Universitatea „Politehnica” Timişoara
Facultatea de Automatică şi Calculatoare

Stadiul actual al dezvoltării sistemelor de securitate pentru rețele de calculatoare de înaltă siguranță

Referat nr. 1

Doctorand:
ing. Valer BOCAN

Conducător ştiințific:
prof. dr. ing. Vladimir CREȚU

Timişoara, 2002

I have not failed. I've just found 10,000 ways that won't work.

Thomas Alva Edison (1847 – 1931)

Ultima versiune a acestui document se poate obține de la adresa *http://www.dataman.ro*
sau scriind autorului la adresa de e-mail *vbocan@dataman.ro*

Cuprins

Capitolul 1

Introducere

Rețelele de calculatoare sunt în general structuri deschise la care se pot conecta un număr mare şi variat de componente. Complexitatea arhitecturală şi distribuția topologică a rețelelor conduce la o lărgire necontrolată a cercului utilizatorilor cu acces nemijlocit la resursele rețelei (fişiere, baze de date, dispozitive periferice, etc.) Putem vorbi despre o *vulnerabilitate* a rețelelor care se manifestă pe două planuri:

- posibilitatea modificării sau distrugerii informațiilor (atac la integritatea fizică);
- posibilitatea folosirii neautorizate a informațiilor; [PATR94]

Se înțelege că această stare de fapt nu poate fi tolerată, proiectarea sistemelor distribuite trebuind să satisfacă unele cerințe fundamentale de fiabilitate, protecție şi securitate. Pe măsură ce rețelele de calculatoare se extind ca număr de resurse conectate şi ca extindere geografică, nevoia de restricționare şi control al accesului creşte.

Sub aspect fizic, resursele unei rețele trebuie protejate într-o manieră adecvată:

- restricționarea accesului la sălile cu echipamente
- protejarea la furt şi distrugere a echipamentelor de comunicație (routere, switch-uri, calculatoare, etc.)
- protejarea căilor fizice de comunicație prin îngroparea în paturi speciale şi plasarea în locuri greu accesibile

Atacurile de tip fizic asupra rețelelor sunt improbabile şi relativ uşor de descoperit. Mai mult, beneficiile atacatorilor sunt minime, având în vedere că de o bună perioadă de timp, valoarea echipamentului este net inferioară informației vehiculată de acesta.

Sub aspect logic, controlul resurselor se face prin asigurarea identității participantului la schimbul de date, denumită generic *autentificare*. De regulă autentificarea se realizează prin ceea ce utilizatorul **ştie** (parolă), prin ce utilizatorul **are** (smart card, cheie), sau prin ce utilizatorul **este** (identificare biometrică, scanare de retină, amprente). [SCHN96]

Lucrarea de față încearcă să facă o prezentare a celor mai importante protocoale de autentificare pe baza unei bogate bibliografii de dată recentă.

Capitolul 1 descrie terminologia folosită în domeniu precum şi mecanismele de securitate implicate în procesul de autentificare.

Capitolul 2 prezintă protocolul Kerberos dezvoltat la Massachusetts Institute of Technology. Se face referire la diferitele versiuni folosite în prezent.

Protocolul SESAME a fost dezvoltat în Europa ca răspuns la Kerberos şi este prezentat în **capitolul 3**.

În **capitolul 4** se prezintă protocolul NetSP care are ca particularitate folosirea unei funcții greu inversabile în locul unui sistem criptografic autentic.

Protocolul SPX a fost creat de Digital Equipment Corporation (DEC) şi este prezentat în **capitolul 5**. Acest protocol foloseşte atât criptografia cu chei publice cât şi cea cu chei secrete.

În Germania s-au pus bazele sistemului TESS, descris în **capitolul 6**. Acesta este un sistem de mecanisme criptografice cooperante.

Capitolul 7 face referire la protocolul SSL pus la punct de Netscape Corporation, iar în **capitolul 8** se prezintă protocolul CHAP.

Capitolul 9 este rezervat concluziilor şi descrierii muncii viitoare cu referire la apărarea împotriva atacurilor de tip DoS – Denial of Service.

1.1. Terminologie

Literatura din domeniul ştiinței calculatoarelor în general şi cea referitoare la securitatea informației în special are un jargon specific. Dată fiind lipsa unui organism lingvistic specializat care să definească precis modalitatea de folosire a termenilor, autorii de texte de specialitate folosesc termenii în moduri deseori contradictorii. În acest capitol ne-am propus prezentarea celor mai importante noțiuni pentru a întrerupe tradiția nefericită amintită.

Termenul *informație* este definit[1] astfel: „cunoştințe comunicate sau recepționate cu privire la un fapt particular sau circumstanță”[2], în general şi „date care pot fi codificate pentru prelucrarea de către un calculator sau un dispozitiv similar”[3], în ştiința calculatoarelor. Definiția este generală, dar serveşte scopului nostru. Shannon dă o definiție formală şi mai precisă în [SHAN48] şi [SHAN49].

Având în vedere definiția anterioară, termenul de tehnologia informației (IT – *information technology*) se referă la orice tehnologie care are legătură cu informația. În particular, IT se referă la stocarea, procesarea şi transmiterea datelor care codifică informația. Similar, termenul de *securitate* IT se referă la chestiuni legate de securitatea informațiilor, în speță securitatea sistemelor şi a comunicației dintre acestea.

- Scopul *securității calculatoarelor* este de a preveni accesul neautorizat la sistemele de calcul şi de a proteja informațiile stocate de distrugeri intenționate, modificare sau deconspirare.
- Scopul *securității comunicațiilor* este de a proteja datele vehiculate într-o rețea de calculatoare sau într-un sistem distribuit. Ca sinonim se foloseşte uneori termenul de *securitatea rețelei.*

[1] Webster's Encyclopedic Unabridged Dictionary of the English Language, Random House Value Publishing, Inc., 1996

[2] Definiție originală în limba engleză: "knowledge communicated or received concerning a particular fact or circumstance"

[3] Definiție originală în limba engleză: "data that can be coded for processing by a computer or similar device"

Termenul de *rețea de calculatoare* se referă la o colecție de sisteme autonome conectate. Două sisteme se numesc conectate dacă sunt capabile să schimbe date între ele printr-o metodă oarecare (rețea, cablu serial / paralel, etc.). În plus, sistemele se numesc autonome dacă între ele nu se poate stabili o relație clară master / slave. Spre exemplu, un sistem cu o unitate de control şi mai multe unități slave nu reprezintă o rețea.

În literatură există o confuzie considerabilă referitoare la ce diferențiază un sistem distribuit de o rețea de calculatoare. Lamport [LAMP78] spune că un *sistem distribuit* este o colecție de procese separate spațial şi care comunică între ele prin schimb de mesaje[1]. Tot în viziunea lui Lamport, în cazul unui sistem distribuit întârzierea introdusă de comunicația cu mesaje nu este neglijabilă în raport cu evenimentele din cadrul unui proces.

În 1988, Tanenbaum arăta distincția cheie dintre o rețea de calculatoare şi un sistem distribuit. În principiu, în cazul unui sistem distribuit existența sistemelor distribuite autonome este transparentă pentru utilizator. În principiu, utilizatorul poate rula un proces iar sistemul de operare poate alege cel mai potrivit procesor pe care să ruleze, iar rezultatele să le direcționeze la locul potrivit. Cu alte cuvinte, utilizatorul nu trebuie să fie conştient de faptul că în scenariu sunt mai multe procesoare, ci sistemul distribuit arată ca un procesor virtual unic. De remarcat că în acest caz, diferența între o rețea de calculatoare şi un sistem distribuit rezidă în software în general şi în sistemul de operare în special, hardware-ul jucând un rol mai puțin important.

Joint Technical Committee 1 (JTC1) care aparține de ISO[2] şi IEC[3], foloseşte termenul de *partener* pentru a se referi la o persoană sau entitate înregistrată şi autentificabilă de către o rețea de calculatoare sau sistem distribuit. Utilizatorii, host-urile şi procesele sunt considerate parteneri.

- Un *utilizator* este responsabil pentru acțiunile sale în cadrul unei rețele.
- Un *host* este o entitate adresabilă în cadrul unei rețele sau unui sistem distribuit. Adresarea se face prin nume sau adresă.
- Un *proces* este o instanțiere a unui program care rulează pe un host anume. Modelul *client / server* se foloseşte în mod curent pentru a distinge un proces client de un proces server.
    - Un *proces client* cere şi în cele din urmă obține un serviciu de rețea, pe când
    - Un *proces server* produce serviciul. În accepțiunea de față serviciul se referă la o funcționalitate abstractă, iar serverul este în mod tipic un fir de execuție care se specializează în această funcționalitate.

Modelul client / server este potrivit pentru proiectarea sistemelor distribuite şi a aplicațiilor corespunzătoare. În forma cea mai simplă, un serviciu este deservit de un sistem unic. Adeseori însă, solicitarea serviciului poate fi atât de mare încât un sistem nu mai face față. Mai mult, se pune uneori problema disponibilității continue a unui serviciu (server web, supraveghere procese vitale, etc.) în condițiile în care hardware-ul este susceptibil de defectare după un număr de ore. În aceste cazuri serviciul poate fi replicat, adică implementat de un sistem distribuit care în acest fel prezintă atât o capacitate de deservire mai mare cât şi o toleranță la defecte mai mare. Studiul şi dezvoltarea tehnicilor de replicare securizată a serviciilor a devenit un domeniu vast de cercetare. [GONG93], [REBI94]

[1] Într-o notă plină de umor, Lamport a definit un sistem distribuit ca "un sistem care te împiedică să-ți termini treaba când un calculator pe care nu l-ai văzut niciodată, cade"

[2] International Organization for Standardization

[3] International Electrotechnical Committee

ISO foloseşte termenul *standard* pentru un document convenit care conține specificații tehnice sau alte criterii precise utilizate consistent ca reguli sau definiții de caracteristici pentru a asigura că materialele, produsele, procesele şi serviciile sunt potrivite pentru scopul propus. În consecință, un *standard de sistem deschis* specifică un *sistem deschis* care permite fabricanților să construiască produse conforme.

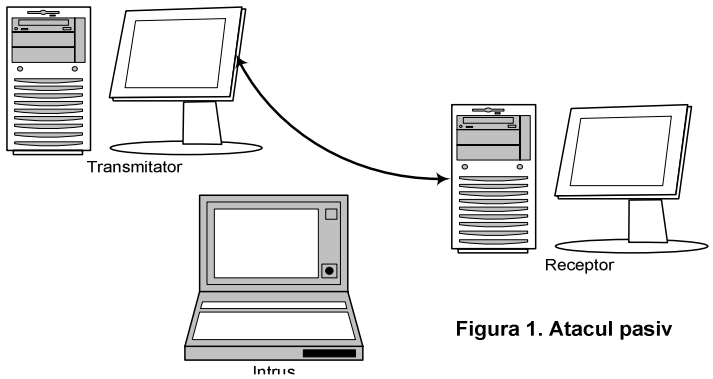

Figura 1. Atacul pasiv

În general, termenul de *vulnerabilitate* se referă la o slăbiciune care poate fi exploatată pentru a viola un sistem sau informațiile pe care acesta le conține. Termenul de *amenințare* se referă la o circumstanță, condiție sau eveniment care are potențialul de a viola securitatea sau de a cauza stricăciuni sistemului. Rețelele de calculatoare sunt sisteme distribuite susceptibile la o varietate de amenințări proferate fie de intruşi[1] fie de utilizatori legitimi, de obicei mai periculoşi decât cei din exterior pentru că au acces la informații nedivulgate în mod normal celor din afară.

În cadrul rețelelor de calculatoare şi a sistemelor distribuite se disting mai multe tipuri de compromitere a bunei funcționări, astfel:

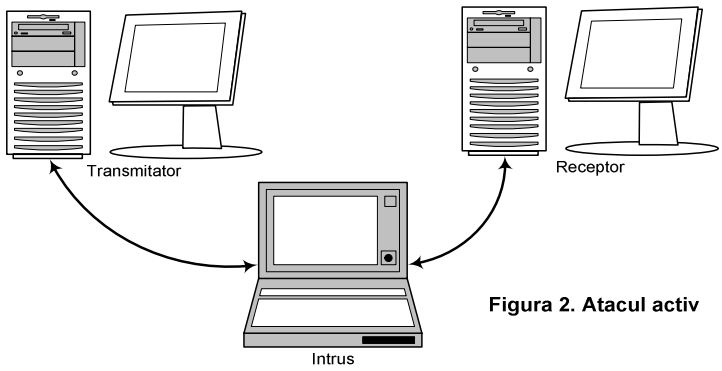

Figura 2. Atacul activ

Compromiterea host-ului ca urmare a subminării sale directe. Rezultatele subminării pot fi de la o simplă modificare a stării proceselor până la controlul total asupra host-ului.

- *Compromiterea comunicării* ca urmare a subminării liniei de comunicație din sistem.

Securitatea căilor de comunicație este foarte importantă deoarece reprezintă un punct foarte vulnerabil în lanțul de comunicație. Putem distinge două tipuri de compromitere a comunicației şi anume prin atac pasiv şi prin atac activ.

Atacul pasiv amenință confidențialitatea datelor transmise, situație ilustrată în figura 1. Datele de la transmițător (stânga) sunt observate de un intrus (mijloc).

[1] Termenul *hacker* este deseori folosit pentru a descrie rău-făcători care pătrund în sisteme.

Fezabilitatea unui astfel de atac depinde de tipul de mediu prin care are loc comunicarea. Astfel, liniile de comunicație mobile sunt relativ uşor de ascultat, pe când liniile fizice necesită acces fizic. Conductorii optici sunt şi ei susceptibili la ascultare, dar cu un efort tehnologic substanțial.

Este interesant de remarcat că atacul pasiv nu se realizează exclusiv la nivelul căilor de comunicație hardware. Pe piață se găsesc pachete software pentru monitorizarea traficului în rețea în special în scop de management. Aceleaşi pachete se pot folosi pentru capturarea parolelor necriptate din rețea.

Atacul activ amenință integritatea şi / sau disponibilitatea datelor, situație ilustrată în figura 2. În acest caz, intrusul poate observa şi controla fluxul de informații, putându-l modifica, extinde, şterge şi re-trimite informații. În plus, intrusul poate inunda receptorul cu informații false pentru a cauza o întrerupere a comunicației, situație denumită uzual Denial of Service.

Adesea, atacul poate să fie atât activ cât şi pasiv. Spre exemplu, prin atac pasiv se obțin parolele de acces la un anumit serviciu, apoi prin atac activ se face autentificarea. Este deci clar că autentificarea prin parole nu este suficientă; despre aceste chestiuni vorbim în lucrarea de față.

1.2. Arhitectura OSI a securității

Arhitectura OSI a securității este o descriere generală a serviciilor de securitate şi a mecanismelor înrudite şi discută relațiile dintre acestea şi cum corespund unei arhitecturi de rețea.

1.2.1. Servicii de securitate

Arhitectura OSI distinge cinci clase de servicii de securitate: autentificarea, controlul accesului, confidențialitatea datelor, integritatea datelor şi non-acceptul.

Serviciile de autentificare permit autentificarea entităților participante la comunicație sau a originii datelor.

- *Serviciul de autentificare a entităților similare* verifică faptul că o entitate dintr-o asociație aparține acesteia şi nu încarcă să-şi falsifice identitatea sau să retransmită copii neautorizate ale identităților din trecut. Acest fel de autentificare se face în faza de stabilire a conexiunilor, şi ocazional în timpul fazei de transfer de date.
- *Serviciul de autentificare a originii datelor* verifică sursele de date, dar nu poate oferi protecție împotriva duplicării sau modificării datelor, în acest caz folosindu-se serviciul de la punctul anterior. Serviciul de autentificare a originii datelor se foloseşte în timpul fazei de transfer de date.

Serviciul de control al accesului protejează împotriva folosirii neautorizate a resurselor sistemelor. Acest tip de serviciu conlucrează strâns cu serviciile de autentificare, deoarece pentru a media accesul la o resursă, utilizatorul trebuie să îşi confirme identitatea.

Serviciul de confidențialitate a datelor protejează sistemul de divulgare neautorizată a datelor.

- *Serviciul de confidențialitate a conexiunii* furnizează confidențialitatea tuturor informațiilor transmise într-o conexiune.
- *Serviciul de confidențialitate fără conexiune* furnizează confidențialitatea unităților de informație.

- *Serviciul de confidențialitate cu câmp selectiv* furnizează confidențialitatea câmpurilor specifice dintr-un flux pe durata unei conexiuni sau a unei unități de informație.
- *Serviciul de confidențialitate a fluxului de trafic* furnizează protecție împotriva analizei traficului.

Serviciile de integritate a datelor protejează datele de modificări neautorizate.

- *Serviciul de integritate a serviciului cu recuperare* furnizează integritatea datelor într-o conexiune. Pierderea integrității este recuperată dacă acest lucru este posibil.
- *Serviciul de integritate a serviciului fără recuperare*, ca şi în cazul precedent furnizează integritatea datelor într-o conexiune. Pierderea integrității nu se poate recupera.
- *Serviciul de integritate a unui câmp desemnat* furnizează integritatea unor câmpuri specifice în cadrul conexiunii.
- *Serviciul de integritate fără conexiune* furnizează integritatea unor unități de date separate.
- *Serviciul de integritate a unui câmp desemnat fără conexiune* furnizează integritatea unor câmpuri specifice din unități de date separate.

Serviciul non-negare furnizează protecție împotriva acceptării ulterioare a trimiterii sau recepționării unui mesaj. Putem distinge două feluri de servicii:

- *Serviciul non-negare cu dovada originii*, oferă receptorului o dovadă a originii mesajului
- *Serviciul non-negare cu dovada livrării*, oferă transmițătorului o dovadă a recepționării mesajului.

Serviciile non-negare devin din ce în ce mai importante în contextul actual de schimb electronic de date şi comerțul electronic pe Internet.

1.2.2. Mecanisme de securitate

Arhitectura OSI de securitate distinge între mecanisme specifice şi mecanisme generale de securitate.

Mecanisme specifice de securitate

OSI enumeră opt mecanisme specifice de securitate:

Mecanisme specifice de securitate (OSI)
1. Codificarea
2. Semnături digitale
3. Controlul accesului
4. Integritatea datelor
5. Schimbul de autentificare
6. Completarea traficului
7. Controlul rutărilor
8. Notarizarea

1. *Codificarea* se foloseşte pentru a proteja confidențialitatea unităților de informație şi deseori se foloseşte complementar cu alte mecanisme. Tehnicile de criptografie sunt prezentate în secțiunile următoare.

2. *Semnăturile digitale* se folosesc în analogie electronică la semnăturile de mână, pentru documente electronice. Similar cu corespondentul real, semnăturile electronice nu trebuie să se poată falsifica, destinatarul să o poată verifica şi emitentul să nu o poată nega mai târziu.
3. *Controlul accesului* forţează aplicarea drepturilor de acces prin utilizarea identităţii autentificate a părţilor. Dacă una dintre părţi încearcă să folosească o resursă neautorizată, serviciul blochează această tentativă şi opţional poate genera o alarmă care să apară în auditul de securitate.
4. *Integritatea datelor* este un mecanism care protejează unităţile de informaţie sau câmpuri din acestea. În general, acest tip de mecanism nu protejează împotriva atacurilor de tip replay.
5. *Schimbul de autentificare* verifică identitatea părţilor. În conformitate cu ITU X.509, un mecanism de autentificare se numeşte puternic dacă se bazează pe tehnici criptografice pentru a codifica schimbul de mesaje.
6. *Completarea traficului* protejează împotriva analizei datelor. Acest mecanism maschează datele reale intercalând date fără valoare, fiind efectiv în conjuncţie cu un mecanism de confidenţialitate.
7. *Controlul rutărilor* se foloseşte pentru alegerea dinamică sau într-un mod predeterminat a rutelor pentru transmiterea informaţiei. Sistemele de comunicaţie pot comanda modificarea rutei la descoperirea unui atac pasiv sau activ. În mod asemănător, anumite informaţii cu etichete de securitate speciale pot fi rutate pe căi speciale.
8. *Notarizarea* se referă la asigurarea transmiterii unor proprietăţi ale informaţiei, cum ar fi integritatea, originea, timpul şi destinaţia. Asigurarea se face de către un terţ într-o manieră verificabilă.

Mecanisme generale de securitate

Mecanismele generale de securitate nu sunt specifice nici unui serviciu, iar unele se confundă cu managementul securităţii. Importanţa mecanismelor generale de securitate este în strânsă legătură cu nivelul de securitate cerut. OSI enumeră opt mecanisme specifice de securitate:

Mecanisme generale de securitate (OSI)
1. Funcţionalitate credibilă
2. Etichete de securitate
3. Detectarea evenimentelor
4. Auditul de securitate
5. Recuperarea de siguranţă

1. Conceptul general de *funcţionalitate credibilă* se poate folosi fie pentru a extinde fie pentru a defini eficienţa altor mecanisme de securitate. Orice funcţionalitate care oferă în mod direct sau mijloceşte accesul la mecanismele de securitate trebuie să fie de încredere.
2. Resursele sistemului pot avea *etichete de securitate* asociate cu acestea. Deseori este necesar ca etichetele să fie transmise împreună cu datele în tranzit, sub forma unor informaţii adiţionale asociate datelor de transferat sau sub formă implicită dată de context, cum ar fi rută sau sursă.
3. *Detectarea evenimentelor* se poate folosi pentru a detecta violarea de securitate.

4. *Auditul de securitate* se referă la o examinare independentă a jurnalelor sistemului pentru a testa bonitatea controalelor de sistem, pentru a se asigura alinierea la politica şi practicile de securitate, pentru a detecta breşele de securitate şi pentru a recomanda schimbări în control, politică şi proceduri.
5. *Recuperarea de siguranță* se ocupă cu manipularea cererilor mecanismelor de securitate şi aplică un set de reguli stabilit apriori.

Arhitectura OSI a securității nu urmăreşte să rezolve o anume problemă a rețelei, ci stabileşte un cadru şi o terminologie care poate fi folosită pentru a descrie şi discuta problemele şi soluțiile lor.

1.2.3. Integrarea criptografiei în arhitectura OSI

Datele codificate într-un fel anume pot trece prin mai multe noduri ale rețelei până să ajungă la destinație. Putem identifica mai multe posibilități de integrare ale punctelor de cifrare în arhitectura unei rețele. [PATR94]

1. *Cifrarea la nivel de legătură de date* presupune codificarea şi decodificarea mesajului la fiecare nod de comunicație. Fiecare nod are propria cheie şi posibil propriul algoritm de criptare. În acest caz, datele sunt protejate doar pe mediul fizic de transmitere.

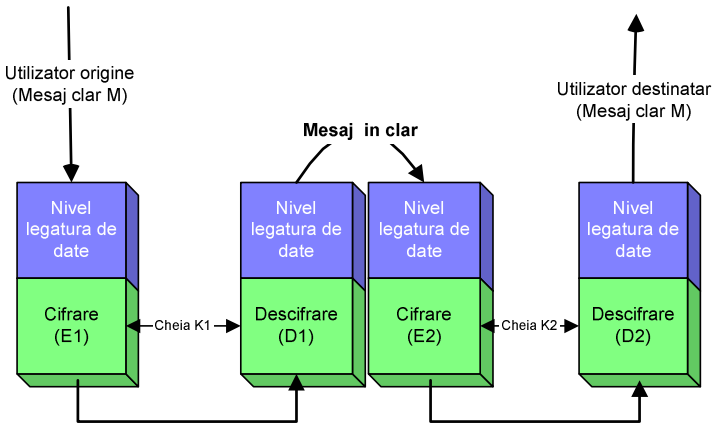

Figura 3. Cifrarea la nivel de legatura de date

2. *Cifrarea la nivel de rețea* este similară cu cea la nivel de legătură, restrângându-se circulația în clar într-un „modul de securitate”. Fiecare legătură utilizează o cheie unică, iar trecerea de la o cheie la alta se face în modulul de securitate.

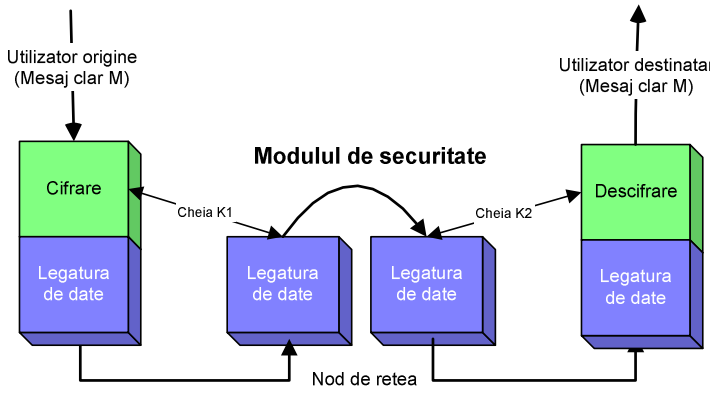

Figura 4. Cifrarea la nivel de retea

3. *Cifrarea la nivel superior* este o soluție ce pare mai naturală într-un sistem orientat prin „sesiuni”, unde schimburile sunt relativ lungi. Toate informațiile care tranzitează prin rețea sunt cifrate în general cu aceeaşi cheie pe durata între sesiuni între entitățile sursă şi destinație ale schimbului. Cifrarea poate fi implementată prin protocoale la nivel de sesiune sau prezentare. Această soluție are avantaje şi dezavantaje. La cifrarea pe legătura de date, utilizatorii au nevoie doar de o singură cheie când comunică cu sistemul local, restul cheilor fiind gestionate automat de sistem. În acest caz însă, utilizatorul trebuie să comunice cu alți utilizatori folosind chei separate. În plus, se pune problema distribuției cheilor prin protocoale speciale în vederea stabilirii unor conexiuni sigure.

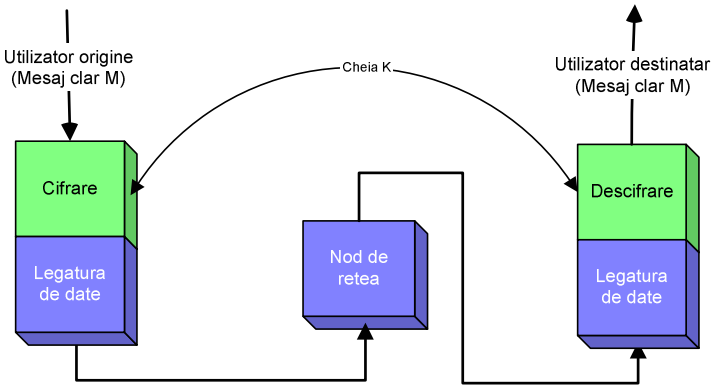

Figura 5. Cifrarea la nivel superior

Cifrarea la nivel superior oferă o securitate sporită, deoarece datele circulă codificat pe tot parcursul, dar adresa destinației trebuie păstrată în clar, ceea ce ar putea fi o posibilă verigă slabă. La cifrarea la nivel de legătură, datele pot fi observate în clar la fiecare nod. Adresa destinației este secretă pe parcursul transmisiei, ceea ce face mai dificilă prognoza rutei.

4. În cazul unor aplicații care necesită un nivel mai ridicat de securitate, pe lângă păstrarea confidențialității datelor este necesar a se păstra şi confidențialitatea destinației. Fiecare pachet are asociat un câmp de adresă cifrat cu cheia nodului anterior. Această schemă se numeşte *sistem de cifrare la nivel superior în combinație cu cifrarea pe legătură a adresei destinație.[PATR94]*

1.3. Autentificarea şi distribuția cheilor

Procesul de autentificare a părților participante la o comunicație presupune folosirea tehnicilor criptografice pentru a împiedica un posibil atac. La rândul lor, tehnicile criptografice folosesc chei pentru codificare / decodificare, ce trebuie distribuite părților. În acest capitol vom face o descriere a celor mai importante concepte.

1.3.1. Tehnici criptografice

În general, *criptologia*[1] se referă la ştiința comunicării secretizate. Aceasta cuprinde atât criptografia cât şi criptanaliza.

Funcții de dispersie cu sens unic

Funcțiile cu sens unic sunt foarte importante în criptologie. Pe înțelesul tuturor, funcțiile cu sens unic sunt uşor de calculat dar greu de inversat. Matematic, acest lucru se spune astfel:

Funcția $f : A \rightarrow B$ este o funcție cu sens unic dacă f(x) se calculează uşor pentru toate valorile $x \in A$, dar este nefezabil din punct de vedere al timpului de calcul când se dă $y \in f(A) = B$ să se găsească un $x \in A$ astfel încât $f(x) = y$. Această definiție nu este precisă în sensul matematic deoarece nu defineşte termenii „uşor" şi „nefezabil". Este important de arătat că existența funcțiilor cu sens unic este o presupunere care nu a fost încă dovedită.

Nu este în general necesar ca o funcție cu sens unic să fie injectivă, valori distincte inițiale putând să conducă la acelaşi rezultat. O funcție cu sens unic $f : A \rightarrow B$ pentru care $|B| << |A|$ se mai numeşte *funcție de dispersie cu sens unic*. Dacă în plus față de aceste condiții este greu să se obțină $x_1, x_2 \in A$ distincte astfel încât $f(x_1) = f(x_2)$, atunci f este o *funcție de dispersie cu sens unic rezistentă la coliziuni*. Exemple de astfel de funcții sunt MD4 (Rivest, 1992), MD5 (Rivest şi Dusse, 1992) şi SHS (Secure Hash Standard) propus de U.S. National Institute of Standards and Technology (NIST).

Criptografia cu chei secrete

În *criptografia cu chei secrete*, o cheie este aleasă între participanți şi este folosită pentru criptarea şi decriptarea mesajelor. De aceea, criptografia cu chei secrete se mai numeşte *criptografie simetrică.*

Criptografia cu chei secrete se folosesc de mii de ani într-o varietate de forme. Implementările moderne iau de obicei forma unui algoritm care se execută în hardware, firmware sau software. Majoritatea de astfel de criptosisteme se bazează pe operații simple cum ar fi permutări şi transpoziții, dar este extrem de dificil să se creeze un sistem rezistent la atacuri, în ciuda faptului că istoria eşecurilor este lungă şi bogată.

Exemple de sisteme de criptografie cu chei secrete utilizate pe scară largă în lume sunt: AES[2], DES[3], triple DES, IDEA[4], RC2, RC4, RC5[5]. Algoritmul FEAL[6] nu se mai foloseşte datorită vulnerabilității la criptanaliză diferențială.

[1] Cuvântul „criptologie" vine din limba greacă din cuvintele kryptos (ascuns) şi logos (cuvânt).

[2] Advanced Encryption Standard, cunoscut sub numele de Rijndael, adoptat de NIST ca standard înlocuitor pentru DES în anul 2001.

[3] Data Encryption Standard, adoptat de NIST în anul 1977.

[4] International Data Encryption Algorithm, 1992

[5] Rivest, 1995

[6] Fast Encryption Algorithm

Criptografia cu chei publice

Ideea funcțiilor cu sens unic a dus la inventarea *criptografiei cu chei publice* de către Diffie şi Hellman în 1976. Din punct de vedere practic, criptografia cu chei publice presupune existența unei perechi de chei legate matematic. Una dintre ele se numeşte *cheie publică* şi trebuie publicată – fără a afecta securitatea sistemului – iar cealaltă este *cheia privată* care nu trebuie să fie deconspirată în nici un fel. [DIFF88] Derivarea cheilor una dintr-alta este nefezabilă din punct de vedere computațional. Cel mai folosit sistem cu chei publice este RSA, inventat în 1978 de Rivest, Shamir şi Adelman la Massachusetts Institute of Technology (MIT).

Aplicarea criptografiei cu chei publice necesită un cadru de autentificare care leagă cheia publică a utilizatorului de identitatea sa. Certificatul cu cheie publică este o dovadă certificată de o *autoritate de certificare* (CA – certificate authority). Folosirea CA evită verificarea de către utilizatorii individuali a cheilor publice a altor utilizatori.

Criptografia cu chei publice este mai convenabilă teoretic decât criptografia cu chei secrete deoarece părțile nu mai trebuie să fie în posesia aceleiaşi chei. Astfel, este necesară o schemă de distribuție a cheilor mult mai simplă.

Totuşi, criptografia cu chei publice necesită în general operații matematice dificile pentru procesoare mici. Spre exemplu, performanța cardurilor inteligente nu este suficient de mare pentru a permite utilizarea criptografiei cu chei publice, dar este interesant de văzut cum calculatorul gazdă poate prelua o parte din sarcini până la atingerea unui nivel de performanță satisfăcător. O alternativă este folosirea criptografiei cu chei publice ca mecanism de distribuție a cheilor pentru un criptosistem cu chei secrete. [OPPL96]

1.3.2. Autentificarea

În general, *autentificarea* se referă la procesul de verificare a identității unei părți. Autentificarea rezultă în autenticitate, însemnând că partea care verifică (*verificatorul*) poate fi sigur că partea verificată (*verificatul)* este cel care spune că este. Uzual, aşa cum am expus în introducere, tehnicile de autentificare se împart în trei categorii fundamentale:

- *Autentificare prin cunoştințe* (ceva ce utilizatorul ştie: coduri PIN, coduri de tranzacție, parole)
- *Autentificare prin posesie* (ceva ce utilizatorul are: chei, carduri de identificare sau alt fel de dispozitive fizice)
- *Autentificare prin proprietăți* (identificarea biometrică a utilizatorului cum ar fi identificarea feței, imagini ale retinei, şabloane vocale, amprente)

În tehnica de calcul actuală autentificarea preponderentă este cea prin cunoştințe.

Autentificarea bazată pe parole

În cele mai multe sisteme distribuite şi rețele de calculatoare, protecția resurselor se realizează prin login direct folosind parole, cu transmiterea în clar a acestora. Această autentificare are mai multe inconveniente, din care amintim doar câteva:

- Utilizatorii tind să selecteze parole neuniform distribuite. Această problemă este binecunoscută şi nu este neapărat legată de rețele de calculatoare şi sisteme distribuite. [FELD90], [KLEI90]
- Nu este convenabil pentru un utilizator care are mai multe conturi pe host-uri diferite să îşi amintească parola pentru fiecare, şi de asemenea să o introducă la fiecare schimbare a host-ului. În schimb, utilizatorul va alege să fie recunoscut de rețea ca întreg şi nu de host-urile individuale.
- Transmisia parolei este expusă la captură pasivă.

În principal datorită acestui ultim punct, autentificarea bazată pe parolă nu este potrivită în rețele de calculatoare şi sisteme distribuite. Parolele trimise prin rețea sunt foarte uşor compromise şi folosite ulterior pentru impersonarea utilizatorului.

În unele situații este chiar supărător că se foloseşte autentificarea prin parolă. Spre exemplu, în Statele Unite ale Americii telefoanele mobile folosesc ca parolă internă la efectuarea unui apel chiar numărul telefonului respectiv pentru ca centrala să poată factura fiecare apel în mod corect. Este evidentă că un atacator poate impersona uşor apelul şi poate efectua convorbiri în contul altei persoane.

Autentificarea bazată pe adresă

O alternativă – nu neapărat mai sigură – la autentificarea prin parolă este autentificarea prin adresă. Aceasta presupune că identitatea unei surse se poate deduce din adresa acesteia conținută în pachete. Ideea de bază este că fiecare host memorează identitatea celorlalte host-uri care au acces la resursele sale. În UNIX, fiecare host are un fişier numit */etc/hosts.equiv*. Utilizatorii cu acelaşi cont pe ambele sisteme pot folosi utilitarele „r" fără a specifica vreo parolă.

Ideea de host-uri credibile nu este o soluție la problema autentificării în rețele de calculatoare. De fapt, acest tip de autentificare chiar pune probleme mai mari din punct de vedere al securității. Dacă un atacator reuşeşte să intre pe contul unui utilizator dintr-un sistem, securitatea este compromisă pe toate sistemele care au încredere în acel sistem. În plus, administratorul de sistem nu poate da drepturi preferențiale unor utilizatori.

În funcție de mediul concret, autentificarea prin adresă este chiar mai puțin sigură decât cea bazată pe parolă. Avantajul îl constituie însă comoditatea în folosire şi de aceea multe sisteme au ales să o implementeze.

Autentificarea criptografică

Ideea din spatele autentificării criptografice este că un A îşi dovedeşte identitatea către B prin efectuarea unei operații criptografice asupra unei entități cunoscute de ambii participanți sau oferită de B. Operația criptografică efectuată de A se bazează pe o cheie criptografică. Aceasta poate fi fie o cheie secretă sau o cheie privată dintr-un sistem cu chei asimetrice.

În general, autentificarea criptografică este mai sigură decât autentificarea bazată pe parolă sau pe adresă. În schimb, noile tehnici bazate pe dovezi *zero-knowledge* pot oferi mecanisme de autentificare chiar mai puternice. [SHAM87, QUIS90] Aceste tehnici necesită calcule matematice destul de complexe dar prezintă mai multe facilități atractive pentru autentificare. În primul rând, permit părții ce se autentifică să dovedească că ştie secretul fără a transfera efectiv informația către verificator. În al doilea rând, multe dintre schemele propuse până acum folosesc aceleaşi informații publice, evitându-se astfel problema distribuției cheilor care apare în cazul mecanismelor ce folosesc DES şi RSA.

În ciuda aparentei simplități, proiectarea sistemelor reale este foarte dificilă. O serie de protocoale publicate au prezentat erori de securitate substanțiale sau subtile. În timpul ultimei decade, eforturile de cercetare s-au concentrat în crearea de utilitare pentru dezvoltarea protocoalelor de autentificare şi distribuție a cheilor cu o anumită asigurare formală a securității. Realizările cele mai notabile sunt logica BAN [BURR89] şi logica GNY [GONG90]. În loc de a produce protocoale specifice, aceste metodologii se utilizează pentru a verifica un set de afirmații presupus adevărate.

1.3.3. Distribuția cheilor

Cele mai multe dintre serviciile de securitate enumerate de arhitectura OSI a securității se bazează pe mecanisme criptografice, iar folosirea acestora necesită un management al cheilor corespunzător. Conform OSI, *managementul cheilor* se ocupă cu „generarea, stocarea, distribuția, ştergerea, arhivarea şi aplicarea cheilor în conformitate cu politica de securitate" (ISO/IEC, 1989). Managementul cheilor se efectuează cu protocoale şi multe dintre proprietățile importante ale acestora nu au nici o legătură cu protocoalele criptografice folosite ci mai degrabă cu structura mesajelor schimbate. Drept urmare, scurgerile de securitate şi vulnerabilitățile nu provin de la algoritmii criptografici slabi ci mai degrabă de la greşeli în proiectarea protocoalelor de la nivelurile mai înalte.

Grupul de lucru 802.10 a *Institute of Electrical and Electronic Engineers* (IEEE) a fost format în mai 1988 pentru a discuta nevoile de securitate a rețelelor locale şi metropolitane. Grupul este sponsorizat *de IEEE Technical Committee on Computer Communications* şi *de IEEE Technical Committee on Security and Privacy*. Lucrul a început în mai 1989 iar rezultatul este standardul IEEE 802.10 care suportă trei clase de tehnici de distribuție a cheilor şi anume distribuția manuală, distribuția bazată pe centru şi distribuția bazată pe certificate. În această lucrare vom adopta aceeaşi clasificare.

Distribuția manuală a cheilor

Distribuția manuală a cheilor foloseşte metode de distribuție off-line pentru a distribui cheile între perechi sau la mai mulți participanți. Distribuția manuală în general este dificilă şi are probleme de scalabilitate, în plus nu oferă o autentificare alta decât cea oferită în mod implicit. De aceea, siguranța metodelor de distribuție off-line este extrem de importantă.

În multe cazuri, distribuția manuală a cheilor este necesară doar o singură dată pentru un utilizator anume. Distribuția materialelor adiționale se face folosind cheia distribuită manual ca o cheie de cifrare a cheilor (KEK – Key Encryption Key). Materialele astfel cifrate se distribuie apoi prin orice canal convenabil.

Distribuția manuală este adesea cea mai eficientă metodă pentru distribuția cheilor de grup, în special în cazul grupurilor mari.

Distribuția bazată pe centru

Tehnicile de distribuție bazate pe centru se folosesc pentru a distribui chei între două sau mai multe părți prin intermediul unui terț credibil. Partea terță poate fi:

- Un centru de distribuție a cheilor (KDC – Key Distribution Center)
- Un centru de translație a cheilor (KTC – Key Translation Center)

Distribuția bazată pe centru acoperă atât KDC-urile cât şi KTC-urile. Acestea depind de KEK pentru a furniza confidențialitatea şi protecția integrității cheilor distribuite.

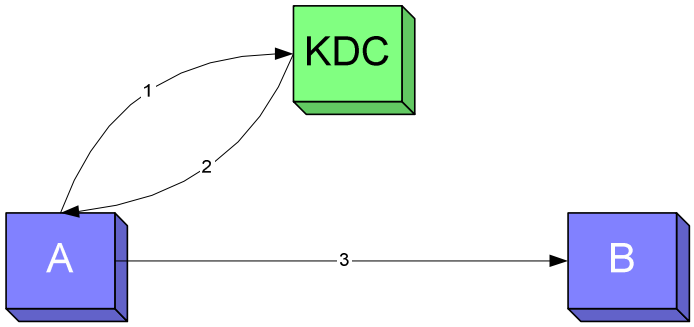

Figura 6. Modelul "trage" pentru distributia bazata pe centru

Cele mai multe metode de distribuție a cheilor au fost proiectate având în vedere scenarii şi aplicații specifice. Spre exemplu, orice schemă care se bazează pe amprente de timp favorizează mediul local, unde toți utilizatorii au acces la un server de timp credibil. Cerința de a avea ceasuri sincronizate în rețele mari nu este absurdă, dar este cu siguranță dificil de asigurat. Mai important este că schemele existente fac presupuneri despre configurația rețelei şi modelele de conectivitate. Spre exemplu, acestea pot cere o anumită paradigmă în contactarea unui server credibil sau KDC. Când A necesită o cheie pentru a comunica cu B, Kerberos – spre exemplu – cere ca A să obțină o cheie de la KDC înainte de a comunica cu B. Această paradigmă se mai numeşte *modelul „trage"*, ilustrat în figura 6.

În contrast, în aceeaşi situație, standardul american pentru managementul cheilor instituțiilor financiare (ANSI X9.17), cere ca A să contacteze mai întâi pe B, apoi B obține cheia necesară de la KDC. Această paradigmă se numeşte *modelul „împinge"*, ilustrat în figura 7.

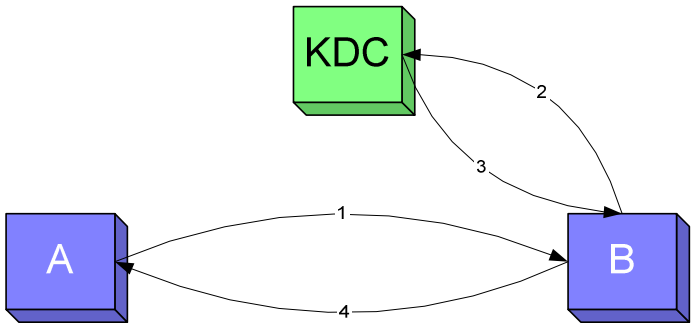

Figura 7. Modelul "impinge" pentru distributia bazata pe centru

Este important de remarcat că nici unul dintre aceste modele nu este superior față de celălalt, fiecare fiind potrivit în mediul său. În rețele locale, pentru care Kerberos a fost proiectat, cerința ca clienții să obțină cheile are sens deoarece distribuie efortul care altfel ar fi preluat de doar câteva servere. Într-o rețea mare, se adoptă metoda opusă deoarece tipic sunt mult mai mulți clienți decât servere şi KDC-urile se află mai apropiate de acestea. În aceste condiții, costul conectivității între clienți şi KDC necesitată de Kerberos devine prohibitiv.

Există şi posibilitatea de a combina cele două metode, ca în figura 8.

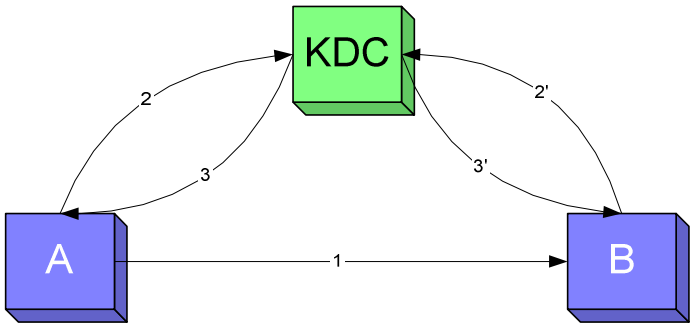

Figura 8. Modelul mixt pentru distributia bazata pe centru

Distribuția bazată pe certificat

Tehnicile de distribuție bazate pe certificat se pot utiliza pentru stabilirea perechilor de chei criptografice. Distingem două tipuri de astfel de tehnici:

- Criptosistem cu chei publice folosit pentru a cripta o cheie generată local pentru a o proteja în timpul transferului către o entitate de management a cheilor. Această metodă se numeşte *transferul cheilor*.
- O cheie criptografică este generată în mod cooperativ atât la entitatea locală cât şi la cea îndepărtată. Această metodă se numeşte *schimb de chei* sau *convenire asupra cheilor*.

În general, distribuția bazată pe certificat nu se poate utiliza pentru stabilirea cheilor într-un mediu multicast. Odată stabilite cheile perechi, acestea se pot folosi totuşi pentru distribuirea cheilor multicast.

Recomandarea X.509 a ITU-T descrie schema distribuției bazată pe certificat, unde o *autoritate de certificare (CA)* autentifică cheile principale. CA-uri diferite se pot autentifica reciproc, rezultând un graf conectat de CA-uri. Punctul inițial de credibilitate pentru un utilizator este CA-ul care l-a înregistrat. [OPPL96]

Capitolul 2

Protocolul Kerberos

În acest capitol ne concentrăm asupra sistemului de autentificare şi distribuţie a cheilor Kerberos. În secţiunea 2.1 amintim dezvoltarea sistemului iar în secţiunea 2.2 arhitectura acestuia. Secţiunea 2.3 descrie protocoalele criptografice implementate de Kerberos, iar în secţiunea 2.4 discutăm anumite extensii şi aplicaţii noi.

2.1. Dezvoltare

Sistemul de autentificare şi distribuţie a cheilor *Kerberos*[1] [STEI88, SCHI94, KOHL93] a fost dezvoltat la Massachusetts Institute of Technology (MIT) pentru a proteja serviciile de reţea oferite de proiectul Athena [CHAM90, CHAM91]. Scopul Kerberos era să extindă noţiunea de autentificare, autorizare şi contabilizare a mediului MIT. În conformitate cu planul tehnic al proiectului Athena, mediul consta în principal din:

- Staţii de lucru publice şi private
  - Staţii de lucru publice în locuri cu securitate fizică minimă sau deloc
  - Staţiile de lucru private sunt sub controlul fizic şi administrativ a indivizilor fără responsabilitate faţă de administraţia centrală a reţelei
- O reţea în campus compusă din multiple LAN-uri de diferite tipuri conectate la un backbone. LAN-urile sunt dispersate spaţial şi vulnerabile la diferite atacuri, pe când backbone-ul este securizat fizic într-o anumită măsură.
- Servere centrale sunt situate în camere cu acces restricţionat şi rulează software care se presupune că nu conţine cod ostil. Unele dintre aceste

[1] În mitologia greacă, cerberul (Kerberos) este numele unui câine de pază cu trei capete al lui Hades, a cărui misiune era să păzească intrarea în lumea de dedesubt.

servere funcționează sub pază strictă, acestea fiind folosite ca servere de securitate.

De remarcat că acest mediu nu este potrivit pentru stocarea, procesarea sau transmiterea de date secrete sau efectuarea de operații cu grad ridicat de risc cum ar fi controlul experimentelor periculoase. Riscurile care se au în vedere în principal sunt utilizarea necontrolată a utilizatorilor neautorizați, violarea integrității resurselor sistemelor precum şi violarea în general a intimității cum ar fi citirea fişierelor personale.

Într-un astfel de mediu, riscurile în securitate provin în principal de la posibilitatea de a falsifica identitatea unui individ pentru a obține acces neautorizat la resursele sistemului. O stație de lucru – incluzând sistemul de operare şi interfața de rețea este sub controlul total al utilizatorului care ar putea sa încerce să se substituie altui utilizator sau host.

Primele trei versiuni ale Kerberos au fost folosite doar în cadrul MIT. Acestea nu mai sunt folosite în prezent, de aceea nu ne vom concentra asupra lor în această lucrare. Prima versiune făcută publica a fost Kerberos V4, versiune ce a cunoscut o răspândire importantă în afara MIT. Ultima actualizare (cu numărul 10) a fost făcută în decembrie 1992, aceasta fiind versiunea finală până în prezent.

Unele medii necesită funcționalități neacoperite de Kerberos V4, iar altele au o structură diferită de modelul MIT. Ca rezultat, în 1989 a început lucrul la Kerberos V5, pe baza experienței acumulate cu Kerberos V4 şi a discuțiilor cu administratorii de sisteme care au implementat Kerberos.

În septembrie 1993, Kerberos V5 a fost specificat ca standard Internet în RFC 1510 [KOHL93]. MIT a dezvoltat şi testat Kerberos V5 pe Ultrix, SunOS, Solaris şi Linux, fiind portat şi pe alte sisteme de către terți.

De remarcat că deşi similare ca şi concept, Kerberos V4 şi V5 sunt substanțial diferite şi chiar sunt în competiție pentru dominația pe piață. Pe scurt, Kerberos V4 are o bază de instalare mai mare, este mai simplu şi are o performanță mai mare decât V5, însă lucrează doar cu adrese IP. Kerberos V5 pe de altă parte, are o bază de instalare mai redusă, este mai complicat şi implicit mai puțin eficient, dar prezintă mai multă funcționalitate decât V4. Diferențele importante între V4 şi V5 le vom discuta ulterior în cadrul acestui capitol.

În ciuda faptului că este disponibil codul sursă pentru Kerberos V4 şi V5, MIT nu-l susține oficial şi nu oferă suport. Unele companii însă oferă contra cost versiuni comerciale de implementări Kerberos. Informații despre versiunile freeware şi comerciale se găsesc în Kerberos FAQ publicat periodic în grupul de ştiri *comp.protocols.kerberos*.

2.2. Arhitectură

În terminologia Kerberos, un domeniu administrativ se numeşte *realm*. Se presupune că orice companie sau organizație care doreşte să ruleze Kerberos poate crea un realm identificat unic printr-un nume. Teoretic, Kerberos poate suporta mai bine de 100.000 de utilizatori, iar – referindu-ne la [SCHI95] – realm-ul ATHENA.MIT.EDU suportă în mod curent 25.000 de utilizatori din care aproximativ 7.000 se autentifică în fiecare zi.

Kerberos se bazează pe modelul client / server. Utilizatorii, clienții şi serviciile de rețea instanțiate pe un host în particular se consideră în mod tipic parteneri. Fiecare partener este identificat în mod unic de un *identificator de partener*. În Kerberos V4, un

identificator de partener are în componență trei câmpuri, fiecare fiind un şir terminat cu null de până la 40 de caractere. Aceste trei câmpuri sunt:

- Numele partenerului, **NAME**
- Numele instanței, **INSTANCE**
- Numele realm-ului, **REALM**

Din punctul de vedere al sistemului Kerberos nu contează cum se folosesc câmpurile **NAME** şi **INSTANCE**. Acestea sunt doar şiruri text. Totuşi în practică, serviciile au un nume şi **INSTANCE** se foloseşte pentru a indica host-ul pe care rulează un serviciu anume. Spre exemplu, serviciul **rlogin** de pe host-ul **asterix** se poate distinge de serviciul **rlogin** de pe host-ul **obelix**.

Scopul sistemului Kerberos este de a permite unui client ce rulează în numele unui utilizator anume să se identifice unui serviciu sau unui server de aplicații corespunzător, fără a se necesita trimiterea unor date secrete care ulterior să poată fi folosite de un atacator la impersonarea utilizatorului. Pentru a realiza acest lucru, modelul Kerberos necesită existența unui terț de încredere care serveşte ca centru de distribuție a cheilor (KDC) în realm-ul Kerberos. KDC-ul constă din două componente şi anume:

- Un *server de autentificare* (AS[1])
- Un număr de *servere pentru acordarea biletelor (TGSs[2])*

Sistemele AS şi TGS sunt componente separate logic dar pot fi procese care rulează pe aceeaşi maşină. Aceasta trebuie să fie protejată cu grijă şi securizată fizic, deoarece un intrus ar putea compromite uşor întreg sistemul de la acest nivel.

KDC menține o bază de date cu informații despre fiecare partener din cadrul sistemului. Deoarece securitatea este foarte importantă, această informație este redusă la un minim posibil pentru a efectua cu succes autentificarea. Astfel, deşi baza de date Kerberos este la nivel de utilizator, aceasta nu conține informații cum ar fi numere de telefon sau adrese, neutilizate în mod curent la autentificare, ci următoarele:

- Identificatorul partenerului
- Cheia master K_p (sau parola dacă este utilizator)
- Data expirării identității
- Data ultimei modificări a înregistrării
- Identitatea partenerului care a operat modificarea
- Timpul maxim de viață a biletelor emise partenerului
- Unele atribute
- Unele date interne de implementare invizibile la exterior cum ar fi versiunea cheilor, versiunea cheii master sau indicatori către valori vechi ale înregistrării

Cheia master (K_p) a fiecărui partener trebuie ținută secretă, de aceea toate aceste chei se codifică cu o cheie master a KDC. Pe lângă o protecție sporită, această metodă permite distribuirea bazei de date între servere fără riscul de a fi capturată de un potențial atacator. Cheia master KDC nu se păstrează în aceeaşi bază de date, ci se operează separat.

[1] Authentication Server
[2] Ticket Granting Servers

Kerberos furnizează mai multe instrumente de management a bazei de date KDC. În domeniile mici, se poate utiliza setul de instrumente manuale de la consola administratorului. În domeniile mari se poate utiliza un sistem automat de management a serviciile (SMS). În fiecare caz, funcțiile de management se realizează la distanță prin rețea, conexiunile la baza de date KDC fiind autentificate tot prin Kerberos. Actualizările se efectuează printr-un protocol care rulează între un client autentificat pe o stație şi baza de date KDC. Există rutine de actualizare pentru adăugarea şi dezactivarea intrărilor, precum şi de a controla schimbări de parolă inițiate de utilizator sau administrator.

Figura 9 ilustrează model Kerberos fundamental şi paşii corespunzători ai protocolului. Situația stă în felul următor: pe partea stângă, un client rulează în numele utilizatorului. Pentru a folosi serviciile serverului aflat în partea dreaptă jos, clientul trebuie să se autentifice cu acesta. În această situație, este în sarcina KDC să ofere clientului credențiale pentru procesul de schimb de autentificare. Este important de remarcat că atât clientul cât şi serverul nu au cunoştință de o cheie inițială de sesiune. De fiecare dată când clientul încearcă să se autentifice serverului, acesta se bazează pe KDC pentru generarea cheii de sesiune şi distribuirea ei părților implicate.

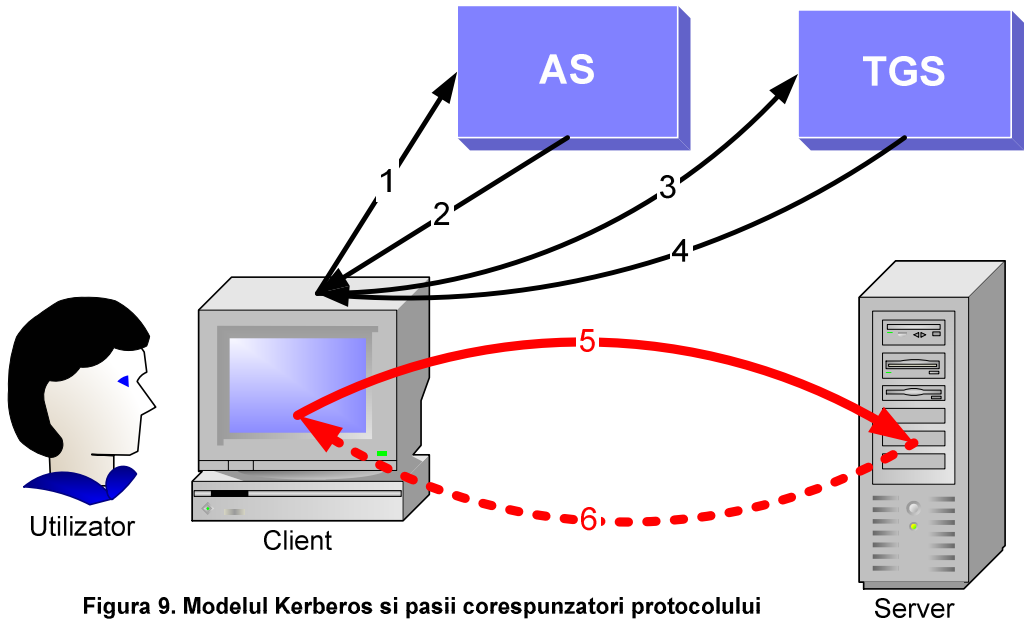

Figura 9. Modelul Kerberos si pasii corespunzatori protocolului

Kerberos foloseşte bilete pentru distribuirea cheilor de sesiune. În general, un bilet este o înregistrare care poate fi folosită la autentificare. În Kerberos, un bilet este un certificat emis de KDC şi criptat cu cheia master a serverului. Printre altele, biletul conține:

- Cheia de sesiune care va fi utilizată pentru autentificarea între client şi server
- Numele partenerului către care cheia de sesiune a fost emisă
- Un timp de expirare după care cheia de sesiune nu mai este validă

Un bilet nu este trimis direct către server, în schimb este trimis clientului care îl înaintează serverului ca parte a schimbului de autentificare. Un bilet Kerberos este întotdeauna criptat cu cheia serverului, cunoscută doar de AS si de serverul în cauză.

Datorită acestei criptări, clientul nu poate modifica mesajul fără ca acest lucru să poată fi depistat.

Când un utilizator se aşează la o staţie de lucru şi încearcă să se logineze, acesta introduce numele său, ca de obicei. Clientul de login furnizează AS numele în pasul 1, iar AS caută intrarea corespunzătoare utilizatorului în baza de date KDC. Parola utilizatorului (sau cheia ce derivă din aceasta) se foloseşte apoi pentru a cripta un bilet de acordare a biletului (TGT[1]) şi TGT-ul este returnat clientului în pasul 2. Clientul cere utilizatorului să îşi introducă parola, cu care se va cripta TGT. Dacă cheia derivată din parola utilizatorului poate decripta cu succes TGT, acestuia i se permite accesul.

Pe partea de client, TGT-ul este memorat pentru folosire ulterioară şi anume pentru a obține bilete pentru autentificarea în servicii de rețea particulare. Scopul principal al TGT este deci să faciliteze o singură autentificare pentru utilizatori. Parola este astfel cerută o singură dată, şi nu de fiecare dată când se cere accesul la un serviciu.

Biletele sunt eliberate de TGS-ul specificat în TGT-ul primit de utilizator. Pentru a obține un bilet, clientul trimite o cerere către TGS în pasul 3. Mesajul include numele serviciului cerut, TGT-ul şi *autentificatorul*. Acesta din urmă este o informație ce poate proba că a fost generată recent utilizând cheia de sesiune de către client şi server împreună. În particular, autentificatorul conține numele utilizatorului, adresa de rețea a clientului şi timpul curent, fiind criptat cu cheia de sesiune returnată în TGT. Autentificatorul este o schemă simplă pentru a descuraja atacurile replay. Dacă TGS consideră atât biletul cât şi autentificatorul valid, biletul este returnat în pasul 4. Clientul creează un alt autentificator şi îl trimite împreună cu biletul de serviciu serverului, în pasul 5. Dacă se cere autentificare reciprocă, serverul returnează un autentificator în pasul 6.

Aşa cum a fost descris până acum, modelul Kerberos nu oferă decât serviciul de autentificare. În sine, el nu oferă informații despre autorizarea clientului în a folosi anumite servicii de rețea. În general, sunt trei posibilități pentru atingerea problemei autorizării, şi anume:

- Baza de date Kerberos ar putea conține informații de autorizare pentru fiecare serviciu şi să emită bilete doar utilizatorilor autorizați.
- Un serviciu dedicat poate menține informațiile de autorizare prin liste de acces pentru fiecare serviciu şi permiterea clientului să obțină certificate sigilate de apartenență la listă. În acest caz clientul ar prezenta serviciului certificarea în locul biletului Kerberos.
- Fiecare serviciu poate menține propria informație de autorizare cu ajutorul opțional al unui serviciu care stochează aceste liste de acces şi oferă certificări de apartenență la listă.

Modelul Kerberos se bazează pe faptul că fiecare serviciu cunoaşte cu exactitate cine sunt utilizatorii săi şi ce formă de autorizare este potrivită pentru aceştia. În consecință Kerberos foloseşte cea de-a treia metodă. În contrast, protocoalele OSF DCE şi SESAME folosesc cea de-a doua metodă.

Pentru simplificarea implementării celei de a treia metode, Kerberos foloseşte modelul de autorizare bazat pe liste de control al accesului. Orice serviciu care consideră că i se potriveşte acest tip de autorizare poate încorpora a bibliotecă cu funcții adecvate. Utilizarea acestui model presupune că serviciul verifică dacă o identitate verificată aparține unei liste de acces.

[1] Ticket Granting Ticket

2.3. Protocoale criptografice

În această secțiune descriem protocoalele criptografice pe care le implementează Kerberos. Începem cu o scurtă descriere a protocolului Needham-Schroeder şi continuăm cu protocoalele utilizate de fapt în Kerberos V4 şi V5. În cele din urmă ne vom concentra atenția asupra autentificării inter-domenii.

2.3.1. Protocolul Needham-Schroeder

Kerberos se bazează pe protocoalele de autentificare şi distribuție a cheilor propuse de Needham şi Schroeder [NEED78, NEED87][1]. Protocolul presupune existența unui terț de încredere care reprezintă un *server de autentificare* (AS). Daca AS are cheile fiecărui partener, doi parteneri arbitrari A şi B pot folosi AS pentru a se autentifica reciproc şi să primească o cheie de sesiune K_{ab}. Primul protocol Needham-Schroeder se poate formaliza după cum urmează:

1. A → AS : A, B, N
2. AS → A : {N, B, K_{ab}, {K_{ab}, A} K_b} K_a
3. A → B : {K_{ab}, A} K_b
4. B → A : {N_b} K_{ab}
5. A → B : {N_b – 1} K_{ab}

Protocolul începe prin comunicația în clar de la A la AS în pasul 1, unde se transmite identitatea pretinsă, identitatea corespondentului dorit B, precum şi o valoare curentă N. La recepționarea mesajului, AS caută cheile secrete ale A şi B (K_a şi K_b) şi alege aleator o cheie de sesiune K_{ab}. În pasul 2, AS returnează lui A {N, B, K_{ab}, {K_{ab}, A} K_b} K_a, mesaj codificat cu cheia lui A, deci numai acesta poate decodifica mesajul şi extrage componentele sale. A poate verifica prezența valorii N pentru a determina dacă mesajul provine într-adevăr de la AS. A memorează valoarea K_{ab} şi trimite lui B valoarea {K_{ab}, A} K_b, în pasul 3. Această valoare este codificată cu cheia lui B, deci numai acesta poate decodifica mesajul şi extrage componentele, în speță cheia K_{ab}. Prin această modalitate B poate afla identitatea corespondentului A, autentificat de AS.

În acest moment, A ştie că orice comunicare codificată cu K_{ab} provine de la B şi orice trimite A codificat cu K_{ab} poate fi înțeles numai de B. Acest lucru este adevărat, deoarece mesajele care au conținut cheia de sesiune au fost legate cheile secrete ale părților A şi B. Acelaşi lucru este valabil şi pentru B. Este important să se asigure securitatea protocolului, în sensul că nici o parte a acestuia să nu poată fi înregistrată şi redată mai apoi de un atacator. Iată cum stau lucrurile din punctul de vedere al celor doi parteneri:

- Pe de o parte, A este conştient că nu a mai folosit K_{ab} şi deci nu are motive să se teamă că materialul codificat cu această cheie este altceva decât un răspuns legitim de la B.
- Pe de altă parte, poziția lui B nu este atât de bună. În afara cazului în care B îşi aminteşte toate cheile folosite anterior de A pentru a verifica dacă K_{ab} este o cheie nouă, B nu este sigur că mesajul primit în pasul 3 provine de la A.

[1] În prima lor lucrare, Needham şi Schroeder propun trei protocoale; două dintre ele folosesc criptografia cu chei secrete, iar cel de-al treilea foloseşte criptografia cu chei publice.

Pentru protecţia împotriva atacurilor replay, B trebuie să aleagă un număr N_b aleator, să-l codifice cu cheia K_{ab} şi să ceară lui A (în pasul 4) să răspundă cu numărul decrementat şi recodificat cu cheia K_{ab}. Dacă răspunsul primit de B este satisfăcător, schimbul ulterior de mesaje poate continua.

Primul protocol Needham-Schroeder constă din 5 paşi. Acest număr se poate reduce la 3 prin menţinerea de către A în cazul partenerilor obişnuiţi de comunicare a perechilor de forma B : K_{ab}, $\{K_{ab}, A\}\ K_b$, astfel eliminându-se paşii 1 şi 2. Noul protocol arată în felul următor:

1. A → AS : A, B, N
2. AS → A : $\{N, B, K_{ab}, \{K_{ab}, A\}\ K_b\}K_a$
3. A → B : $\{K_{ab}, A\}K_b$, $\{N_a\}K_{ab}$
4. B → A : $\{N_a - 1, N_b\}K_{ab}$
5. A → B : $\{N_b - 1\}K_{ab}$

O vulnerabilitate majoră a ambelor versiuni ale protocolului Needham-Schroeder constă în posibilitatea ca un terţ C care este capabil de a urmări dialogul să lanseze un atac off-line asupra K_{ab}. Daca C poate determina o cheie de sesiune K'_{ab} folosită în trecut, C poate impersona pe A. De fapt, C poate folosi biletul corespunzător împreună cu valoarea codificată cu K'_{ab} in pasul 3. B provoacă pe A în pasul 4 şi deoarece C cunoaşte pe K'_{ab}, poate răspunde corect în pasul 5. Slăbiciunea subtilă din spatele acestui protocol este că mesajul din pasul 3 nu conţine nici o informaţie pentru B pentru a-i verifica prospeţimea. De fapt acest lucru este cunoscut doar de AS şi A. Această slăbiciune a fost descoperită de Denning şi Sacco [DENN81]. Ca măsură reparatorie, ei au propus înlocuirea valorii cu amprente de timp şi folosirea următorului protocol în loc:

1. A → AS : A, B
2. AS → A : $\{B, K_{ab}, T, \{K_{ab}, A, T\}\ K_b\}\ K_a$
3. A → B : $\{K_{ab}, A, T\}\ K_b$
4. B → A : $\{N_b\}K_{ab}$
5. A → B : $\{N_b - 1\}\ K_{ab}$

Protocoalele sunt mai mult asemănătoare decât diferite. În protocolul Denning-Sacco, A trimite lui AS pe A şi B în pasul 1 şi AS returnează lui A $\{B, K_{ab}, T, \{K_{ab}, A, T\}\ K_b\}\ K_a$ în pasul 2. În acest caz T reprezintă amprenta de timp, folosită pentru a se determina prospeţimea unui mesaj. A şi B efectuează această verificare în paşii 2 şi 3 prin simpla comparare cu timpul local al sistemului. Având în vedere acest lucru, odată cu introducerea amprentelor de timp apar alte două probleme:

- Utilizarea amprentelor de timp necesită ceasuri sincronizate global
- Definirea intervalelor acceptabile de timp este o sarcină dificilă

Ambele probleme nu sunt independente şi au creat o nouă arie în cercetare, vezi [GONG92] şi [LAMB92].

2.3.2. Protocolul Kerberos V4

Protocolul utilizat în Kerberos V4 se bazează în parte pe protocolul Needham-Schroeder cu unele modificări menite să-l adapteze la mediul de rețea pentru care a fost proiectat inițial. Printre aceste schimbări, amintim:

- Utilizarea amprentelor de timp, aşa cum au propus Denning şi Sacco
- Adăugarea unui serviciu de oferire a biletelor pentru a permite autentificarea suplimentară fără a fi necesar ca utilizatorul să-şi reintroducă parola
- O abordare diferită a autentificării inter-domenii

Având în vedere figura nr. 9, paşii protocolului Kerberos V4 se poate formaliza după cum urmează:

1. C → AS : U, TGS, T, L
2. AS → C : $\{K, N, T_{c,tgs}\}\ K_u$
3. C → TGS : $S, N, T_{c,tgs}, A_{c,tgs}$
4. TGS → C : $\{T_{c,s}, K'\}\ K$
5. C → S : $T_{c,s}, A_{c,s}$
6. S → C : $\{T + 1\}\ K'$

În această descriere, termenul $T_{c,tgs} = \{U, C, TGS, T, L, K\}\ K_{TGS}$ se utilizează pentru a ne referi la un TGT care a emis de AS pentru a fi folosit de un client C pentru a se autentifica cu un TGS şi a cere un bilet corespunzător. În mod similar, termenul $T_{c,s} = \{U, C, S, T', L', K'\}\ K_s$ desemnează un bilet emis de TGS şi utilizat de C pentru a se autentifica cu un server S. În ambele expresii, T şi T' se referă la amprente de timp, iar L şi L' desemnează timpul de viață al acestora. O amprentă de timp reprezintă numărul de secunde de la ora 00:00:00 GMT, 1 ianuarie 1970 şi se codifică pe 4 octeți. Aceasta este o reprezentare comună a timpului într-un sistem UNIX. În mod similar, timpul de viață al unui bilet se specifică în unități de cinci minute şi se codifică într-un singur octet, drept urmare timpul maxim de viață care se poate exprima în Kerberos V4 este de puțin peste 21 de ore.

În plus față de acestea, termenii $A_{c,tgs} = \{C, T\}\ K$ şi $A_{c,s} = \{C, T'\}K'$ desemnează autentificatorii pentru $T_{c,tgs}$ şi $T_{c,s}$.

Cei şase paşi ai protocolului se pot grupa în trei schimburi:

- Schimbul AS între client şi AS (paşii 1 şi 2)
- Schimbul TGS între client şi TGS (paşii 3 şi 4)
- Schimbul AP între client şi serverul de aplicații (paşii 5 şi 6)

În cele ce urmează vom discuta aceste schimburi.

Schimbul AS

Schimbul AS al protocolului Kerberos V4 este ilustrat în figura 10. Pe scurt, un client C utilizează un serviciu de nume (*name service)* pentru a localiza în termeni de topologie a rețelei cel mai apropiat TGS disponibil. C trimite apoi un mesaj KRB_AS_REQ (*Kerberos authentication server request*) către AS în pasul 1. Mesajul include identificatorul de utilizator U, identificatorul TGS-ului selectat, o amprentă de timp T şi durata de viață dorită pentru TGT.

După primirea mesajului KRB_AS_REQ, AS caută şi extrage cheile secrete asociate cu U şi TGS. AS selectează apoi o cheie de sesiune aleatoare K şi-i răspunde lui C cu mesajul KRB_AS_REP (*Kerberos authentication server reply*) în pasul 2. Mesajul include K, N şi $T_{c,tgs}$. În plus, mesajul este codificat cu K_u.

După recepționarea mesajului KRB_AS_REP, C trebuie să ceară utilizatorului U parola sa. De fapt, protocolul Kerberos V4 adoptă o regulă general valabilă în securitate, aceea de a ştii parola utilizatorului un timp minim posibil. Totuşi, aşteptarea pentru câteva secunde a mesajului KRB_AS_REP nu creşte siguranța în mod semnificativ, de aceea Kerberos V5 cere parola înainte ca C să trimită mesajul KRB_AS_REP. Un alt motiv pentru care această parolă se cere înainte este că C trebuie să dovedească faptul că ştie parola utilizatorului pentru ca AS să trimită mesajul KRB_AS_REP, lucru care îngreunează atacul prin ghicire a parolelor.

În Kerberos V4, dacă U introduce corect parola pwd_U, C poate folosi o funcție *h* de dispersie cu sens unic cunoscută pentru a calcula cheia secretă a lui U, adică $K_U = h(pwd_U)$.

Echipat cu această cheie, C poate decodifica $\{K, N, T_{c,tgs}\}\ K_U$ şi extrage K, N, şi Tc,tgs. Dacă C are succes, acesta este în posesia unui TGT care poate fi utilizată pentru a cere bilete de la TGS.

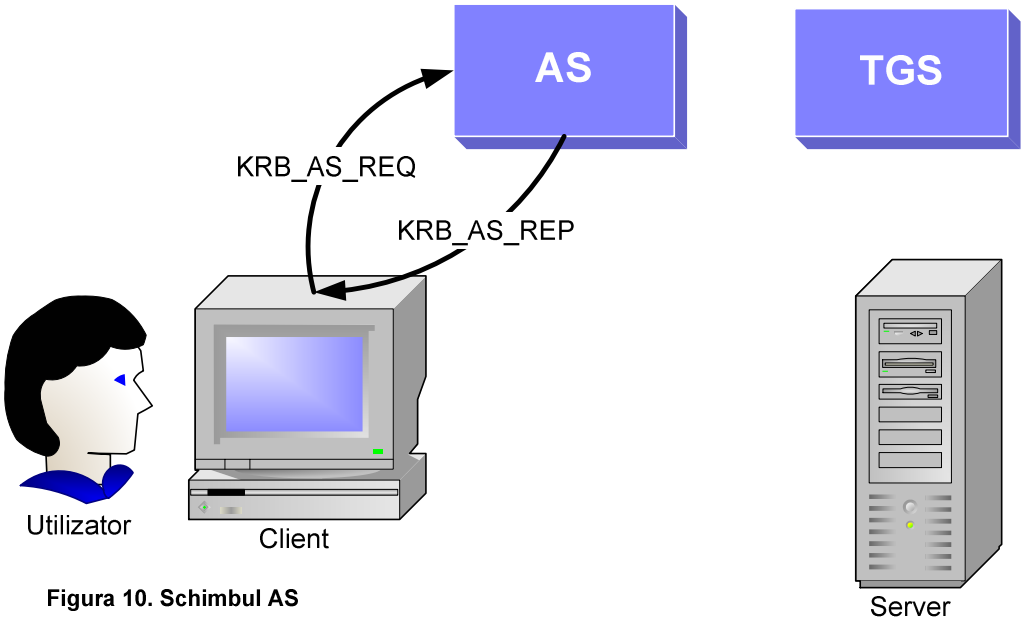

Figura 10. Schimbul AS

De remarcat că într-un TGT, timpul de viață se foloseşte ca o parolă de expirare a timpului. Limitarea timpului de viață al TGT limitează astfel avariile produse în sistem în cazul compromiterii TGT. În Kerberos, în general nu este posibil să se revoce un TGT odată ce a fost emis.

Schimbul TGS

Schimbul TGS al protocolului Kerberos V4 este ilustrat în figura 11. Formatul mesajului este aproape identic cu cel din schimbul AS. Diferența constă în faptul că mesajul este codificat cu cheia de sesiune K (cunoscută de C şi TGS) în loc de cheia utilizatorului K_U.

Înainte de a iniția un schimb TGS, C trebuie să determine în care domeniu a fost înregistrat serverul de aplicații pentru care se cere un bilet. Dacă C nu are deja un TGT pentru acel domeniu, trebuie să obțină unul. Prima dată se încearcă a se obține un TGT

pentru domeniul destinație de la un server Kerberos local, prin trimiterea mesajului KRB_TGS_REQ. Acesta poate returna un TGT pentru domeniul în cauză, caz în care C îl poate folosi. Alternativ, serverul Kerberos poate returna un TGT pentru un domeniu mai apropiat de cel solicitat, caz în care procesul trebuie repetat pentru acest domeniu, specificat în TGT. În cazul în care nu se primeşte nici un răspuns, se încearcă obținerea TGT-ului de la un domeniu mai sus în ierarhie.

În pasul 3, C trimite mesajul KRB_TGS_REQ (*Kerberos ticket granting server request*) către TGS. Mesajul include S, N, $T_{c,tgs}$ şi un autentificator $A_{c,tgs}$ = {C, T} K, cu T amprentă de timp. De remarcat că $T_{c,tgs}$ poate avea o viață relativ lungă şi poate fi capturat şi redat. Scopul autentificatorului este să arate că C ştie cheia secretă şi deci să contracareze un astfel de atac. Utilizarea autentificatorilor necesită o sincronizare relativ precisă a ceasurilor, diferența admisibilă fiind configurabilă la fiecare server. De regulă, aceasta este configurată la 5 minute, dar în practică aceasta s-a dovedit mult mai problematică. Odată cu introducerea protocoalelor de sincronizare temporală a fost posibilă sincronizarea mult mai precisă.

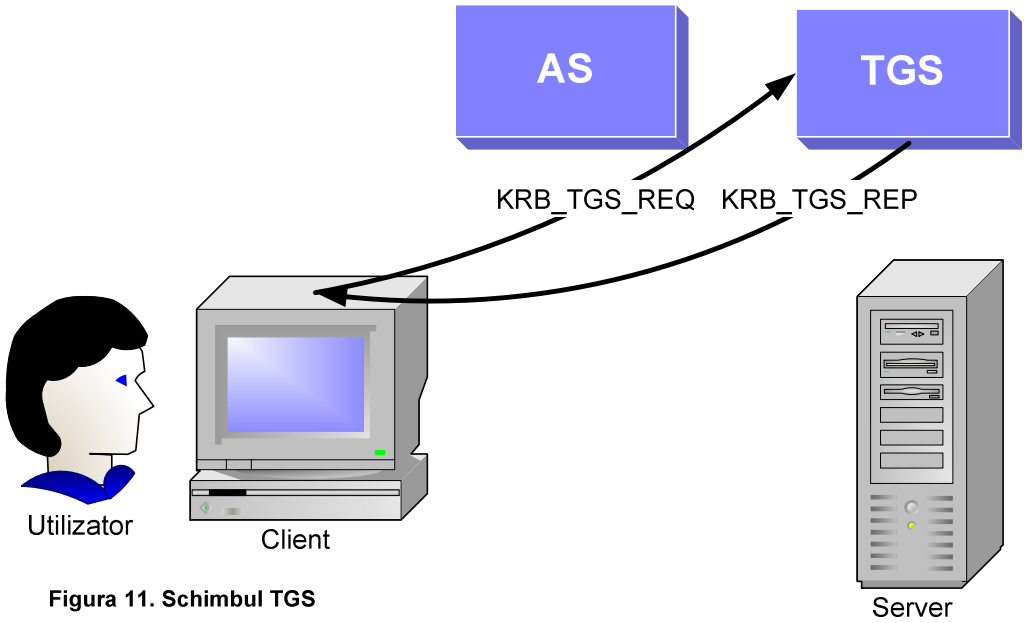

Figura 11. Schimbul TGS

Mesajul KRB_TGS_REQ este procesat într-o manieră similară cu mesajul KRB_AS_REQ dar se fac o serie de verificări suplimentare. În pasul 4, TGS-ul returnează un mesaj KRB_TGS_REP *(Kerberos ticket granting server reply)* care are acelaşi format cu mesajul KRB_AS_REP. Acesta include un bilet $T_{c,s}$ pentru serverul cerut S şi o nouă cheie de sesiune K', ambele codificate cu K. Când mesajul KRB_TGS_REP este recepționat de C, este procesat în aceeaşi manieră cu KRB_AS_REP, aşa cum s-a descris mai înainte. Principala diferență este că partea cifrată a răspunsului trebuie decodificată cu cheia de sesiune cunoscută de TGS şi nu cu cheia utilizatorului.

Schimbul AP

Schimbul AP din protocolul Kerberos V4 este ilustrat în figura 12. Acesta este utilizat de aplicații de rețea fie pentru a autentifica un client față de un server sau să se autentifice reciproc. Clientul trebuie să aibă deja credențiale pentru server prin utilizarea schimburilor AS şi TGS.

În pasul 5, C trimite mesajul KRB_AP_REQ *(Kerberos Application Request)* către serverul S. Mesajul include $T_{c,s}$, $A_{c,s}$ = {C, T'} K' şi unele informaţii de management. Autentificarea se bazează pe timpul curent al serverului, biletul $T_{c,s}$ şi autentificatorul $A_{c,s}$.

Pentru a ne se asigura că mesajul KRB_AP_REQ nu este o redare a unui mesaj recent, este îndeajuns ca S să memoreze toate amprentele de timp într-o fereastră dată şi să se asigure că mesajul nu are o amprentă de timp identică cu cele anterior recepţionate. Orice autentificator mai vechi decât fereastra de timp permisă ar fi respins oricum, deci nu ar fi folositor a se memora amprente mai vechi decât această fereastră.

Kerberos V4 însă nu se complică să memoreze amprente de timp, aceasta deoarece dacă S este un serviciu replicat în care toate instanţele împart aceeaşi cheie master, ar fi relativ uşor pentru un atacator să trimită un mesaj de la o instanţă S_1 către o altă instanţă S_2. Rezolvarea simplă a acestei probleme ar fi ca autentificatorul să conţină şi identificatorul nivelului de reţea a instanţei S în cauză.

Dacă nu apare nici o eroare şi dacă este necesară autentificarea reciprocă, S trebuie să returneze lui C mesajul KRB_AS_REP *(Kerberos application reply)* în pasul 6. Din nou, acest mesaj este codificat cu cheia de sesiune K', cunoscută de C şi S. Din moment ce această cheie a fost în biletul codificat cu cheia secretă a serverului, posesia acesteia este dovada că S este partenerul dorit. Mai precis, S trebuie să incrementeze amprenta de timp inclusă în autentificatorul mesajului KRB_AP_REQ şi să-l recodifice cu K'.

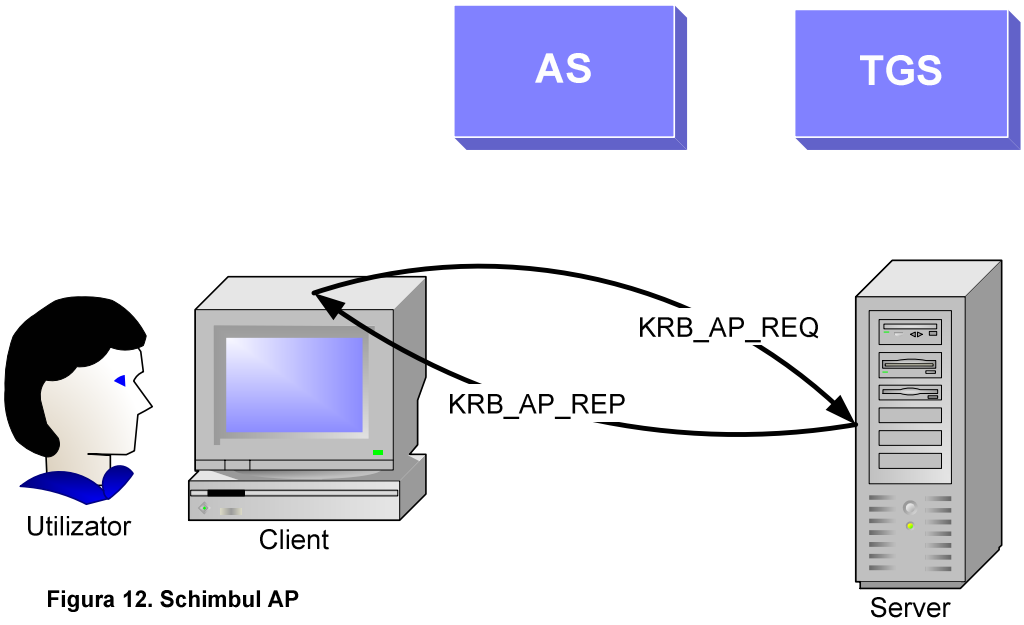

Figura 12. Schimbul AP

Confidenţialitatea datelor şi servicii de integritate

Aşa cum a fost descris anterior, Kerberos furnizează servicii de autentificare. Totuşi, un „produs" auxiliar al protocolului este schimbul unei chei de sesiune K', cunoscută de client şi de server. Această cheie poate fi utilizată de aplicaţie pentru a proteja confidenţialitatea şi integritatea comunicaţiilor. De fapt, sistemul Kerberos oferă două tipuri de mesaje, şi anume:

- Mesajul KRB_PRIV pentru a proteja confidenţialitatea comunicaţiilor.
- Mesajul KRB_SAFE pentru a proteja integritatea comunicaţiilor.

În ciuda acestor tipuri de mesaje oferite de sistem, o aplicație este liberă să utilizeze orice metodă potrivită pentru tipul particular de date care se transmite. Formatul acestor mesaje nu va fi detaliat mai mult aici.

2.3.3. Protocolul Kerberos V5

În această secțiune ne vom concentra asupra Kerberos V5 în general şi a diferențelor importante față de Kerberos V4. În Kerberos V5 au fost operate următoarele modificări:

- *Identificatorii părților:* În Kerberos V5, identificatorii părților constau în două repere: numele domeniului (realm) şi un rest. Numele domeniului este separat pentru a facilita implementarea uşoară a rutinelor de traversare şi a verificărilor de acces la domeniu. Restul este o secvență de câte componente sunt necesare pentru numele părții. De remarcat că identificatorul din Kerberos V4 este similar cu un rest cu două componente în Kerberos V5.
- *Utilizarea criptării:* Pentru a îmbunătăți modularitatea lui Kerberos, criptarea a fost separată în module software care pot fi schimbate sau îndepărtate la nevoie. Când se foloseşte criptarea, textul respectiv este marcat în aşa fel încât receptorul să poată identifica modulul de care are nevoie pentru a înțelege mesajul. Algoritmii de criptare de asemenea au sarcina de a oferi siguranța integrității, prin folosirea unor metode adecvate, cum ar fi suma de control.
- *Adrese de rețea:* În Kerberos V5, adresele sunt etichetate cu un tip şi lungime, astfel că dacă un host suportă mai multe protocoale de rețea să le poată oferi bilete.
- *Codificarea mesajelor:* Mesajele sunt descrise utilizând notația de sintaxă abstractă 1 (ASN 1) şi codificate după regulile fundamentale de codificare (BER). Astfel, se evită specificarea codificării pentru cantități compuse din mai mulți octeți, aşa cum era cazul în Kerberos V4.
- *Formatul biletului:* Biletul are un format extins pentru noile opțiuni. Acesta este împărțit în două părți, una codificată iar alta nu. Numele serverului este necodificat pentru a permite unui server cu identități multiple să selecteze cheia potrivită. În plus față de această schimbare, timpul de viață al biletului este reprezentat ca timp de start şi timp de expirare, permițând timp de viață aproape nelimitat.
- *Suport inter-domeniu:* În Kerberos V5, domeniile pot coopera printr-o ierarhie bazată pe numele realm-ului. Un domeniu sursă este interoperabil cu domeniul destinație dacă cele două cunosc o cheie pentru domeniul destinație.

În plus față de acestea, Kerberos V5 mai aduce o serie de îmbunătățiri. Spre exemplu, câmpurile indicatoare menționate anterior permit o flexibilitate mai mare în utilizarea biletelor decât în versiunea V4. Fiecare bilet emis de AS în schimbul inițial este marcat ca atare. Aceasta permite serverelor să ceară clientului să prezinte un bilet obținut prin folosirea directă a cheii secrete a clientului în locul cheii obținute folosind un TGT. O astfel de cerință împiedică un atacator să schimbe parola unui utilizator de la o stație neutilizată dar loginată.

În Kerberos V5, biletele pot fi eliberate cu două date de înnoire. Biletul expiră ca de obicei la primul timp, dar dacă TGS primeşte o cerere de înnoire înainte de expirarea acestuia, biletul emis ca înlocuitor este valabil pentru încă o perioadă de timp. Totuşi, TGS-ul nu va reînnoi un bilet după expirarea celui de al doilea timp. Acest mecanism prezintă avantajul că deşi credenţialele se pot utiliza o perioadă lungă de timp, TGS-ul ar putea refuza să reînnoiască bilete raportate ca furate şi astfel să împiedice folosirea lor în continuare.

Kerberos se ocupă în principal cu autentificarea şi nu este în mod direct preocupat de servicii de securitate înrudite cum ar fi autorizarea şi contabilitatea. Pentru a sprijini implementarea acestor servicii de securitate, Kerberos V5 oferă un mecanism de transmisie sigură a informaţiilor de autorizare şi contabilitate ca parte a unui bilet. Acestea pot restricţiona utilizarea unui bilet.

Când se solicită un bilet, restricţiile sunt specificate şi trimise unui TGS unde sunt incluse în bilet, codificate şi protejate de modificare. În forma cea mai generală a protocolului, un client cere ca KDC să includă sau adauge astfel de informaţii la bilet. TGS-ul la rândul lui nu îndepărtează informaţia de autorizare din TGT, ci o copiază în fiecare bilet nou şi adaugă informaţii adiţionale. La decodificarea biletului, informaţiile de autorizare sunt disponibile la serverul de aplicaţii. Kerberos nu face nici o interpretare a acestor informaţii, în schimb serverul de aplicaţii trebuie să le folosească pentru a restricţiona accesul clienţilor la resursele prezentate în bilet.

Având în vedere figura 9, paşii protocolului Kerberos V5 se pot formaliza după cum urmează:

1. C → AS : U, TGS, L_1, N_1
2. AS → C : U, $T_{c,tgs}$, {TGS, K, T_{start}, T_{expire}, N_1} K_U
3. C → TGS : S, L_2, N_2, $T_{c,tgs}$, $A_{c,tgs}$
4. TGS → C : U, $T_{c,s}$, {S, K', T'_{start}, T'_{expire}, N_2} K
5. C → S : $T_{c,s}$, $A_{c,s}$
6. S → C : {T'} K'

Analog cu Kerberos V4, $T_{c,tgs}$ şi $T_{c,s}$ se referă la un TGT respectiv un bilet, iar $A_{c,tgs}$ şi $A_{c,s}$ se referă la autentificatorii respectivi.

În pasul 1, C selectează un TGS şi trimite un mesaj KRB_AS_REQ la AS. Mesajul include identificatorul utilizatorului (U), identificatorul TGS-ului selectat, durata de viaţă solicitată L_1 pentru TGT şi o valoare N_1. În plus faţă de acestea, un mesaj poate specifica un număr de opţiuni, cum ar fi:

- Dacă pre-autentificarea trebuie realizată
- Dacă biletul solicitat se poate re-înnoi sau re-trimite.
- Dacă biletele derivate ar trebui să fie postdatate sau să permită postdatarea.
- Dacă un bilet re-înnoibil va fi acceptat în locul unuia ne-înnoibil (dacă timpul de expirare nu poate fi satisfăcut)

După recepţionarea mesajului KRB_AS_REQ, AS caută şi extrage cheile secrete pentru U şi TGS. Dacă este necesar, AS pre-autentifică cererea iar dacă aceasta eşuează, un mesaj de eroare corespunzător este returnat lui C. Altfel, AS selectează aleator o nouă cheie de sesiune K' şi returnează un mesaj KRB_AS_REP lui C în pasul 2. Mesajul include U, un $T_{c,tgs}$ = {U, C, TGS, K, T_{start}, T_{expire}} K_{tgs} şi {TGS, K, T_{start}, T_{expire}, N_1} K_U. Timpii de start şi de expirare T_{start} respectiv T_{expire} ale TGT-ului sunt setaţi în concordanţă

cu politica de securitate a domeniului în aşa fel încât se încadrează în timpul de viață L_1 specificat în mesajul KRB_AS_REQ.

După recepționarea mesajului KRB_AS_REP, C aplică o funcție cu sens unic *h* la parola utilizatorului pwd_U pentru a calcula cheia master a acestuia. $K_U = h(pwd_U)$. Echipat cu această cheie, C poate decodifica $\{TGS, K, T_{start}, T_{expire}\}\ K_U$, şi extrage TGS, K, T_{start}, T_{expire}. Acum poate folosi acest TGT pentru a solicita un bilet de la TGS.

Schimbul TGS este inițiat ori de câte ori C doreşte să obțină un bilet pentru server, re-înnoiască său valideze un bilet existent sau chiar să obțină un bilet proxy. Cel puțin în primul caz, clientul trebuie să fi obținut deja un TGT prin schimbul AS. În pasul 3, C trimite mesajul KRB_TGS_REQ către TGS. Din nou, mesajul include informații pentru autentificarea clientului plus cereri pentru credențiale. Informațiile de autentificare constau în $T_{c,tgs}$ şi un autentificator corespunzător $A_{c,tgs} = \{C, T\}\ K$ generat de C cu amprenta de timp T şi cheia de sesiune K. În pasul 4, TGS returnează un mesaj KRB_TGS_REP care include un bilet $T_{c,s} = \{U, C, S, K', T'_{start}, T'_{expire}\}\ K_s$ pentru serverul S, precum şi $\{S, K', T'_{start}, T'_{expire}\}\ K$.

Din nou, schimbul AS se utilizează fie pentru a autentifica un client unui server, fie pentru a se autentifica reciproc un client şi un server. În pasul 5, C trimite un mesaj KRB_AP_REQ către serverul S. Mesajul conține biletul $T_{c,s}$ şi un autentificator corespunzător $A_{c,s} = \{C, T'\}\ K'$, împreună cu unele informații de management. Autentificarea se bazează pe timpul curent al serverului, a autentificatorului şi a biletului. Dacă nu apare nici o eroare şi dacă se cere autentificare reciprocă, S returnează un mesaj KRB_AP_REP în pasul 6. Acest mesaj include amprenta de timp T', codificată cu cheia de sesiune K' pe care S o cunoaşte împreună cu C.

2.3.4. Autentificarea inter-domenii

În general, un mediu de rețea constă din mai multe organizații, cum ar fi companii în competiție, agenții guvernamentale, instituții financiare său universități. Într-un astfel de mediu, ar fi dificil de găsit o entitate credibilă pentru toată lumea care să ruleze un Kerberos KDC. Problema este că oricine are acces la KDC cunoaşte cheia master a utilizatorului şi are acces la orice are acces şi utilizatorul. Chiar şi dacă o astfel de entitate s-ar găsi, replicile KDC-ului ar trebui să fie de asemenea credibile. Mai mult, o entitate credibilă ar fi foarte ocupată cu utilizatorii şi serviciile care intră şi iasă din rețea.

Din acest motiv, rețeaua este împărțită în domenii, fiecare dintre acestea având propriul KDC. Acestea pot fi mai multe în cadrul unui domeniu, dar sunt identice din punct de vedere funcțional, având aceeaşi cheie master şi aceeaşi bază de date. Pe de altă parte, KDC-urile din domenii diferite sunt total diferite, fiind responsabile pentru un set diferit de participanți.

Kerberos a fost proiectat să funcționeze şi în afara granițelor domeniului, fiind posibil ca în principiu, un client într-un domeniu să se autentifice unui server din alt domeniu. În Kerberos, o autentificare peste granițele domeniului se numeşte *autentificare inter-domeniu*.

O *cheie inter-domeniu* este o cheie secretă cunoscută de KDC-urile a două domenii Kerberos. Prin stabilirea acestor chei, administratorul domeniilor permite ca un utilizator autentifica într-un domeniu să se autentifice şi la distanță. Schimbul de chei inter-domeniu se face prin înregistrarea tuturor TGS-urilor fiecărui domeniu ca utilizator în domeniul curent. Un client este capabil atunci să obțină un TGT pentru TGS-ul domeniului îndepărtat. Când se foloseşte acel TGT, TGS-ul îndepărtat foloseşte cheia inter-domeniu pentru a decodifica TGT-ul şi este atunci sigur că a fost emis de TGS-ul

clientului. Biletele emise de TGS-ul îndepărtat vor indica unui serviciu că autentificarea clientului a fost făcută în alt domeniu.

Se spune că un domeniu comunică cu altul dacă cele două cunosc aceeaşi cheie inter-domeniu sau dacă domeniul local cunoaşte cheia unui domeniu intermediar care la rândul său comunică cu domeniul îndepărtat. În terminologia Kerberos, o cale de autentificare se referă la o secvență de domenii tranzitate în proces.

Aşa cum s-a menționat anterior, Kerberos V4 necesita ca AS-ul să comunice cu fiecare domeniu unde era necesară autentificarea inter-domeniu. Acest procedeu nu duce la o scalare bună din moment ce o interconectare completă necesită schimbul a n^2 chei, cu n fiind numărul total de domenii.

În contrast, Kerberos V5 suportă autentificarea inter-domenii cu treceri, cheile fiind distribuite ierarhic. Aici, fiecare domeniu menține chei cu domeniul părinte şi domeniile fii. Dacă cheia inter-domeniu nu este distribuită de două domenii, sistemul ierarhic permite stabilirea unei căi de autentificare. Această cale este memorată în cadrul biletului pentru a se decide nivelul de încredere acordat procesului de autentificare.

2.4. Concluzii

Utilizarea sistemului Kerberos îmbunătățeşte securitatea aplicațiilor în rețea prin îngreunarea impersonării unui utilizator. Totuşi, Kerberos nu este o rezolvare a tuturor problemelor din rețea, iar Kerberos are de asemenea propriile probleme. Câteva dintre limitările şi slăbiciunile lui Kerberos V4 au fost descrise pentru prima dată de Bellovin şi Merritt în [BEME] iar mai apoi reluate de Kohl, Neuman şi Ts'o în [KOHL94]. În cele ce urmează vom descrie rezultatele acestor studii:

Kerberos V4 a fost conceput în principal pentru necesitățile proiectului Athena, astfel că în anumite locuri face presupuneri şi abordează lucrurile într-un mod nepotrivit în toate cazurile:

Deficiențe de mediu

- *Dependența de criptosistem:* Implementarea de referință de la MIT pentru Kerberos V4 utilizează DES pentru a codifica mesajele. Înainte ca restricțiile de export ale sistemelor criptografice să fie ridicate, răspândirea utilizării lui Kerberos a fost destul de dificilă.
- *Dependența față de protocolul Internet:* Kerberos V4 necesită utilizarea adreselor de tip IP, ceea ce îl face nepotrivit în anumite medii.
- *Ordinea octeților în mesaj:* În Kerberos V4, host-ul care trimite mesajul ordonă octeții în conformitate cu ordinea sa naturală. Receptorul este responsabil de a re-ordona octeții pentru a se potrivi modului său nativ. În timp ce acest lucru va uşura comunicarea între host-uri cu aceeaşi ordine a octeților, comunicarea cu un tip diferit de host poate afecta inter-operabilitatea.
- *Timpul de viață al biletului:* S-a menționat deja că timpul maxim de viață al unui bilet este puțin peste 21 de ore. Această limită s-a dovedit un neajuns major în Kerberos V4, din moment ce împiedică acordarea biletelor unor job-uri care rulează multă vreme.
- *Înaintarea autentificării:* Kerberos V4 nu permite ca un bilet emis unui client de pe un host să fie trimis altui host sau utilizat de alt client.
- *Numirea utilizatorilor:* În Kerberos V4, numele sunt compuse din trei componente: nume, instanță şi domeniu, fiecare având maxim 40 de caractere. Aceste dimensiuni s-au dovedit prea mici pentru unele aplicații.

- *Autentificarea inter-domenii:* Kerberos V4 necesită păstrarea unui număr de n^2 chei ale celor n domenii. Chiar şi pentru n relativ mic, sarcina de a le distribui este scumpă.

Deficiențe tehnice

- *Dubla criptare:* TGT-ul emis de AS este codificat de două ori când este returnat la client şi numai o dată când este trimis la TGS. În general nu este necesar a se codifica biletul de două ori, fiind o pierdere de timp.
- *Codificarea PCBC:* Kerberos V4 utilizează DES în versiune nestandardizată a modului CBC.
- *Autentificarea mesajului:* Algoritmul de sumă de control din Kerberos V4 nu este documentat sau publicat, de aceea nu există dovezi despre slăbiciunea sau tăria sa. Totuşi, faptul că un algoritm nu a fost spart până în prezent nu înseamnă că algoritmul este sigur.

Cele mai multe dintre probleme expuse anterior au fost luate în seamă la design-ul lui Kerberos V5.

Probabil că cea mai evidentă problemă este că sistemul Kerberos încă se bazează pe parole bine alese şi pe faptul că acestea sunt ținute secrete. Dacă un atacator poate primi acces la parola unui utilizator, Kerberos nu poate distinge între cei doi. Aici putem face două afirmații:

- Prima, că oamenii în general aleg parole slabe care sunt vulnerabile la atacuri prin dicționar sau ghicire. Similar, forțarea oamenilor să aleagă parole complexe nu este o măsură fericită.
- A doua, Kerberos este expus atacurilor de tip „parolă verificabilă”. În pasul 2 al protocolului, AS-ul returnează un mesaj clientului codificat cu o cheie derivată din parola utilizatorului. Dacă un atacator este capabil să solicite acest mesaj sau este capabil de a-l captura, acesta poate lansa un atac de ghicire a parolelor. În acest caz, atacul se poate lansa simultan împotriva tuturor mesajelor cunoscute de către atacator. In acest caz nu există un sistem similar cu „UNIX salt”.

În principiu, atacurile de tip „parolă verificabilă” sunt o formă specială de atacuri ostile numite *atacuri de text verificabil*. În timp ce vulnerabilitățile la atacuri de text cunoscut sunt relativ uşor de detectat şi corectat, atacurile de text verificabil sunt mult mai subtile şi mult mai dificil de evitat. Trebuie remarcat că aceste atacuri sunt o amenințare serioasă doar dacă parolele sunt alese necorespunzător.

Pentru a face AS-ul rezistent la atacuri de tip „parolă verificabilă” au fost propuse independent două tehnici. Una dintre acestea a fost propusă Lomas, Gong, Saltzer şi Needham în [GONG93] şi [LOMA89], iar cealaltă de către Bellovin şi Merritt în [BELL92] şi [BELL93]. Aceste propuneri se bazează pe criptografia cu chei publice, o abordare respinsă de la început în design-ul original.

O problemă oarecum înrudită a Kerberos V5 este faptul că nu e rezistent la atacuri reply. În afara cazului în care serverul ține o listă cu autentificatorii folosiți în timpul ferestrei de timp, un intrus ar putea lansa un atac de tip off-line. Dacă atacatorul ar putea convinge host-ul asupra unui timp incorect, host-ul ar putea oferi un număr de autentificatori postdatați, ceea ce creează premisa unui atac viitor. De fapt, Kerberos

presupune o relativă sincronizare a host-urilor, adică acces din partea acestora la un serviciu de timp, ceea ce induce un risc suplimentar. [OPPL96]

Capitolul 3

Protocolul SESAME

În acest capitol ne vom îndrepta atenția asupra proiectului european SESAME *(Secure European System for Applications in a Multi-vendor Environment).* În secțiunea 3.1 vom descrie proiectul, iar în secțiunea 3.2 vom face prezentarea arhitecturii şi a sistemului de distribuție a cheilor. În secțiunea 3.3 ne vom opri pe scurt asupra protocolului criptografic folosit de SESAME, iar în 3.4 vom analiza aplicabilitatea acestui protocol.

3.1. Prezentare

Asociația fabricanților europeni de calculatoare (ECMA[1]) a fost fondată în 1961 pentru a lucra la standardizarea sistemelor informatice şi de comunicații. Unul dintre proiectele asociației este SESAME (*Secure European System for Applications in a Multi-vendor Environment),* obiectivul primar fiind să producă tehnologie pentru controlul accesului în sisteme distribuite.

Proiectul SESAME primeşte o jumătate din fonduri de la CEC[2] prin programul RACE R2051, iar membrii cuprind atât dezvoltatori (incluzând comercianții) cât şi utilizatorii (incluzând companiile de telefonie).

SESAME a fost organizat ca un proiect în două faze:

- Faza întâi, dezvoltarea lucrului ECMA cu finalitate o implementare care se demonstreze că ideile de arhitectură şi principiile sunt atât fezabile cât şi practice. Acest lucru s-a realizat în 1991 şi prototipul rezultat se numeşte SESAME V1 [PARK91].

[1] European Computer Manufacturer Association
[2] Commission of the European Communities

- Faza a doua, dezvoltarea componentelor de securitate pentru construirea produselor comerciale. Această fază a dus la dezvoltarea unei versiuni anterioare numite SESAME V2 care a fost lansată pentru testare beta limitată membrilor SESAME User Group. Pentru a facilita răspândirea sistemului, SESAME V2 a fost distribuit gratuit pentru folosință necomercială. Astăzi, la sfârşitul fazei a doua, varianta SESAME V3 este disponibilă gratuit pentru uz necomercial şi sub licență pentru includerea în produse comerciale.

Ca toate proiectele finanțate de CEC, SESAME este cercetat îndeaproape şi supus unui proces de audit. În 1994, lucrul la SESAME a fost auditat iar în 1995 s-a primit aprobarea pentru continuarea lucrului.

Codul sursă SESAME este dezvoltat în colaborare de către:

- Bull SA
- International Computers Ltd (ICL)
- Siemens Nixdorf Informationssysteme (SNI) AG
- Software and Systems Engineering (SEE) Ltd

Proiectul SESAME a adoptat şi modificat uşor implementări existente ale Kerberos, DES, RSA şi MD5. Totuşi, efortul de dezvoltare major este direcționat către extensii ale protocolului Kerberos V5. Deoarece tehnologia SESAME se intenționează a fi disponibilă global, codul său sursă trebuie publicat într-o formă care să permită exportul între diferite națiuni. Pentru a se conforma cu diferite legi de import şi export, SESAME permite schimbarea algoritmilor criptografici şi a funcțiilor de dispersie. În general, responsabilitatea de a obține licențe aparține companiilor sau organizațiilor care intenționează să folosească sau să exporte tehnologia.

3.2. Arhitectură

Proiectul SESAME a adoptat terminologia introdusă de cadrele de securitate ISO/IEC. În particular, se foloseşte termenul de *partener* pentru a referi o persoană sau entitate înregistrată şi autentificabilă. Când joacă un rol activ (spre exemplu când solicită acces), partenerul se numeşte *inițiator*. Când joacă un rol pasiv (spre exemplu este accesat), partenerul se numeşte *țintă*.

Un serviciu este un set abstract de funcționalități care poate fi implementat de un număr de servere separate. Referindu-ne la modelul client / server, componentele aplicației client joacă rolul de inițiatori comunicând cu componentele aplicației server care joacă rolul de țintă.

S-a menționat deja că obiectivul primar al proiectului SESAME este producerea de tehnologie pentru controlul accesului în sisteme distribuite, adică să ofere servicii de autentificare, control al accesului, confidențialitate şi integritate a datelor. Aceste deziderate se ating printr-un serviciu Kerberos V5 extins şi un serviciu de privilegii în stilul ECMA:

- *Serviciul de autentificare extinsă Kerberos V5* suportă atât mecanismul de autentificare bazat pe parolă cât şi un mecanism mai sofisticat bazat pe criptografia cu chei publice.
- *Serviciul de privilegii în stilul ECMA* se poate utiliza pentru implementarea politicii de control al accesului bazat pe roluri. În SESAME, drepturile de

acces sunt numite *privilegii* şi sarcina de a le specifica şi asocia unui utilizator care în general în sarcina unui administrator.

Să ne amintim că în modelul Kerberos fundamental un client solicită un TGT de la AS iar clientul poate utiliza acest TGT pentru a solicita mai departe bilete de la TGS. În modelul SESAME se foloseşte o tehnică similară pentru autorizare şi controlul accesului. Dacă un client doreşte să utilizeze un serviciu, el nu trebuie doar autentificat de AS, ci drepturile sale autentificate de un server de privilegii (PAC). În principiu, un PAC constă atât din privilegiile utilizatorilor cât şi informaţia de control corespunzătoare. Privilegiile utilizatorilor sunt date precum identitatea utilizatorului, rolul, grupul organizaţional, permisiunile de securitate, iar informaţiile de control spun unde şi când se poate folosi PAC-ul şi dacă poate fi delegat sau nu. PAC este conceptual similar cu un certifica de control al accesului, aşa cum se specifică în (ISO / IEC, 1993b). În modelul SESAME, un PAS generează un PAC la prezentarea unei dovezi de autentificare, iar PAC-ul este semnat digital cu cheia privată a PAS-ului corespunzător.

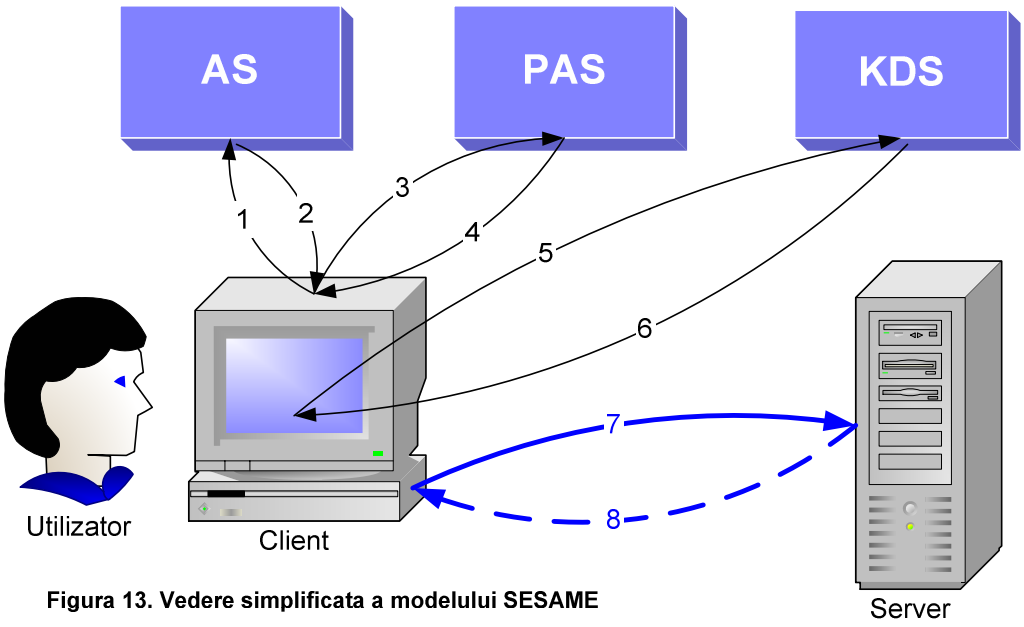

Figura 13. Vedere simplificata a modelului SESAME

Figura 13 ilustrează modelul SESAME simplificat. Se observă că acesta este foarte similar cu modelul Kerberos, diferenţa fiind că PAS este adăugat ca o a treia componentă a serverului de securitate iar TGS-ul se numeşte server de distribuţie a cheilor (KDS).

Pe scurt, un client solicită un bilet PAS de la serverul Kerberos V5 extins AS. Dacă clientul primeşte biletul, îl foloseşte pentru a solicita un PAC şi un bilet pentru KDS de la PAS. Dacă clientul primeşte atât PAC-ul cât şi biletul KDS, acesta îl va folosi pentru a obţine un bilet de serviciu şi informaţiile pentru utilizare. În cele din urmă clientul se poate autentifica serverului, iar dacă este necesară autentificarea reciprocă, serverul se poate autentifica clientului.

Mesajele schimbate între client şi serverele de aplicaţii şi securitate sunt enumerate mai jos:

Pasul	Mesaj
1.	KRB_AS_REQ
2.	KRB_AS_REP

3.	KRB_PAS_REQ
4.	KRB_PAS_REP
5.	KRB_TGS_REQ
6.	KRB_TGS_REP
7.	SES_INIT_CTXT
8.	SES_INIT_CTXT_COMPLETE

Arhitectura SESAME V3 este de fapt mai complicată decât modelul simplificat arătat în figura 13. Într-un domeniu SESAME, AS,PAS şi KDS sunt procese care rulează pe un host în particular, acesta fiind numit server de securitate a domeniului SESAME (DSS).

Un DSS menține o bază de informații a managementului securității (SMIB[1]) care memorează toate datele legate de securitate legate de AS, PAS şi KDS. În plus față de aceasta, fiecare domeniu de securitate SESAME rulează propria autoritate locala de înregistrare (LRA[2]) pe care utilizatorii o pot contacta pentru a obține informații de la autoritatea de certificare (CA). De remarcat că SESAME V3 foloseşte criptografia cu chei publice, deci trebuie să ofere o posibilitate de a genera şi distribui certificate de chei publice.

În SESAME V3, CA-ul se presupune că rulează într-un domeniu propriu care poate să nu aibă nici o legătură cu alte domenii de securitate. Mai mult, CA-ul poate lucra off-line şi ar putea sa nu fie accesibil direct din rețea. În schimb, legăturile între LRA-uri și CA-ul off-line sunt stabilite prin agenți de certificare (CAA[3]) localizați în domeniul CA.

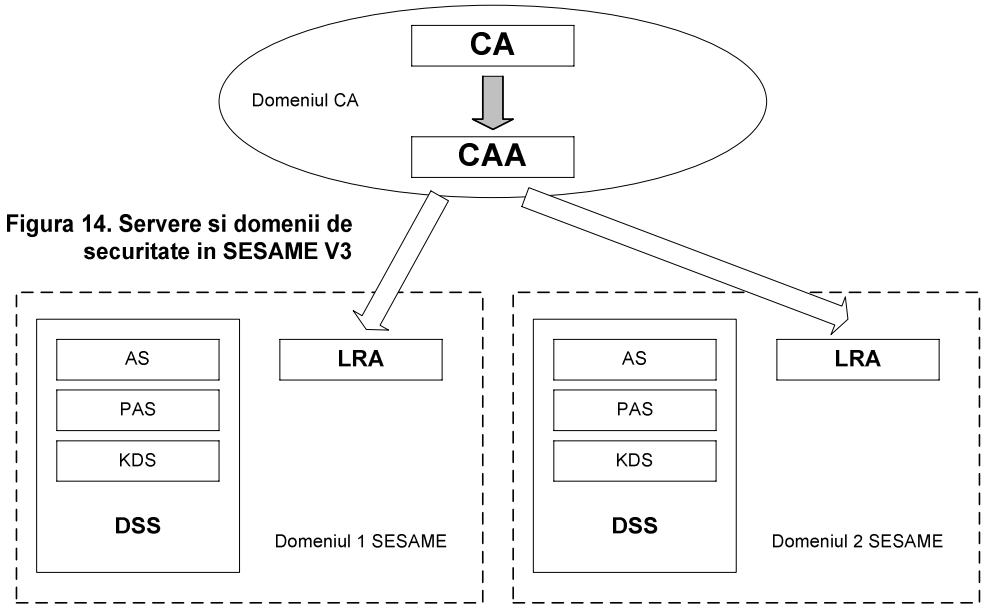

Figura 14. Servere si domenii de securitate in SESAME V3

Văzut din punctul de vedere al CA, un server CAA furnizează şi primeşte date de la administratorul CA. Din punctul de vedere al LRA, un server CAA se comportă atât ca un tampon de solicitări semnate către CA cât şi ca agent pentru alimentarea cu perechi de chei certificate de CA. În principiu, serverele CA şi CAA pot folosi orice mecanism de

[1] Security Management Information Base
[2] Local Registration Authority
[3] Certificate Authority Agent

transfer de fişiere pentru a schimba date. În SESAME V3, transferul de fişiere are loc utilizând medii cum ar fi discurile flexibile.

Diferitele servere şi domenii de securitate sunt ilustrate în figura 14.

Forma ovală reprezintă domeniul CA, în timp ce cele două dreptunghiuri reprezintă două domenii SESAME. Domeniul CA constă dintr-un server CA (off-line) şi un server CAA (online), precum şi o linie de comunicaţie asincronă între cele două. Pe de altă parte, fiecare domeniu de securitate SESAME constă dintr-un server DSS şi unul LRA, cu DSS-ul constituit din AS, PAS şi KDS. Conexiunea dintre LRA şi CAA se stabileşte printr-o cale de comunicaţie sincronă.

Din punctul de vedere al aplicației de rețea, arhitectura SESAME V3 este ilustrată în figura 15.

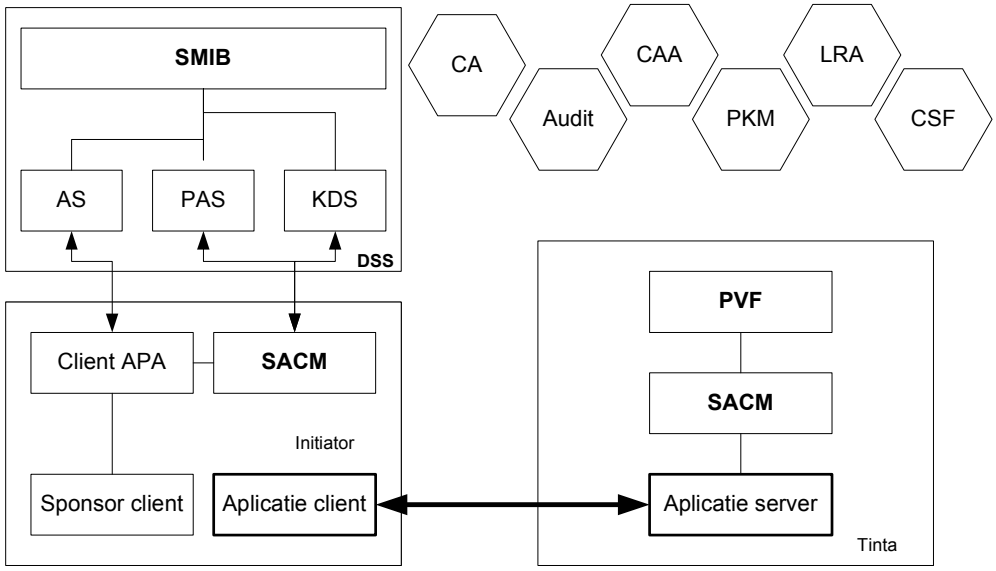

Figura 14. Arhitectura SESAME V3 din punctul de vedere al unei aplicatii de retea

Să presupunem că o aplicaţie client care rulează pe maşina iniţiator vrea să se autentifice maşinii ţintă. Atât iniţiatorul cât şi ţinta pot (dar nu trebuie) să fie înregistraţi în acelaşi domeniu de securitate SESAME. Aici sunt implicate mai multe componente arhitecturale ale sistemului SESAME, acestea trebuind a fi instalate pe partea iniţiatorului:

- Sponsorul utilizatorului (US[1]) furnizează o interfaţă cu sistemul. În SESAME V3 este furnizată doar o linie de comandă simplă care permite utilizatorului să se logineze (seslogin), să schimbe atribute de control şi privilegiile (chattr) şi să termine sesiunea (seslogout). US poate fi schimbat la nevoie de o componentă mai sofisticată.
- Atributul de autentificare şi privilegiu (APA[2]) este o componentă arhitecturală utilizată de US pentru a ascunde detaliile autentificării iniţiale şi achiziţia PAC implicită.
- În general, managerul de context (SACM[3]) este responsabil pentru stabilirea şi menţinerea unui context de securitate între clientul şi serverul unei aplicaţii particulare.

[1] User Sponsor
[2] Authentication and Privilege Attribute
[3] Secure Association Context Manager

Evident, componenta SACM trebuie instalată şi pe partea țintă. În plus față de acestea, trei dintre componente sunt necesare în diferite locuri în tabloul general SESAME V3:

1. Managementul cheilor publice (PKM[1]) constă într-o bibliotecă ce oferă acces la certificate variate şi funcții de manipulare ale cheilor, precum şi un set de instrumente în linia de comandă.
2. Facilitatea de suport criptografic (CSF[2]) implementează algoritmi criptografici utilizați fie de componentele SESAME sau de alte aplicații. Algoritmii implementați curent şi utilizați în SESAME V3 sunt DES-CBC, RSA, MD5 şi DES-MD5. CSF-ul a fost proiectat în aşa fel încât algoritmii pot fi înlocuiți iar mărimile cheilor ajustate în funcție de legislația locală. Din motive de control al exportului, versiunea publică a sistemului SESAME foloseşte un simplu XOR pentru a codifica datele.
3. Evenimentele de audit sunt acțiuni de securitate relevante care apar într-un sistem şi merită să fie înregistrate pentru analiză ulterioară. În SESAME V3, facilitatea de audit memorează evenimentele printr-o conductă numai pentru scriere către un proces care rulează sub identificare proprie. Astfel, fişierul jurnal este protejat de modificare de către aplicațiile proces. Analiza jurnalului este în afara scopului sistemului SESAME.

SESAME foloseşte o ierarhie de chei cu două niveluri, constând în chei simple şi chei de dialog.

- O *cheie simplă* este stabilită şi utilizată între un SACM inițiator şi PVF-ul SACM-ului țintă, pentru a proteja PAC-urile corespunzătoare precum şi informațiile de stabilire a cheilor.
- O *cheie de dialog* este derivată din cheia simplă folosind o funcție de dispersie cu sens unic. Scopul acesteia este de a proteja datele schimbate într-un context de securitate. Pentru protecția integrității şi a confidențialității se pot stabili chei de dialog separate, permițând ca mecanisme cu puteri de criptare diferite să fie utilizate conform cu legislația locală.

3.3. Protocolul criptografic

Referindu-ne la figura 13 şi tabelul mesajelor SESAME, vom prezenta acum pe scurt protocolul criptografic pe care se bazează sistemul SESAME V3. În loc de o prezentare formală a protocolului vom face o descriere a sa.

Când un client se află la o stație de lucru pentru a se logina, el trebuie să prezinte sponsorului US, numele, parola şi rolul solicitat. US înaintează această informație clientului APA în pasul 1 al protocolului SESAME, acesta trimițând mesajul KRB_AS_REQ lui AS, solicitând un TGT. Formatul mesajului este aproximativ acelaşi cu cel din cadrul protocolului Kerberos. AS generează un bilet PAS şi o cheie simplă corespunzătoare şi returnează mesajul KRB_AS_REP lui APA în pasul 2.

Schimbul Kerberos AS are ca rezultat achiziția unui bilet PAS şi a cheii simple corespunzătoare, care sunt depozitate în SACM la partea inițiatoare. De acum înainte, toate interacțiunile între DSS şi client sunt efectuate de SACM-ul inițiator.

[1] Public Key Management
[2] Cryptographic Support Facility

În pasul 3, SACM-ul inițiator trimite un mesaj KRB_PAS_REQ către PAS. Mesajul include atât biletul PAS cât şi rolul utilizatorului. Luând în calcul rolul solicitat, PAS-ul generează un PAC, îl semnează cu cheia sa privată şi în plus generează un bilet KDS.

În pasul 4, PAS-ul returnează un mesaj KRB_PAS_REP SACM-ului inițiator, iar acest mesaj include atât PAC cât şi biletul KDS. În plus față de acestea, mesajul poate include valori de control necesare dacă PAC-ul este delegabil. Mesajul KRB_PAS_REP este codificat cu cheia simplă cunoscută de SACM-ul inițiator şi de PAS.

SACM-ul inițiator păstrează toate informațiile primite până acum, în speță biletele PAS şi KDC, PAC-ul, valorile de control şi cheia simplă. Un program de manipulare a utilizatorilor poate accesa atributele în PAC prin apeluri API corespunzătoare, aşa încât spre exemplu, un utilizator poate fi informat cu ce privilegii lucrează în mod curent. Cu condiția ca aceşti paşi de inițializare au fost parcurşi cu succes, orice aplicație pe partea inițiatorului poate invoca componenta SACM prin apeluri corespunzătoare. Dacă pentru orice motiv PAC-ul memorat de SACM nu este valid sau este nepotrivit, SACM solicită un nou PAC de la PAS.

Dacă un client doreşte să folosească un server de aplicații specific pe partea țintă, el cere SACM-ului inițiator să solicite un bilet corespunzător prin trimiterea mesajului KRB_TGS_REQ către KDS în pasul 5. Dacă KDS are o cheie secretă cu PVF-ul corespunzător SACM-ului țintă, KDS-ul returnează un mesaj KRB_TGS_REP către SACM-ul inițiator în pasul 6.

În pasul 7, SACM-ul inițiator generează un mesaj SES_INIT_CTXT care conține un bilet de serviciu, un pachet conținând semințele cheilor de integritate şi confidențialitate, precum şi PAC-ul. Când protocolul de comunicație transmite marcajul de stabilire a contextului de la SACM-ul inițiator la cel țintă, acesta din urmă îl înaintează PVF-ului său pentru verificare. PVF-ul procesează informația pentru a extrage cheia simplă şi utilizează pachetul cheii de dialog pentru a genera două chei de dialog, şi anume cheia de confidențialitate şi cheia de integritate. Dacă biletul este valid şi dacă se solicită autentificare reciprocă, SACM-ul țintă returnează SACM–ului inițiator în pasul 8, mesajul SES_INIT_CTXT_COMPLETE. Acum este stabilit un context de securitate între SACM-urile inițiator şi țintă, orice aplicație putând folosi acest context pentru transmisie de date. După ce aplicația a terminat, se solicită terminarea contextului prin trimiterea SES_CTXT_ABORT.

3.4. Concluzii

În acest capitol am arătat cum munca la SESAME a extins autentificarea Kerberos şi sistemul de distribuție a cheilor prin utilizarea criptografiei cu chei publice şi cum aceasta prezintă avantaje în privința managementului şi a scalabilității. Sistemul de autentificare şi distribuție a cheilor care a rezultat, SESAME V3, nu oferă doar autentificare, confidențialitatea şi integritatea datelor, ci şi servicii de autorizare şi control al accesului. Drept urmare, SESAME este o alternativă interesantă la Kerberos.

Un alt avantaj este faptul că SESAME nu este un sistem proprietar. În schimb, rădăcinile sale sunt în munca ECMA şi nu este legat de o platformă specifică sau protocol anume. De la început, SESAME a fost proiectat pentru un mediu multi-vendor, iar acest obiectiv se dovedeşte util când se realizează integrarea unui sistem în organizații mari sau clustere de organizații cooperante. [OPPL96]

Capitolul 4

Protocolul NetSP

În acest capitol vom discuta despre sistemul de autentificare şi distribuţie a cheilor NetSP[1], dezvoltat de IBM. Facilitatea distinctivă este folosirea unei funcţii de dispersie cu sens unic în locul unui criptosistem real. În secţiunea 4.1 vom face prezentarea sistemului, iar în 4.2 ne concentrăm asupra protocoalelor criptografice implementate de NetSP. În cele din urmă, în secţiunea 4.3 vom analiza aplicabilitatea protocoalelor criptografice în afara scopului propus iniţial de NetSP.

4.1. Prezentare

În general, o *funcţie de dispersie cu sens unic cu cheie* este o funcţie parametrizată de o cheie secretă. O astfel de funcţie se poate utiliza pentru a adăuga redundanţă controlată informaţiei şi pentru a proteja astfel integritatea acesteia. Folosirea funcţiilor de dispersie cu sens unic cu cheie a fost propusă independent de Cohen pentru a proteja integritatea software-ului [COHE87] şi de Gong şi Tsudik pentru a calcula şi verifica codurile de autentificare ale mesajelor [GONG89, TSUD92]. Unele încercări preliminare pentru folosirea acestor funcţii au fost in SNMP[2], versiunea 2.

Înainte de aceste propuneri, metoda obişnuită de calcul şi verificare a unui MAC era de a utiliza un criptosistem cu chei secrete, cum ar fi DES, pentru a codifica mesajul în modul CBC şi utilizarea ultimului bloc ca MAC. Ideea folosirii funcţiei de dispersie cu sens unic constă în utilizarea unei funcţii rezistente la coliziuni cum ar fi MD4, MD5 sau SHS pentru a genera MAC-ul. Pentru concatenare se pot folosi mai multe funcţii, iar prefixul şi sufixul secret au fost propuse în [TSUD92]. Studiile au arătat că aceste metode au slăbiciuni care pot fi exploatate de unele atacuri sofisticate [KALI95, PREN95].

Un avantaj substanţial al funcţiilor de dispersie cu sens unic faţă de tehnicile care folosesc criptarea pentru autentificarea mesajului este viteza. Un alt mare avantaj este că sursele şi implementarea lor nu sunt supuse restricţiilor de export ale Statelor Unite.

[1] Network Security Program
[2] Simple Network Management Protocol

Ambele avantaje sunt foarte atractive din punctul de vedere al fabricantului şi a comerciantului.

Ideea utilizării funcțiilor de dispersie pentru autentificarea mesajelor şi utilizarea MAC-ului pentru autentificare şi distribuția cheilor a condus la crearea unui proiect comun între IBM Research Laboratories din Zürich, Elveția şi Thomas J. Watson Research Center din Yorktown Heights, New York. Obiectivul proiectului era să proiecteze o familie de protocoale de autentificare şi distribuție a cheilor care să fie minimale, flexibile şi scalabile, şi să poată fi folosite în diferite medii de rețea, precum şi să arate rezistența acestor protocoale la atacuri specifice.

În general este de dorit a se obține protocoale minimale şi flexibile datorită părerii generale că mecanismele de securitate sunt scumpe, cu performanță slabă şi cumva greoaie. De fapt această viziune face ca utilizatorii singulari sau chiar organizațiile ajung să evite, dezactiveze sau scurtcircuiteze soluțiile de securitate implementate.

Minimalitatea protocolului se poate exprima în mai multe moduri, cum ar fi numărul de mesaje ce se schimbă în timpul execuției protocolului, dimensiunea fiecărui mesaj sau numărul de operații criptografice. Minimalitatea protocolului este răsplătită când măsurile de securitate trebuie să fie compatibile cu sistemul anterior şi se cere – spre exemplu – ca anumite facilități să fie introduse într-un spațiu restrâns, cum ar fi un header.

Familia de protocoale uşoare de autentificare şi distribuție a cheilor a fost descrisă prima dată în [BIRD92] şi discutată ulterior în [BIRD93, BIRD95]. Protocoalele sunt eficiente ca dimensiune a mesajelor şi încărcarea computațională şi minime ca tehnici criptografice. În plus, protocoalele s-au arătat rezistente la un set larg de atacuri cunoscute sub numele de *atacuri de întrețesere*[1]. Pe scurt, un astfel de atac se traduce prin posibilitatea unui atacator de a folosi mesaje legitime capturate din execuții anterioare ale protocolului sau mesaje primite de la participanți autorizați.

Unul dintre protocoalele majore din familie se numeşte KryptoKnight [MOLV92]. Datorită numelui, familia de protocoale se numeşte *KryptoKnight* sau *protocoalele KK*.

Astăzi, protocoalele KK sunt folosite de programul de securitate a rețelei (NetSP) dezvoltat de IBM. Acesta constă de fapt din mai multe produse de securitate, cum ar fi:

- NetSP SNG – Secured Network Gateway
- NetSP SLC – Secured Logon Coordinator

NetSP SNG este un firewall în timp ce NetSP SLC este un întreg sistem de autentificare şi distribuție a cheilor. Pentru că această lucrare se referă la astfel de sisteme, când ne referim la NetSP, ne referim implicit la NetSP SLC.

NetSP versiunea 1 (V1) a fost lansată oficial în ianuarie 1994 pentru sistemele de operare AIX/6000, OS/2 şi MS-DOS. În plus față de protocoalele TCP/IP pentru aceste platforme, NetBIOS a fost suportat pe AIX, OS/2 şi MS-DOS, iar LU 6.2 pe AIX şi OS/2. În august 1994 a apărut versiunea 1.2 a sistemului NetSP care suporta IPX/SPX pe OS/2 şi DOS/Windows pentru serverele de autentificare şi clienții care rulau Novell Netware.

4.2. Protocoale criptografice

În această secțiune vom face o prezentare generală a protocoalelor KK. În particular, ne vom îndrepta atenția în autentificarea a două părți şi distribuția cheilor,

[1] Interleaving Attacks

distribuția cheilor a trei părți, distribuția cheilor inter-domeniu şi protocoalele *single sign-on*.

4.2.1. Autentificarea a două părți

Protocolul de autentificare a două părți (2PAP[1]) este la baza protocoalelor KK. Acesta specifică protocolul provocare – răspuns care permite ca doi participanți A şi B să se autentifice reciproc. 2PAP poate fi formalizat după cum urmează:

1. A → B : N_a
2. B → A : N_b, (N_a, N_b, B) K_{ba}
3. A → B : (N_a, N_b) K_{ab}

În această descriere a protocolului, originea şi receptorul mesajului sunt omise în corpul mesajelor deoarece pot fi deduse din informația de adresare din rețea. N_a şi N_b sunt valori alese aleator de către A respectiv B. Notația (m) K înseamnă un MAC calculat din mesajul m şi codificat cu cheia K. Similar, (N_a, N_b, B) K_{ba} se referă la un MAC calculat prin concatenarea componentelor N_a, N_b şi B şi codificarea acestora cu cheia secretă K_{ba}. Dacă atât (N_a, N_b, B) K_{ba} cât şi (N_a, N_b) K_{ab} sunt implementate cu un sistem de chei private, K_{ba} şi K_{ab} sunt acelaşi lucru. Dacă in schimb se utilizează DES în modul CBC pentru autentificare, atunci termenul (N_a, N_b, B) K_{ba} abreviază de fapt {{{N_a}K_{ba} $\oplus$ N_b} K_{ba} $\oplus$ B} K_{ba}, iar (N_a, N_b) K_{ab} abreviază {{N_a} K_{ab} $\oplus$ N_b} K_{ab}. În loc de a utiliza un criptosistem pentru a calcula şi verifica MAC-urile, se poate utiliza o funcție de dispersie cu sens unic cu cheie în loc, aşa cum s-a menționat anterior. În acest caz, (N_a, N_b, B) K_{ba} se referă la h(K_{ba}, N_a, N_b, B) pentru metoda prefixului secret, h(N_a, N_b, B, K_{ba}) pentru metoda sufixului secret şi h(K_{ba}, N_a, N_b, B, K_{ba}) pentru metoda plicului.

Întorcându-ne la 2PAP, A oferă lui B o valoare N_a în pasul 1 şi B răspunde cu o altă valoare Nb împreună cu (N_a, N_b, B) K_{ba} în pasul 2. A poate utiliza acum K_{ba} pentru a verifica MAC-ul şi pentru a-l autentifica pe B. Dacă B se consideră a fii autentic, A îi oferă lui B cantitatea (N_a, N_b) K_{ab} care este un alt MAC calculat din concatenarea lui N_a şi N_b cu cheia secretă K_{ab} în pasul 3. B poate folosi acum K_{ab} pentru a verifica MAC-ul şi pentru a-l autentifica pe A.

Dezvoltatorii protocolului 2PAP original au arătat că acesta este într-adevăr rezistent la atacuri prin întrețesere şi toate protocoalele KK moştenesc această proprietate. u alte cuvinte, dacă se poate dovedi că un protocol este echivalent cu 2PAP, putem spune că moşteneşte proprietățile acestuia, inclusiv rezistența la atacuri de întrețesere.

4.2.2. Distribuția cheilor între două părți

Obiectivul protocolului de distribuție a cheilor între două părți (2PKDP) este să permită unei părți A care o cheie K_a cunoscută de un centru de distribuție a cheilor (KDC) să obțină o nouă cheie K_a^{new}. Presupunerile care au condus la proiectul protocoalelor KK de distribuție a cheilor sunt după cum urmează:

- O cheie este întotdeauna generată de o singură parte
- O cheie care se distribuie trebuie să fie o cantitate aleatoare şi imprevizibilă. Dacă A şi B se angajează într-un protocol de distribuție a cheilor şi A generează o cheie, atunci nici B şi nici altă parte nu trebuie să fie capabilă de a deduce cheia.
- Cheile nu se reutilizează niciodată.

[1] Two-party Authentication Protocol

Aceste presupuneri au dus la observația importantă că o cheie este conceptual similară cu o valoare unică. De fapt, singura diferență între o cheie nouă distribuită în 2PKDP şi o valoare folosită in 2PAP este că valoarea se comunică în clar în timp ce cheia trebuie comunicată secretizat. Familia de protocoale KK cuprinde astfel două 2PKDP-uri: primul oferă confidențialitatea cheilor şi altul oferă integritatea şi autenticitatea cheilor.

Protocolul de distribuție a cheilor între două părți (2PKDP[1])

Având în vedere cele expuse anterior, protocolul 2PKDP se poate deriva uşor din 2PAP prin simpla înlocuire a lui B cu KDC. Protocolul rezultat se poate formaliza după cum urmează:

1. A → KDC : N_a
2. KDC → A : N_k, $(N_a, N_k, KDC)\ K_a \oplus K_a^{new}$
3. A → KDC : $(N_a, N_k)\ K_a$

Similar cu protocolul 2 PAP, A oferă KDC o valoare N_a în pasul 1. KDC selectează aleator o altă valoare N_k şi o nouă cheie secretă K_a^{new} pentru A, calculând apoi un MAC din ambele valori şi identificatorul propriu, rezultatul adunându-se modulo 2 la noua cheie secretă K_a^{new}. În pasul 2, KDC trimite lui A atât N_k cât şi $(N_a, N_k, KDC)\ K_a \oplus K_a^{new}$. De remarcat că în acest punct că un atacator care e capabil să captureze mesajul nu poate să extragă nici o informație dacă nu cunoaşte cheia secretă a lui A, K_a. A poate utiliza acum K_a şi N_k pentru a calcula $(N_a, N_k, KDC)\ K_a$ şi foloseşte acest MAC pentru a extrage K_a^{new} din $(N_a, N_k, KDC)\ K_a \oplus K_a^{new}$. În pasul 3, A poate trimite lui KDC un alt MAC calculat din ambele valori N_a şi N_k cu K_a. De remarcat că spre deosebire de 2PAP, pasul 3 nu este neapărat necesar în 2PKDK.

Protocolul de distribuție a cheilor între două părți autentificate (2PAKDP[2])

Protocolul 2PKDP descris anterior este sigur în sensul confidențialității cheilor, aceasta însemnând că nimeni în afară de KDC şi A nu poate obține K_a^{new} [TSUD93]. Totuşi, protocolul nu este nu este sigur în sensul integrității cheilor şi al autenticității. Cu alte cuvinte, un intrus ar putea modifica mesajul returnat de KDC către A în pasul 2 şi să provoace extragerea unei chei false din mesaj. Problema rezidă în faptul că A nu are nici o metodă să afle dacă mesajul recepționat este întreg şi autentic, sau mai mult, KDC-ul ar putea să nici nu fie disponibil şi întreg mesajul să fi fost produs de un intrus.

În acest moment trebuie remarcat că această vulnerabilitate nu este un eşec al protocolului, deoarece un intrus nu poate schimba cheia la o valoare cunoscută. Tot ce poate face intrusul este să modifice cheia cu o cantitate aleasă aleator, dar această vulnerabilitate este rezolvată în protocolul 2PAKDP:

1. A → KDC : N_a
2. KDC → A : MAC(A), $\{MAC(A)\}\ K_a \oplus N_k$
3. A → KDC : $(N_a, N_k)\ K_a$

Protocolul 2PAKDP este foarte similar cu 2PKDP. Notația MAC(A) abreviază $(N_a, N_k, KDC)\ K_a$.

[1] Two-Party Key Distribution Protocol
[2] Two-Party Authenticated Key Distributed Protocol

4.2.3. Distribuția cheilor între trei părți

Obiectivul general al protocolului de distribuție a cheilor între trei părți (3PKDP) este distribuirea unei chei de sesiune de la un KDC către A şi B. Un protocol 3PKDP sigur trebuie să satisfacă aceleaşi condiții ca şi protocolul 2PKDP cu condiția suplimentară ca nici una dintre părți să nu fie capabilă să altereze cheia distribuită.

Din moment ce inițial A şi B nu au în comun nici o cheie secretă care ar putea fi folosită ca o cheie de sesiune, aceştia trebuie să contacteze KDC, cu care fiecare are o cheie secretă. KDC-ul oferă părților A şi B bilete. În NetSP, un *bilet* se referă la un mesaj trimis de un KDC într-un mesaj al 3PKDP. Această terminologie diferă uşor de cea folosită de Kerberos.

În legătură cu distribuția cheilor celor trei părți, în general nu este necesar ca ambele părți să contacteze KDC-ul. În funcție de cine contactează primul KDC-ul şi în funcție de modelul folosit (împinge sau trage), au fost proiectate mai multe protocoale de tip 3PKDP, pe care le vom prezenta în continuare.

Scenariul A-B-KDC

În scenariul A-B-KDC se presupune că A este fie incapabil, neautorizat său nu vrea să contacteze KDC, şi că B trebuie să îl înlocuiască pe A. Acest scenariu se poate realiza prin combinarea a două execuții 2PAKDP (una între A şi KDC, una între B şi KDC) şi o execuție a 2PAP (între A şi B). Protocolul 3PKDP este formalizat după cum urmează:

1. A → B : N_a
2. B → KDC : N_a, N_b, A
3. KDC → B : MAC(A), {MAC(A)}$K_a \oplus K_{ab}$: MAC(B), {MAC(B)}$K_b \oplus K_{ab}$
4. B → A : MAC(A), {MAC(A)}$K_a \oplus K_{ab}$: N_b, (N_a, N_b, B) K_{ab}
5. A → B : (N_a, N_b) K_{ab}, [(N_a, K_{ab}) K_a]
6. B → KDC : [(N_a, K_{ab}) K_a, (N_b, K_{ab}) K_b]

Scenariul KDC-A-B

În scenariul KDC-A-B, A îl contactează pe B înainte de a solicita bilete de la KDC. Acest scenariu se poate formaliza după cum urmează:

1. A → B : N_a
2. B → A : N_b, (N_a, N_b, B) K_b
3. A → KDC : N_a, N_b, B, (N_a, η_b, B) K_a, A unde η_b = (N_a, N_b, B) K_b
4. KDC → A : MAC(A), {MAC(A)} Ka $\oplus$ Kab, MAC(B), {MAC(B)} Kn $\oplus$ Kab
5. A → B : MAC(B), {MAC(B)}$K_b \oplus K_{ab}$, [(N_a, N_b, B) K_{ab}]

4.2.4. Distribuția cheilor inter-domenii

În principiu, toate protocoalele 3PKDP se pot extinde pentru a suporta distribuția cheilor inter-domenii. Dacă ne amintim de modelul Kerberos, este sarcina clientului să solicite toate biletele necesare pentru a contacta un server într-un domeniu îndepărtat. Acest lucru presupune relativ multă muncă din partea clienților, ceea ce face Kerberos potrivit pentru medii locale. Totuşi, există unele medii unde această abordare se dovedeşte

greoaie sau nu se poate aplica deloc. Obiectivul fixat inițial, acela de maximă flexibilitate a condus la proiectarea protocolului KK de distribuție inter-domeniu a cheilor.

Asemănător cu modelul Kerberos, se presupune că există deja cheile inter-domeniu. O cheie inter-domeniu se referă la o cheie cunoscută de KDC-uri şi de domeniile administrative şi de autentificare. Dacă centrele de distribuție ale cheilor se numesc KDC_1 şi KDC_2, termenul K_{12} se referă la cheia inter-domeniu folosită de cele două în autentificarea şi distribuția cheilor. Deşi folosirea literelor majuscule implică folosirea criptografiei cu chei secrete, nu este nici un motiv pentru care criptografia cu chei publice să nu poată fi folosită. De fapt, utilizarea criptografiei cu chei publice poate fi transparentă pentru părți şi doar KDC-ul să fie implicat. Pe lângă multele avantaje, cel mai clar dezavantaj este mărirea lungimii mesajelor şi a dimensiunii pachetelor.

Comunicarea fără / cu KDC

În cazul cel mai simplu, se presupune că KDC-urile nu participă la comunicația inter-domeniu. De remarcat că în ciuda simplității, această presupunere este realistă deoarece în mod normal KDC-ul este un server bine protejat şi comunicarea cu lumea exterioară ar fi doar un risc asumat gratuit. În plus, comunicarea inter-domenii este mai apare de mai puține ori şi este mai nesigură decât comunicația principală. Astfel, încărcarea asupra KDC-ului poate fi redusă mult prin interzicerea comunicației inter-domenii.

Dacă comunicarea directă între KDC-uri nu este posibilă, soluția evidentă este ca unul dintre ele să emită ambele bilete, iar celălalt să le translateze în bilete valabile în propriul domeniu.

Sunt situații în care comunicația între KDC-uri este necesară. Spre exemplu, una dintre părți ar putea să nu aibă o conexiune directă cu KDC-ul său sau altul străin. Comunicația între KDC-uri este mai avantajoasă dacă legătura este mai rapidă, mai ieftină sau chiar mai securizată, astfel că ele pot comunica mai bine decât cu părțile.

Modificările aduse protocolului 3PKDP pentru a suporta comunicarea prin KDC-uri sunt două mesaje adiționale între KDC-urile implicate. Paşii suplimentari sunt transparenți pentru părți iar acestea nu trebuie să diferențieze între KDC-ul propriu sau cel din alt domeniu. Spre exemplu, protocolul 3PKDP se modifică astfel pentru a suporta comunicarea inter-domenii:

1. A → B : N_a
2. B → KDC_2 : N_a, N_b, A
3. KDC_2 → KDC_1 : N_a, N_b, A, B
4. KDC_1 → KDC_2 : MAC(A), {MAC(A)} $K_a \oplus K_{ab}$, MAC(12), {MAC(12)} $K_{12} \oplus K_{ab}$
5. KDC_2 → B : MAC(A), {MAC(A)} $K_a \oplus K_{ab}$, MAC(B), {MAC(B)} $K_b \oplus K_{ab}$
6. B → A : MAC(A), {MAC(A)} $K_a \oplus K_{ab}$, (N_a, N_b, B) K_{ab}, N_a, N_b, A
7. A → B : (N_a, N_b) K_{ab}

Abrevierile MAC(A), MAC(B), MAC(12) sunt aceleaşi cu cele din protocolul 3PKDP anterior.

4.3. Concluzii

Protocoalele KK se pot utiliza pentru a furniza servicii de autentificare şi distribuţie ale cheilor, acestea putând fi folosite ulterior pentru asigurarea confidenţialităţii datelor şi servicii de integritate.

La proiectarea protocoalelor KK s-a urmărit ca acestea să fie compacte, flexibile iar exportabilitatea să constituie o soluţie atractivă pentru securizarea aplicaţiilor existente şi ale sistemelor de comunicaţie la orice strat al protocolului, fără a ţine seama de configuraţia reţelei sau paradigma de comunicaţie.

IBM a propus o arhitectură care să furnizeze securitate la nivelul de reţea. Inima acestei arhitecturi sunt două protocoale, şi anume IPST[1] şi MKMP[2]:

- IPST urmează spiritul discuţiilor în grupul de lucru IPSEC[3] din IETF[4]. Pentru a proteja integritatea şi autenticitatea unei datagrame IP, la header-ul acesteia se calculează şi se ataşează un MAC. Pentru a proteja confidenţialitatea unei datagrame IP se utilizează încapsularea IP. În consecinţă, o datagramă IP este codificată şi plasată în altă datagramă înaintea transmisiei. Receptorul se presupune că deţine cheile necesare să decodifice datagrama şi să verifice MAC-ul în consecinţă.
- MKMP se referă la un protocol sau un set de protocoale pentru managementul cheilor criptografice aşa cum este cerut de managementul asociaţiilor de securitate în IPST. Protocoalele oferă mecanisme de securitate pentru reîmprospătarea periodică a cheilor şi derivarea cheilor de lucru.

MKMP abordează lucrurile într-o manieră ierarhică. In particular, se presupune că două entităţi IPST A şi B deja cunosc împreună o cheie master K_{AB} de lungă durată şi că pot folosi această cheie pentru a negocia o cheie de sesiune de scurtă durată K_{ab}. De remarcat că protocolul de negociere a cheii de sesiune rezolvă o problemă foarte similară cu cea din protocoalele 2PKDP şi 2PAKDP din familia KK.

Să presupunem că A şi B cunosc împreună o cheie de sesiune K_{AB} şi o valoare N_b dintr-o execuţie anterioară a protocolului. În acest caz, părţile pot utiliza protocolul MKMP pentru a negocia o cheie de sesiune de scurtă durata, K_{ab}, după cum urmează:

1. A → B : N_a, (N_a, N_b) K_{AB}
2. B → A : N_b', (N_b', N_a) K_{AB}

În pasul 1, A selectează aleator o valoare N_a, utilizează K_{AB} să calculeze (N_a, N_b) K_{AB} şi trimite lui B valoarea anterioară şi N_a. În pasul 2, B selectează altă valoare N_b' şi foloseşte KAb să calculeze (N_b', N_a) K_{AB} şi trimite totul înapoi lui A. A şi B înlocuiesc N_b şi N_b' pentru a fi folosite la următoarele execuţii ale protocolului. Pentru a deriva cheia de sesiune se foloseşte o funcţie f , $K_{AB} = f_{AB}(N_b', N_a)$. Funcţia f este legată de cheia master de lungă durată K_{AB}. Acest lucru este necesar deoarece ambele argumente ale funcţiei sunt transmise în clar în paşii 1 şi 2 ai protocolului MKMP.

IBM a implementat IPST şi MKMP [CHEN95]. Implementarea se vinde ca parte integrantă a NetSP SNG, care este un produs firewall. In plus, IBM a propus MKMP

[1] IP Secure Tunnel Protocol
[2] Modula Key Management Protocol
[3] Internet Protocol Working Group
[4] Internet Engineering Task Force

grupului de lucru IPSEC al IETF pentru o posibilă includere ca standard Internet, însă propunerea a fost respinsă în favoarea altor protocoale cum ar fi SKIP (Simple Key-Management Protocol) şi Photuris[1] Key Management Protocol.

Nevoia de a interconecta domenii eterogene de administrare care utilizează protocoale diferite pentru autentificare şi distribuție a cheilor este o realitate, dar şi o sarcină dificilă. Proiectarea unui astfel de mecanism nu este o sarcină uşoară, iar dacă eterogenitatea afectează şi securitatea, problema interconectării devine şi mai grea. Într-un fel, construirea unui gateway care translatează mesaje criptografice este o contradicție în sine, din moment ce acesta este implicat în transferul securizat de la un capăt la celălalt, făcându-l mai puțin sigur. O primă propunere de model de interconectare între Kerberos şi NetSP a fost prezentată în [PIES93].

[1] *Photuris* este denumirea latină pentru licurici (engl. firefly). Pe de altă parte, Firefly este numele unui protocol secret proiectat de NSA pentru telefonul securizat STU-III.

Capitolul 5

Protocolul SPX

În acest capitol ne vom ocupa de sistemul de autentificare şi distribuţie a cheilor SPX, proiectat şi prototipizat de DEC. În secţiunea 5.1 vom face descrierea dezvoltării sistemului, iar în secţiunea 5.2 ne vom opri asupra vederii de ansamblu a arhitecturii sale. În secţiunile 5.3 şi 5.4 vom descrie protocoalele criptografice implementate de sistemul SPX.

5.1. Prezentare

La sfârşitul anilor 1980, Digital Equipment Corporation (DEC) a propus o arhitectură distribuită de securitate (DSSA[1]) care s-a axat pe servicii de securitate într-un sistem distribuit. [GASS89, LINN90]. În legătură cu folosirea tehnicilor criptografice, DSSA a urmărit o abordare hibridă, adică criptografia cu chei secrete se foloseşte pentru autentificarea originii datelor şi servicii de integritate şi confidenţialitate, iar criptografia cu chei publice pentru servicii de autentificare.

Serviciul de autentificare al DSSA a fost numit DASS[2]. Serviciul a fost prototipizat într-un sistem de autentificare şi distribuţie a cheilor numit *Sphinx* [TARD90], redenumit mai târziu SPX [TARD91]. Chiar şi azi, SPX se pronunţă Sphinx şi nimeni nu a putut oferi semnificaţia reală a acronimului.

SPX foloseşte DES ca sistem de codificare cu chei secrete şi RSA ca sistem de codificare cu chei publice. Sistemul a fost conceput în aşa fel încât înlocuirea sistemului de codificare să se facă cu minime modificări la protocol, posibil nici una. SPX este implementat în reţele TCP/IP, deci ar trebui să fie portabil pe un număr mare de sisteme.[ALAG91a, ALAG91b]

În ciuda faptului că SPX a atras atenţia comunităţii securităţii reţelelor, dezvoltarea sistemului a fost întreruptă după lansarea versiunii 2.4, în decembrie 1992. Astăzi, DEC promovează o implementare proprie a Kerberos V4 şi produse OSF DCE, pentru a satisface nevoile clienţilor.

[1] Distributed System Security Architecture
[2] Distributed Authentication Security Service

Astăzi, SPX este disponibil gratuit şi liber. Atât codul sursă cât şi documentația se poate descărca de la crl.dec.com. Ca şi Kerberos, SPX foloseşte tehnici criptografice care până de curând erau restricționate la export.

În ianuarie 1993, SPX a fost propus ca protocol experimental pentru securizarea serviciului **telnet** în RFC 1412. [ALAG93]. În septembrie 1993 a fost propusă o versiune revizuită în RFC 1507 [KAUF93] care defineşte doar un protocol experimental.

5.2. Arhitectură

SPX distinge între două clase de participanți, şi anume utilizatori umani şi servere. Ambele pot juca rol de pretinzător şi de verificator. Dacă o parte joacă rolul unui *pretinzător*, trebuie să fie recunoscută ca autentică de către un verificator, în timp ce ca *verificator* o parte caută să-l identifice pe pretinzător.

În general, utilizatorii sunt răspunzători pentru ce fac procesele în numele lor în cadrul rețelelor şi al sistemelor distribuite. Procesele se consideră reprezentantele utilizatorilor, deci acestea trebuie autentificate corespunzător. Reprezentarea unui utilizator într-un proces este legată de un set de credențiale, acestea incluzând identitatea utilizatorului şi un o cheie criptografică utilizată la autentificare. În principiu, orice proces care poate arăta că are cunoştință despre această cheie poate să ruleze în contul utilizatorului. Dacă procesul rulează concomitent şi pe alt host, credențialele ar putea fi necesare şi pe host-ul îndepărtat. Ar putea fi necesar să se permită procesului îndepărtat să efectueze accese în numele utilizatorului, ceea ce în SPX se numeşte *delegare*.

În funcție dacă partea în discuție este pretinzător sau verificator, SPX foloseşte credențiale diferite:

- *Credențialele pretinzătorului* sunt valide pentru o perioadă de timp relativ scurtă (de obicei opt ore). Acestea sunt în mod tipic generate şi instalate când utilizatorul se loginează la stația client. Credențialele constau dintr-o cheie privată de delegare pe termen scurt şi un bilet corespunzător. Acesta este un certificat pentru cheia publică de delegare, semnată digital cu cheia pe termen lung al utilizatorului.
- *Credențialele verificatorului* sunt generate şi instalate pe server şi acestea includ în mod direct cheile private corespunzătoare pe termen lung. Aceste credențiale sunt valid pe o lungă perioadă de timp.

Utilizarea criptografiei cu chei publice necesită în general existența şi operarea unei ierarhii de certificări, aceasta fiind în mod ideal cuplată cu un serviciu director şi o modalitate de denumire corespunzătoare. Când SPX a fost proiectat, s-a presupus că o autoritate de certificare a cheilor va fi disponibilă comunității Internet.

Cu excepția reprezentării sintactice ale numelor care e legată de recomandarea X.500 a ITU-T, DASS a fost proiectat să fie independent de serviciul director şi cel de nume. SPX implementează propriul serviciu de distribuție a certificatelor, proiectanții nedorind să introducă dependență de serviciul X.500.

În SPX, certificatele X.509 pentru cheile publice sunt create de o autoritate de certificare (CA) credibilă şi foarte sigură. Pentru a reduce impactul disponibilității reduse, CA-ul generează certificate care sunt stocate şi distribuite de centre specializate (CDC – Certificate Distribution Center). CDC-urile nu trebuie să fie credibile, acestea putând fi replicate pentru o disponibilitate crescută. În principiu, CDC-ul reprezintă un „depozit” de certificate sau alte informații de autentificare pentru părți, simulând multe dintre proprietățile unui serviciu director.

CDC-ul este accesat fie în timpul procesului de login, când se creează credențialele pretinzătorului pentru un client sau când se creează credențialele verificatorului pentru un server. În fiecare caz, CDC-ul furnizează codificat cheia privată a părții. Un alt moment în care CDC-ul este accesat este la crearea sau verificarea semnelor de autentificare în timpul schimbului SPX. În cazul în care CDC-ul este compromis, rezultatul este cel mult o întrerupere a revocării certificatelor sau în întreruperea serviciului pentru părți, nefiind posibilă compromiterea schemei de autentificare per total.

În plus față de CDC-uri, SPX foloseşte LEAF[1] pentru a oferi părților copii codificate ale cheilor lor private pe termen lung şi pentru a le permite înrolarea în CDC. Similar cu CDC-urile, LEAF a fost proiectat într-un mod în care dacă este compromis, permite expunerea cheilor private unui atac de tip ghicire, dar nu ar permite atacatorului să compromită autentificarea per total.

SPX suportă relații de încredere multiple între părți şi CA-uri. Într-o instanță particulară a autentificării, pretinzătorul şi verificatorul nu trebuie să aibă încredere în acelaşi CA, şi nici nu trebuie să aibă încredere în toate CA-urile. Dacă o parte are încredere într-un CA anume, acela se va numi autoritate credibilă (TA – trusted authority) pentru partea respectivă. TA-urile pentru o parte corespunde astfel tuturor CA-urilor credibile pentru aceasta.

Părțile SPX folosesc *simbol de autentificare* pentru a se autentifica reciproc. Asemeni unui bilet Kerberos, un simbol de autentificare transferă în mod securizat cheia DES de sesiune. Dar, spre deosebire de Kerberos, această cheie de sesiune nu este generată de un terț credibil şi nici nu este codificată cu cheia secretă a receptorului. În SPX, un simbol de autentificare este generat de pretinzător şi codificat cu cheia publică a verificatorului. Astfel, părțile trebuie să-şi cunoască reciproc cheile publice recepționate de la CDC. Părțile care s-au autentificat corect pot folosi serviciile de integritate, confidențialitate şi origine a datelor la nivel de mesaj folosind codificarea DES cu cheia de sesiune.

Din punct de vedere al utilizatorului, Kerberos şi SPX au o funcționalitate similară, deci interfețele cu utilizatorul sunt comparabile. Tabela de mai jos cuprinde comenzile SPX implementate de instrumentele „r" de la Berkeley.

Comanda SPX	Acțiune
cdb_destroy	Distruge baza de date CDC
cdb_edit	Editează baza de date CDC
cdb_init	Inițializează baza de date CDC
cdb_list	Listează baza de date CDC
createcertif	Generează un certificat X.509
createkey	Generează un fişier cu cheia RSA
install_server	Instalează credențialele serverelor
spxdestroy	Distruge credențialele
spxinit	Stabileşte credențialele pretinzătorului
spxlist	Listează conținutul credențialelor pretinzătorului
fcp	Copiere fişier (cu autentificare SPX)
flogin	Login (cu autentificare SPX)
fsh	Shell (cu autentificare SPX)

[1] Login Enrollment Agent Facility

5.3. Protocoale criptografice

În următoarele secțiuni ne oprim în detaliu asupra inițializării credențialelor, schimbul de autentificare şi protocolul de înscriere.

5.3.1. Inițializarea credențialelor

Dacă un utilizator doreşte să se delege la stația sa, el trebuie să posede cheia sa privată pentru a instala credențialele pretinzătorului. În general nu este realist să cerem utilizatorilor să îşi reamintească cheile private şi nici să poate *smart card*-uri sau discuri flexibile cu fişiere. S-a menționat deja că SPX are un LEAF care permite utilizatorilor să-şi obțină cheia privată în formă codificată. Rolul LEAF se diminuează pe măsură ce smart card-urile sau alte dispozitive cu funcționalitate similară devin disponibile.

LEAF nu expune în mod direct cheile private ci doar le expune atacurilor de tip ghicire, deoarece acesta nu cunoaşte în mod direct aceste chei ci le memorează codificate cu DES şi derivate din cheia utilizatorului (KEK – key encryption key) cu ajutorul unei funcții cu sens unic. Pentru a ajunge să aibă acces la cheia privată, un client trebuie să se preautentifice, această însemnând că trebuie să arate dovada cunoaşterii parolei. Pentru a permite o astfel de preautentificare, LEAF păstrează cheia privată împreună cu semnătura de dispersie a acesteia, utilizând o altă funcție decât cea utilizată pentru a forma KEK care criptează cheia privată. Ideea este ca dezvăluirea cheii private să se facă doar dacă valoarea de dispersie prezentată de client coincide cu cea memorată.

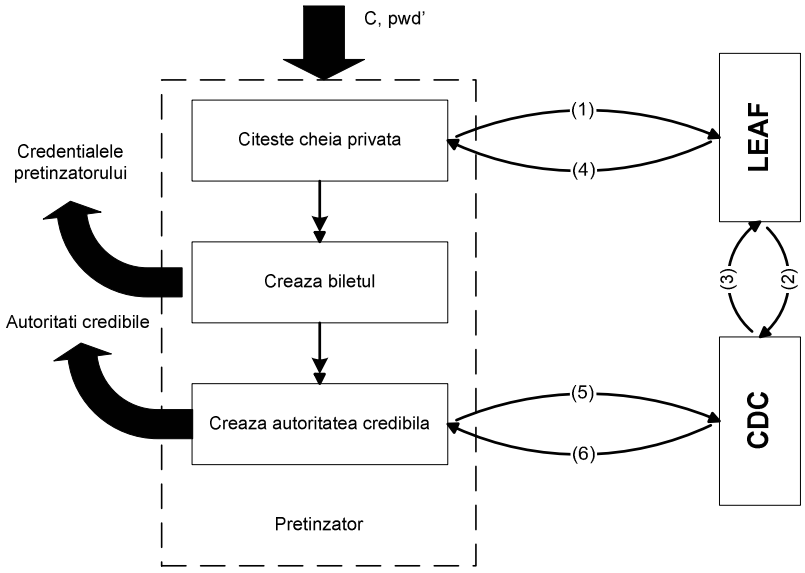

Figura 15. Protocolul SPX de initializare a credentialelor unui pretinzator

Schema protocolului de inițializare a credențialelor este ilustrată în figura 15. Protocolul constă din şase paşi care se pot formaliza după cum urmează:

1. C → LEAF : C, $\{T, N_c, h_1(pwd')\}k_{LEAF}$
2. LEAF → CDC : C
3. CDC → LEAF : $\{\{k_C^{-1}\}h_2(pwd), h_1(pwd)\}K$, $\{K\}_{LEAF}$
4. LEAF → C : $\{\{k_C^{-1}\}h_2(pwd)\}N_c$

5. C → CDC : C
6. CDC → C : C << (TA_C) >>

Protocolul începe cu presupunerea că utilizatorul care vrea se inițializeze credențialele pretinzătorului şi-a introdus deja identificarea sa C şi parola pwd'. C nu este folosit doar pentru a indica identitatea clientului ci este folosit şi în procesul de inițializare. Acest proces se numeşte login. De remarcat apostroful de la pwd, care înseamnă că parola nu e neapărat corectă în acest moment.

În pasul 1, C trimite lui LEAF identificatorul său şi un mesaj codificat cu cheia publică de lungă durată k_{LEAF}. Această cheie se presupune că este instalată pe sistemul folosit de C, cum ar fi într-un fişier de configurare. Mesajul criptat conține amprenta de timp T, o valoare N_c, şi h_1(pwd'), o dispersie a parolei furnizată de C.

În pasul 2, LEAF cere unui CDC să i se furnizeze informațiile de autentificare pentru C. Această informație constă din $\{\{k_C^{-1}\}h_2(pwd)$ şi $h_1(pwd)$.

În pasul 3, CDC sigilează cu DES această informație cu o cheie K aleasă aleator, şi furnizează lui LEAF cantitățile $\{\{k_C^{-1}\}h_2(pwd), h_1(pwd)\}K$ şi $\{K\}_{LEAF}$. Evident, LEAF-ul poate folosi această cheie privată pe termen lung k_{LEAF}^{-1} pentru a decodifica $\{K\}k_{LEAF}$ şi a extrage cheia DES corespunzătoare. LEAF poate folosi K pentru decodificarea $\{\{k_C^{-1}\}h_2(pwd), h_1(pwd)\}K$ şi extragerea componentelor $\{\{k_C^{-1}\}h_2(pwd)$ şi $h_1(pwd)$. Dacă dispersia cu sens-unic a parolei din CDC (care este $h_1(pwd)$) se potriveşte cu valoarea primită de la C (care este $h_1(pwd')$), LEAF ştie că C cunoaşte parola şi este autentic. Altfel se afişează un mesaj de eroare şi se înregistrează acest lucru într-un jurnal. De remarcat că atacul prin ghicire a parolelor necesită cunoaşterea cheii private k_{LEAF}^{-1} sau contactarea LEAF-ului chiar şi pentru o singură încercare. În consecință, SPX se presupune că este mult mai rezistent la atacuri prin ghicire decât alte protocoale.

În pasul 4, LEAF codifică $\{k_C^{-1}\}h_2(pwd)$ cu valoarea N_c primită în pasul 1 şi transmite rezultatul $\{\{k_C^{-1}\}h_2(pwd)\}N_c$ lui C. Acesta poate acum utiliza N_c şi $h_2(pwd')$ pentru a decodifica cheia sa privată pe termen lung k_C^{-1} şi să folosească această cheie pentru a instala credențialele pretinzătorului. Pentru a face acest lucru, C selectează aleator o pereche de chei publice de delegare (k_c, k_c^{-1}) şi generează un bilet de login corespunzător $Ticket_C = \{C, k_c, L\}k_C^{-1}$. Acesta include identificatorul lui C, cheia de delegare de scurtă durată şi durata de viață a biletului. În plus, biletul este semnat digital cu cheia privată de termen lung al lui C. De acum înainte, C nu-şi mai foloseşte cheia privată de termen lung ci cheia de delegare k_c^{-1}.

Ultimul lucru pe care C trebuie să-l facă pentru instalarea credențialelor este să instaleze certificate pentru autoritățile credibile. În pasul 5, C furnizează CDC-ului identificatorul său iar în pasul 6 CDC-ul returnează C << (TA_C) >>, care este o listă de certificate pentru autoritățile credibile ale lui C. Certificatele sunt emise de către C şi semnate digital cheia privată a lui C, k_C-1. După recepționarea C << (TA_C) >>, C are câte o cheie publică pentru fiecare dintre TA-urile sale. C poate utiliza aceste chei pentru a verifica semnăturile digitale de la TA-uri.

5.3.2. Schimbul de autentificare

Schimbul de autentificare SPX este ilustrat în figura 16. Protocolul este formalizat după cum urmează:

1. C → CDC : V
2. CDC → C : $(TA_C, V, L_V, k_V)k_{TAc}^{-1}$
3. C → V : C, $Ticket_C$, $\{K\}k_V$, $\{k_c^{-1}\}K$ | $\{K\}k_c^{-1}$, $Auth_K$

4. V → CDC : C
5. CDC → V : $(TA_V, C, L_C, k_C)k_{TAv}^{-1}$
6. V → C : $Auth_K$'

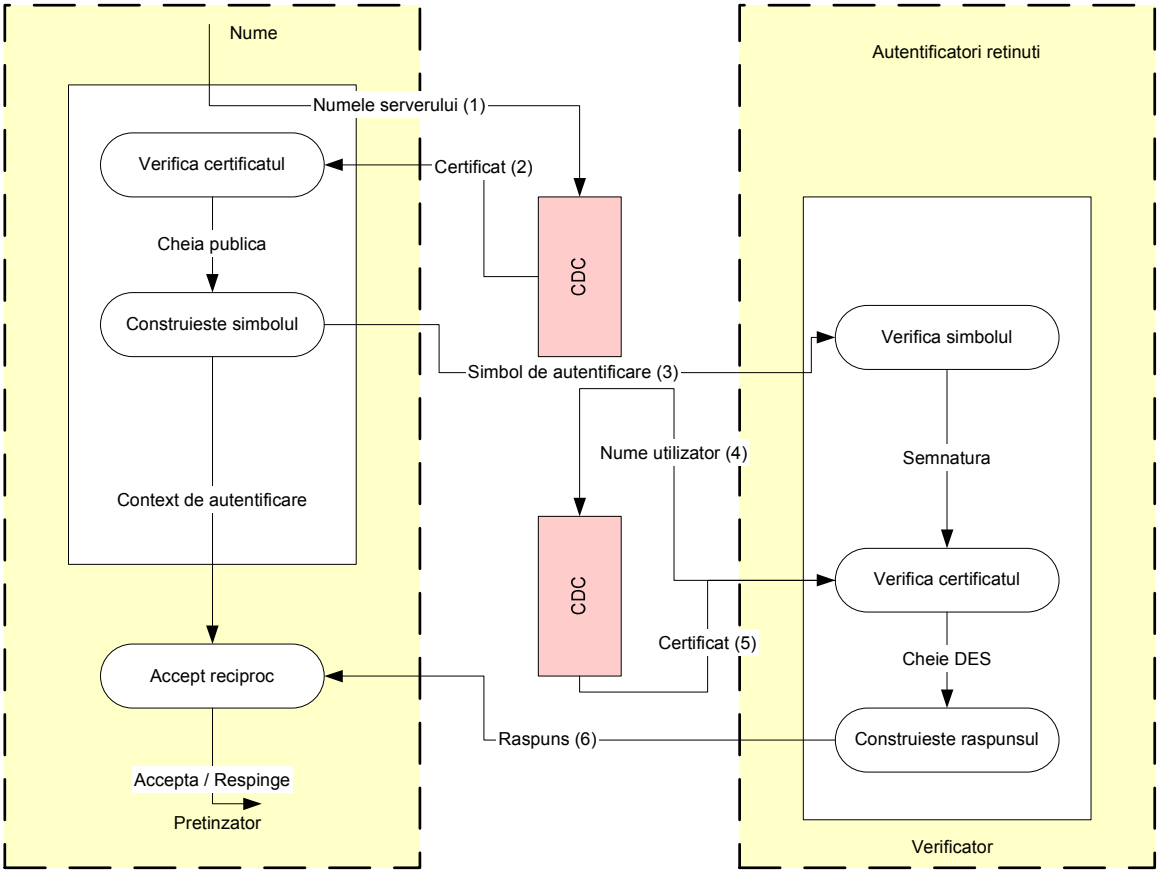

Figura 16. Vedere de ansamblu a schimbului de autentificare SPX

În pasul 1, pretinzătorul C furnizează CDC-ului identificatorul verificatorului V căruia doreşte să se autentifice iar în pasul 2, CDC furnizează lui C un certificat corespunzător $(TA_C, V, L_V, k_V)k_{TAc}^{-1}$. De remarcat că TA_C este o autoritate credibilă pentru C, iar C are cheia publică ce se foloseşte pentru verificarea semnăturii digitale care se adaugă la certificatul anterior. Dacă semnătura digitală este validă, C extrage cheia publică a lui V (k_V) din certificat. În plus față de aceasta, C selectează aleator o cheie DES de sesiune K şi foloseşte această cheie pentru a genera un simbol de autentificare. Acest simbol consta din:

- C
- $Ticket_C = \{C, k_c, L)k_C$-1
- $\{K\}k_V$
- $\{k_c^{-1}\}K$ dacă se solicită delegarea, sau $\{K\}k_c^{-1}$ altfel
- un autentificator $Auth_K$ care constă dintr-o amprentă, indicator de direcție şi legături de canal opționale, cum ar fi adrese de rețea sau informații de context ale aplicației. $Auth_K$ este autentificat K.

În pasul 3, C îi furnizează lui V semnul de autentificare generat prin intermediul protocolului de aplicație, care în general este diferit de protocolul de rețea situat mai jos în

ierarhie. V utilizează credenţialele sale pentru a decodifica simbolul şi pentru a decodifica şi recupera cheia K. În continuare, V foloseşte K şi timpul curent pentru verificarea validităţii autentificatorului. Dacă se solicită delegarea, V decodifică cheia privată de delegare K şi o verifică cu cheia publică din bilet. Altfel, V verifică semnătura cheii codificate folosind cheia publică din bilet.

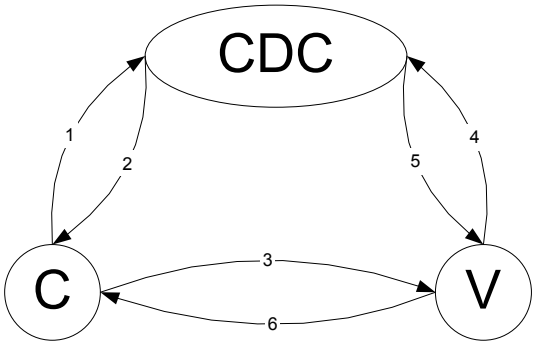

Figura 17. Traseul mesajelor in schimbul de autentificare SPX

În acest moment, V a verificat validitatea simbolului de autentificare dar încă trebuie să îi verifice autenticitatea. În pasul 4, V îi furnizează lui CDC identitatea pretinsă a lui C, iar în pasul 5, CDC-ul returnează $(TA_V, C, L_C, k_C)k_{TAv}^{-1}$. Remarcăm faptul că acest certificat este analog cu certificatul care a fost returnat lui C în pasul 2. În consecinţă, V poate folosi cheia publică a TA_V pentru a verifica semnătura digitală ataşată certificatului şi pentru a extrage k_C. Echipat cu această cheie, V poate verifica semnătura digitală care este asociată cu $Ticket_C$. Dacă semnătura este validă, V presupune că C este autentic. Dacă se solicită delegarea mai există un pas suplimentar pentru instalarea credenţialelor pretinzătorului folosind biletul şi cheia privată de delegare.

Dacă este necesară autentificarea reciprocă, V returnează lui C un alt autentificator $Auth'_K$ in pasul 6. În orice caz, C şi V cunosc împreună cheia K ce poate fi utilizată pentru autentificarea originii datelor, a confidenţialităţii şi servicii de integritate a datelor. SPX detectează atacurile replay în schimbul de autentificare prin utilizarea amprentelor de timp în cadrul autentificatorilor cu rememorarea mesajelor anterioare acceptate până când acestea se învechesc. Unul dintre motivele pentru care s-a ales această abordare a fost compatibilitatea cu Kerberos.

5.4. Concluzii

SPX este un sistem interesant de autentificare şi distribuţie a cheilor în principal pentru că a fost istoric primul sistem care nu a urmat abordarea hibridă ci a folosit noile standarde în dezvoltare ITU-T X.509 şi ISO 9594-8. Dacă s-ar fi folosit criptografia cu chei publice, sistemul SPX ar fi furnizat şi servicii non-negare.

În privinţa răspândirii sistemului SPX se constată că acesta nu a fost utilizat pe scară largă în afara laboratoarelor de cercetare de la DEC şi nu a mai fost dezvoltat de la versiunea 2.4, şi dat fiind faptul că majoritatea dezvoltatorilor SPX au părăsit compania şi că lista de discuţii a încetat să mai existe, este puţin probabil că sistemul SPX va mai prezenta interes în viitor.

Capitolul 6

Protocolul TESS

Sistemul de securitate exponențială este un ansamblu de mecanisme şi funcţii criptografice diferite dar cooperante, bazate pe exponențierea discretă. Sistemul este dezvoltat de către European Institute for System Security (EISS) de la University of Karlsruhe, Germania. În acest capitol ne vom îndrepta atenția asupra utilizării TESS la autentificare şi distribuția cheilor. În secțiunea 6.1 vom aborda dezvoltarea sistemului, în secțiunea 6.2 vom detalia arhitectura sa, iar în 6.3 vom descrie protocoalele criptografice pe care se bazează TESS.

6.1. Prezentare

Dându-se un grup finit G, *exponențierea discretă* în G se referă la compunerea de n ori a unui element $\alpha \in G$ cu el însuşi:

$$N \rightarrow \alpha^n = \alpha \cdot \alpha \cdot \ldots \cdot \alpha$$

Aici folosim înmulțirea pentru a ne referi la compunere, iar α^0 este elementul neutru din G. Inversa exponențierii discrete se numeşte *logaritm discret* şi este în general dificil de calculat. Spre exemplu, dându-se un număr prim mare *p* şi un element primitiv α în G, este uşor a se calcula $y = \alpha^x$ pentru $x \in G$, în schimb este dificil a se calcula x, dându-se y, α şi p.

Pentru toate scopurile practice, exponențierea discretă într-un grup finit G este considerată un candidat bun pentru o funcție cu sens unic în sens criptografic. În plus, este demn de remarcat faptul că exponențierea discretă, spre deosebirea de funcția cu sens unic RSA, nu are o metodă ascunsă de a sparge sistemul.

Istoric, primul grup propus ca bază pentru criptosistemele cu chei publice a fost grupul multiplicativ Z_p de întregi modulo un număr prim p care de fapt este grupul multiplicativ a câmpului finit GF(p). Operația fundamentală în cadrul grupului este înmulțirea întregilor modulo p şi din moment ce aceasta se efectuează eficient, Z_p se consideră o alegere bună pentru implementarea software. Totuşi, există şi alternative cum

ar fi grupul multiplicativ a unui câmp de extensie $GF(p)^n$ care se poate construi din polinoame peste GF(p), grupuri de curbe eliptice şi subgrupuri ale acestora.

O serie de primitive de securitate sunt bazate pe exponențierea discretă. Cele mai populare exemple sunt:

- *Schimbul de chei Diffie-Hellman* (Diffie şi Hellman, 1976)
- *Schema de semnătură ElGamal* (ElGamal, 1984, 1985) şi variațiunile propuse de Agnew, Mullin şi Vanstone [AGNE90]
- *Digital Signature Standard* (DSS) propus de U.S. National Institute of Standards and Technology (NIST) în Federal Information Processing Standard (FIPS) 186 (NIST, 1994)
- *Schema zero-knowledge* propusă de Chaum, Evertse şi Van de Graaf [CHAU88], precum şi variantele propuse de Beth [BETH89] şi Schnorr [SCHN90].

Schimbul de chei Diffie – Hellman foloseşte proprietatea de sens unic a exponențierii discrete pentru a permite două părți A şi B să schimbe o cheie secretă folosind un canal public. Dacă A şi B au convenit asupra unui număr prim p şi un element primitiv $\alpha \in GF(p)$, A (B) poate selecta un număr aleator x_a (x_b) şi publică $y_a = \alpha^{xb} \pmod p$). În concordanță cu terminologia folosită în criptografia cu chei publice, x_a (x_b) se referă la cheile private ale lui A, respectiv B, iar y_a (y_b) se referă la cheile publice ale acestora.

Dacă A şi B îşi comunică reciproc cheia publică corespunzătoare, părțile se pot pune de acord asupra unei chei comune de sesiune $K = K_{ab} = K_{ba}$, după cum urmează:

$$K_{ab} = y_b^{xa} = \alpha^{xbxa} = \alpha^{xaxb} = y_a^{xb} = K_{ba} \pmod p$$

Se cunoaşte faptul că schimbul de chei Diffie – Hellman este vulnerabil la atacurile „omul din mijloc". Dacă un atacator C controlează comunicația între A şi B, acesta poate trimite $y_c = \alpha^{xc} \pmod p$ lui B când recepționează y_a de la A şi trimite y_c către A la recepționarea y_c de la B. Dacă A şi B acceptă mesajele contrafăcute, în continuare C trimite cheia $K' = \alpha^{xaxc} \pmod p$ lui A şi altă cheie $K'' = \alpha^{xbxc} \pmod p$ lui B. Desigur, C poate utiliza aceste chei pentru a decodifica, citi, eventual modifica şi recodifica orice mesaj trimis între A şi B şi invers.

Problema care stă la baza vulnerabilității schimbului de chei se datorează faptului că nu este asigurată autenticitatea cheilor publice y_a şi y_b. De fapt, această problemă a fost cunoscută încă de la publicarea originală de către Diffie şi Hellman, mai multe posibilități de ocolire fiind publicate în [RIVE84].

O posibilitate interesantă a fost descoperită independent de Günther [GUNT90] şi Bauspiess şi Knobloch [BAUS90]. Ideea de bază este folosirea cheilor publice certificate de o autoritate de certificare (CA). După [GIRA91], a cheie publică *auto-certificată* este o cheie ce poate fi calculată din identificatorul deținătorului şi unele informații publice. Noțiunea de cheie publică auto-certificată este similară cu noțiunea lui Shamir de criptosistem bazat pe identitate [SHAM85].

Protocolul Günther – Bauspiess – Knobloch a fost numit KATHY, un acronim de la „*KeY exchange with embedded AuTHentication*". Având definit protocolul KATHY. EISS de la Universitatea din Karlsruhe, Germania a început să dezvolte alte protocoale care utilizează chei publice auto-certificate [HORS91, HORS92]. Efortul de dezvoltare a condus la TESS (The Exponential Security System) care este astăzi un set de mecanisme

criptografice diferite dar cooperante, bazate pe exponențierea discretă [BETH94a, BETH94b, BETH95, DANI95].

6.2. Arhitectură

TESS constă în principal din cinci module implementate în limbajul C [KNOB92]:

- Un *modul de aritmetică cu întregi mari*, care suportă operanzi de lungimi de până la 4096 de biți. Modulul oferă un set de funcții de nivel scăzut pentru adunare, scădere, înmulțire, împărțire, radical de ordinul 2, comparare şi altele. În plus, mai oferă funcții de nivel înalt cum ar fi înmulțire modulară, exponențiere modulară, inversiune modulară şi altele. Unele dintre aceste funcții sunt disponibile şi codificate în limbaj de asamblare pentru a creşte performanța.
- Un *modul de cifrare* care constă din două părți, una care manipulează cifruri stream bazate pe registre de deplasare cu feedback liniar, cealaltă care manipulează cifruri bloc. Cu referire la cea de-a doua parte, cifrurile suportate sunt DES, FEAL şi IDEA, în modul ECB (electronic code book). Totuşi, un modul special poate fi folosit la nivelul superior pentru a adăuga suport în modurile CBC (cipher block chaining), OFB (output feedback) şi CFB (cipher feedback).
- Un *modul de chipcard*, care implementează un protocol în trei straturi pentru comunicarea cu chipcard-uri. Primul strat constă dintr-un modul dependent de sistem care asigură comunicația între un host şi chipcard printr-o legătură serială standard. Al doilea strat se utilizează pentru comanda cititorului de chipcard prin interfața serială iar al treilea strat implementează protocoalele de transport.
- Un *modul de autentificare* care implementează protocoalele KATHY descrise în secțiunea următoare. Modulul cuprinde trei clase de funcții şi anume funcții pentru inițierea autentificării şi terminarea unei sesiuni codificate, funcții de stare care furnizează aplicației informații despre starea parametrilor autentificării şi a partenerului de comunicație autentificat, precum şi funcții filtru pentru procesarea datelor de intrare şi ieşire.
- Un *modul de suport* care conține funcții cu suport pentru o gamă largă de platforme hardware şi sisteme de operare.

Pentru dezvoltarea modulelor s-a ales limbajul de programare C pentru a creşte portabilitatea software-ului. Daca modulele trebuie portate pe o nouă platforma, versiunea în limbajul C se compilează fără sau cu puține modificări. Odată implementată aplicația, funcțiile de bază se pot rescrie în limbaj de asamblare pentru spor de viteză.

Până acum, TESS a fost folosit pentru a scrie două pachete de aplicații, şi anume SELANE (Secure Local Area Network Environment) şi EES (Exponential Electronic Signature).

SELANE

SELANE[1] este o aplicație TESS care reprezintă un sistem de autentificare şi distribuție a cheilor similar cu alte sisteme descrise în această lucrare. De fapt, SELANE este motivul principal pentru care am dedicat acest capitol protocolului TESS.

Aşa cum sugerează numele, SELANE a fost conceput inițial pentru medii de rețele locale, dar vom vedea că acest nume este înşelător deoarece nimic din proiectarea sistemului sau a algoritmilor criptografici nu restricționează utilizarea sistemului în rețeaua locală. De fapt, SELANE poate oferi servicii de autentificare, confidențialitatea datelor şi integritate în orice rețea sau mediu distribuit.

SELANE a fost implementat în medii de lucru comune pe UNIX, VMS, DOS şi Apple Macintosh. EISS foloseşte SELANE pentru a securiza accesul **telnet** din Internet şi rețeaua locală.

EES

EES este o altă aplicație TESS care suportă generarea şi verificarea semnăturilor ElGamal.

Sistemul EES a fost la origine dezvoltat pentru schimb electronic de date (EDI – electronic data interchange) şi aplicații bancare în 1989 când după un studiu aprofundat al sistemelor de semnături, metoda exponențială a părut superioară. Mai recent, după anunțul lui DSS de către NIST, rolul exponențierii discrete a recăpătat recunoaştere, fiind încă o primitivă de securitate pe lângă RSA.

6.3. Protocoale criptografice

Utilizarea TESS necesită un terț de încredere care serveşte ca autoritate de certificare (CA – Certification Authority) sau autoritate de emitere a cheilor sigure (SKIA – Secure Key Issuing Authority). În implementarea curentă a TESS, SKIA este un program de sine stătător care rulează pe un calculator personal fără conectivitate la rețea. Cheia secretă a SKIA este memorată pe disc, fiind codificată cu DES.

Ideea generală a TESS este ca SKIA să genereze un descriptor de identitate m_A pentru participantul A şi să calculeze semnătura ElGamal corespunzătoare (r_A, s_A) pentru m_A. Perechea (m_A, r_A) reprezintă atunci cheia publică auto-certificată a lui A, în timp ce s_A reprezintă cheia privată corespunzătoare. În general, legătura între cele două chei este verificarea ecuației în schema semnăturii ElGamal. În TESS totuşi, în loc de verificarea semnăturii ElGamal (r_A, s_A) ecuația se utilizează pentru a calcula r_A^{sA} din date publice fără a cunoaşte cheia privată s_A.

În secțiunile următoare vom descrie protocoalele utilizate pentru inițializarea SKIA, pentru înregistrarea utilizatorilor şi pentru autentificarea reciprocă a acestora, precum şi generarea şi verificarea semnăturilor digitale.

6.3.1. Inițializarea SKIA

Dacă un număr mare prim p de cel puțin 512 biți în lungime şi o rădăcină primitivă $\alpha \in GF(p)$ care poate servi ca generator pentru GF(p) sunt cunoscute de toate SKIA-urile, un SKIA particular poate fi inițializat prin selectarea aleatoare a unei chei private $X \in GF(p)$ şi calcularea cheii publice corespunzătoare:

[1] Numele SELANE se referă de asemenea şi la Selene, o zeiță antică a lunii. Asemeni Selenei care a avut posibilitatea de a-şi alege iubitul Endymion, utilizatorul sistemului SELANE are posibilitatea de a-şi alege algoritmul de criptare.

$$Y = \alpha^X \pmod p$$

Tripletul (p, α, Y) reprezintă cheia publică a lui SKIA, unde X reprezintă cheia privată corespunzătoare. De remarcat că proprietatea de sens unic a exponențierii discrete sugerează că nu se poate deduce X dacă se cunoaşte Y. Din moment ce acest triplet este necesar la verificarea ecuației pentru semnăturile generate de SKIA, tripletul este şi o parte esențială a tuturor cheilor generate de acest SKIA.

6.3.2. Înregistrarea utilizatorilor

Cu privire la înregistrarea unui utilizator A, SKIA generează un descriptor de identitate m_A care este suficient de precis pentru a identifica în mod unic pe A. O cale pentru a obține acest lucru este a descrie pe A cu nume, adresă, număr de telefon, data naşterii şi codul numeric personal, a converti toate acestea într-o reprezentare orientată pe biți sau octeți şi calcularea dispersiei care este un număr din GF(p).

După pasul inițial, SKIA alege un număr aleator $k_A \in Z_{p-1}^* \setminus \{1\}$, calculează $r_A = \alpha^{kA} \pmod p$ şi rezolvă congruența:

$$X_{rA} + s_A k_A \equiv m_A \pmod{p-1}$$

pentru s_A. Pentru a realiza acest lucru, SKIA trebuie să calculeze

$$s_A = (m_A - X_{rA})\, k_A^{-1} \pmod{p-1}$$

Perechea (r_A, s_A) reprezintă atunci o semnătură ElGamal pentru m_A, cu (m_A, r_A) fiind cheia publică a lui A şi s_A fiind cheia privată a lui A. De remarcat că SKIA poate să nu dorească ca A să-i cunoască cheia privată, altfel utilizatorul ar putea dezvălui această cheie accidental sau intenționat. Pentru aplicații de securitate de nivel înalt nu este acceptabil ca cheile private să fie compromise, deci A trebuie să fie capabil să-şi folosească cheia privată fără să o citească. Această contradicție aparentă se poate rezolva folosind dispozitive hardware protejate fizic, precum chipcard-uri şi simboluri personale.

Valoare k_A trebuie distrusă de SKIA imediat după înregistrarea lui A. Problema se leagă de faptul că A ar putea rezolva congruența pentru cheia privată a SKIA, X dacă A ar cunoaşte s_A şi k_A. În plus trebuie să se asigure că numărul aleator k_A nu s-a utilizat de două ori şi că s_A nu este egal cu 0.

Am menționat anterior că Agnew, Mullin şi Vanstone au propus o variație a schemei de semnătură ElGamal. Această variație este cunoscută uzual ca variația AMV [AGNE90]. Ideea de bază este inversarea rolurilor lui r şi s în semnătura ElGamal, avantajul fiind că pentru a calcula semnătura prin rezolvarea congruenței pentru s, cel care semnează trebuie să calculeze y^{-1} în Z_{p-1}^* numai o dată în loc de fiecare dată pentru fiecare semnătură.

În TESS, folosirea variației AMV a dus la un protocol KATHY puțin modificat, şi anume protocolul r^r –KATHY. În acest protocol, SKIA rezolvă congruența

$$Xs_A + r_A k_A \equiv m_A \pmod{p-1}$$

pentru s_A, şi deci calculează

$$s_A = (m_A - r_A k_A) X^{-1} \pmod{p-1}$$

De remarcat că acest calcul necesită inversul lui X care este acelaşi pentru toţi utilizatorii, în timp ce în cazul anterior era necesar k_A^{-1} care este diferit pentru fiecare utilizator. În consecinţă, utilizarea protocolului r^r –KATHY permite ca X^{-1} să fie calculat o singură data pentru toţi utilizatorii.

Un punct care trebuie avut în vedere este faptul că în timpul procesului de înregistrare al unui client A, SKIA are cunoştinţă de cheia secretă a acestuia, s_A. Drept urmare, SKIA ar putea folosi această cheie fie pentru a recalcula orice cheie de sesiune pe care A o schimbă cu alt utilizator B, fie să falsifice semnături în contul lui A. SKIA ar putea chiar permite altora să facă acest lucru.

Pentru a evita această situaţie delicată, procedura de înregistrare a utilizatorului se modifică puţin. În principiu, A trebuie să-şi semneze digital şirul m_A într-un mod în care nu se poate afla valoarea de fapt a s_A. Conceptul de bază este parametrul ascuns în semnătură, o variantă a schemei de semnătură oarbă introdusă de Chaum [CHAU92]. Într-o schemă cu parametru ascuns ElGamal, SKIA cunoaşte m_A şi r_A, dar nu poate afla nimic despre s_A.

În TESS, un semnătură ElGamal cu parametru ascuns se mai numeşte şi *mărturie*. În principiu, SKIA depune mărturie că s_A aparţine lui A, fără a cunoaşte valoarea de fapt a lui s_A. Mărturia se generează după cum urmează: A selectează un număr aleator $t_A \in Z_{p-1}^*$ şi calculează $\beta = \alpha^{tA} \pmod p$. A trimite această valoare către SKIA, care foloseşte β în loc de α în toţi paşii următori. În particular, SKIA selectează aleator $k_A \in Z_{p-1}^*$ cu $(k_A, p-1) = 1$, calculează $r_A = \beta^{kA} \pmod p$ şi rezolvă congruenţa

$$X_{rA} + b_A k_A \equiv m_A \pmod{p-1}$$

pentru b_A, şi furnizează lui A tripletul (m_A, r_A, b_A). Din nou, perechea (r_A, b_A) reprezintă o semnătură ElGamal pentru m_A. Pentru a extrage s_A din b_A, A trebuie să calculeze $s_A = b_A t_A^{-1} \pmod{p-1}$. De remarcat că fără cunoaşterea valorilor t_A şi respectiv t_A^{-1}, SKIA nu este capabil să deducă valoarea s_A.

6.3.3. Autentificarea

Dacă A şi B sunt înregistraţi în acelaşi domeniu administrativ, ambii cunosc cheia publică Y a SKIA. Mai mult, dacă A a folosit protocolul de bază pentru înregistrare, următorul protocol se poate folosi pentru iniţierea unui schimb de chei cu B, cu autentificare înglobată.

1. A → B : m_A, r_A
2. B → A : $v_a = r_A^{za} \pmod p$

În pasul 1, A transmite lui B parametrii săi publici m_A şi r_A. B selectează aleator $z_A \in Z_{p-1}^*$, calculează $v_a = r_A^{za} \pmod p$ şi îi trimite lui A această valoare în pasul 2. Dacă A calculează $K_a' = v_a^{sA} \pmod p$ iar B calculează $K_a'' = (\alpha^{mA} Y^{-rA})^{za} \pmod p$, de fapt au calculat aceeaşi cheie de sesiune $K_a = K_a' = K_a''$.

Dacă A utilizează protocolul r^r, atunci A şi B trebuie să comunice utilizând protocolul KATHY uşor modificat. Acesta se poate formaliza după cum urmează:

1. A → B : m_A, r_A
2. B → A : $v_a = Y^{za} \pmod p$

În acest caz, B generează $v_a = Y^{za} \pmod p$ în loc de $v_a = r_A^{za} \pmod p$. Restul protocolului rămâne acelaşi, iar A şi B convin asupra aceleiaşi chei de sesiune K_a.

Dacă A şi B doresc să se autentifice reciproc, trebuie ca protocolul să fie aplicat şi în sens invers. În cazul protocolului KATHY, paşii adiționali sunt după cum urmează:

3. B → A : m_B, r_B
4. A → B : $vb = r_B^{zb} \pmod p$

În cazul protocolului r^r –KATHY, paşii adiționali sunt:

3. B → A : m_B, r_B
4. A → B : $v_b = Y^{zb} \pmod p$

În ambele cazuri, A şi B pot conveni asupra altei chei de sesiune $K_b = K_b' = K_b''$. Următorul protocol de tip provocare-răspuns se poate folosi pentru ca A şi B să se asigure că sunt în posesia aceloraşi chei K_a şi K_b:

1. A → B : $\{N_a\} K_a$
2. B → A : $\{N_b\} K_b$
3. A → B : $\{N_b\} K_a$
4. B → A : $\{N_a\} K_b$

În pasul 1, A selectează o valoare N_a aleatoare, o codifică cu cheia K_a şi trimite rezultatul lui B. Similar, B selectează o valoare N_b aleatoare, o codifică cu cheia K_b şi trimite rezultatul lui A în pasul 2. În pasul 3, A decodifică $\{N_b\}K_b$ cu cheia sa K_b, o recodifică cu K_a şi trimite rezultatul lui B. Similar, B decodifică $\{N_a\}K_a$ cu cheia sa K_a, o recodifică cu K_b şi trimite rezultatul lui A în pasul 4. Dacă la sfârşitul protocolului A şi B recepționează aceleaşi valori pe care le-au selectat inițial, înseamnă că sunt în posesia aceloraşi chei de sesiune. Fiecare poate folosi fie K_a fie K_b ca o cheie de sesiune pentru servicii de autentificare a originii, confidențialitatea datelor, integritate sau pot folosi o funcție f pentru dispersarea cheilor într-una singură: $Kab = f(K_a, K_b)$.

Dacă A şi B doresc să se reautentifice după un timp, pot utiliza acelaşi protocol fără a mai fi nevoie să treacă prin paşii 1 – 4.

6.3.4. Semnături digitale

Dacă un utilizator A s-a înregistrat după cum s-a descris anterior, acesta îşi poate folosi cheile publică şi privată pentru a semna documente. Dacă A doreşte să semneze digital un mesaj m, trebuie să aleagă o valoare aleatoare $k \in Z_{p-1}^*$, să calculeze $t = r_A^k$ şi să rezolve congruența $m \equiv s_A t + ku \pmod{p-1}$ pentru u. În acest caz, (m, m_A, r_A, t, u) reprezintă mesajul semnat autentic m. Semnătura se poate verifica calculând

$$r_A^m = (\alpha^{mA} Y^{-rA})^t t^u$$

Pentru variația KATHY care foloseşte schema AMV, t trebuie calculat ca $t = Y^k$ şi congruența de rezolvat este $m \equiv s_A u + kt \pmod{p-1}$. Aceasta duce la ecuația de verificare

$$Y_m = (\alpha^{mA} r_A^{-rA})^u t^t$$

De remarcat că verificatorul unei astfel de semnături trebuie doar să aibă încredere în parametri publici ai SKIA şi nu în parametri fiecărui participant. Ambele scheme de semnătură digitală sunt încorporate în pachetul de aplicații EES.

6.4. Concluzii

În acest capitol ne-am familiarizat cu TESS, un set de mecanisme criptografice diferite dar cooperante. Utilizarea TESS pentru autentificare şi distribuție a cheilor necesită existența unui SKIA off-line care joacă rolul unui terț de încredere.

Similar cu un birou de paşapoarte, este la latitudinea SKIA să înregistreze noi utilizatori şi să le acorde credențialele corespunzătoare. Credențialele unui utilizator constau în principal dintr-o cheie publică auto-certificată şi o cheie privată corespunzătoare. Calea uzuală de a furniza unui utilizator credențiale este oferirea unui dispozitiv hardware asupra căruia nu se poate interveni, cum ar fi un chipcard sau un obiect personal, iar acest dispozitiv să memoreze valorile în numele utilizatorului. În cazul în care costul dispozitivului este prohibitiv sau nu se justifică, se pot utiliza simple fişiere codificate cu chei derivate din parola utilizatorului.

În orice caz, cheile publice auto-certificate a doi utilizatori oarecare se pot utiliza la efectuarea unui schimb de chei cu autentificare înglobată. Scopul acestui schimb este obținerea unei chei secrete care se poate folosi pentru protejarea pe mai departe a autenticității, confidențialității şi integrității mesajelor schimbate.

Dezvoltarea EISS s-a concentrat pe posibilitatea de a folosi exponențierea discretă pentru serviciul de însoțire a cheilor [BETH94] şi modificarea protocolului KATHY pentru a adăuga suport la DSS. În plus, la TESS s-a adăugat o schemă verificabilă de comunicare a secretului [BETH93]. Folosind această schemă a devenit fezabilă distribuția securizată a cheilor.

În legătură cu managementul securității, cercetarea s-a îndreptat către investigarea relațiilor de încredere în certificarea arbitrară şi ierarhiile SKIA [KLEI93]. În general, este posibil să se grupeze SKIA-urile ierarhic în aşa fel încât SKIA de pe nivelurile superioare să le autorizeze pe cele de pe nivelurile inferioare.

Deoarece TESS este simplu şi eficient, el reprezintă o alternativă atractivă în cadrul rețelelor şi al sistemelor distribuite. Totuşi, trebuie avute în vedere următoarele la evaluarea utilizării TESS:

1. Sistemul nu este răspândit pe larg în aplicațiile de rețea.
2. Sistemul este dezvoltat la o universitate. Deşi acest lucru nu poate fi considerat un dezavantaj, trebuie avută în vedere lipsa de suport tehnic profesional pe diferite platforme.
3. Faptul că SKIA este off-line evită o gâtuire evidentă şi separă disponibilitatea serviciului de autentificare de disponibilitatea serverelor corespunzătoare. Totuşi, revocarea credențialelor unui utilizator se face greu, deoarece SKIA nu este implicat direct în procesul de autentificare. În momentul de față problema revocării este rezolvată parțial printr-o dată de expirare. [OPPL96]

Capitolul 7

Protocolul SSL

Conform definiției din documentul oficial, protocolul SSL (v 3.0) este un protocol de securitate care oferă comunicare secretă prin Internet. Protocolul permite aplicațiilor client / server să comunice într-un fel în care se împiedică capturarea, modificarea sau falsificarea mesajelor.

7.1. Prezentare

Scopul principal al protocolului SSL este de a oferi confidențialitate şi încredere între două aplicații care comunică. Protocolul constă din două straturi: La cel mai de jos nivel, plasat deasupra unui protocol sigur de comunicație (spre exemplu TCP) se află protocolul *SSL Record*. Acesta este folosit pentru încapsularea protocoalelor de nivel mai înalt, cum ar fi protocolul *SSL Handshake* care permite serverului şi clientului să negocieze un algoritm de criptare şi cheile criptografice corespunzătoare înainte ca aplicațiile să schimbe vreun mesaj. Un nivel superior al protocolului ar putea sta deasupra acestuia în mod transparent.

Protocolul SLL oferă securitatea conexiunii, având trei proprietăți:

- Conexiunea este privată. Criptarea se utilizează după un înțelegerea inițială pentru definirea cheii secrete. Criptografia simetrică se utilizează pentru codificarea datelor (DES, RC4, etc.)
- Identitatea părților se autentifică utilizând criptografie asimetrică (RSA, DSS, etc.)
- Conexiunea este de încredere. Transportul mesajului include verificarea acestuia cu un MAC parametrizat. Pentru calcularea MAC-ului se folosesc funcțiile de dispersie SHA, MD5, etc.

Scopurile protocolului SSL, în ordinea priorităților sunt:

1. **Securitatea criptografică**: SSL ar trebui să se utilizeze pentru conexiune sigură între două părți.

2. **Inter-operabilitate**: Programatori independenți ar trebui să fie capabili de a dezvolta aplicații SLL care să funcționeze cu succes fără ca aceştia să aibă cunoştință de codul scris de altcineva.
3. **Extensibilitatea**: SSL încearcă să furnizeze un cadru în care să se integreze metode noi de criptare (simetrică sau asimetrică), acest lucru contribuind la evitarea creării unui protocol nou (riscând implicit să se introducă noi slăbiciuni) şi evitarea scrierii unei biblioteci de securitate noi.
4. **Eficiența relativă**: Operațiile criptografice tind să fie puternic procesor intensive, în particular criptografia cu chei publice. Din acest motiv, SSL a înglobat un mecanism de caching al sesiunii pentru a reduce numărul de conexiuni ce trebuie stabilite în totalitate.

SSL este un protocol stratificat. Pe fiecare strat, mesajele pot conține câmpuri pentru lungime, descriere şi conținut. SSL primeşte mesajele de transmis, fragmentează informația în blocuri prelucrabile, opțional comprimă datele, aplică un MAC, codifică şi în final transmite rezultatul. Datele recepționate sunt decriptate, verificate, decomprimate şi reasamblate, apoi oferite protocolului superior ierarhic.

O sesiune SSL are stare. Este responsabilitatea protocolului SSL Handshake să coordoneze stările clientului şi ale serverului, permițând ca sistemele să funcționeze în mod consistent, în ciuda faptului că stările nu sunt exact paralele. În mod logic, starea este reprezentată de două ori, o dată ca starea curentă în operare şi (în timpul protocolului de handshake) ca starea în aşteptare. În plus, se mențin stări separate ale scrierii şi citirii. Când clientul sau serverul recepționează un mesaj de schimb a specificației cifrului, se copiază starea în aşteptare în starea curentă de citire. Când negocierea este completă, serverul şi clientul interschimbă mesajele de specificare a cifrului şi comunică prin noul cifru asupra căruia s-a convenit.

O sesiune SSL poate include mai multe conexiuni sigure, în plus participanții pot avea mai multe sesiuni simultane. Starea sesiunii poate cuprinde următoarele elemente:

- **Identificator de sesiune**: O secvență arbitrară de octeți aleasă de server pentru a identifica o sesiune activă sau reluabilă.
- **Certificat al egalului:** Certificat X.509 al părții cu care se comunică. Acest element poate să fie nul.
- **Metoda de compresie:** Algoritmul utilizat pentru a comprima datele înainte de codificare.
- **Specificația cifrului:** Specifică algoritmul de codificare a datelor (cum ar fi nimic, DES, AES, etc.) şi un algoritm MAC (cum ar fi MD5 sau SHA). De asemenea defineşte atributele criptografice cum ar fi dimensiunea tabelei de dispersie.
- **Secretul master:** O valoare secretă de 48 de octeți cunoscută de client şi de server.
- **Capacitatea de reluare:** Un indicator care arată dacă o sesiune se poate utiliza pentru a inița noi conexiuni.

Starea conexiunii poate include următoarele elemente:

- **Valori aleatoare pentru client şi server**: Secvențe de octeți alese de client şi de server pentru fiecare conexiune.

- **Secret MAC pentru scrierea pe server:** Secretul utilizat în operații MAC pe datele scrise de server.
- **Secret MAC pentru scrierea pe client:** Secretul utilizat în operații MAC pe datele scrise de client.
- **Cheia de scriere a clientului:** Cheia pentru datele codificate de server şi decodificate de client.
- **Cheia de scriere a serverului:** Cheia pentru datele codificate de client şi decodificate de server.
- **Vectori de inițializare:** Când se utilizează un cifru bloc în modul CBC, se păstrează un vector de inițializare pentru fiecare cheie. Acest câmp este inițializat de protocolul SSL Handshake.
- **Numere de secvență:** Fiecare participant menține numere de secvență pentru mesajele transmise şi recepționate pentru fiecare conexiune. Când un participant trimite sau recepționează un mesaj de modificare a specificațiilor cifrului, acest număr se pune pe zero. Numere de secvență sunt de tipul *uint64* şi nu trebuie să depăşească valoarea $2^{64} – 1$. [FREI96]

7.2. Arhitectură

Protocolul SSL foloseşte o combinație de criptări simetrice şi cu chei publice. Criptarea cu chei simetrice este mult mai rapidă decât cea cu chei publice, dar aceasta din urmă oferă metode de autentificare mai bune. O sesiune SSL începe întotdeauna cu un schimb de mesaje numit SSL Handshake. Acesta permite serverului să se autentifice clientului folosind criptografia cu chei publice, apoi permite clientului şi serverului să coopereze în crearea cheilor simetrice utilizate pentru codificarea şi decodificarea rapidă precum şi detecția modificării mesajelor. Opțional, protocolul permite şi clientului să se autentifice serverului.

7.2.1. SSL Handshake

În cele ce urmează vom descrie paşii urmați de un sistem în timpul protocolului SSL Handshake, fără a intra în detalii de programare:

1. Clientul trimite serverului versiunea sa SSL, setările cifrurilor, date generate aleator şi alte informații necesare în cadrul viitoarei comunicații securizate SSL.
2. Serverul trimite clientului versiunea sa SSL, setările cifrurilor, date generate aleator şi alte informații necesare în cadrul viitoarei comunicații securizate SSL. De asemenea, serverul trimite propriul certificat şi dacă clientul cere o resursă care necesită autentificarea clientului, cere certificatul acestuia.
3. Clientul utilizează unele dintre informațiile primite pentru autentificarea cu serverul. Dacă serverul nu poate fi autentificat, utilizatorul este atenționat asupra faptului că nu se poate stabili o comunicație autentificată şi securizată.
4. Utilizând toate informațiile deținute până acum, clientul (în cooperare cu serverul, în funcție de cifrul utilizat) creează un **secret premaster** pentru sesiune, îl criptează cu cheia publică a serverului (obținut din certificatul serverului în pasul 2) şi trimite acest secret serverului.
5. Dacă serverul a cerut autentificarea clientului (un pas opțional în cadrul protocolului), clientul mai semnează o informație unică cunoscută de cele

două părți. În acest caz, clientul trimite atât informația semnată cât şi certificatul său serverului, precum şi secretul premaster codificat.

6. Dacă serverul a cerut autentificarea clientului, acesta încearcă să autentifice clientul. Dacă clientul nu poate fi autentificat, sesiunea se încheie. Dacă clientul se autentifică cu succes, serverul decriptează secretul premaster folosind cheia sa privată apoi efectuează o serie de paşi (ca de altfel şi clientul) pentru a genera **secretul master.**
7. Atât clientul cât şi serverul folosesc secretul master pentru a genera chei de sesiune, care sunt chei simetrice folosite la codificare şi decodificarea informației precum şi la verificarea integrității datelor în timpul sesiunii SSL.
8. Clientul trimite un mesaj către server informându-l că mesajele viitoare de la acesta vor fi codificate cu cheia de sesiune. Apoi trimite un mesaj codificat prin care se specifică faptul că partea de client a protocolului de handshake s-a încheiat.
9. Serverul trimite un mesaj către client informându-l că mesajele viitoare de la acesta vor fi codificate cu cheia de sesiune. Apoi trimite un mesaj codificat prin care se specifică faptul că partea de server a protocolului de handshake s-a încheiat.
10. Protocolul SSL Handshake este acum încheiat, iar sesiunea SSL a început. Clientul şi serverul folosesc cheile de sesiune pentru a codifica şi decodifica datele care se schimbă între cele două părți.

Autentificarea serverului

Software-ul client cu capabilități SSL necesită întotdeauna autentificarea serverului. Aşa cum s-a arătat în pasul 2, serverul trimite clientului certificatul pentru autentificarea sa. Clientul foloseşte acest certificat în pasul 3.

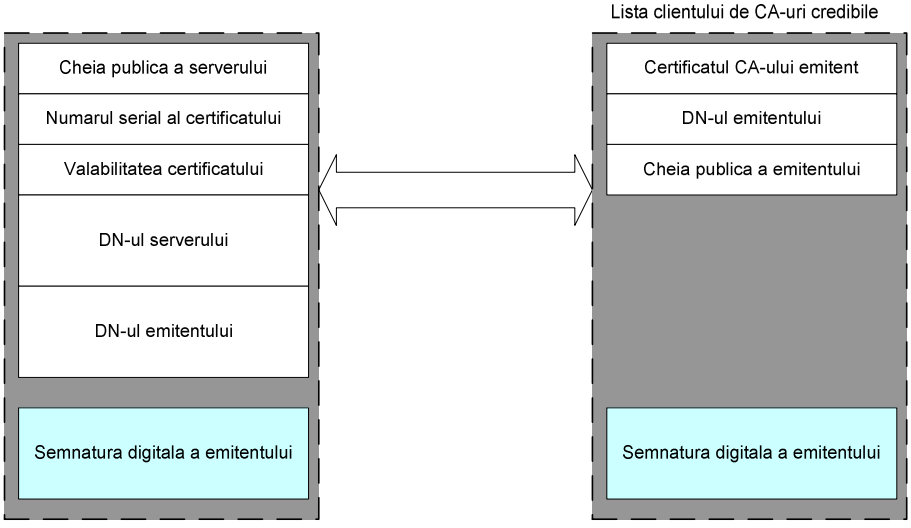

Figura 18. Modalitatea de autentificare a certificatului unui client

Pentru a autentifica legătura dintre o cheie publică şi un server identificat de un certificat care conține o cheie publică, clientul SSL trebuie să recepționeze un răspuns „da" la toate cele patru întrebări ilustrate mai jos. Deşi a patra întrebare nu este în mod

tehnic parte din protocolul SSL, cade în responsabilitatea clientului să suporte această cerință care oferă asigurarea identității serverului şi protejează împotriva unei forme de atac numită „omul din mijloc”.

Un client SSL urmează paşii descrişi în continuare pentru a autentifica identitatea serverului:

1. **Data de astăzi este în perioada de valabilitate?** Clientul verifică perioada de validitate a certificatului serverului. Dacă data şi timpul curent sunt în afara perioadei de valabilitate, procesul de autentificare nu mai continuă, altfel se trece la pasul 2.
2. **CA-ul emitent este credibil?** Fiecare client menține o listă de CA-uri credibile, reprezentată în partea dreaptă a figurii 18. Această listă determină care dintre certificate vor fi acceptate de client. Dacă numele (DN – distinguished name) al CA-ului emitent se potrivește cu numele unui CA de pe listă, răspunsul la întrebare este „da” şi se trece la pasul 3. Altfel, serverul nu este autentificat până când clientul nu verifică faptul că CA-ul se află într-un lanț credibil, aflat pe listă.
3. **Cheia publică a CA-ului emitent verifică semnătura digitală a emitentului?** Clientul utilizează cheia publică din certificatul CA-ului pentru a valida semnătura digitală a acestuia pe serverul în discuție. Dacă informația în certificatul serverului s-a modificat de la momentul în care a fost semnat de CA sau dacă cheia publică din certificatul CA-ului nu corespunde cu cheia publică folosită pentru semnarea certificatului serverului, clientul nu va autentifica identitatea serverului. Dacă semnătura digitală a CA-ului se poate verifica, serverul consideră certificatul utilizatorului ca o „scrisoare de introducere” validă de la CA şi continuă. În acest punct clientul a determinat că certificatul serverului este valid.
4. **Numele de domeniu din certificatul serverului se potriveşte cu numele serverului în sine?** Acest pas confirmă că serverul este localizat la adresa de rețea specificată de numele de domeniu în certificatul serverului. Deşi cest pas nu face parte – în mod tehnic – din protocolul SSL, acesta oferă singura protecție împotriva atacului „omul din mijloc”. Clienții trebuie să efectueze acest pas şi să refuze să autentifice un server dacă numele de domeniu nu se potrivesc.
5. **Serverul este autentificat.** Clientul continuă cu protocolul SSL handshake. Dacă pentru orice motiv nu se ajunge până în acest punct, utilizatorul este informat că nu se poate crea o conexiune autentificată şi codificată. Dacă serverul cere autentificarea clientului, acesta va efectua paşii specificați în paragraful următor.

Autentificarea clientului

Serverele cu capabilități SSL pot fi configurate să ceară la rândul lor autentificarea clientului sau validarea criptografică a identității acestuia. Când serverul astfel configurat cere autentificarea clientului (pasul 6), clientul trimite serverului atât certificatul cât şi o informație separată semnată digital pentru identificarea sa. Serverul utilizează datele semnate pentru validarea cheii publice din certificat şi pentru autentificarea identității pretinse de certificat.

Protocolul SSL cere clientului să creeze o semnătură digitală dintr-o dispersie cu sens unic aplicată unor date aleatoare generate în timpul handshake-ului şi cunoscute doar de client şi de server. Dispersia datelor este apoi codificată cu cheia privată care corespunde cheii publice din certificatul prezentat serverului.

Pentru a autentifica legătura dintre cheia publică şi persoana sau altă entitate identificată de certificatul conține cheia, serverul trebuie să primească răspunsul „da” la primele patru întrebări de mai jos.

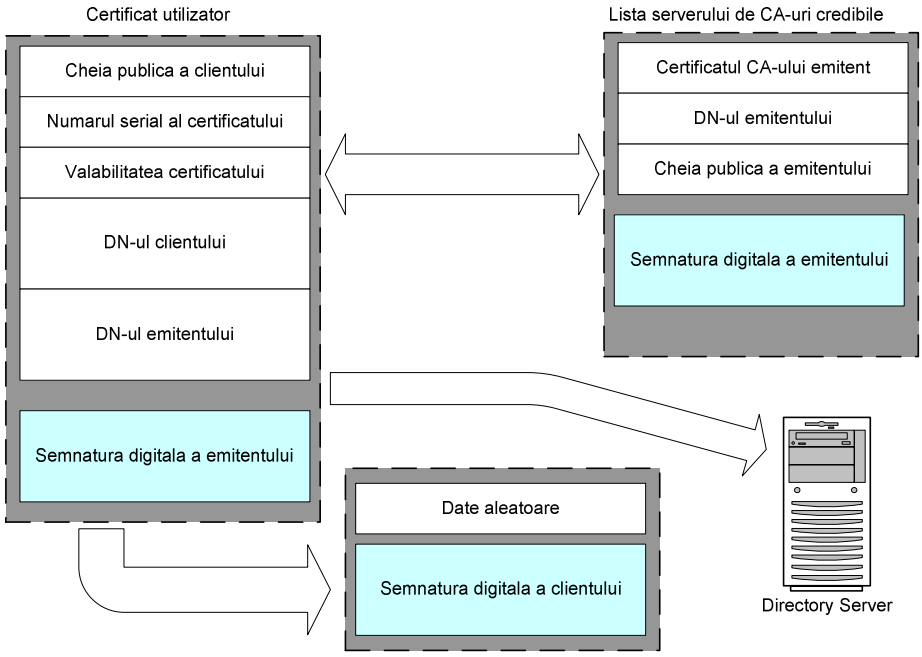

Figura 19. Modalitatea de autentificare a certificatului unui client

Serverul parcurge următorii paşi pentru autentificarea identității clientului:

1. **Cheia publică a utilizatorului validează semnătura sa digitală?** Serverul verifică dacă semnătura digitală a utilizatorului poate fi validată cu cheia publică a certificatului. Dacă este aşa, serverul a stabilit că cheia publică a utilizatorului în cauză se potriveşte cu cheia privată folosită la crearea semnăturii şi datele nu au fost modificate de la momentul semnăturii.
2. **Data de astăzi este în perioada de valabilitate?** Serverul verifică perioada de validitate a certificatului clientului. Dacă data şi timpul curent sunt în afara perioadei de valabilitate, procesul de autentificare nu mai continuă, altfel se trece la pasul 3.
3. **CA-ul emitent este credibil?** Fiecare sever menține o listă de CA-uri credibile, reprezentată în partea dreaptă a figurii 19. Această listă determină care dintre certificate vor fi acceptate de server. Dacă numele (DN – distinguished name) al CA-ului emitent se potriveşte cu numele unui CA de pe listă, răspunsul la întrebare este „da” şi se trece la pasul 4. Altfel, clientul nu este autentificat până când clientul nu verifică faptul că CA-ul se află într-un lanț credibil, aflat pe listă. Administratorii pot controla care CA-uri sunt incluse pe listă.
4. **Cheia publică a CA-ului emitent verifică semnătura digitală a emitentului?** Serverul utilizează cheia publică din certificatul CA-ului pentru a valida semnătura digitală a acestuia. Dacă informația din certificat s-a modificat de la momentul în care a fost semnat de CA sau dacă cheia publică din certificatul CA-ului nu corespunde cu cheia publică folosită

pentru semnarea certificatului, serverul nu va autentifica identitatea clientului. Dacă semnătura digitală a CA-ului se poate verifica, serverul consideră certificatul utilizatorului ca o „scrisoare de introducere" validă de la CA şi continuă. În acest punct serverul a determinat că certificatul clientului este valid.

5. **Clientul autentificat are acces la resursele solicitate?** Serverul verifică ce resurse poate accesa clientul în conformitate cu listele de control al accesului (ACL) şi stabileşte o conexiune cu drepturile potrivite. Dacă serverul nu ajunge la pasul 5 pentru orice motiv, utilizatorul identificat de certificat nu poate fi autentificat şi utilizatorului nu i se permite acces la resurse.

7.3. Protocoale criptografice

Schimbul de chei, autentificarea, criptarea şi algoritmii MAC sunt determinate de suita de cifruri selectate în mesajul *hello* de la server. În cadrul SSL se folosesc două tipuri de sisteme de criptografie şi anume simetrică şi asimetrică. Înainte însă de a le prezenta, ne vom referi mai întâi la protocolul Handshake, care se află la baza SSL.

7.3.1. Protocolul Handshake

Parametrii criptografici ai sesiunii sunt produşi de protocolul SSL Handshake, care operează deasupra stratului SSL Record. Când un client şi un server SSL comunică pentru prima dată, aceştia convin asupra unei versiuni a protocolului, selectează algoritmii criptografici, opțional se autentifică reciproc şi utilizează tehnici de criptare cu chei publice pentru a genera secrete. Aceste procese se realizează în cadrul protocolului Handshake, care se rezumă după cum urmează:

Clientul trimite un mesaj **client hello** la care serverul trebuie să răspundă cu un mesaj **server hello**, altfel survine o eroare fatală şi conexiunea eşuează. Valorile din aceste două mesaje sunt folosite pentru stabilirea caracteristicilor de securitate între client şi server, şi anume atributele: versiunea de protocol, identificatorul de sesiune, suita de cifruri şi metoda de compresie. În plus se generează şi se interschimbă două valori**: ClientHello.random** şi **ServerHello.random**. Ca răspuns la mesajele hello, serverul va trimite certificatul său dacă urmează a fi autentificat. În plus, se poate trimite un mesaj de schimb ale cheilor dacă este necesar (spre exemplu dacă serverul nu are certificat sau certificatul este numai pentru semnătură). Dacă serverul este autentificat, acesta poate cere un certificat de la client, dacă suita de cifruri aleasă cere acest lucru. Acum serverul va trimite un mesaj **server hello done** care indică faptul că această fază a protocolului s-a încheiat. Serverul aşteaptă acum un răspuns de la client. Dacă serverul a emis un mesaj de cerere a certificatului, clientul trebuie să trimită fie un mesaj cu certificatul său fie o alertă de lipsă a acestuia. Acum se trimite mesajul de schimb al cheilor de către client, conținutul acestuia depinzând de algoritmul cu chei publice selectat. Dacă clientul a trimis un certificat cu capacități de semnare, se trimite un mesaj de verificare semnat digital.

În acest moment clientul trimite un mesaj pentru schimbarea specificațiilor criptografice, clientul copiind specificațiile în aşteptare peste cele curente. Imediat după aceea, clientul trimite mesajul de terminare codificat cu algoritmul selectat. Ca răspuns, serverul va trimite propriul mesaj de schimbare a specificațiilor criptografice şi va trimite mesajul de terminare codificat cu algoritmul selectat de specificații. În acest moment, strângerea de mână este completă, iar clientul şi serverul pot începe schimbul securizat de date la nivel de aplicație. Întregul proces este descris schematic în figura 20:

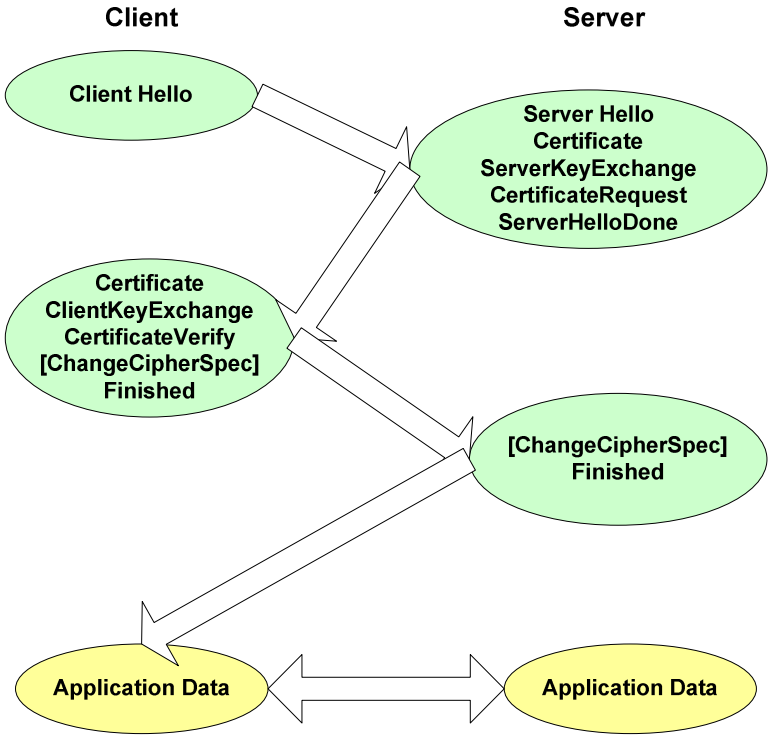

Figura 20. Schimbul de mesaje in protocolul SSL Handshake

Când clientul şi serverul decid să reia o sesiune anterioară sau doresc să duplice o sesiune existentă (în loc să negocieze noi parametri de securitate), mesajele schimbate sunt următoarele:

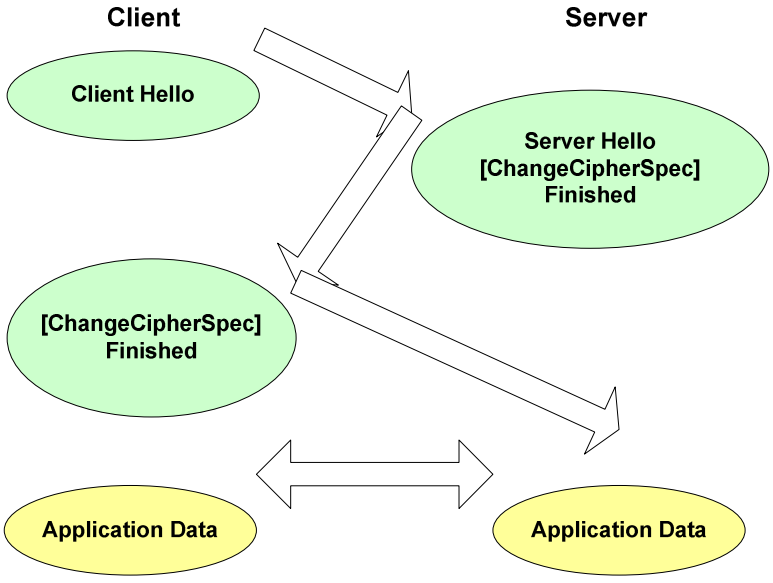

Figura 21. Reluarea unei sesiuni SSL

Clientul trimite un mesaj ClientHello cu identificatorul sesiunii de reluat. Serverul verifică cache-ul iar dacă identificatorul este găsit şi se doreşte reluarea sesiunii, se trimite înapoi mesajul ServerHello cu acelaşi identificator de sesiune. În acest punct, atât clientul cât şi serverul trebuie să-şi trimită mesaje de specificare a cifrurilor şi continuă direct cu mesajele de terminare. Odată restabilirea încheiată, clientul şi serverul pot schimba date la nivelul de aplicații. Dacă identificatorul de sesiune nu este găsit în cache, serverul

generează un nou identificator de sesiune, iar clientul şi serverul efectuează un handshake complet. Conținutul şi semnificația fiecărui mesaj se găseşte în [FREI96].

7.3.2. Calcule criptografice asimetrice

Algoritmii asimetrici sunt utilizați în cadrul protocolului Handshake pentru autentificarea părților şi pentru generarea cheilor şi secretelor.

Pentru Diffie-Hellman, RSA şi FORTEZZA se foloseşte acelaşi algoritm pentru a converti secretul premaster în secretul master. Primul ar trebui eliminat din memorie odată ce secretul master a fost calculat.

```
      master_secret =
      MD5(pre_master_secret + SHA('A' + pre_master_secret +
ClientHello.random + ServerHello.random)) +
      MD5(pre_master_secret + SHA('BB' + pre_master_secret +
ClientHello.random + ServerHello.random)) +
      MD5(pre_master_secret + SHA('CCC' + pre_master_secret +
ClientHello.random + ServerHello.random));
```

RSA

Când se foloseşte RSA pentru autentificarea serverului şi schimbul de chei, clientul generează un secret premaster de 48 de octeți, îl codifică cu cheia publică a serverului şi îl trimite acestuia. La recepție, serverul face uz de cheia sa privată şi decodifică secretul, apoi ambele părți calculează secretul master, aşa cum s-a arătat mai sus.

Diffie-Hellman

Se efectuează un calcul Diffie-Hellman convențional. Cheia negociată Z se foloseşte ca secret premaster şi se calculează secretul master ca mai sus. De remarcat că parametrii Diffie-Hellman sunt fie conținuți în certificatul serverului fie generați pe loc.

FORTEZZA

Se generează un secret de 48 de octeți care se trimite codificat cu TEK serverului. Acesta decodifică secretul şi îl converteşte în secret master, ca mai sus. Cheile de codificare se generează de către client şi schimbate în cadrul mesajelor de schimb; secretul master se foloseşte numai pentru calcule MAC.

7.3.3. Calcule criptografice simetrice

Tehnica utilizată pentru criptarea şi verificarea integrității înregistrărilor SSL este stabilită de specificația criptografică curentă (CipherSpec). Un exemplu tipic ar fi criptarea datelor cu DES şi generarea codurilor de autentificare cu MD5. Algoritmii de codificare şi algoritmii MAC sunt poziționați pe SSL_NULL_WITH_NULL_NULL la începutul protocolului SSL Handshake, indicând că nu se efectuează criptare sau verificare de date. Protocolul handshake se foloseşte pentru a negocia un set de protocoale mai sigure şi pentru a genera chei criptografice.

Secretul master

Înainte ca verificarea şi criptarea să se poată efectua asupra înregistrărilor, clientul şi serverul trebuie să genereze o informație secretă cunoscută numai de aceştia. Această valoare este o cantitate de 48 de octeți numită secretul master. Acesta este utilizat pentru

generarea cheilor şi secretelor pentru criptare şi calcule MAC. Unii algoritmi, cum ar fi FORTEZZA, pot avea propriile proceduri de generare a cheilor, în acest caz secretul master folosindu-se doar pentru calcule MAC.

Convertirea secretului master în chei şi secrete MAC

Secretul master este dispersat într-o secvență de octeți securizați atribuiți secretelor şi cheilor MAC, în conformitate cu specificațiile în vigoare (CipherSpec). Specificația necesită:

- Secret MAC de scriere al clientului
- Secret MAC de scriere al serverului
- Cheie de scriere a clientului
- Cheie de scriere a serverului
- Vector de inițializare (IV[1]) al clientului
- Vector de inițializare (IV) al serverului

Toate acestea sunt generate din secretul master, în această ordine. Cheile nefolosite sunt goale, cum este cazul cheilor FORTEZZA comunicate în mesajul KeyExchange.

Următoarele intrări sunt disponibile procesului de definire a cheilor:

- opaque MasterSecret[48]
- ClientHello.random
- ServerHello.random

Când se generează chei şi secrete MAC, secretul master este folosit ca sursă de entropie, iar valorile aleatoare furnizează material necodificat numit şi „sare”.

Pentru a genera material pentru chei, se calculează entitatea:

```
key_block =
      MD5(master_secret + SHA('A' + master_secret + ServerHello.random +
ClientHello.random)) +
      MD5(master_secret + SHA('BB' + master_secret + ServerHello.random
+ ClientHello.random)) +
      MD5(master_secret + SHA('CCC' + master_secret + ServerHello.random
+ ClientHello.random)) + [...];
```

până când datele calculate sunt suficiente. Entitatea key_block se partiționează după cum urmează:

```
client_write_MAC_secret[CipherSpec.hash_size]
server_write_MAC_secret[CipherSpec.hash_size]
client_write_key[CipherSpec.key_material]
server_write_key[CipherSpec.key_material]
client_write_IV[CipherSpec.IV_size] /* non-export ciphers */
server_write_IV[CipherSpec.IV_size] /* non-export ciphers */
```

Orice valori generate în plus se ignoră.

Algoritmii exportabili (pentru care *CipherSpec.is_exportable* este adevărată) necesită procesare suplimentară pentru derivarea cheilor finale:

```
final_client_write_key =
      MD5(client_write_key + ClientHello.random + ServerHello.random);
final_server_write_key =
```

[1] IV – Initialization Vector

```
MD5(server_write_key + ServerHello.random + ClientHello.random);
```

Algoritmii de criptare exportabili îşi derivă vectorii de iniţializare din mesajele aleatoare:

```
client_write_IV = MD5(ClientHello.random + ServerHello.random);
server_write_IV = MD5(ServerHello.random + ClientHello.random);
```

Ieşirile MD5 sunt ajustate la dimensiunile corespunzătoare prin eliminarea celor mai nesemnificativi octeţi.

7.4. Concluzii

Pentru ca SSL să fie capabil să ofere o conexiune securizată, atât clienţii cât şi serverele, precum şi aplicaţiile şi cheile trebuie să fie securizate. În plus, trebuie ca implementarea să fie ferită de erori de securitate.

Sistemul este la fel de puternic ca cel mai slab algoritm de autentificare şi de distribuţie a cheilor. Astfel trebuie să se utilizeze doar funcţii criptografice sigure, verificate. Chei publice scurte, chei simetrice de 40 de biţi sau mai puţin şi servere anonime trebuie utilizate cu precauţie. Implementările şi utilizatorii trebuie să aleagă cu grijă autorităţile de certificare acceptabile; o autoritate mincinoasă poate provoca pagube imense.

Capitolul 8

Protocolul CHAP

Challenge Handshake Authentication Protocol (CHAP) este folosit la autentificarea ambelor capete ale unei linii de comunicație. Acesta se utilizează în special de către host-uri şi routere care se conectează la un server de rețea PPP (Point-to-Point Protocol) prin intermediul circuitelor comutate sau liniilor dial-up, putându-se folosi chiar şi în cazul liniilor închiriate. CHAP foloseşte un mecanism de răspuns la provocare bazat pe o valoare unică aleatoare şi un secret cunoscut doar de părțile implicate în autentificare.

8.1. Prezentare

Pentru a stabili o comunicație printr-un canal punct la punct, fiecare terminal al legăturii PPP trebuie să trimită pachete LCP pentru configurarea legăturii de date în timpul fazei de stabilire a conexiunii. După stabilirea acesteia, se trece – opțional – la faza de autentificare pentru ca mai apoi să se treacă la faza protocolului la nivel de rețea.

În mod implicit, autentificarea nu este obligatorie. Dacă se doreşte autentificare, implementarea trebuie să specifice opțiunea de configurare Authentication-Protocol în timpul fazei de stabilire a conexiunii.

Protocolul CHAP se utilizează pentru a verifica periodic identitatea părților prin folosirea unui dialog, ca mai jos. De remarcat că acest proces are loc inițial la stabilirea parametrilor legăturii de date şi se poate repeta ori de câte ori se consideră necesar.

1. La terminarea fazei de stabilire a conexiunii, autentificatorul trimite omologului său un mesaj de „provocare”.
2. Omologul răspunde cu o valoare calculată utilizând o funcție de dispersie cu sens unic.
3. Autentificatorul compară răspunsul cu ceea ce a calculat el însuşi; dacă valorile coincid, autentificarea s-a realizat cu succes, altfel conexiunea trebuie încheiată.
4. La intervale aleatoare de timp, autentificatorul trimite o nouă provocare repetând paşii 1 – 3.

Avantaje

CHAP oferă protecție împotriva atacurilor cu redare de către omolog prin utilizarea unui identificator care se incrementează şi o valoare de „provocare" aleatoare. Provocările multiple sunt destinate limitării expunerii în cazul unui atac. Autentificatorul decide frecvența şi momentul provocărilor.

Acest tip de autentificare depinde de existența unui secret cunoscut doar de autentificator şi de omologul său. Secretul în cauză nu se trimite pe linia de comunicație. Deşi autentificarea este cu sens unic, prin negocierea CHAP în ambele direcții se poate utiliza acelaşi secret pentru autentificarea reciprocă.

Din moment ce CHAP se poate utiliza pentru a autentifica mai multe sisteme diferite, câmpurile de nume se pot folosi ca index pentru a localiza un secret anume într-o tabelă mare de secrete. Acest lucru permite de asemenea existența mai multor perechi nume / secret pentru un sistem anume precum şi posibilitatea de a schimba cheia oricând în timpul unei sesiuni.

Dezavantaje

CHAP necesita ca secretul să fie disponibil în formă clară. Bazele de date cu parole codificare ireversibil care se găsesc în mod curent nu se pot folosi.

Acest protocol nu se pretează la instalări de mari dimensiuni deoarece secretele trebuie să fie disponibile la toate capetele căilor de comunicație, lucru ce introduce un risc deloc de neglijat. [SIMP96]

8.2. Arhitectură

Algoritmul CHAP cere ca lungimea secretului să fie cel puțin un octet. Secretul ar trebui să fie cel puțin la fel greu de ghicit ca o parolă bine aleasă. Se preferă ca secretul să fie cel puțin la fel de lung ca valoarea de dispersie a algoritmului folosit (spre exemplu 16 pentru MD5). Această cerință oferă protecție împotriva căutărilor exhaustive.

Algoritmul de dispersie este ales în aşa fel încât să fie dificil şi nefezabil să se deducă secretul din valoarea provocării şi valorile de răspuns.

Fiecare provocare ar trebui să fie unică din moment ce repetarea unei valori în conjuncție cu acelaşi secret ar permite unui atacator să răspundă cu un mesaj interceptat anterior. Din moment ce este de aşteptat ca acelaşi secret să fie folosit la autentificarea cu servere situate în regiuni geografice separate, valoarea de provocare trebuie să prezinte unicitate temporală şi globală.

Valoarea de provocare trebuie să fie de asemenea imprevizibilă, deoarece un atacator ar putea să răspundă cu o valoarea presupus viitoare ceea ce ar duce evident la compromiterea sistemului.

Mesajul de configurare

Negocierea parametrilor de comunicație se face prin intermediul unui mesaj cu formatul special, descris mai jos. Câmpurile sunt trimise de la stânga la dreapta.

[0x03]	[0x05]	[0xC223]	[0x05]
Tip	Lungime	Protocol Autentificare (CHAP)	Algoritm (MD5)

Câmpul *Protocol Autentificare* conține valoarea 0xC223 pentru a semnaliza protocolul CHAP, iar câmpul *Algoritm* conține metoda de autentificare utilizată. Implementarea minimală cere algoritmul MD5 (identificat cu valoarea 0x05).

Formatul pachetelor CHAP

Formatul pachetelor CHAP este prezentat în cele ce urmează:

Cod	Identificator	Lungime	Date

Cod – acest câmp are lungimea de un octet şi identifică tipul pachetului CHAP. Codurile posibile sunt următoarele:

- 1 – Provocare
- 2 – Răspuns
- 3 – Succes
- 4 – Eşec

Identificator – are lungimea de un octet şi ajută la potrivirea provocărilor şi răspunsurilor.

Lungime – câmpul are o lungime de doi octeți şi indică lungimea totală a pachetului, inclusiv câmpurile de cod, identificator, lungime şi date. Datele recepționate peste această valoare se consideră umplutură din cadrul legăturii de date şi trebuie ignorate.

Date – câmpul de date este de lungime 0 sau mai mare. Formatul acestuia este dependent de semnificația codului. [SIMP96]

8.2.1. Provocare şi răspuns

Pachetul de provocare se foloseşte pentru a începe protocolul CHAP. Autentificatorul trebuie să trimită un pachet CHAP cu câmpul de cod setat pe 1 (provocare). În lipsa unui răspuns valid, pachetul se retransmite până la depăşirea unui număr de încercări prestabilit.

Un pachet de provocare poate să fie trimis de asemenea în orice moment pentru a evita modificarea ulterioară a parametrilor.

Omologul trebuie să se aştepte să primească pachete de provocare, iar la primirea acestora trebuie să răspundă cu pachete de răspuns (codul 2).

La primirea unui pachet de răspuns, autentificatorul compară valorile de răspuns cu cele calculate local şi decide dacă autentificarea s-a făcut cu succes. În funcție de rezultat se trimite un pachet de succes sau de eşec. [SIMP96]

8.2.2. Succes şi eşec

Dacă valoarea recepționată în răspuns este egală cu valoarea calculată local, atunci autentificatorul trebuie să trimită un pachet cu câmpul Cod setat pe 3 (succes). In caz contrar, se trimite un pachet cu câmpul Cod setat pe 4 (eşec) şi se iau măsurile care se impun pentru terminarea conexiunii. [SIMP96]

8.3. Concluzii

CHAP este un protocol simplu şi uşor de implementat. Specificația din RFC 1994 [SIMP96] este permisivă, lăsând la latitudinea implementatorului anumite decizii. Spre exemplu, când autentificarea eşuează, se poate opta fie pentru terminarea conexiunii fie

pentru limitarea tipului de trafic (spre exemplu, utilizatorul poate trimite e-mail administratorului de sistem informându-l despre probleme).

Nu există nici o cerință ca protocolul să fie implementat în dublu sens. Este perfect acceptabil să se folosească protocoale diferite în sensuri diferite. De asemenea, secretul nu trebuie să fie acelaşi în ambele direcții.

Compania Microsoft a dezvoltat propriile extensii ale acestui protocol, modificări ce sunt publicate în RFC 2433 [ZORN98]. Noul protocol, numit MS-CHAP este folosit în autentificarea sistemelor Windows prin dial-up.

Capitolul 9

Concluzii

Pe parcursul a opt capitole am prezentat situaţia actuală în domeniul protocoalelor de autentificare. Sub nici o formă nu putem considera că am epuizat acest subiect, aici rezumându-ne doar la cele mai cunoscute protocoale existente pe piaţă. Fiecare dintre acestea are avantajele şi dezavantajele sale, iar aria de aplicabilitate depinde şi ea într-o mare măsura de caracteristicile fiecăruia.

La modul ce mai simplu, problema autentificării se pune în momentul când două entităţi doresc să comunice pe o cale aleasă de comun acord. Problema e că într-un mediu electronic, falsificarea identităţii se poate realiza foarte uşor mai ales în lipsa unor măsuri speciale de protecţie. Dacă în cazul unei convorbiri telefonice între două persoane autentificarea o reprezintă chiar recunoaşterea vocii, în cazul unor sisteme electronice nu mai există nici o proprietate comparabilă care să ajute la autentificare. În schimb se apelează la o serie de schimburi de mesaje, dependente de multe ori de un terţ credibil, o sursă de numere pseudo-aleatoare şi în unele cazuri de sincronismul în timp al celor două sisteme.

9.1. Slăbiciuni existente actual

Se cunoaşte faptul că siguranţa unui sistem este determinată de cea mai puţin sigură parte a acestuia. Protocoalele de care am vorbit în această lucrare nu fac excepţie. Un atacator nu va alege niciodată spre exemplu să decodifice o comunicaţie securizată cu 1024 de biţi când poate instala un program de interceptare a apăsărilor de taste dacă i se dă ocazia. Dacă nu poate face acest lucru, va apela la metode de îngreunare a comunicaţiei între cele două părţi sau va încerca chiar compromiterea totală a canalului de comunicaţie. Aceasta acţiune se numeşte blocare distribuită a serviciului (***DDoS*** – *Distributed Denial of Service*).

Atacurile de tip blocare a serviciilor au fost semnalate în urmă cu decenii. Atacurile distribuite sunt mai recente, fiind semnalate pe la jumătatea anului 1999. Primul atac bine documentat a avut loc în august 1999 când un program numit **Trinoo** a fost instalat pe cel puţin 227 de sisteme cu scopul de a ataca prin cereri false un calculator al universităţii din Minnesota. Sistemul în cauză a fost oprit pentru mai bine de două zile.

Primul atac mediatizat în presa publică a avut loc în februarie 2000. Pe 7 februarie, Yahoo! a fost victima unei blocări a serviciului timp în care site-ul său a fost inaccesibil timp de trei ore. A doua zi, pe 8 februarie, site-urile *amazon.com*, *buy.com*, *cnn.com* şi *ebay.com* au fost atacate simultan cauzând fie oprirea completă sau încetinirea semnificativă. În cele trei ore în care nu a fost disponibil, Yahoo! a pierdut 500,000 de dolari din venituri provenite din vânzare de produse şi reclame. Similar, vânzătorul de cărți Amazon a pierdut 600,000 de dolari în cele 10 ore de nefuncționare. Fără îndoială, cifrele prezentate sunt alarmante, mai ales având în vedere perspectiva dezvoltării accelerate a comerțului electronic.

9.2. Munca viitoare

Am văzut anterior că în general serverele cu legătură directă la Internet sunt expuse şi descoperite în fața unui atac distribuit. De aceea, se impun măsuri de contracarare a acestui fenomen. Tendința actuală în domeniu este de a folosi aşa numitele **client puzzles** pentru autentificare. Ideea de bază constă în cerința ca un client candidat la autentificare să rezolve corect o problemă înainte ca serverul să aloce resurse pentru comunicația cu acesta.

Ideea de puzzle criptografic i se datorează lui Merkle [MERK78]. Totuşi, Merkle l-a utilizat pentru schimbul de chei în loc de controlul accesului. Puzzle-urile client au fost aplicate la inundarea prin TCP SYN de Juels şi Brainard [JUEL99], care au menționat că şi SSL are aceeaşi problemă. Aura, Nikander şi Leiwo aplică puzzle-urile la protocoalele de autentificare în general în [AURA00].

În ordinea importanței, scopurile în cercetarea viitoare sunt după cum urmează:

1. Împiedicarea atacurilor de tip DoS – *Denial of Service* asupra serviciilor de autentificare securizate.
2. Protocoalele de autentificare modificate să rămână compatibile în jos pe cât posibil, fără a afecta securitatea dovedită până în prezent.
3. Minimizarea încărcării serverului prin implementarea acestor măsuri.

Compatibilitatea în jos este mai importantă decât încărcarea serverului din moment ce în prezent puterea procesoarelor este în continuă creştere.

9.2.1. Împiedicarea atacurilor de tip DoS

Atacurile de tip DoS funcționează după un scenariu pe cât de simplu pe atât de eficient. În principiu, pe un număr oarecare (preferabil mare) de calculatoare conectate la Internet se instalează – fără ştiința şi aprobarea utilizatorului – un program controlabil de la distanță. La comandă sau după un timp predeterminat, programul face cereri către site-ul sau serviciul supus atacului, generând trafic sau încărcare mare. Pentru a fi încununat de succes, atacul trebuie să genereze trafic mai mare decât lățimea de bandă şi capacitatea de procesare a serverului, oricare este mai mică.

Luând exemplul unui site de comerț electronic, un atacator poate bloca vânzările site-ului prin atacul asupra serverului care procesează plăți prin cărți de credit, în timp ce website-ul rămâne disponibil. Acest lucru este posibil deoarece în forma sa actuală protocolul SSL/TLS acceptă ca serverul să efectueze o decriptare RSA, fără nici un control asupra identității clientului. După cum se ştie, o decriptare RSA este o operație relativ scumpă din punct de vedere a timpului de execuție, iar un site de comerț electronic de dimensiuni mari, la ora actuală (septembrie 2002) poate efectua până la 4000 de decriptări RSA pe secundă. Dacă presupunem că un handshake parțial al protocolului SSL are nevoie de aproximativ 200 de octeți rezultă că un trafic de 800 KB/sec. este suficient pentru a paraliza site-ul.

Ideea în spatele unui sistem de împiedicare a atacurilor de tip DoS este de a încetini suficient de mult un atacator până când atacul nu mai are succes, prin refuzul de a efectua orice fel de prelucrări de date în absența unei minime autentificări a clientului.

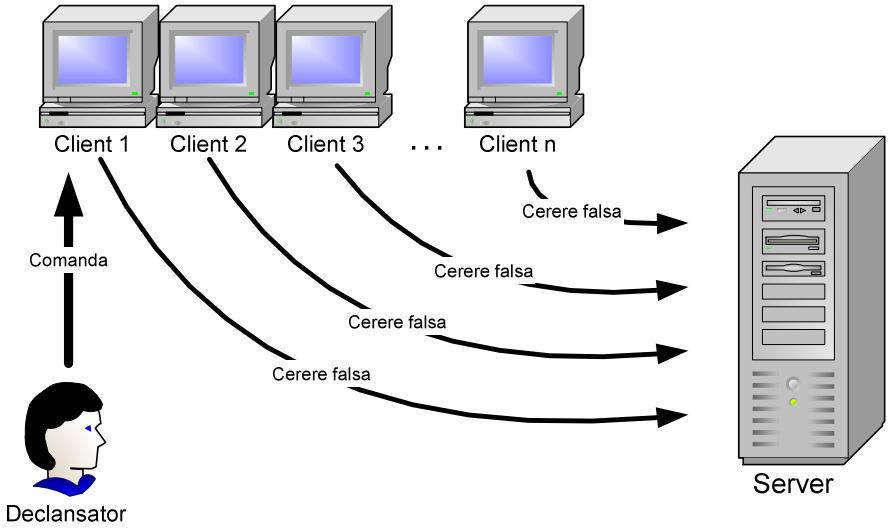

Figura 22. Modelul general al unui atac de tip Denial-of-Service

Protocoalele prezentate în această lucrare sunt în general slab sau deloc protejate împotriva atacurilor de tip Denial-of-Service. Se impune în această situație intervenția asupra unor părți din cadrul protocoalelor în discuție pentru a le conferi rezistență sau preferabil imunitate la atacuri de genul celor descrise anterior. Este încă devreme să ne pronunțăm cu exactitate asupra modulelor în care se va interveni în cadrul protocoalelor, acesta fiind subiectul unui referat ulterior. Totuşi, putem enunța un model general, ca în figura 23.

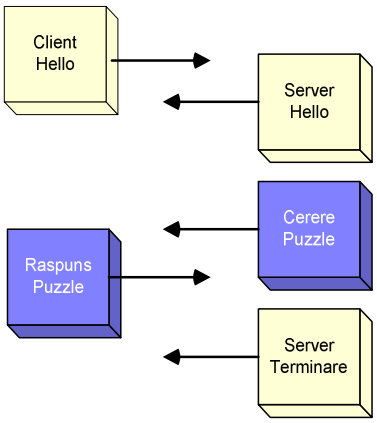

Figura 23. Modelul general al metodei client-puzzle

Modelul prezentat aici se pretează pentru majoritatea protocoalelor expuse în lucrare. Părțile în culoare închisă sunt adăugate la modelul general existent, reprezentat de fiecare protocol în parte.

9.2.2. Compatibilitatea cu sistemele existente

Modificarea protocoalelor de autentificare şi distribuție a cheilor în vederea obținerii unei rezistențe la atacuri de tip DoS trebuie să aibă în vedere şi compatibilitatea cu parcul de software instalat în prezent. O modificare – oricâte beneficii ar aduce aceasta – va fi cu greu implementată în masă dacă presupune schimbări majore în software sau ruperea compatibilității cu versiunea anterioară.

Acest deziderat poate sau nu să fie posibil, în funcție de evoluția modificărilor propuse pentru protocoale.

9.2.3. Minimizarea încărcării serverului

Se impune ca puzzle-urile să fie rezolvabile într-un timp finit şi în limite rezonabile, pentru ca serverul să nu fie ocupat inutil cu activități auxiliare. În lucrările viitoare vom experimenta în acest domeniu, deoarece timpul de rezolvare determină gradul de siguranță în cazul unui atac.

Bibliografie

1. [AURA00] Tuomas Aura, Pekka Nikander, Jussipekka Leiwo – *DoS-resistant Authentication with Client Puzzles*. Proceedings of the Cambridge Security Protocols Workshop 2000, LNCS, Cambridge, UK, April 2000, Springer-Verlag
2. [AGNE90] G. B. Agnew, R. C. Mullin, S. A. Vanstone – *Improved Digital Signature Scheme based on Discrete Exponentiation*, Electronic Letters, Vol. 26, 1990
3. [ALAG91a] K. Alagappan – *SPX Installation*, Digital Equipment Corporation, February 1991
4. [ALAG91b] K. Alagappan, J. Tardo – *SPX Guide – A Prototype Public Key Authentication Service*, Digital Equipment Corporation, February 1991
5. [ALAG93] K. Alagappan – *Telnet authentication: SPX (RFC 1412)*, Digital Equipment Corporation, 1993
6. [BAUS90] F. Bauspiess, H. J. Knobloch – *How to Keep Authenticity Alive in A Computer Network,* Proceedings of EUROCRYPT' 89, Springer-Verlag, Berlin, 1990
7. [BELL92] S. M. Bellovin, M. Merritt – *Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks*, Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, 1992
8. [BELL93] S. M. Bellovin, M. Merritt – *Augumented Encrypted Key Exchange*, Proceedings of the 1st ACM Conference on Communications and Computing Security, November 1993
9. [BEME] Steven M. Bellovin, Michael Merritt – *Limitations of the Kerberos Authentication System,* AT&T Bell Labs
10. [BETH89] Th. Beth – *Efficient Zero-Knowledge Identification Scheme for Smart Cards*, Proceedings of EUROCRYPT '88, Springer-Verlag, Berlin, 1989
11. [BETH93] Th. Beth, H. J. Knobloch, M. Otten – *Verifiable Secret Sharing for Monotone Access Structures*, Proceedings of the 1st ACM Conference on Communication and Computing Security, November 1993
12. [BETH94] Th. Beth, H. J. Knobloch, M. Otten, G. J. Simmons, P. Wichmann – *Towards Acceptable Key Escrow System*, Proceedings of the 2nd ACM Conference on Communication and Computing Security, November 1994
13. [BETH94a] T. Beth, H. J. Knobloch, S. Stempel, P. Wichmann – *Authentifikationsdienst SELANE – Modularisierung und Einsatz*, Report 94/3, Univeristy of Karlsruhe, EISS, 1994

14. [BETH94b] T. Beth, D. Gollmann – *Security Systems Based on Exponentiation Primitives: TESS – The Exponential Security System*, Proceedings of IFIP SEC ’94, May 1994
15. [BETH95] T. Beth – *Sichere offene Datennetze*, Spektrum der Wissenschaft, May 1995
16. [BIRD92] R. Bird, I. Gopal, A. Herzberg, P. A. Janson, S. Kutten, R. Molva, M. Yung – *Systematic Design of Two-Party Authentication Protocols*, Advances in Cryptology, CRYPTO’ 91, Springer-Verlag, 1992
17. [BIRD93] R. Bird, I. Gopal, A. Herzberg, P. A. Janson, S. Kutten, R. Molva, M. Yung – *Systematic Design of a Family of Attack-Resistent Authentication Protocols”, IEEE Journal on Selected Areas in Communications,* Vol. 11, 1993
18. [BIRD95] R. Bird, I. Gopal, A. Herzberg, P. A. Janson, S. Kutten, R. Molva, M. Yung – *The KryptoKnight Family of Light-Weight Protocols for Authentication and Key Distribution”*, IEEE / ACM Transactions on Networking, Vol. 3, 1995
19. [BORM93] D. Borman – *Telnet Authentication: Kerberos Version 4 (RFC 1411)*, Cray Research, Inc., 1993
20. [BURR89] M. Burrows, M. Abadi, R. Needham – *“A Logic of Authentication*”, ACM Operating Systems Review, Vol. 23, 1989
21. [CAMP99] Roy H. Campbell, M. Dennis Mickusas, Monika Chandak – *Sesame Authentication Protocol*, University of Illinois at Urbana-Champaign, 1999
22. [CHAM90] G. A. Champine, D. E. Geer, W. N. Ruh – *“Project Athena as a Distributed Computer System”,* IEEE Computer, Vol. 23, September 1990
23. [CHAM91] G. A. Champine – *“MIT Project Athena – A Model for Distributed Computing”*, Digital Press, 1991
24. [CHAU88] D. Chaum, J. H. Evertse, J. van de Graaf – *An Improved Protocol for Demonstrating Possesion of Discrete Logarithms and some Generalizations*, Proceedings of EUROCRYPT ’87, Springer-Verlag, Berlin, 1988
25. [CHAU92] D. Chaum – *Achieving Electronic Privacy*, Scientific American, August 1992
26. [CHEN95] P. C. Cheng, J. A. Garay, A. Herzberg, H. Krawczyk – *Design and Implementation of Modular Key Management Protocol and IP Secure Tunnel on AIX*, Proceedings of the USENIX UNIX Security V Symposium , USENIX Association, Berkeley, CA, 1995
27. [COHE87] F. B. Cohen – *A Cryptographic Checksum for Integrity Protection*, Computers & Security, Vol. 6, 1987
28. [DANI95] H. Danisch – *The Exponential Security System TESS: An Integrity-Based Cryptographic Protocol for Authenticated Key-Exchange*, E.I.S.S./IAKS, 1995
29. [DANI95] H. Danisch – *The Exponential Security System TESS: An Identity-Based Cryptographic Protocol for Authenticated Key-Exchange*, RFC 1824, August 1995
30. [DELG] Delgado – *Kerberos vs. SSL***,** Stanford University
31. [DENN81] D. E. Denning, G. Sacco - *“Timestamps in Key Distribution Protocols”*, Communications of the ACM, Vol. 24, 1981
32. [DIFF88] W. Diffie – *The First Ten Years of Public-Key Cryptography*, Proceedings of the IEEE, Vol. 76, 1988
33. [FELD90] D. C. Feldmeier, P. R. Karn – *UNIX Password Security – Ten Years Later*, Advances in Cryptology – CRYPTO ’89, Springer-Verlag, Berlin, 1990
34. [FREI96] Alan O. Freier, Philip Karlton, Paul C. Kocher – *The SSL Protocol, Version 3.0 (Internet Draft)*, Transport Layer Security Working Group, 1996

35. [GASS89] M. Gasser, A. Goldstein, C. Kaufman, B. Lampson – *The Digital Distributed System Security Architecture*, Proceedings of the 12th National Computer Security Conference, 1989
36. [GIAM97] Giampaolo Bella – *Formal Analysis of the Kerberos Authentication System,* University of Cambridge, 1997
37. [GIRA91] M. Girault – *Self-certified Public Keys,* Proceedings of EUROCRYPT' 91, Springer-Verlag, Berlin, 1991
38. [GONG89] L. Gong – *Using One-Way Functions for Authentication*, ACM Computer Communication Review, Vol. 22, 1989
39. [GONG90] L. Gong, R. Needham, R. Yahalom – *"GNY Logic Fill In"*, Proceedings of the IEEE Symposium on Security and Privacy, 1990
40. [GONG92] L. Gong – *"A Security Risk of Depending on Synchronized Clocks"*, ACM Operating Systems Review, Vol. 26, 1992
41. [GONG93] L. Gong – *Increasing Availability and Security of an Authentication Service*, IEEE Journal of Selected Areas in Communications, Vol. 11, June 1993
42. [GONG93] L. Gong, T. M. A. Lomas, R. M. Needham, J. H. Saltzer – *Protecting Poorly Chosen Secrets from Guessing Attacks*, IEEE Journal on Selected Areas in Communications, Vol. 11, June 1993
43. [GUNT90] C. G. Günther – *An Identity-Based Key-Exchange Protocol*, Proceedings of EUROCRYPT ,89, Springer-Verlag, Berlin, 1990
44. [HORS91] P. Horster, H. J. Knobloch – *Discrete Logarithm Based Protocols*, Proceedings of EUROCRYPT' 91, Springer-Verlag, Berlin, 1991
45. [HORS92] P. Horster, H. J. Knobloch – *Cryptographic Protocols and Network Security*, Proceedings of the IFIP SEC '92, May 1992
46. [JABL97] David P. Jablon – *Strong Password-Only Authenticated Key Exchange*, Integrity Sciences, Inc., 1997
47. [JUEL99] Ari Juels, John Brainard – *Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks*, Proceedings of NDIS 1999.
48. [KALI95] B. Kaliski, M. Robshaw – *Message Authentication with MD5*, RSA Laboraties' Cryptobytes, Vol. 1, 1995
49. [KAUF93] C. Kaufman – *DASS – Distributed Authentication Security Service, RFC 1507,* September 1993
50. [KLEI90] D. V. Klein – *"Foiling the Cracker": A Survey of, and Improvements to, Password Security,* Proceedings of the USENIX UNIX Security II Symposium, USENIX Association, Berkeley 1990
51. [KLEI93] B. Klein – *Authentifikationsdienste für sichere Informationssysteme*, Ph.D. Thesis, University of Karlsruhe, Germany, 1993
52. [KNOB92] H. J. Knobloch, P. Horster – *Eine Krypto-Toolbox für Smartcards*, Datenschutz und Datensicherung, Vol. 16, July 1992
53. [KOHL93] J. Kohl – *The Kerberos Network Authentication Service (V5) (RFC 1510)*, Digital Equipment Corporation, 1993
54. [KOHL94] J. T. Kohl, B. C. Neuman, T. Y. Ts'o – *The Evolution of the Kerberos Authentication System*, Distributed Open Systems, IEEE Computer Society Press, Los Alamitos, CA, 1994
55. [LAMB92] K. Y. Lam, T. Beth – *Timely Authentication in Distributed Systems",* ESORICS '92 – European Symposium on Research in Computer Security, Springer-Verlag, Berlin, November 1992
56. [LAMP78] L. Lamport – *Time, Clocks and the Ordering of Events in a Distributed System*, Communications of the ACM, Vol. 21, July 1978

57. [LINN90] J. Linn – *Practical Authentication for Distributed Computing*, Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos, 1990
58. [LINN96] J. Linn – *The Kerberos Version 5 GSS-API Mechanism(RFC 1964),* OpenVision Technologies, 1996
59. [LOMA89] T. M. A. Lomas, L. Gong, J. H. Saltzer, R. M. Needham – *Reducing Risks from Poorly Chosen Keys*, ACM Operating Systems Review, Vol. 23, 1989
60. [MEDV99] A. Medvinsky – *Addition to Kerberos Cipher Suites to Transport Layer Security (TLS) (RFC 2712)*, Excite, 1999
61. [MERK78] R. C. Merkle – *Secure Communications Over Insecure Channels*, Communications of the ACM, April 1978
62. [MOLV92] R. Molva, G. Tsudik, E. Van Herreweghen, S. Zatti – *KryptoKnight Authentication and Key Distribution System,* ESORICS '92 – European Symposium on Research in Computer Security, Springer-Verlag, 1992
63. [NEED78] R. M. Needham, M. D. Schroeder – *"Using Encryption for Authentication in Large Networks of Computers"*, Communications of the ACM, Vol. 21, December 1978
64. [NEED87] R. M. Needham, M. D. Schroeder – *"Authentication Revisited"*, ACM Operating Systems Review, Vol. 21, 1987
65. [NETS98] Netscape Corporation – *Introduction to SSL,* http://developer.netscape.com
66. [OPPL96] Rolf Oppliger – *Authentication Systems for Secure Networks*, Artech House, Inc., 1996
67. [ORA95] Oracle White Paper – *Client/Server Authentication*, 1995
68. [PARK91] T. A. Parker – *A Secure European System for Applications in a Multi-vendor Environment (The SESAME Project)*, Proceedings of the 14th National Computer Security Conference, 1991
69. [PARK95] Tom Parker, Denis Pinkas – *Sesame Technology Version 4*, 1995
70. [PATR94] Victor-Valeriu Patriciu – *Criptografia şi securitatea rețelelor de calculatoare*, Editura Tehnică, 1994
71. [PIES93] F. Piessens, B. De Decker, P. Jason – *Interconnecting Domains with Heterogeneous Key Distribution and Authentication Protocols*, Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, 1993
72. [PREN95] B. Preneel, P. C. van Oorschot – *MDx-MAC and Building Fast MACs from Hash Functions*, Advances in Cryptology – CRYPTO '95, Springer Verlag, 1995
73. [QUIS90] J. J. Quisquater, L. Guillou – *"How to Explain Zero-Knowledge Protocols to Your Children"*, Advances in Crytoplogy – CRYPTO '89, Springer – Verlag, Berlin 1990
74. [REBI94] M. K. Reiter, K. P. Birman – *How to Securely Replicate Services*, ACM Transactions on Programming Languages and Systems, Vol. 16, 1994
75. [RIVE84] R. L. Rivest, A. Shamir – *How to Expose an Eavesdropper,* Communications of the ACM, Vol. 27, 1984
76. [SCHI94] J. I. Schiller – *"Secure Distributed Computing"*, Scientific American, November 1994
77. [SCHI95] J. I. Schiller, D. A. Atkins – "*Scaling the Web of Trust: Combining Kerberos and PGP to Provide Large Scale Authentication",* Proceedings of the Technical Conference on UNIX and Advanced Computing Systems, USENIX Association, Berkeley CA, January 1995

78. [SCHMD] Bruce Schneier, Mudge – *Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)*
79. [SCHN90] C. P. Schnorr – *Efficient Identification and Signatures for Smart Cards*, Advances in Cryptology – CRYPTO '89, Springer-Verlag, Berlin, 1990
80. [SCHN96] Bruce Schneier – *Applied Cryptography*, John Wiley & Sons, Inc., 1996
81. [SHAM85] A. Shamir – *Identity-based Cryptosystem and Signature Schemes*, Advances in Cryptography – CRYPTO' 84, Springer-Verlag, Berlin, 1985
82. [SHAM87] A. Fiat, A. Shamir – *"How to Prove Yourself: Practical Solutions to Identification and Signature Problems"*, Advances in Crytoplogy – CRYPTO '86, Springer – Verlag, Berlin 1987
83. [SHAN48] C.E. Shannon – *A Mathematical Theory of Communications*, The Bell System Technical Journal, Vol. 27, July / October 1948
84. [SHAN49] C.E. Shannon – *Communication Theory of Secrecy Systems*, The Bell System Technical Journal, Vol. 28, October 1949
85. [SIMP96] W. Simpson – *PPP Challenge Handshake Authentication Protocol (CHAP) (RFC 1994)*, DayDreamer, 1996
86. [STEI88] J. G. Steiner, B. C. Neuman, J. I. Schiller – *"Kerberos: An Authentication Service for Open Network Systems"*, Proceedings of the USENIX UNIX Security Symposium, 1988
87. [TARD90] J. Tardo, K. Alagappan, R. Pitkin – *Public-Key Based Authentication Using Internet Certificates*, Proceedings of the USENIX UNIX Security II Symposium, USENIX Association, Berkeley, CA, 1990
88. [TARD91] J. Tardo, K. Alagappan – *SPX: Global Authentication Using Public Key Certificates*, Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, 1991
89. [TJW98] Thomas Wu – *A Real-World Analysis of Kerberos Password Security*, Computer Science Department, Stanford University, 1998
90. [TKS00] Johnathan Trostle, Irina Kosinovsky, Michael M. Swift – *Implementation of Crossrealm Referral Handling in the MIT Kerberos Client*, University of Washington, 2000
91. [TSUD92] G. Tsudik – *Message Authentication with One-Way Hash Functions*, ACM Computer Communication Review, Vo. 22, 1992
92. [TSUD93] G. Tsudik, E. Van Herreweghrn – *On Simple and Secure Key Distribution*, Proceedings of the 1sr ACM Conference on Communications and Computing Security, November 1993
93. [TTS00] T. Ts'o – *Telnet Authentication: Kerberos Version 5 (RFC 2942),* VA Linux Systems, 2000
94. [VANG] Mark Vandenwauver, René Govaerts, Joos Vandewalle – *Role Based Access Control in Distributed Systems,* Katholieke Universiteit Leuven
95. [WAGS] David Wagner, Bruce Schneier – *Analysis of the SSL 3.0 protocol*
96. [ZORN98] G. Zorn, S. Cobb - *Microsoft PPP CHAP Extensions (RFC 2433)*, Microsoft Corporation, 1998

Glosar

- **AES** (AES): Criptosistem cu chei secrete (Advanced Encryption Standard)
- **Amenințare** (Threat): Circumstanță, condiție sau eveniment cu potențial de a viola securitatea resurselor sistem.
- **Analiza traficului** (Traffic analysis): Deducerea de informații prin observarea traficului de date (prezență, absență, cantitate, direcție şi frecvență).
- **Arhitectură de securitate** (Security architecture): Descriere de nivel înalt a structurii unui sistem cu funcții de securitate atribuite componentelor din cadrul acesteia.
- **Atac cu redare** (Replay attack): Un atac asupra unui sistem de autentificare prin înregistrarea şi redarea mesajelor valide trimise anterior. Orice informație constantă de autentificare cum ar fi parole, dispersia unei parole sau chiar informații biometrice transmise digital se pot înregistra şi mai târziu reda pentru ca un mesaj fals să apară autentic.
- **Atac de întrețesere** (Interleaving attack): Atac bazat pe abilitatea atacatorului de a utiliza secvențe de mesaje înregistrate de la execuții trecute ale protocolului.
- **Atribut de securitate** (Security attribute): Informație de securitate asociată cu un participant într-un sistem distribuit.
- **Audit de securitate (**Security audit): Examinare independentă a înregistrărilor din sistem şi a activităților pentru a testa eficiența controalelor în sistem, pentru a asigura alinierea la politica şi procedurile operaționale adoptate, pentru detectarea breşelor în securitate şi pentru recomandarea schimbărilor în politică şi procedurile de control.
- **Autentificare (A**uthentication): Procesul de verificare a identității pretinse de un participant.
- **Autentificare inter-domenii (**Inter-realm authentication): Autentificare dincolo de granițele domeniului.
- **Autentificator (**Authenticator): O înregistrare de date care conține informații care se pot dovedi că au fost generate recent folosind o cheie de sesiune cunoscută doar de un client şi serverul solicitat.
- **Autoritate de certificare (**Certification authority): Terț de încredere care creează, atribuie şi distribuie certificate.
- **Autoritate de certificare încrucişată (**Cross certifying CA): Autoritate de certificare care emite certificate pentru participanți arbitrari şi autorități peste care nu are jurisdicție imediată.
- **Autorizare (**Authorization): Procesul de acordare a drepturilor, ce include acordarea accesului pe baza drepturilor de acces.

- **Bilet (**Ticket): Înregistrare de date ce pot fi utilizate pentru autentificare.
- **Blocarea serviciului (**Denial of service): Împiedicarea accesului autorizat la resurse sau întârzierea operațiilor critice.
- **Cale de certificare (**Certification path): O secvență de certificate începând cu un certificat emis de autoritatea de certificare a unui participant şi terminând cu certificatul pentru un alt participant, unde fiecare certificat din cale conține cheia publică pentru verificarea certificatului următor.
- **Capabilitate (**Capability): Înregistrare de date ce poate servi ca identificator pentru o resursă aşa încât posesia să confere drepturi de acces la resursă.
- **Certificat (**Certificate): Înregistrare de date care furnizează cheia publică a unui participant, împreună cu alte informații legate de numele participantului şi autoritatea de certificare care l-a emis. Certificatul nu se poate falsifica deoarece poartă semnătura digitală a autorității.
- **Cheie (**Key): Secvență de simboluri care controlează operațiile de cifrare şi descifrare.
- **Cheie de delegare (**Delegation key): O cheie utilizată la delegare.
- **Cheie de sesiune (**Session key): O cheie temporară cunoscută de două părți, cu timp de viață limitat.
- **Cheie inter-domeniu (**Inter-realm key): Cheie secretă cunoscută de două centre de distribuție ale cheilor (KDC) în domenii Kerberos diferite.
- **Cheie privată (**Private key): Cheie criptografică utilizată în criptografia cu chei publice pentru a semna şi / sau decodifica mesaje.
- **Cheie publică (**Public key): Cheie disponibilă public, utilizată în criptosisteme asimetrice.
- **Cheie publică auto-certificată (**Self-certified public key): Cheie publică ce poate fi calculată din identificare deținătorului şi alte informații publice.
- **Cheie secretă (**Secret key): Cheie utilizată într-un criptosistem simetric, cunoscută de părțile implicate în comunicare.
- **Cifrare (**Encipherment): Transformare criptografică a datelor pentru a produce text cifrat.
- **Client (**Client): Un proces care cere şi în cele din urmă obține un serviciu de rețea. Un client de regulă acționează în numele utilizatorului.
- **Completarea traficului (**Traffic padding): Generarea de informații nefolositoare în cadrul unităților de date.
- **Compromiterea comunicației (**Communication compromise): Rezultatul subversiunii unei linii de comunicație într-o rețea de calculatoare sau sistem distribuit.
- **Compromiterea gazdei (**Host compromise): Rezultatul subversiunii unei gazde individuale în cadrul unei rețele sau sistem distribuit.
- **Confidențialitate (**Confidentiality): Proprietatea că informația nu este făcută disponibilă sau distribuită părților neautorizate.
- **Contabilizare (**Accounting): Procesul de măsură al utilizării resurselor unui participant oarecare.
- **Context de autentificare (**Authentication context): Informație transmisă în timpul unei instanțe particulare ale autentificării.
- **Controlul accesului (**Access control): Procesul de împiedicare a utilizării neautorizate ale resurselor, incluzând utilizarea într-o manieră neautorizată.

- **Credenţiale (**Credentials): Înregistrare de date necesare pentru a stabili identitatea pretinsă a unui participant. În modelul Kerberos, credenţialele se referă la un bilet plus cheia secretă de sesiune necesară pentru a folosi cu succes biletul pentru autentificare.
- **Criptologie (**Cryptology): Ştiinţa comunicaţiilor securizate.
- **Delegare (**Delegation): Procesul prin care un participant îi permite altuia să acţioneze în numele său.
- **DES (**DES): Criptosistem cu chei secrete (Data Encryption Standard)
- **Descifrare (**Decipherment): Inversa cifrării.
- **Disponibilitate (**Availability): Proprietatea de a fi accesibil şi utilizabil la cererea unei entităţi autorizate.
- **Domeniu**[1] **(**Realm): Domeniu de autentificare în Kerberos.
- **ECMA (**ECMA): *European Computer Manufacturer Association* – este o asociaţie europeană fondată în 1961 şi dedicată standardizării informaţiei şi a sistemelor de comunicaţii.
- **Etichetă (**Label): Informaţii legate de securitate asociate unui obiect.
- **Gazdă (**Host): Entitate adresabilă în cadrul unei reţele de calculatoare sau sistem distribuit. Entitatea este adresată tipic fie prin numele său sau prin adresa din nivelul de reţea.
- **IDEA (**IDEA): Criptosistem cu chei secrete (International Data Encryption Algorithm)
- **Identificator de participant (**Principal identifier): Identificator utilizat pentru a identifica în mod unic un participant.
- **Informaţie (**Information): Cunoştinţe comunicate sau recepţionate cu privire la un fapt sau o circumstanţă în general, şi datele care pot fi codificate în scopul procesări de către un calculator sau dispozitiv similar.
- **Informaţie de autentificare (**Authentication information): Informaţie folosită la autentificare.
- **Iniţiator (**Initiator): Participant care joacă un rol activ, spre exemplu cere accesul.
- **Integritate (**Integrity): Proprietate care asigură că datele sunt transmise de la sursă la destinaţie fără alterări nedetectate.
- **ISO (**ISO): *International Organization for Standardization* este un organism non-guvernamental înfiinţat în anul 1947. Misiunea sa este de a promova dezvoltarea standardizării şi a activităţilor conexe pentru facilitarea schimbului internaţional de bunuri şi servicii, precum şi dezvoltarea cooperării în sferele intelectuale, ştiinţifice, tehnologice şi economice.
- **ITU (**ITU): *International Telecommunications Union* este o organizaţie internaţională în cadrul căruia guvernele şi sectorul privat coordonează reţelele şi serviciile de telecomunicaţii. Activităţile organizaţiei includ coordonarea, dezvoltarea, regularizarea şi standardizarea telecomunicaţiilor.
- **Înlocuire (**Masquerade): Faptul că un participant pretinde în mod neautorizat că este un alt participant.
- **Kerberos (**Kerberos): Sistem de autentificare şi distribuţie a cheilor dezvoltat la MIT.
- **Limitare (**Limitation): Facilitate indisponibilă de cele mai multe ori.
- **Managementul cheilor (**Key management): Generarea, stocarea, distribuţia, ştergerea, arhivarea şi aplicarea cheilor în acord cu politica de securitate.

[1] Deşi traducerea termenului *realm* în limba română este *tărâm*, am avut în vedere adoptarea unui alt termen apropiat ca înţeles dar cu o mai bună legătură cu domeniul calculatoarelor.

- **Mesaj audit de securitate (**Security audit message): Un mesaj generat ca urmare a apariției unui eveniment auditat.
- **Negare (**Repudiation): Negare a unei entități implicate în comunicare că ar fi luat parte la o parte sau la toată comunicarea.
- **NetSP (**NetSP): Sistem de autentificare şi distribuție a cheilor dezvoltat de IBM.
- **Non-negare** (Non-repudiation): Proprietatea ca un receptor să fie capabil să dovedească faptul că transmițătorul unor informații le-a trimis într-adevăr, chiar dacă ulterior neagă acest lucru.
- **Participant (**Principal): Persoană sau sistem înregistrat şi autentificabil într-o rețea de calculatoare sau sistem distribuit.
- **Pereche de chei (**Key pair): Un set de două chei (una publică şi una privată) care au sens doar împreună.
- **Politică de autorizare (**Authorization policy): Un set de reguli, parte a unei politici de control al accesului prin care accesul la obiecte este permis sau interzis. O politică de autorizare se poate defini ca liste de control al accesului, capabilități sau atribute asociate obiectelor.
- **Preautentificare (**Preauthetication): Autentificare înainte schimbului de autentificare propriu-zis.
- **Pretinzător (**Claimant): Participant care caută să fie recunoscut ca autentic.
- **Proces (**Process): Instanțiere a unui program, rulând pe o gazdă particulară.
- **RC2, RC4 şi RC5 (**RC2, RC4 and RC5): Criptosisteme cu chei secrete.
- **Responsabilitate (**Accountability): Proprietate care asigură că acțiunile unui participant oarecare pot fi urmărite înapoi către acesta.
- **Rețea de calculatoare (**Computer network): Colecție de sisteme de calcul autonome şi interconectate.
- **Revocarea certificării (**Certification revocation): Anunț că o cheie privată a fost compromisă şi că certificatul care aparține cheii publice corespunzătoare nu ar mai trebui folosit pentru autentificare.
- **RSA (**RSA): Criptosistem cu chei publice (Rivest, Shamir, Adelman)
- **Schimb de autentificare (**Authentication exchange): O secvență de unul sau mai multe mesaje trimise pentru autentificare.
- **Semnătură digitală (**Digital signature): Transformare criptografică a unei unități de date care permite receptorului să dovedească sursa şi integritatea unității şi să o protejeze de falsificare.
- **Server (**Server): Proces care furnizează un serviciu de rețea.
- **Serviciu (**Service): Set coerent de funcționalitate abstractă.
- **SESAME (**SESAME): Sistem de autentificare şi distribuție a cheilor dezvoltat ca parte a unui proiect de cercetare şi dezvoltare.
- **Simbol de autentificare (**Authentication token): O înregistrare de date care conține informația de autentificare.
- **Sistem deschis (**Open system): Sistem care se conformă la standarde deschise.
- **Sistem distribuit (**Distributed system): Rețea de calculatoare în care existența mai multor sisteme autonome este transparentă şi deci nu neapărat vizibilă pentru utilizator.
- **SPX (**SPX): Sistem de autentificare şi distribuție a cheilor proiectat de DEC.
- **Standard (**Standard): Înțelegere documentată conținând specificații tehnice sau alte criterii precise care să fie utilizate ca reguli, linii directoare sau definiții de caracteristici, în aşa fel încât materialele, produsele, procesele şi serviciile se potrivesc scopului lor.

- **Standard deschis (**Open standard): Standard care specifică un sistem deschis şi care permite oricărui fabricant construirea produselor corespunzătoare.
- **Tehnologia informației (**Information technology – IT): Tehnologie care se ocupă cu informația.
- **Terţ credibil (**Trusted third party): O autoritate sau agentul său, în care au încredere alte entități în aspecte legate de securitate.
- **TESS (**TESS): Un set de funcții şi mecanisme diferite dar cooperante bazate pe exponențierea discretă.
- **Text cifrat (**Cipher text): Rezultatul unei funcții de criptare. Criptarea transformă textul clar în text cifrat.
- **Text clar (**Plain text): Intrarea unei funcții de criptare sau ieşirea unei funcții de decriptare. Decriptarea transformă textul cifrat în text clar.
- **Țintă (**Target): Participant care joacă rol pasiv, spre exemplu fiind accesat.
- **Urme de audit de securitate (**Security audit trail): Date colectate şi utilizate pentru a facilita un audit de securitate.
- **Utilizator (**User): Participant care este făcut responsabil pentru activitățile sale dintr-o rețea de calculatoare sau sistem distribuit.
- **Valoare (**Nonce): Număr aleator şi imprevizibil generat recent.
- **Verificator (**Verifier): Participant care caută să autentifice un pretinzător.
- **Vulnerabilitate (**Vulnerability): Slăbiciune care poate fi exploatată pentru a viola un sistem sau informațiile pe care acesta le conține.

Abrevieri şi acronime

- **ACL** Access Control List
- **AES** **A**pplication Environment Specification
- **ANSI** American National Standards Institute
- **APA** **A**uthentication and Privilege Attribute
- **API** **A**pplication Programming Interface
- **AS** Authentication Server
- **ASN.1** Abstract Syntax Notation 1
- **ATM** Asynchronous Transfer Mode

- **BAN** Burrows, Abadi, Needham
- **Bellcore** Bell Communications Research
- **BER** Basic Encoding Rules
- **BFI** Swiss Federal Office of Information Technology and Systems

- **CA** Certification Authority
- **CAA** Certification Authority Agent
- **CAE** Common Applications Environment
- **CAT** Common Authentication Technology
- **CBC** Cipher Block Chaining
- **CCITT** Consultative Committee on International Telegraphy and Telephony (now ITU-T)
- **CD** Compact Disc
- **CDC** Certificate Distribution Center
- **CDMF** Commercial Data Masking Facility
- **CDS** Cell Discovery Service
- **CEC** Commission of the European Communities
- **CFB** Cipher Feedback
- **CSF** Cryptographic Support Facility
- **CV** Control Value

- **DAC** Discretionary Access Control
- **DASS** Distributed Authentication Security Service

- **DCE** Distributed Computing Environment
- **DEC** Digital Equipment Corporation
- **DES** Data Encryption Standard
- **DIT** Directory Information Tree
- **DNS** Domain Name Service
- **DoC** U.S. Department of Commerce
- **DoD** U.S. Department of Defense
- **DoD** U.S. Department of State
- **DOS** Disk Operating System
- **DSA** Digital Signature Algorithm
- **DSS** Digital Signature Standard
- **DSS** Domain Security Standard
- **DSSA** Distributed System Security Architecture
- **DTI** Directory Information Tree

- **ECB** Electronic Code Book
- **ECMA** European Computer Manufacturers Association
- **EDI** Electronic Data Interchange
- **EES** Exponential Electronic Signature
- **EISS** European Institute for System Security
- **EKE** Encrypted Key Exchange
- **EPAC** Extended Privilege Attribute Certificate
- **EU** European Union

- **FAQ** Frequently Asked Questions
- **FEAL** Fast Encryption Algorithm
- **FIPS** Federal Information Processing Standard
- **FTP** File Transfer Protocol

- **GDA** Global Domain Agent
- **GDS** Global Domain Service
- **GNY** Gong, Needham, Yahalom
- **GSS-API** Generic Security Service API

- **HP** Hewlett-Packard

- **IAM** Institute for Computer Science and Applied Mathematics
- **IBM** International Business Machines Corporation
- **ICL** International Computers Limited
- **ICSI** International Computers Science Institute
- **IDEA** International Data Encryption Algorithm
- **IDS** Inter Domain Service
- **IEC** International Electrotechnical Committee
- **IEEE** Institute of Electrical and Electronic Engineers
- **IETF** Internet Engineering Task Force
- **IP** Internet Protocol
- **IPSEC** IP Security Protocol

- **IPST** IP Secure Tunnel Protocol
- **IS** International Standard
- **ISO** International Organization for Standardization
- **ISODE** ISO Development Environment
- **IT** Information Technology
- **ITU-T** International Telecommunication Union

- **JTC1** Joint Technical Committee 1

- **KDC** Key Distribution Center
- **KDS** Key Distribution Server
- **KEK** Key Encryption Key
- **KTC** Key Translation Center

- **LAN** Local Area Network
- **LEAF** Login Enrollment Agent Facility
- **LLC** Logical Link Control
- **LRA** Local Registration Authority

- **MAC** Message Authentication Code
- **MAN** Metropolitan Area Network
- **MD** Message Digest
- **MDC** Modification Detection Code
- **MHS** Message Handling System
- **MIB** Management Information Base
- **MIC** Message Integrity Code
- **MIT** Massachusetts Institute of Technology
- **MKMP** Modular Key Management Protocol

- **NBS** National Bureau of Standards
- **NCSC** National Computer Security Center
- **NCSL** National Computer Systems Laboratory
- **NetSP** Network Security Program
- **NII** National Information Infrastructure
- **NIST** National Institute of Standards and Technology
- **NLSP** Network Layer Security Protocol
- **NSA** National Security Agency

- **OFB** Output Feedback
- **OSF** Open Software Foundation
- **OSI** Open Systems Interconnection
- **OSI-RM** OSI Reference Model
-

- **PAC** Privilege Attribute Certificate
- **PAS** Privilege Attribute Server
- **PC** Personal Computer
- **PEM** Privacy Enhanced Mail

- **PGP** Pretty Good Privacy
- **PIN** Personal Identification Number
- **PKCS** Public Key Cryptography Standard
- **PKM** Public Key Management
- **PKP** Public Key Partners
- **PPID** Primary Principal Identifier
- **PT** Privilege Ticket
- **PTGT** Privilege Ticket Granting Ticket
- **PV** Protection Value
- **PVF** PAC Validation Facility

- **RACF** Resource Access Control Facility
- **RFC** Request for Comments
- **RFT** Request for Technology
- **ROM** Read Only Memory
- **RPC** Remote Procedure Call
- **RSA** Rivest, Shamir, Adelman

- **SACM** Secure Association Context Manager
- **SELANE** Secure Local Area Network Environment
- **SESAME** Secure European System for Applications in a Multi-vendor Environment
- **SHA** Secure Hash Algorithm
- **SHS** Secure Hash Standard
- **SKIA** Secure Key Issuing Authority
- **SLC** Secured Logon Coordinator
- **SMIB** Security Management Information Base
- **SMS** Service Management System
- **SNG** Secured Network Gateway
- **SNI** Siemens Nixdorf Informationssysteme
- **SNMP** Simple Network Management Protocol
- **SNP** Secure Network Programming
- **SPKM** Simple Public-key GSS-API Mechanisms
- **SSE** Software and Systems Engineering (SSE) Ltd.
- **SSO** Single Sign-On

- **TA** Trusted Authority
- **TAN** Transaction Authentication Number
- **TCB** Trusted Computing Base
- **TCP** Transport Control Protocol
- **TESS** The Exponential Security System
- **TGS** Ticket Granting Server
- **TGT** Ticket Granting Ticket
- **TLSP** Transport Layer Security Protocol
- **TVP** Time-Variant Parameter

- **UID** User Identification

- **UUID** Universal Unique Identifier
- **URL** Uniform Resource Locator
- **US** User Sponsor
- **U.S.** United States

- **WG** Working Group
- **WWW** World Wide Web