarrow{\alpha^a} \xleftarrow{\alpha^{b'}} O \xrightarrow{\alpha^{a'}} \xleftarrow{\alpha^b} B$$

unde $a', b' \in Z_q^*$ sunt generate aleator de O.

O interceptează mesajele lui A şi B şi le înlocuieşte cu ale sale. La sfârşitul protocolului, el a stabilit o cheie de comunicaţie $\alpha^{ab'}$ cu A şi o cheie $\alpha^{a'b}$ cu B.
Când A doreşte să trimită un mesaj lui B, el va utiliza cheia pe care o împarte cu O; acesta poate decripta mesajul şi apoi să îl cripteze cu cheia comună cu B.
B primeşte mesajul, fără să realizeze că a fost citit de O.

Acelaşi lucru se întâmplă în cazul unui mesaj trimis de B către A.

5.5.3 Punerea de acord MTI

Protocolul MTI propus de Matsumoto, Takashima şi Imai, poate fi îmbunătăţit astfel ca să reziste unui atac *man-in-the-middle*.

1. A generează aleator $x \in Z_{p-1}^*$, calculează
$$R_A = \alpha^x \pmod p$$
şi trimite lui B perechea $(Cert(A), R_A)$.
2. B generează aleator $y \in Z_{p-1}^*$, calculează
$$R_B = \alpha^y \pmod p$$
şi trimite lui A perechea $(Cert(B), R_B)$.
3. A calculează $K_{A,B} = R_B^a \cdot Y_B^x \pmod p$, iar B calculează $K_{B,A} = R_A^b \cdot Y_A^y \pmod p$.

Cheia comună de sesiune este deci

$$K = K_{A,B} = K_{B,A} = \alpha^{ay+bx} \pmod p$$

Exemplul 5.3. *Să luăm* $p = 27803$ *şi* $\alpha = 5$.
Dacă A *alege* $a = 21131$, *el va calcula* $Y_A = 5^{21131} = 21420 \pmod{27803}$, *pe care îl pune în certificatul său.*
La fel, dacă B *alege* $b = 17555$, *va avea* $Y_B = 5^{17555} = 17100 \pmod{27803}$

Presupunem că A *selectează* $x = 169$; *el va trimite lui* B

$$R_A = 5^{169} = 6268 \pmod{27803}$$

Dacă B *alege* $y = 23456$, *el trimite lui* A

$$R_B = 5^{23456} = 26759 \pmod{27803}$$

Acum se poate calcula

$$K_{A,B} = R_B^a \cdot Y_B^x \ (mod\ p) = 26759^{21131} \cdot 17100^{169} = 21600 \ (mod\ 27803)$$

Aceeaşi cheie o obţine şi B.

Semnătura arbitrului elimină posibilitatea interpunerii lui *Oscar*.

Într-adevăr, dacă un intrus O se interpune între A şi B, va avea loc scenariul

$$A \ \underset{Cert(B),\alpha^{y'}}{\overset{Cert(A),\alpha^{x}}{\rightleftarrows}} \ O \ \underset{Cert(B),\alpha^{y}}{\overset{Cert(A),\alpha^{x'}}{\rightleftarrows}} \ B$$

În acest moment, A şi B vor calcula chei diferite: A calculează $K = \alpha^{xb+y'a} \ (mod\ p)$, iar B calculează $K = \alpha^{x'b+ya} \ (mod\ p)$.

În plus, nici una din cheile lui A sau B nu poate fi calculată de O, pentru că aceasta ar necesita cunoaşterea exponentului secret "a" respectiv "b". Deci, deşi A şi B obţin chei distincte, nici una din ele nu poate fi calculată şi de O (în ipoteza că problema logaritmului discret este dificilă).

Altfel spus, A şi B sunt siguri că nici o altă persoană nu poate calcula cheia. Avem deci o *autentificare implicită* a cheii de sesiune.

Pentru atacuri pasive, securitatea sistemului MTI se reduce tot la dificultatea problemei Diffie - Hellman.

5.5.4 Protocoale AK înrudite cu protocolul Diffie - Helman

Vom prezenta câteva protocoale de punere de acord a cheilor cu autentificare (AK) utilizate frecvent în standarde criptografice. În toate aceste cazuri, confirmarea cheii este făcută prin protocoale separate, cuprinse în standarde.

KEA

Protocolul KEA (Key Exchange Algorithm) a fost propus de NSA şi declasificat în 1998.

1. A şi B obţin – din certificate – cheile publice Y_B respectiv Y_A.
2. A generează aleator $x \in Z_q^*$ şi trimite lui B valoarea $R_A = \alpha^x$.
3. B generează aleator $y \in Z_q^*$ şi trimite lui A valoarea $R_B = \alpha^y$.

4 A verifică
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$$
Dacă relaţiile sunt îndeplinite, calculează cheia
$$K_{A,B} = (Y_B)^x + (R_B)^a \pmod p$$

5 B procedează similar, obţinând – în caz de succes – cheia
$$K_{B,A} = (Y_A)^y + (R_A)^b \pmod p$$

6 A şi B calculează cheia de sesiune (de 80 biţi) $K = e(K_{A,B}) = e(K_{B,A})$, unde e este funcţia de criptare a sistemului simetric SKIPJACK ([84]).

Din calcule rezultă imediat
$$K_{A,B} = K_{B,A} = \alpha^{ay} + \alpha^{bx}$$

Exemplul 5.4. *Fie $p = 43$ şi $q = 7$ (divizor al lui $p - 1 = 42$). Se alege apoi $\alpha = 4$ (element de ordin 7 în Z_{43}^*).*

Să presupunem că $a = 5$ şi $b = 2$; deci cheile publice sunt $Y_A = 4^5 = 35 \pmod{43}$ respectiv $Y_B = 4^2 = 16 \pmod{43}$.

Protocolul KEA va decurge astfel:

1. *A şi B obţin – din certificate – $Y_B = 16$ respectiv $Y_A = 35$.*
2. *A alege $x = 6$ şi trimite lui B valoarea $R_A = 4^6 = 11 \pmod{43}$.*
3. *B alege $y = 3$ şi trimite lui A valoarea $R_B = 4^3 = 21 \pmod{43}$.*
4. *A verifică condiţiile $1 < R_B < 21 < 43$ şi $R_B^q = 21^7 = 1 \pmod{43}$.*
   Cum ele sunt îndeplinite, A calculează cheia
   $$K_{A,B} = 16^6 + 21^5 = 39 \pmod{43}.$$
5. *B verifică condiţiile $1 < R_A < 11 < 43$ şi $R_A^q = 11^7 = 1 \pmod{43}$.*
   Cum ele sunt îndeplinite, B calculează cheia
   $$K_{B,A} = 35^3 + 11^2 = 39 \pmod{43}.$$

Este foarte important ca A (B) să verifice relaţiile de la pasul 3 (respectiv 4).

În caz contrar, sunt posibile unele atacuri, cum ar fi:

1. Dacă A nu verifică relaţia $(R_B)^q \equiv 1 \ (mod\ p)$.
   Atunci utilizatorul B poate afla informaţii despre cheia secretă a a lui A.
   Să presupunem că $p-1$ are un divizor s de lungime mică (până în 40 biţi) şi fie $\beta \in Z_p^*$ de ordin s. Dacă B trimite $R_B = \beta$, atunci A calculează $K_{A,B} = \alpha^{bx} + \beta^a \ (mod\ p)$ şi $K = e(K_{A,B})$.

   Să presupunem acum că A trimite lui B un mesaj criptat $c = e_K(m)$, unde e_K este funcţia de criptare (cu cheia K) a unui sistem de criptare simetric, iar mesajul m are o structură standard. Pentru fiecare $d \in [0, s-1]$, B va calcula $K'_{B,A} = \alpha^{bx} + \beta^d \ (mod\ p)$ şi $K' = e(K'_{B,A})$, după care va decripta $m' = e^{-1}_{K'}(c)$.
   Dacă m' are structura standard, atunci B trage concluzia că $d = a \ (mod\ s)$, deci obţine informaţie parţială despre a.

   Acest lucru se poate repeta pentru diferiţi factori primi mici ai lui $p-1$.

2. Dacă A nu verifică relaţiile $1 < Y_B < p, \ \ 1 < R_B < p$.
   Atunci *Oscar* certifică $Y_O = 1$ drept cheia sa publică, după care trimite lui B cheia publică temporară R_A a lui A, spunând că este a sa. După ce B replică cu R_B, *Oscar* trimite $R'_B = 1$ lui A, spunând că vine de la B. A calculează $K_{A,B} = \alpha^{bx} + 1$ şi B calculează $K_{B,O} = \alpha^{bx} + 1$. Deci B are – fără să ştie – o cheie de sesiune comună cu A.

Construirea cheii de sesiune prin criptarea cu un sistem de criptare simetric a cheii comune, are cel puţin două argumente în favoarea sa:

i. În acest fel se amestecă biţii "tari" ai sistemului cu eventuali biţi "slabi" – biţi care oferă informaţie despre cheia comună $K_{A,B}$, sau pot fi ghiciţi cu o probabilitate semnificativă.

ii. Se distrug relaţiile algebrice dintre cheia comună şi cheile publice Y_A, Y_B, R_A, R_B. Acest lucru previne un atac cu cheie cunoscută, numit *atacul triunghiular al lui Burmester*, care poate fi descris astfel:

   C este un utilizator cu perechea de chei (c, α^c). El observă o execuţie a protocolului între A şi B, în care aceştia inter-schimbă cheile publice temporare $R_A = \alpha^x, \ R_B = \alpha^y$; cheia comună rezultată este $K_{A,B} = K_{B,A} = \alpha^{ay} + \alpha^{bx}$.
   C iniţiază cu A o execuţie a protocolului, folosind drept cheie publică temporară $R_C = \alpha^y$. Dacă A foloseşte acum $R_A = \alpha^{x'}$, va rezulta o cheie $K_{A,C} = \alpha^{ay} + \alpha^{cx'}$ pe care o poate calcula numai A.
   Similar, C iniţiază cu B o execuţie a protocolului, folosind $R_C = \alpha^x$; rezultă o cheie $K_{B,C} = \alpha^{bx} + \alpha^{cy'}$ care poate fi calculată numai de B.
   Dacă C poate afla prin alte mijloace $K_{A,C}$ şi $K_{B,C}$ (faza de "cheie cunoscută" a atacului), el va putea determina

$$K_{A,B} = K_{A,C} + K_{B,C} - \alpha^{cx'} - \alpha^{cy'}.$$

Modelul unificat

Modelul unificat (The Unified Model) ([6]) este un protocol utilizat în prima versiune a standardelelor ANSI $X9.42$, ANSI $X9.63$ şi IEEE $P1363$. Avantajul său este simplitatea – şi deci uşurinţa analizei sale.

1. A generează aleator $x \in Z_q^*$; trimite lui B elementele $R_A = \alpha^x$ şi $Cert(A)$.
2. B generează aleator $y \in Z_q^*$; trimite lui A elementele $R_B = \alpha^y$ şi $Cert(B)$.
3. A verifică relaţiile
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$$
Dacă ele sunt îndeplinite, A calculează cheia de sesiune
$$K = h((Y_B)^a \parallel (R_B)^x)$$
4. B procedează similar. Cheia de sesiune este obţinută de B cu formula
$$K = h((Y_A)^b \parallel (R_A)^y)$$

Cheia comună calculată de A şi B este $K = h(\alpha^{ab} \| \alpha^{xy})$.

MQV

Protocolul MQV (Menezes, Qu, Solinas) este inclus în 1998 în versiunile iniţiale ale standardelor ANSI $X9.42$, ANSI $X9.63$ şi IEEE $P1363$. El are multe puncte comune cu *Modelul unificat*, diferenţele apărând în calculul cheii de sesiune.

1. A generează aleator $x \in Z_q^*$; trimite lui B elementele şi $R_A = \alpha^x$ şi $Cert(A)$.
2. B generează aleator $y \in Z_q^*$; trimite lui A elementele şi $R_B = \alpha^y$ şi $Cert(B)$.
3. A verifică relaţiile
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$$
Dacă ele sunt îndeplinite, A calculează
$$s_A = (x + a \cdot \overline{R}_A) \pmod q, \qquad K_A = \left(R_B \cdot (Y_B)^{\overline{R}_B}\right)^{s_A}.$$
Dacă relaţiile nu sunt verificate sau $K_A = 1$, atunci A opreşte protocolul.

4 B procedează similar şi calculează
$$s_B = (y + b \cdot \overline{R}_B) \pmod q, \qquad K_B = \left(R_A \cdot (Y_A)^{\overline{R}_A}\right)^{s_B}.$$

5 Cheia comună de sesiune este $K = h(K_A) = h(K_B) = h(\alpha^{s_A s_B})$.

Raportul tehnic care defineşte protocolul foloseşte o notaţie suplimentară: Dacă $X \in Z_p^*$, atunci $\overline{X} = (X \pmod{2^{80}}) + 2^{80}$. Mai general,

$$\overline{X} = (X \pmod{2^{\lceil f/2 \rceil}}) + 2^{\lceil f/2 \rceil}$$

unde f este lungimea lui q, exprimată în biţi. De remarcat că $\overline{X} \pmod q \neq 0$.

Observaţia 5.4. *Utilizarea lui $\overline{R}_A$ foloseşte doar jumătate din biţii lui R_A; în acest fel numărul de operaţii pentru calculul lui $(Y_A)^{\overline{R}_A}$ se înjumătăţeşte fără a afecta securitatea protocolului.*
În plus, definiţia lui $\overline{R}_A$ asigură $\overline{R}_A \neq 0$, deci contribuţia cheii private "a" la calculul lui s_A nu poate fi ignorată.

Verificarea $K_A = 1$ asigură condiţia ca ordinul lui K_A să fie q.
Asupra protocolului MQV poate fi lansat un atac *on-line*:
Să presupunem că C interceptează cheia R_A trimisă spre B; pe baza ei:

1. C calculează $R_C = R_A \cdot (Y_A)^{\overline{R}_A} \cdot \alpha^{-1}$, $c = \left(\overline{R}_C\right)^{-1}$ şi $Y_C = \alpha^c$.
2. C determină cheia Y_C şi o certifică (lucru posibil, deoarece C cunoaşte cheia secretă c).
3. C transmite lui B elementele R_C şi $Cert(C)$.
4. B răspunde cu R_B, pe care C îl forwardează lui A.

În acest moment, A şi B au o cheie de sesiune comună K, pe care B crede că o împarte cu C (nu cu A).

Variante ale protocoalelor AK într-o singură trecere

Principalul scop al protocoalelor AK într-o trecere este punerea de acord a lui A şi B asupra unei chei de sesiune necesare transmiterii unui singur mesaj – de la A la B; acest lucru presupune evident faptul că A are acces la $Cert(B)$. Astfel de protocoale sunt utile în aplicaţii în care numai unul din utilizatori este on-line.

Toate protocoalele prezentate (KEA, *Modelul unificat*, MQV) pot fi convertite în protocoale cu o singură trecere, prin simpla înlocuire a cheii temporare R_B a lui B cu cheia publică Y_B.
Exemplificăm acest lucru cu transformarea protocolului MQV într-un protocol cu o singură trecere.

1. A generează aleator $x \in Z_q^*$; trimite lui B elementele şi $R_A = \alpha^x$ şi $Cert(A)$.
2. A calculează

$$s_A = (x + a \cdot \overline{R}_A) \pmod q, \qquad K_A = \left(Y_B \cdot (Y_B)^{\overline{Y}_B}\right)^{s_A}$$

   Dacă $K_A = 1$, atunci A opreşte execuţia (cu eşec).
3. B verifică relaţiile

$$1 < R_A < p, \qquad (R_A)^q \equiv 1 \pmod p$$

   Dacă aceste condiţii sunt îndeplinite, atunci B calculează

$$s_B = (b + b \cdot \overline{Y}_B) \pmod q, \qquad K_B = \left(R_A \cdot (Y_A)^{\overline{R}_A}\right)^{s_B}$$

   Dacă $K_B = 1$, atunci B opreşte protocolul (cu eşec).
4. Cheia de sesiune este $K = h(K_A) = h(K_B) = h(\alpha^{s_A s_B})$.

5.5.5 Protocoale AKC înrudite cu protocolul Diffie - Helman

Protocoalele explicite de punere de acord a cheii se obţin de obicei din protocoale implicite însoţite de un mesaj de autentificare (un MAC). În general ele sunt preferate protocoalelor AK, deoarece confirmarea asigură un plus de securitate.

Protocoale AKC derivate din protocoale AK

Vom construi un protocol în trei treceri derivat din *Modelul unificat*, la care se adaugă un MAC care autentifică numărul trecerii, utilizatorii şi cheile temporare.
Se mai folosesc două funcţii de dispersie h_1, h_2. În practică, ele sunt definite

$$h_1(m) = h(10, m), \quad h_2(m) = h(01, m)$$

unde h este o funcţie de dispersie criptografică.

1. A generează aleator $x \in Z_q^*$; trimite lui B elementele $R_A = \alpha^x$ şi $Cert(A)$.
2. (a) B verifică
$$1 < R_A < p, \qquad (R_A)^q \equiv 1 \pmod p$$
Dacă relaţiile nu sunt îndeplinite, B opreşte execuţia (cu eşec).
   (b) B generează $y \in Z_q^*$ şi calculează
$$k' = h_1\left((Y_A)^b \,\|\, (R_A)^y\right), \quad K = h_2\left((Y_A)^b \,\|\, (R_A)^y\right),$$
$$R_B = \alpha^y, \qquad m_B = MAC_{k'}(2, B, A, R_B, R_A).$$
   (c) B trimite lui A tripletul $(Cert(B),\ R_B,\ m_B)$.
3. (a) A verifică
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$$
Dacă relaţiile nu sunt îndeplinite, B opreşte execuţia (cu eşec).
   (b) A calculează
$$k' = h_1\left((Y_B)^a \,\|\, (R_B)^x\right), \quad m'_B = MAC_{k'}(2, B, A, R_B, R_A)$$
şi verifică egalitatea $m'_B = m_B$.
   (c) A calculează
$$m_A = MAC_{k'}(3, A, B, R_A, R_B), \quad K = h_2\left((Y_B)^a \,\|\, (R_B)^x\right)$$
şi trimite lui B valoarea m_A.
4. B calculează $m'_A = MAC_{k'}(3, A, B, R_A, R_B)$ şi verifică egalitatea $m'_A = m_A$.
5. Cheia de sesiune este K.

În mod similar se pot construi protocoale AKC derivate din KEA şi MQV. Variantele AKC ale protocoalelor MQV şi *Model Unificat* sunt incluse în standardul ANSI $X9.63$.

Protocol între staţii

Tot cu scopul de a evita atacurile de tip *man-in-the-middle*, Diffie, Van Oorschot şi Wiener au propus un protocol numit STS (*protocol între staţii*).
Implementat în diverse variante, structura sa de bază este definită astfel:

1. A generează aleator un număr $a \in Z_{p-1}^*$; apoi calculează numărul $\alpha^a \ (mod\ p)$ pe care îl trimite lui B.
2. B generează aleator un număr $b \in Z_{p-1}^*$.
3. B calculează $\alpha^b \ (mod\ p)$, apoi
$$K = (\alpha^a)^b \ (mod\ p), \quad y_B = sig_B(\alpha^b, \alpha^a)$$
Trimite lui A mesajul $(Cert(B), \alpha^b \ (mod\ p), y_B)$.
4. A calculează
$$K = \left(\alpha^b\right)^a \ (mod\ p)$$
şi verifică $Cert(V)$ cu ver_T, apoi y_B cu ver_B extras din $Cert(B)$.
5. A calculează $y_A = sig_A(\alpha^a, \alpha^b)$ şi trimite lui B mesajul $(Cert(A), y_A)$.
6. B verifică $Cert(A)$ cu ver_T, apoi y_A cu ver_A extras din $Cert(A)$.

În această schemă, numărul prim p şi elementul primitiv $\alpha \in Z_p$ sunt publice. Fiecare utilizator A dispune de un protocol privat de semnătură sig_A şi unul public de verificare ver_A.
Arbitrul T are de asemenea asociate funcţiile sig_T respectiv ver_T. Certificatul lui A este

$$Cert(A) = (ID(A), ver_A, sig_T(ID(A), ver_A))$$

Informaţiile schimbate în cadrul protocolului STS simplificat sunt schematizate de diagrama următoare:

$$A \quad \xrightarrow{\alpha^a} \quad \xleftarrow{\alpha^b, sig_B(\alpha^b, \alpha^a)} \quad \xrightarrow{sig_A(\alpha^a, \alpha^b)} \quad B$$

Prin acest protocol, un intrus O nu se poate interpune între A şi B.
Într-adevăr, dacă O interceptează α^a şi îl înlocuieşte cu $\alpha^{a'}$, el va trebui să înlocuiască de asemenea şi $sig_B(\alpha^b, \alpha^{a'})$ cu $sig_B(\alpha^{b'}, \alpha^a)$, ceea ce nu poate decât în cazul $a = a'$ şi $b = b'$ (pentru că nu cunoaşte sig_B).

La fel, O nu poate înlocui $sig_A(\alpha^{a'}, \alpha^b)$ cu $sig_A(\alpha^a, \alpha^{a'})$, pentru că nu cunoaşte sig_A.

Varianta aceasta de protocol nu oferă totuşi o confirmare a cheii. Pentru aceasta trebuie modificat $y_B = e_K(sig_B(\alpha^b, \alpha^a))$ în pasul 3 şi $y_A = e_K(sig_A(\alpha^a, \alpha^b))$ în pasul 5.

În acest fel – ca la *Kerberos* – se obţine o confirmare a cheii decriptând o parte cunoscută a cheii de sesiune. Acesta este protocolul STS complet.

5.6 Chei auto-certificate

Metoda punerii de acord prezentată în acest paragraf este construită de Girault; ea nu necesită certificat. Valoarea cheii publice asigură o autentificare implicită.

Protocolul lui Girault combină proprietăţile sistemului RSA cu cele ale logaritmilor discreţi.

Fie $n = pq$ unde $p = 2p_1 + 1,\ q = 2q_1 + 1$ sunt numere prime tari (p_1, q_1 sunt numere prime distincte).
Cum grupul multiplicativ Z_n^* este izomorf cu $Z_p \times Z_q$, ordinul maxim al unui element din Z_n^* este $cmmmc(p-1, q-1) = 2p_1q_1$.

Fie $\alpha \in Z_n^*$ de ordin $2p_1q_1$. Vom utiliza problema logaritmului discret în subgrupul ciclic al lui Z_n^* generat de α.

În acest protocol, factorizarea $n = pq$ este cunoscută numai de către arbitrul T. Valorile n, α sunt publice, iar p, q sunt secrete (deci şi p_1, q_1).

T alege un exponent de criptare RSA public, să spunem e. Exponentul de decriptare $d = e^{-1}\ (mod\ \phi(n))$ este secret.

Fiecare utilizator A are un identificator $ID(A)$ şi primeşte de la T o cheie publică auto-certificată R_A, conform următorului protocol:

1. A alege un exponent secret a şi calculează $Y_A = \alpha^a\ (mod\ n)$.
2. A trimite lui T valorile (a, Y_A).
3. T calculează $R_A = (Y_A - ID(A))^d\ (mod\ n)$, pe care îl trimite lui A.

De remarcat aportul arbitrului în calculul lui R_A. Se observă că

$$Y_A = R_A^e + ID(A) \quad (mod\ n)$$

poate fi calculat folosind numai informaţiile publice.

Protocolul lui Girault este dat mai jos. Schematizat, schimbul de informaţii arată în felul următor:

$$A \quad \xrightarrow{\quad ID(A), R_A, \alpha^x\ (mod\ n) \quad} \quad B$$
$$A \quad \xleftarrow{\quad ID(B), R_B, \alpha^y\ (mod\ n) \quad} \quad B$$

La sfârşitul acestui protocol, A şi B dispun de aceeaşi cheie:

$$K = \alpha^{bx+ay} \quad (mod\ n)$$

1. A generează aleator x şi calculează $S_A = \alpha^x \ (mod\ n)$;
   tripletul $(ID(A), R_A, S_A)$ este trimis lui B.

2. B alege aleator y şi calculează $S_B = \alpha^y \ (mod\ n)$;
   tripletul $(ID(B), R_B, S_B)$ este trimis lui A.

3. A calculează cheia

$$K_{A,B} = S_B^a \cdot (R_B^e + ID(B))^x \quad (mod\ n).$$

   Cheia calculată de B este

$$K_{B,A} = S_A^b \cdot (R_A^e + ID(A))^y \ (mod\ n)$$

Exemplul 5.5. *Să presupunem* $p = 839$ *şi* $q = 863$. *Vom avea* $n = 724057$ *şi* $\phi(n) = 722356$. *Elementul* $\alpha = 5$ *are ordinul* $2p_1q_1 = \phi(n)/2$.

Dacă arbitrul T *alege* $e = 84453$ *drept exponent de criptare, vom avea* $d = 125777$.

Dacă $ID(A) = 500021$ *şi* $a = 111899$, *vom avea* $Y_A = 488889$ *şi* $R_A = 650704$.
În mod similar, considerăm $ID(B) = 500022$ *şi* $b = 123456$, *deci* $Y_B = 111692$, $R_B = 683556$.

Dacă A *şi* B *vor să stabilească o cheie de sesiune şi* A *alege numărul* $x = 56381$, *iar* B *numărul* $y = 356935$, *vom avea* $S_A = 171007$, $S_B = 320688$.

După protocol, cei doi utilizaotri vor dispune de cheia comună $K = 42869$.

5.6.1 Securitatea sistemului cu chei auto-certificate

Cum valorile $Y_A, R_A, ID(A)$ nu sunt semnate de autoritatea T, nimeni nu poate verifica direct autenticitatea lor.

Să presupunem că ele provin de la O (fără ajutorul arbitrului), care vrea să se dea drept A.

Dacă O furnizează $ID(A)$ şi dacă R_A conduce la un Y'_A greşit, nu se poate calcula exponentul a' asociat (dacă problema logaritmului discret este dificilă).
Fără a', O nu poate determina cheia.

O situaţie similară apare dacă O se interpune între A şi B.
El poate împiedica pe A şi B să obţină o cheie comună, dar nu poate efectua calculele lor. Are loc deci o autentificare implicită.

O întrebare ar fi: *De ce* A *trebuie să comunice arbitrului valoarea* a *?*
T poate determina R_A plecând de la Y_A, fără să cunoască a. Acest lucru se face pentru ca arbitrul să fie convins – înainte de a calcula R_A – că A posedă cheia secretă a.

Dacă T nu face această verificare înainte de calculul lui R_A, sistemul poate fi atacat.

Să presupunem că O alege un a' fals şi determină $Y'_A = \alpha^{a'} \ (mod\ n)$.

Cu aceste elemente, el stabileşte o cheie publică falsă $R'_A = (Y'_A - ID(A))^d \ (mod\ n)$ în felul următor:

O calculează $Y'_O = Y'_A - ID(A) + ID(O)$ şi trimite lui T perechea $(Y'_O, ID(O))$.

Presupunem că arbitrul calculează efectiv pentru O valoarea

$$R'_O = (Y'_O - ID(O))^d \ (mod\ n).$$

Atunci, din

$$Y'_O - ID(O) \equiv Y'_A - ID(A) \quad (mod\ n)$$

se obţine imediat $R'_O = R'_A$.

Să presupunem acum că A şi B efectuează protocolul, iar O se interpune conform schemei următoare:

$$A \xrightarrow[\xleftarrow{ID(B), R_B, \alpha^y \ (mod\ n)}]{ID(A), R_A, \alpha^x \ (mod\ n)} O \xrightarrow[\xleftarrow{ID(B), R_B, \alpha^y \ (mod\ n)}]{ID(A), R'_A, \alpha^{x'} \ (mod\ n)} B$$

B calculează deci cheia $K' = \alpha^{x'b+ya'} \ (mod\ n)$, iar A calculează $K = \alpha^{xb+ya} \ (mod\ n)$.

O obţine K' calculând $K' = S_B^{a'} \cdot (R_B^e + ID(B)))^{x'} \ (mod\ n)$.

O şi B posedă deci aceeaşi cheie, în timp ce B crede că o împarte cu A.

În acest moment, O poate decripta mesajele trimise de B pentru A.

5.7 Exerciţii

5.1. *Considerăm protocolul Blom (cazul $k = 1$) pentru utilizatorii A, B, C, D. Se dau valorile*

$$p = 7873, \quad r_A = 2365, \quad r_B = 6648, \quad r_C = 1837, \quad r_D = 2186.$$

Polinoamele secrete sunt:

$$g_A(x) = 6018 + 6351x, \quad g_B(x) = 3749 + 7121x,$$
$$g_C(x) = 7601 + 7802x, \quad g_D(x) = 635 + 6828x.$$

1. *Calculaţi cheile de sesiune pentru fiecare pereche de utilizatori (şi verificaţi de exemplu că $K_{A,B} = K_{B,A}$).*

2. *Arătaţi cum C şi D pot calcula $K_{A,B}$.*

5.2. *Considerăm protocolul Blom având $k = 2$, construit pentru 5 utilizatori: A, B, C, D, E. Datele iniţiale sunt*

$$p = 97, \quad r_A = 14, \quad r_B = 38, \quad r_C = 92, \quad r_D = 69, \quad r_E = 70.$$

Polinoamele secrete sunt

$$g_A(x) = 15 + 15x + 2x^2, \quad g_B(x) = 95 + 77x + 83x^2, \quad g_C(x) = 88 + 32x + 18x^2,$$
$$g_D(x) = 62 + 91x + 59x^2, \quad g_E(x) = 10 + 82x + 52x^2.$$

1. *Calculaţi cheia de sesiune* $K_{A,B} = K_{B,A}$.

2. *Arătaţi cum calculează alianţa* C, D, E *cheia* $K_{A,B}$.

5.3. *În protocolul Diffie - Hellman de perioadă scurtă avem* $p = 11$ *şi* $\alpha = 2$.

1. *Dacă* $R_A = 9$, *cât este a ?*

2. *Dacă cheia de sesiune este* $K = 3$, *cât a fost* R_B *?*

5.4. *În protocolul Diffie - Hellman avem* $p = 27001$, $\alpha = 101$. *Dacă* $a = 21768$ *şi* $b = 9898$, *determinaţi cheia de sesiune* $K_{A,B}$.

5.5. *Să presupunem că* A, B, C *doresc să stabilească o cheie comună de sesiune folosind o variantă a protocolului Diffie - Hellman. Cheia pe care o doresc este de forma*

$$\alpha^{abc} \ (mod\ p)$$

unde a *este ales de* A, b *este ales de* B *şi* c *este ales de* C.
Efectuaţi calculele parcurgând toate cele şase comunicări.

5.6. *Fie următoarea generalizare a protocolului Diffie - Hellman pentru stabilirea unei chei de sesiune între* $t \geq 2$ *utilizatori* $A_0, \ldots A_{t-1}$*:*

1. *Fiecare* A_i *generează aleator* $x_i \in Z_{p-1}^*$ *şi trimite* $y_i = \alpha^{x_i} \ (mod\ p)$ *la ceilalţi* $t-1$ *utilizatori.*

2. *Fiecare* A_i *calculează* $z_i = \left(y_{i+1} \cdot y_{i-1}^{-1}\right)^{x_i} \ (mod\ p)$ *şi trimite* z_i *la ceilalţi* $t-1$ *utilizatori.*

3. *Fiecare* A_i *calculează cheia de sesiune*

$$K = y_{i-1}^{tx_i} \cdot z_i^{t-1} z_{i+1}^{t-2} \ldots z_{i+t-3}^2 \cdot z_{i+t-2} \quad (mod\ p)$$

(toţi indicii de la paşii 2 *şi* 3 *sunt calculaţi modulo* t*).*

1. *Arătaţi că la sfârşitul protocolului toţi utilizatorii au aceeaşi cheie.*

2. *Arătaţi că pentru un adversar pasiv, a afla cheia de sesiune este echivalent cu a rezolva problema Diffie - hellman.*

5.7. *A şi B efectuează protocolul MTI cu* $p = 30113$ *şi* $\alpha = 52$. *Presupunem că A are cheia secretă* $a = 8642$ *şi alege* $x = 28654$, *iar B are cheia* $b = 24673$ *şi alege* $y = 12385$. *Verificaţi paşii protocolului şi determinaţi cheia de sesiune* $K_{A,B} = K_{B,A}$.

5.8. *Dacă un adversar pasiv încearcă să calculeze cheia K construită de A şi B pe baza protocolului MTI, el trebuie să rezolve o apariţie a problemei MTI reprezentată sub forma*

Enunţ: *Fie* $I = (p, \alpha, \beta, \gamma, \delta, \epsilon)$, *unde p este număr prim,* $\alpha \in Z_p$ *este primitiv, şi* $\beta, \gamma, \delta, \epsilon \in Z_p^*$.
Cerinţă: *Să se determine* $\beta^{log_\alpha \gamma} \delta^{log_\alpha \epsilon} \pmod p$.

Să se arate că un oracol care rezolvă problema MTI poate rezolva problema Diffie - Hellman şi reciproc.

5.9. *Se consideră protocolul Girault cu* $p = 167$, $q = 179$; *deci* $n = 29893$. *Presupunem* $\alpha = 2$ *şi* $e = 11101$.

1. *Calculaţi exponentul de decriptare d.*
2. *Ştiind* $ID(A) = 10021$ *şi* $a = 9843$, *calculaţi* Y_A *şi* R_A.
   Similar, ştiind $ID(B) = 10022$, *calculaţi* Y_B, R_B.
3. *Verificaţi cum se poate determina* Y_A *folosind* $R_A, ID(A)$ *şi e.*
   Similar pentru Y_B.
4. *Presupunem că A alege* $x = 15556$, *iar B alege* $y = 6420$. *Calculaţi* S_A, S_B *şi arătaţi cum A şi B obţin o cheie de sesiune comună.*

Bibliografie

[1] C. A. Asmuth, J. Bloom – *A modular approach to key safeguarding*, IEEE Trans on IT 29 (1983), pp. 208-210.

[2] Atanasiu, A. – *Securitatea Informaţiei, vol. 1, Criptografie*, ed. InfoData, Cluj, 2008.

[3] D. Bayer, S.Haber, W.Stornetta – *Improving the efficiency and reliability of digital time-stamping. Sequences II*, Methods in Communication, Security and Computer Science, Springer Verlag (1993), 329-334.

[4] J. Benaloh – *Verifiable sectret-ballot elections*, Ph.D thesis, Yale University, Technical report 561, 1987.

[5] J. Benaloh, J. Leichter – *Generalized secret sharing and monotone functions*, în "Advances in Cryptology – CRYPTO 8", S. Goldwasser, ed., LNCS 403 (1989), pp. 27-35.

[6] S. Blake - Wilson, A. Menezes – *Authenticated Diffie - Hellman Key Agreement Protocols*, Proc. of the 5-th Intern. Workshop on Security Protocols, LNCS 1361, 1997, pp. 137-158.

[7] G. R. Blakley – *Safeguarding cryptographic keys*, în "Proc. of the National Computer Conference, 1979", American Federation of Information Processing Societies Proc. 48 (1979), pp. 313-317.

[8] Boneh, D., Franklin, M. – *Identity Based Encryption from the Weil Pairing*, SIAM Journal of Computing, vol. 32, no. 3, pp. 586-615.

[9] Boneh, D., Boyen, X. – *Efficient Selective - ID Secure Identity Based Encryption Without Random Oracles*, Proc. of EUROCRYPT 2004, Interlaken, Elveţia, 2-6 May 2004, pp. 223-238.

[10] J. N. Bos, D. Chaum - *Provably unforgable signatures*, LNCS, 740 (1993), pp. 1-14.

[11] S. Brands – *An Efficient Off-Line Electronic Cash System Based On The Representation Problem*, Technical Report CS-R9323, 1993, CWI, Amsterdam, Netherlands.

[12] D. Chaum – *Untraceable electronic mail, return addresses, and digital pseudonyms*, Commun. ACM, 24 (2), 1981, pp. 8490.

[13] D. Chaum, H. van Antwerpen – *Undeniable signatures*, LNCS, 435 (1990), pp. 212-216.

[14] D. Chaum, A. Fiat, M. Naor – *Untraceable Electronic Cash*, Advances in Cryptology, CRYPTO 8, S. Goldwasser (Ed.), Springer-Verlag, pp. 319-327.

[15] D. Chaum, E. van Heyst – *Group signatures*, Advances in Cryptology, EUROCRYPT 91, LNCS, vol. 547, Springer-Verlag (1991), pp. 257-265.

[16] D. Chaum, E. van Heijst, B. Pfitzmann – *Cryptographically strong undeniable signatures, unconditionally secure for the signer*, LNCS, 576 (1992), pp. 470-484.

[17] Chen, L. ş.a – *An Efficient ID-KEM Based on the Sakai-Kasahara Key Construction*, IEEE Proc. Information Theory, vol. 153, no. 1, (2006), pp. 19-26.

[18] T-S Chen, K-H Huang, Y-F Chung – *Digital Multi - Signature Scheme based on the Elliptic Curve Cryptosystem*, J. Comp. Sci & Technol. vol. 19 (2004), no. 4, pp. 570-573.

[19] X. Chen, F. Zang, K. Kim – *A new ID - based group signature scheme from bilinear pairings*, *http* : *//eprint.iacr.org*/2003/100.*pdf*, 2003.

[20] B. Chor, S. Goldwasser, S. Micali, B. Awerbuch – *Verifiable secret sharing and achieving simultaneity in the presence of faults*, Proc. of the 26th IEEE Symposium on the Foundations of Computer Science, (1985), pp. 383-395

[21] Cocks, C. – *an Identity Based Encryption Scheme Based on Quadratic Residues*, Proc. of the Eighth IMA Intern. Conf. on Cryptography and Coding, Cirencester, 17-19 Dec. 2001, pp. 360-363.

[22] J.S. Coron, D. Naccache, J. Stern – *On the security of RSA Padding*, In Advances of Cryptology CRYPTO 99, LNCS 1666, Springer - Verlag, 1999, pp. 1-18.

[23] R. Cramer, R. Gennaro, B. Schoenmakers – *A secure and optimally efficient multiauthority election scheme*, în EUROCRYPT 1997, pp. 103 118.

[24] I.B. Damgard –*A design principle for hash functions*, LNCS, 435 (1990), pp. 516-427.

[25] H. Delfs, H. Knebl – *Introduction to Cryptography, Second edition*, Springer Verlag, 2007.

[26] W. Diffie, M.E. Hellman – *Multiuser cryptographic techniques*, AFIPS Conference Proceedings, 45(1976), 109 – 112

[27] W. Diffie, M.E. Hellman – *New Directions in Cryptography*, IEEE Trans. on Information Theory, vol. IT-22 (1976), pp 644-654

[28] H. Dobbertin – *Cryptanalysis of MD4*, Journal of Cryptology, 11 (1998), pp. 253-271.

[29] T. ElGamal – *A public key cryptosystem and a signature scheme based on discrete algorithms*, IEEE Trans. on Information Theory, 31 (1985), pp. 469-472.

[30] M. J. Farsi – *Digital Cash*, Masters Thesis in Computer Science, Dpt of Mathematics and Computing Science, Goteborg University 1997.

[31] P. Feldman – *A practical scheme for non-interactive verifiable secret sharing*, Proc. of the 28th IEEE Symposium on the Foundations of Computer Science, (1987), pp. 427-437.

[32] N. Ferguson – *Single Term Off-Line Coins*, Advances in Cryptology - EUROCRYPT 93, Springer-Verlag, pp. 318-328.

[33] A. Fujioka, T. Okamoto, K. Ohta – *A practical secret voting scheme for large scale elections*, în J. Seberry şi Y. Zheng, (eds), ASIACRYPT, LNCS 718 (1992), pp. 244251.

[34] E. Fujisaki, T. Okamoto – *Secure Integration of Assymetric and Simmetric Encryption Schemes*, Proc. of Crypto 99, Santa Barbara, CA, August 20-24, 1999, pp. 537-554.

[35] S. Galbraith – *Pairings*, în "Advances in Elliptic Curve Cryptography" ed. T. Blake, G. Seroussi şi N. Smart; London Math. Society, Lecture Notes Series 317 (2005), pp. 183-214.

[36] T. El Gamal – *A public key cryptosystem and a signature scheme based on discrete algorithms*, IEEE Trans on Inf. Theory, 31 (1985), pp. 469-472.

[37] J. Gibson – *Discrete logarithm hash function that is collision free and one way*, IEEE Proceedings-E, 138 (1991), 407-410.

[38] H. Ghodosi, J. Pieprzyk, Safavi-Naini – *Remarks on the multiple assignment secret sharing scheme*, LNCS 1334 (1997), 72-82.

[39] S. Haber, W. Stornetta; How to timestamp a digital document. Journal of Cryptology, 3(1991), 99-111.

[40] E. van Heyst, T.P.Petersen – *How to make efficient fail-stop signatures*, LNCS, 658 (1993), 366-377.

[41] S. Iftene – *A generalisation of Mignottes secret sharing scheme*, în Proc. of the 6th Intern. Symposium on Symbolic and Numeric Algorithms for scientific computing, Timişoara, T. Jebelean, V. Negru, D. Petcu, D. Zaharia (eds), Sept. 2004, pp. 196-201, Mirton Publ. House (2004).

[42] I. Ingermarsson, G. D. Simmons – *A protocol to set up shared secret schemes without assistance of mutually trusted party*, EUROCRYPT 90, LNCS vol. 473, Springer - verlag 1991, pp. 266-282

[43] W. Jackson, K.M. Martin, C. M. O′ Keefe – *Muttualy trusted authority - free secret sharing schemes*, Journal of Cryptology, 10 (4), 1997, pp. 261-289.

[44] E. D. Karnin, J. W. Greene, M. E. Hellman – *On secret sharing systems*, IEEE Trans. on Information Theory 29 (1983), pp. 35-41.

[45] V. Klima – *Tunnels in Hash Functions: MD5 Collisions within a Minute*, Cryptology ePrint Archive, http://eprint.iacr.org, Report 105 (2006).

[46] Klitz, E. – *On the Limitations of the Spread of an IBE-to-PKE Transformation*, Proc of PKC 2006, LNCS 3958, Springer, 2006, pp. 274-289.

[47] H. Krawczyk, M. Bellare, R. Canetti – *HMAC: Keyed - Hashing for Message Authentication*, RFC 2104, 1997.

[48] E. Kranakis – *Primality and Cryptography*, Wiley-Teubner Series in Computer Science (1986).

[49] H. Krawczyk – *Secret sharing made short*, în Advances in Cryptology – CRYPTO 93, D. R. Stinson, ed., LNCS 773 (1994), pp. 136-146.

[50] L. Law, S. Sabett, J. Solinas – *How to make a mint: The Cryptography of Anonymous Electronic Cash*, http://jya.com/nsamint.htm

[51] C.I. Lei, C.I. Fan – *A universal single-authority election system*, IEICE Transactions on Fundamentals E81-A (10) (1998), 2186-2193

[52] Mambo, Masahiro, Keisyke, Usuda, Okamoto – *Proxy signatures: Delegation of the power to sign mesages*, IEICE Trans. Fundamentals, ET9-A (1996), pp. 1338 - 1354.

[53] Martin, L – *Introduction to Identity – Based Encryption*, Artech House, Information Security and Privacy Series, 2008.

[54] T. Matsumoto, Y. Takashima, H. Imai – *On seeking smart public-key distribution systems*, The Trans. of the IECE of Japan, E69 (1986), pp. 99-106.

[55] C. Meadows – *Some threshold schemes without central key distributors*, Congressus Numerantium, 46 (1985), pp. 187-199.

[56] R. J. McEliece, D. Sarwate – *On sharing secrets and Reed-Solomon codes*, Comm. of the ACM 24 (1981), pp. 583-584.

[57] A. Menezes, P. van Oorschot, S. Vanstome – *Handbook of Applied Cryptography*, CRC Press Inc (1997)

[58] R.C. Merkle – *A fast software one-way functions and DES*, LNCS, 435 (1990), pp. 428-446.

[59] M. Mignotte – *How to share a secret*, în Cryptography Proc., Burg Feuerstein 1982, T. Beth, ed., LNCS 149 (1983), pp. 371-375.

[60] C. J. Mitchell, F. Piper, P. Wild – *Digital signatures*, Contemporary Cryptology, The Science of Information Integrity, IEEE Press, (1992), pp. 325-378.

[61] Y.Mu, V. Varadharajan – *Anonymous e-voting over a network*, Proc. of the 14th Annual Computer Security Applications Conference, ASAC8 (1998) 293-299

[62] R. Needham, M. Schroeder – *Using encryption for authentication in large networks of computers.*, Comm. of the ACM 21, 12 (1978), pp. 993 999.

[63] A. M. Odlyzco – *Cryptanalytic attacks on the multiplicative knapsack cryptosystems and on Shamirs fast signature scheme*, IEEE Trans. on Information Theory, IT30 (1984), pp. 594-601.

[64] T. Okamoto – *Receipt - free electronic voting scheme for large scale elction*, Proc. of Workshop on Security Protocols, LNCS 1361 (1997).

[65] T. Okamoto, K. Ohta – *Universal electronic cash*, J. Feigenbaum (Ed.), Advances in cryptology CRYPTO91, LNCS 576, Springer-Verlag (1992).

[66] B. Preneel, R. Govaerts, J. Vandewalle – *Hash functions based on block ciphers: a syntetic approach*, LNCS, 773 (1994), pp. 368-378.

[67] M.O. Rabin – *Efficient dispersal of information for security, load balancing, and fault tolerance*, Journal of ACM, 36(2), 1989, pp. 335-348.

[68] R.L. Rivest – *The MD4 message digest algorithm*, LNCS, 537, (1991), pp. 303-311.

[69] R. L. Rivest – *The MD5 Message Digest Algorithm*, RFC 1321 (1992).

[70] A. Salomaa – *Criptografie cu chei publice*, Ed. Militară, 1994.

[71] B. Schoenmakers – *A simple publicly verifiable secret sharing scheme and its application to electronic voting*, Advanced in Cryptology, LNCS 1666 (1999), pp. 148-164.

[72] A. Shamir – *Identity-Based Cryptosystems and Signature Schemes*, Proc. of CRYPTO 84, Santa Barbara, CA. August 19-22, 1984, pp. 47-53.

[73] A. Shamir – *How to share a secret*, Comm of the ACM 22 (1979), 612-613.

[74] G.J. Simmons – *The prisoners problem and the subliminal channel*, Advances in Cryptology - Crypto, 83 (1984), pp. 51-67.

[75] M. E. Smid, D. K. Branstad – *Response to comments on the $NIST$ proposed digital signature standard*, LNCS, 740 (1993), pp. 76-88.

[76] M. Stadler – *Publicly verifiable secret sharing*, Advances in Cryptology - EUROCRYPT 96, LNCS vol. 1070, Springer verlag (1996), pp. 190-199.

[77] D. R. Stinson – *Decomposition constructions for secret sharing schemes*, IEEE Trans. on Information Theory 40 (1994), pp. 118-125.

[78] D. Stinton – *Cryptographie, theorie et pratique*, Intern. Thompson Publ. France, 1995.

[79] Z. Tan, Z. Liu, C. Tan – *Digital proxy Blind Signature Schemes based on DLP and ECDLP*, MM Research Preprints, 212-217, Academia Sinica, Beijing, no. 21, Dec. 2002.

[80] S. Vaudenay – *A Classical Introduction to Cryptography*, Springer Verlag 2006.

[81] H.C.Williams – *Some public-key criptofunctions as intractable as factorisation*, Cryptologia, 9 (1985), pp. 224-237.

[82] ISO/IEC 9796; Information technology – Security Techniques – Digital Signature Scheme Giving message recovery, Intern. Organisation for Standardisation, Geneva, 1991.

[83] *Secure Hash Standard*, National Bureau of Standards, FIPS Publications 180, 1993.

[84] *SKIPJACK and KEA Algorithm Specifications*, versiunea 2.0, http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf

[85] *Digital signature standard*, National Bureau of Standards, FIPS Publications 186, 1994

Index