

# O analiză comparativă a soluțiilor de securitate la nivel de aplicație oferite de Sun și Microsoft

Data: 27.03.2002

Autori: Cristian Mihăescu, Sorin Scorțan, SecureNet S.R.L, Craiova ([cm@secure-net.ro](mailto:cm@secure-net.ro), [ss@secure-net.ro](mailto:ss@secure-net.ro))

Compania: SecureNet S.R.L., Craiova, tel.: 051/410555, <http://www.secure-net.ro>

## 1. Securitatea informației electronice

În primele decenii ale informației electronice Internetul era folosit doar de câteva universități pentru trimiterea poștei electronice. În aceste condiții, securitatea informației electronice nu dădea dureri de cap nimănui. În zilele noastre, informația electronică și mai ales schimbul de date în această formă este din ce în ce mai prezentă în viața a milioane de oameni. Statisticile arată ca la începutul anului 2002 existau în întreaga lume peste 800 de milioane de utilizatori Internet.

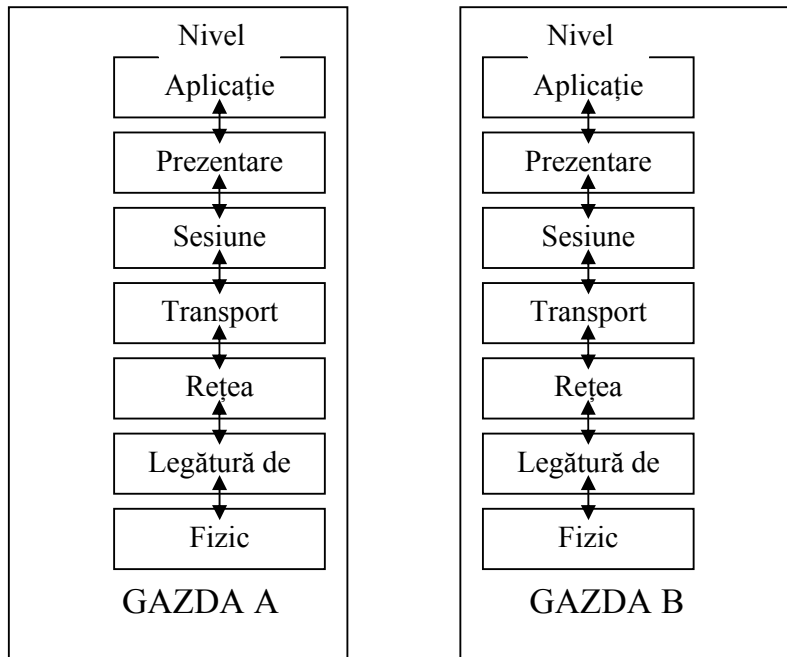
În decursul acestei dezvoltări, activități care trebuie să se bucure de securitate maximă (transmiterea de bani dintr-un cont în altul, semnare de contracte, transmitere de informații confidențiale, licitații) și care erau efectuate în mod tradițional, cu ajutorul telefonului, faxului, prezenței personale la locul / institutia „faptei”, organizarea de transporturi de bani însoțite de bodyguarzi profesioniști și poliție) au început să se regăsească din ce în ce mai mult printre aplicațiile software contemporane, care ne ușurează pur și simplu pe nesimțite viața de toate zilele – atât privat cât și în afaceri.

În noul context, ca și în cel vechi, securitatea informației este un element fundamental. Indiferent dacă o licitație are loc prin trimiterea de scrisori sigilate sau de oferte în formă electronică, participarea la licitație nu au voie să vadă de exemplu cine este și ce prețuri propune concurența! A garanta integritatea unor asemenea procese este o știință în sine, care cuprinde atât know-how în domeniu organizatoric cât mai ales în domeniul tehnic, acesta cuprinzând transferul și menținerea datelor în rețelele de calculatoare cât și accesul autorizat, asigurând nemanipularea, al acestora. Pentru a dezvolta un sistem de securitate al informației electronice trebuie să știm cine pot fi atacatorii, cum pot interveni ei în sistem și ce stricăciuni pot produce acestuia. Deoarece oricine poate fi un posibil atacator iar gama de stricăciuni produse este foarte largă, trebuie să ne îndreptăm atenția asupra felului în care sistemul poate fi agresat și cum putem preveni acest lucru.

Cele patru scopuri de bază în ceea ce privește securitatea unui sistem informatic sunt:

- a) confidențialitatea - păstrarea informației departe de utilizatorii neautorizați;
- b) autentificarea - determinarea identității partenerului înainte de comunicație;
- c) nerepudierea - dovedirea partenerului că el este acela care a trimis datele;
- d) controlul integrității - siguranța faptului că datele nu au fost modificate;

În general ne referim la două calculatoare conectate în rețea (LAN, WAN, Internet) care doresc să schimbe date între ele. În Fig.1 este prezentat modelul de referință OSI (International Standards Organization), model general valabil pentru două sau mai multe calculatoare conectate în rețea (LAN, WAN, Internet).



**Fig. 1. Modelul de referință OSI**

Atacurile se pot efectua la oricare dintre cele 7 nivele, începând cu cel fizic și terminând cu nivelul aplicație. De asemenea, la fiecare din cele 7 nivele se poate implementa o strategie de securitate care să dea un plus de siguranță rețelei.

Chiar la nivel fizic, liniile pot fi închise în tuburi sigilate conținând gaz de argon la presiuni înalte, orice încercare de a atenta la integritatea tubului ducând la pierderi de gaz și deci la alertarea celor ce urmăresc acest lucru.

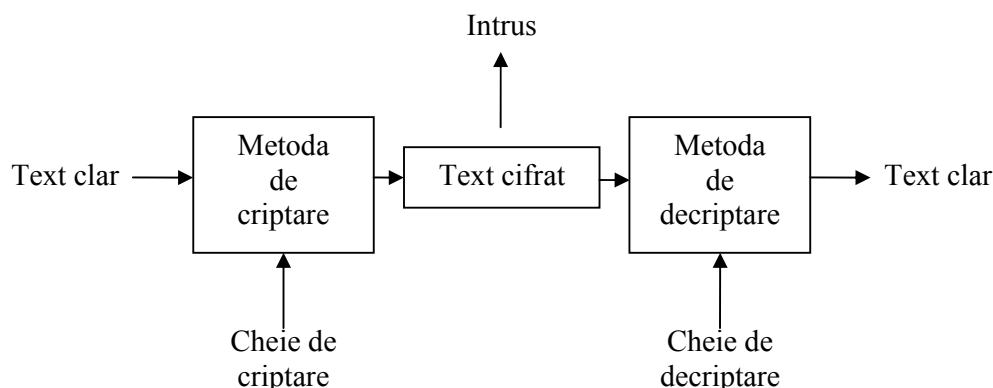
Securitatea se implementează la nivelul rețea prin *firewalls* pentru a păstra anumite pachete în interiorul sau în afara acestuia, conform cu necesitățile impuse de administrarea rețelei. La nivelul transport, conexiunile pot fi criptate de la un capăt la celălalt, adică de la un proces la celălalt.

Cu toate că aceste soluții conduc la realizarea confidențialității și chiar a integrității, nici una din ele nu soluționează problema autentificării sau nerepudierii. Pentru a asigura aceste cerințe, soluțiile trebuie să se găsească la nivelul aplicație. Atacurile la nivelul aplicație sunt cele mai periculoase deoarece pot trece cu destulă ușurință de sistemele de securitate implementate la nivelele inferioare (de exemplu la nivelul rețea - *firewalls*).

Practic, aplicațiile trebuie ele însele să se ocupe de asigurarea celor 4 scopuri de bază ale unui sistem de securitate. Pentru aceasta, aplicațiile ce rulează în rețea trebuie să se găsească într-un cadru de securitate. Cadrul general de securitate funcționează după un principiu foarte simplu. Înainte ca datele să parăsească nivelul aplicație spre nivelele inferioare trebuie să fie criptate. Operațiunea inversă are loc la destinație, imediat ce datele ajung la nivelul aplicație. Dacă ne asigurăm că acest sistem funcționează, din momentul în care datele părăsesc nivelul aplicație al sursei până în momentul în care acestea ajung la nivelul aplicație al destinației acestea vor fi criptate și deci orice atac la nivele inferioare va fi sortit eșecului.

În general, criptarea și decriptarea decurge ca în figura următoare:  
Algoritmii care sunt folosiți în modelul general de criptare sunt publici, însă sunt foarte complecși. Forța acestor algoritmi constă în imposibilitatea generării cheilor de criptare și decriptare. Lungimea acestora (64, 128, 256 biți) determină un spațiu foarte mare al cheilor ceea ce conduce practic la

imposibilitatea determinării lor.



**Fig. 2. Modelul general de criptare**

## **2. Soluții de securitate oferite de Sun in Java SDK1.4**

Sun propune trei extensii, ca parte integrantă din pachetul SDK 1.4 și care dau o nouă perspectivă asupra securității. Cele trei extensii sunt **JCE** (*Java Cryptography Extension*), **JSSE** (*Java Secure Socket Extension*) și **JAAS** (*Java Authentication and Authorization Service*).

Extensia JCE reprezintă cadrul în care se regăsesc implementații:

- algoritmi de criptare
- algoritmi pentru generarea de chei

JCE API acoperă următoarele tipuri de algoritmi de criptare

- algoritmi simetrici ( cu cheie privată ) - DES, RC2, RC4, IDEA
- algoritmi asimetrici ( cu cheie publică ) - RSA
- criptare cu parolă, PBE ( Password Based Encryption )

Mai mult decât atât, pe lângă serviciile criptografice legate de algoritmi și chei, JCE oferă și posibilitatea conversiilor între diferitele tipuri de chei.

Serviciile ( engine classes ) care se regăsesc în JCE sunt :

- fabrici de chei - se pot genera chei pentru tipurile amintite de algoritmi;
- crearea și managementul bazelor de date în care sunt ținute cheile;
- crearea și managementul parametrilor algoritmilor de criptare;
- fabrici de certificate.

Un serviciu criptografic oferit de JCE are următoarele proprietăți :

- este întotdeauna asociat cu un algoritm;
- implementează cele două funcții de bază;
- de dispersie (*digest*) a mesajului;
- de semnătură a mesajului;
- generează materialul criptografic reprezentat de chei și parametri pentru algoritmi;
- generează certificate și baze de date în care să fie ținute cheile (keystore);

Serviciul JCE este reprezentat de următoarele clase de bază :

- 1) Clasa *MessageDigest* - implementează funcția de dispersie (*digest*) a mesajului
- 2) Clasa *Signature* - semnează un mesaj sau verifică sau verifică semnătura dintr-un mesaj
- 3) Clasa *KeyPairGenerator* - generează o pereche de chei, una publică iar cealaltă privată pentru a fi folosite de un anumit algoritm.
- 4) Clasa *CertificateFactory* - generează certificate și CRL ( *Certificate Revocation Lists* )
- 5) Clasa *KeyStore* - generează și face managementul obiectelor "keystore" , adică a bazelor de date în care sunt ținute cheile.
- 6) Clasa *AlgorithmParameters* - face managementul parametrilor pentru un anumit algoritm, inclusiv codificarea și decodificarea acestora.
- 7) Clasa *AlgorithmParameterGenerator* - generează parametrii pentru un anumit algoritm
- 8) Clasa *SecureRandom* - generează numere aleatoare .
- 9) Clasa *CertPathBuilder* - generează înlănțuiri de certificate .
- 10) Clasa *CertPathValidator* - validează înlănțuiri de certificate
- 11) Clasa *CertStore* - folosit pentru obținerea certificatelor sau a CRL-urilor.

Pachetele în care se găsesc aceste clase sunt *java.security*, *java.security.spec*, *java.security.cert*, *java.security.interfaces*.

În ceea ce privește criptografia, Java pune la dispoziția noastră *tool*-uri, și anume *keytool* și *jarsigner* care se pot folosi cu succes la generarea de chei și certificate cu posibilitatea de a semna arhivele Java (*jar files*). Următoarea secvență de comenzi se poate folosi la generarea unui certificat și la semnarea arhivelor *jar*.

- 1) *keytool -genkey -alias signLegal -keystore senderKeystore*  
Generează o bază de date, numită *senderKeystore* în care se găsește cheia publică.
- 2) *keytool -export -keystore senderKeystore signLegal -file sender.cer*  
Generează certificatul *sender.cer* ce conține cheia publică din *senderKeystore* .
- 3) *jarsigner -keystore senderKeystore -signedJar Ssecurity.jar security.jar signLegal*  
Semnează arhiva *security.jar* cu cheia privată din *senderKeystore*, generând astfel *Ssecurity.jar*

În acest moment, certificatul împreună cu arhiva semnată (*Ssecurity.jar*), pot fi trimise partenerului (în exemplu *receiver*), iar acesta poate verifica arhiva cu următoarele comenzi:

- 1) *keytool -import -alias sender sender.cer -keystore receiverKeystore*  
Importă cheia publică din *sender.cer* în baza de date a partenerului (*receiverKeystore*).
- 2) *jarsigner -verify -verbose -keystore receiverKeystore Ssecurity.jar*

Verifică arhiva *Ssecurity.jar* folosind cheia publică din baza de date *receiverKeystore*, unde a fost importată.

Presupunând că problemele de criptografie au fost rezolvate, nu ne rămâne decât să ne asigurăm că putem comunica în siguranță. JSSE implementează protocoalele **SSL V3** (*Secure Socket Layer*) și **TLS 1.0** (*Transport Layer Security*). Această extensie asigură de asemenea suport pentru protocolul HTTPS și algoritmul de criptare RSA.

SSL, ca strat nou între aplicație și nivelul de transport, folosește criptografia cu cheie publică pentru autentificare. În acest context criptografia cu cheie privată și semnăturile digitale asigură integritatea și confidențialitatea datelor.

Pașii care sunt urmați în procesul criptografic al implementării SSL sunt :

- 1) Se criptează codul cu cheia secretă a expeditorului
- 2) Se criptează codul cu cheia publică a destinatarului

3) Se generează certificate care transportă cheia publică folosită în criptografia asimetrică. Un certificat conține :

- fabricantul, adică CA (Certificate Authority)
- perioada de valabilitate
- informații despre entitatea (persoana, firma etc. ) pe care certificatul o reprezintă
- cheia publică a entității
- semnătura digitală

Esența procesului de comunicare prin SSL constă în stabilirea unor parametri criptografici înainte de transmiterea efectivă a datelor. Stabilirea parametrilor poartă numele de "*SSL handshake*". În primul rând, clientul comunică serverului ce "*cipher suites*" are disponibile . Un "*cipher suite*" reprezintă o combinație de parametri criptografici ce definesc algoritmul și cheile folosite pentru autentificare și criptare. Apoi, serverul se poate autentifica (acest pas este opțional), permițându-i clientului să fie sigur că entitatea server este ceea cu care clientul se așteaptă să comunice. Pentru aceasta, serverul prezintă clientului un certificat ce conține cheia sa publică. Verificând acest certificat clientul poate fi sigur de identitatea serverului.

Din acest moment, pot fi schimbate date între client și server.

Pachetele ce implementează acest serviciu sunt *javax.net*, *javax.ssl.net*, *javax.security.cert*.

Serviciul JSSE este reprezentat de următoarele clase de bază :

1. Clasele *SocketFactory* și *SSLSocketFactory*

Creează diverse tipuri de socluri (*sockets*). Astfel, folosind mai multe tipuri de fabrici, aplicația poate crea și folosi mai multe tipuri de socluri.

2. *SSLSocket* și *SSLServerSocket*

Aceste clase extind *javax.net.Socket*. Obiectele de acest tip au acces la contextual în care au fost create, adică la *SSLContext*.

3. Interfața *SSLSession*

Reprezintă contextual negociat între cele două părți care comunică .

4. Clasa *HttpsURLConnection*

HTTPS este un protocol similar cu HTTP. HTTPS, înainte de a începe comunicația, stabilește un canal sigur între server și client prin socluri SSL/TLS. Înainte de obținerea unei conexiuni de tip *HttpsURLConnection*, se pot configura parametrii de comunicație ai HTTP/HTTPS.

5. Clasa *SSLContext*

Un obiect de acest tip conține informații despre toate soclurile create în respectivul context. Configurarea unui obiect de acest tip este realizată prin manageri de chei și manageri de încredere (*key and trust managers*). Acești manageri asigură suportul pentru autentificare și pentru stabilirea de comun acord a cheilor, acestea fiind necesare în definirea cifrelor (*cipher suites*) suportate de context.

6. Clasa *TrustManager*

Această clasă determină dacă autentificarea este reușită sau nu. În caz că autentificarea nu reușește, conexiunea va fi distrusă.

Este momentul să ne oprim atenția asupra extensiei JAAS. Această extensie permite serviciilor care rulează pe un server să se autentifice și asigură controlul asupra utilizatorilor ce folosesc serviciile respective.

Autentificarea are scopul de a determina cine execută la un moment dat codul Java. Modul de implementare pentru un anumit tip de autentificare este determinat la *runtime* și este specificat într-un fișier de configurare (*login configuration file*). Pentru aceasta există:

- obiectul *LoginContext* ce descrie metodele de bază folosite în autentificare.
- obiectul *Configuration* ce descrie tehnologiile de autentificare
- obiectul *LoginModule* care realizează efectiv autentificarea

Extensia JAAS conține module de autentificare ce folosesc JNDI (Java Naming and Directory Interface), Unix Operating Environment, Windows NT, Kerberos, Keystore (baze de date ce conțin chei publice sau private). Administratorul sistemului este responsabil de configurarea unui fișier de configurare ce conține una sau mai multe directive care stabilesc ce se întâmplă când o aplicație încearcă să logheze un utilizator. Aceste directive reprezintă modulele de login care sunt folosite pentru autentificarea unui utilizator. Administratorul sistemului poate instala câte module de login dorește și le poate defini ca opționale sau obligatorii, conform cu politica de securitate adoptată.

Deciziile referitoare la controlul accesului sunt luate în baza a două criterii. Primul criteriu se referă la locul unde se află codul executat și semnăturile care sunt efectuate asupra lui. Al doilea criteriu are în vedere utilizatorul sau după caz serviciul care execută codul (*Subject*).

Autorizarea, din punctul de vedere al JAAS extinde arhitectura de securitate din Java, care înseamnă o politică de securizare (*security policy*) specificând drepturile de acces ale codului care se execută. JAAS adaugă la controlul centrat în jurul codului care se execută (*code-centric access control*), controlul userului care execută (*user-centric access control*). Permișiunile garantate nu se mai referă decât evidența codului ce se execută dar și pe evidența entităților ce îl execută.

Serviciul JAAS este reprezentat de următoarele clase de bază :

- Clasa *Subject*

Reprezintă o grupare de informații referitoare la o singură entitate (persoană, firma, etc.). Aceasta cuprinde informații despre *Principals*, credite publice, și credite private (*public and private credentials*)

- Clasa *Principal*

Această clasă reține identitatea entității. Clasa *Subject* se ocupă de actualizarea identității pentru respective entitate.

Referitor la autentificare, JAAS pune la dispoziție următoarele clase:

- Clasa *LoginContext*

Această clasă pune la dispoziția noastră metode de autentificare a entităților (*Subject*). Ea se pune la curent cu setările din fișierul de configurare, sau cu modulele de login (*LoginModules*) configurate pentru o anumită aplicație.

- Interfața *LoginModule*

Această interfață asigură dezvoltatorilor posibilitatea de a implementa diferite tehnologii de autentificare ce se pot folosi în alte aplicații, în mod *plugin*. Modulele de login pot asigura interfața cu diverse componente hardware, cum ar fi *smart cards*.

Referitor la autorizare, JAAS pune la dispoziție următoarea clasă:

- Clasa *Policy*

Această clasă este abstractă, reprezentând politica de securitate a sistemului. În mod implicit, Java 2 SDK oferă o implementare a acestei clase reprezentată de fișierul *java.policy*. Java 2 SDK v1.4 permite configurarea fișierului *java.policy* astfel încât să suporte înregistrări care au în componența lor obiecte de tip *Principal*.

Așa cum pentru criptare Java 2 SDK pune la dispoziția noastră utilitare precum *keytool* sau *jarsigner*, pentru autorizare și autentificare putem folosi utilitarul *policytool*.

### **3. Soluții de securitate oferite de Microsoft**

Microsoft a elaborat o arhitectură numită Microsoft Internet Security Framework (MISF) ce furnizează un set de tehnologii de securitate multiplatformă pentru aplicații de comunicații online și de comerț electronic.

Printre tehnologiile MISF implementate se numără:

- Authenticode - interfață pentru autentificarea componentelor software pe Internet

- CryptoAPI - interfață pentru criptografie
- Servicii de securizare canale (SSL și PCT)
- Implementarea protocolului SET (Secure Electronic Transactions) pentru tranzacții online cu cărți de credit

Tehnologia MISF este distribuită prin intermediul sistemelor de operare Windows și Windows NT și prin aplicații ca Microsoft Internet Explorer, Microsoft Certificate Server și Microsoft Internet Information Server.

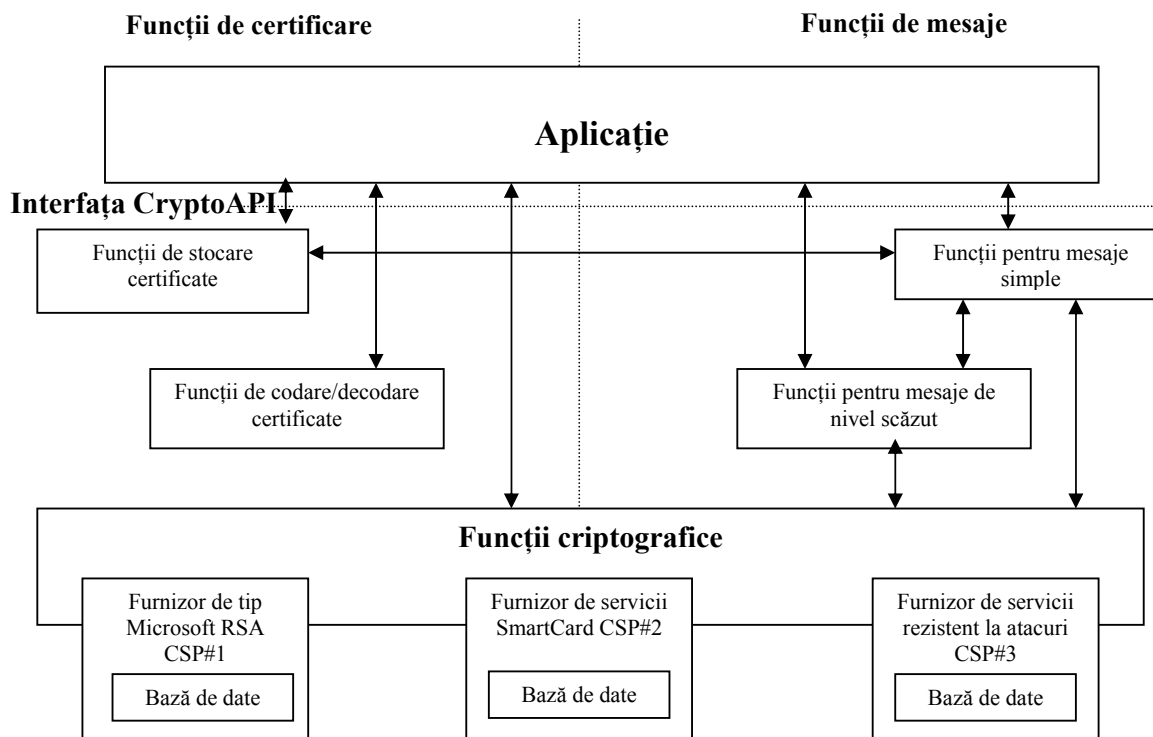
La baza arhitecturii MISF stă tehnologia Microsoft CryptoAPI, un API toolkit care oferă funcționalități de codificare/decodificare folosind protocolul ASN.1, hashing, criptare/decriptare date, autentificare folosind certificate digitale și gestiunea certificatelor în depozite de certificate. Criptarea și decriptarea se poate realiza cu chei de sesiune sau cu perechi de chei publice și private.

Versiunea CryptoAPI 1.0 suportă operații cu chei publice și chei simetrice ca: generare de chei, gestiunea cheilor, schimbul cheilor, criptare/decriptare, hashing, semnături digitale și verificarea semnăturilor.

Versiunea CryptoAPI 2.0 adaugă funcționalități de gestionare a certificatelor, precum și funcții de codare/decodare a mesajelor folosind protocolul ASN.1

Microsoft CryptoAPI utilizează un model în care criptografierea este furnizată de Cryptographic Service Providers (CSP). Acest model permite utilizarea în aplicații a algoritmilor de criptografie. Algoritmii suportați sunt: RC2 și RC4 pe 40 de biți, MD2 și MD5, SHA-1 pe 160 de biți. De asemenea suportă certificate de tip X.509 v3 și codificarea ASN.1, precum și încapsularea de tip PKCS#7 și PKCS#10.

Arhitectura CryptoAPI se prezintă astfel:



**Fig.3 Arhitectura CryptoAPI**

Arhitectura CryptoAPI se compune din cinci zone funcționale:

- funcții criptografice de bază (funcții de conectare la un furnizor de servicii criptografice, funcții de generare a cheilor și funcții de schimbare a cheilor)
- funcții de codificare/decodificare a certificatelor
- funcții de stocare a certificatelor
- funcții de criptare/decriptare a mesajelor și a datelor
- funcții pentru mesaje de nivel scăzut

Microsoft pune la dispoziție dezvoltatorilor și un set de unelte CryptoAPI (instalate în directorul mssdk\bin) care permit producătorilor de software să semneze fișiere de tip .exe, .cab, .cat, .ocx, .dll, .stl. Acest set de unelte CryptoAPI se compune din:

- MakeCert.exe – creează un certificat X.509 de test
- Cert2SPC.exe – creează un certificat de publicare de software
- SignCode.exe – semnează și marchează temporal un fișier
- ChkTrust.exe – verifică validitatea unui fișier
- MakeCTL.exe – creează o listă de certificate de încredere (certificate trust list – CTL)
- CertMgr.exe – gestionează certificatele, listele de certificate de încredere și lista cu certificatele revocate (certificate revocation list – CRL)
- SetReg.exe – setează cheile din registru pentru verificarea certificatelor
- MakeCat.exe – creează un fișier catalog nesemnlat cu codurile hash ale unui set de fișiere și cu atributele asociate fiecărui fișier

Microsoft pune la dispoziția dezvoltatorilor și un client COM ce suportă Automation, numit CAPICOM, care execută funcții criptografice folosind Microsoft ActiveX și obiecte COM. Se poate utiliza în aplicații dezvoltate cu Microsoft Visual Basic, Visual Basic Scripting Edition și C++ și poate semna digital date, verifica semnături digitale, cripta și decripta informații diverse.

### **Concluzii:**

Având în vedere că aplicațiile software implementează din ce în ce mai mult servicii în care securitatea este o componentă majoră (plăți *on-line*, transmitere de documente strict secrete, etc.) putem spune că folosirea unei arhitecturi de securitate (fie cea propusă de Java, fie cea propusă de Microsoft) va fi o necesitate din ce în ce mai stringentă. De asemenea nu putem să nu amintim că pierderile cauzate de hackeri sunt deosebite, acest fapt determinând companiile să cheltuiască sume importante pentru asigurarea securității. Costurile care revin unei întreprinderi pentru garantarea siguranței informației electronice trebuie întotdeauna comparate cu costurile de garantare a siguranței informației în forma tradițională. Dacă este vorba de a urmări cine intra într-o instituție este suficient un post de portar 24 de ore din 24 sau se poate instala un sistem electronic cu cartela care rezolvă problema în mod evident mai bine și văzut pe o durată mai lungă precis și mai ieftin. Dacă am nevoie să asigur informațiile despre trezoreria unei banci, desigur ca am nevoie în viața normală de un seif în interiorul unei clădiri păzită de mai mulți gardieni etc. Pentru a transplanta acest scenariu în forma electronică va fi necesară dezvoltarea și menținerea unei infrastructuri similare (firewall, zona demilitarizată, autentificare pe baza de smart card sau cu mijloace biometrice, asigurarea sistemului de distribuție a acestor smart carduri etc).

În ceea ce privește dezvoltarea de aplicații cu un grad ridicat de securizare, putem spune cu siguranță că arhitecturile prezentate sunt suficient de puternice și flexibile pentru a satisface necesitățile de implementare ale pieței de software actuale.

**Referințe:**

- “Java Security”, Scout Oaks, O’Reilly USA, May 2001
- Java JCE - <http://java.sun.com/products/jce>
- Java JSSE - <http://java.sun.com/products/jsse>
- Java JAAS - <http://java.sun.com/products/jaas>
- Microsoft Security – <http://www.microsoft.com/security>
- MSDN Library Security -  
<http://msdn.microsoft.com/nhp/default.asp?contentid=28001191>

**Despre autori:**

Cristian Mihăescu este preparator la Catedra de Inginerie Software, Facultatea de Automatică, Calculatoare și Electronică, Universitatea din Craiova și inginer software cu jumătate de normă la SecureNet S.R.L. din Craiova. Poate fi contactat pe email la [cm@secure-net.ro](mailto:cm@secure-net.ro). Sorin Scorțan lucrează ca inginer software la SecureNet S.R.L. din Craiova și poate fi contactat pe email la [ss@secure-net.ro](mailto:ss@secure-net.ro).