Capitolul 1

Infrastructură cu chei publice (PKI)

1.1 Prezentare generală

Într-un sistem de criptare cu cheie publică, cheia de criptare nu este secretă; deci este necesară o autentificare a sa, pentru a-i garanta integritatea şi a elimina o serie de atacuri, cum ar fi *man-in-the-middle*.

Cheia publică a unui utilizator trebuie autentificată (semnată) de o autoritate de certificare (CA). Rolul acesteia este de a garanta autenticitatea şi integritatea unei chei publice (unică) pentru fiecare utilizator. După certificare, utilizatorul poate trimite cheia sa publică oricărui alt utilizator, care îi poate verifica autenticitatea.

Un certificat se poate elibera nu numai pentru autentificarea unei chei, ci – în general – pentru orice bloc de identificare.

Definiţia 1.1. *Un bloc de identificare este o structură asociată unui utilizator.*
Ea poate conţine numărul serial al PC-ului, numărul de telefon, numărului reţelei, numărul de certificare, data de expirare a certificatului, diverse autorizaţii.

Certificatul este creat prin legarea cheii publice cu blocul de identificare. Atunci când doi utilizatori stabilesc între ei o comunicare, ei îşi trimit unul altuia certificatele, şi fiecare validează identitatea partenerului.

Ca mod de implementare. componentele cheii publice şi blocul de identificare al unui utilizator pot fi scrise pe un smartcard. Cu ajutorul acestuia, componentele pot fi transportate fizic (electronic) la o autoritate de certificare. Aceasta generează un certificat care va fi încărcat de asemenea pe smartcard.
Ulterior toate aceste componente sunt încărcate pe calculator.

Obiectivele pe care trebuie să le îndeplinească un sistem atunci când apelează la CA sunt în mare:

- Să fie capabil să estimeze scopul şi valoarea certificatelor şi a procesului de certificare;

- Să înţeleagă durata unui certificat şi cum gestionează un PKI acest certificat pe durata valabilităţii lui;
- Să fie capabil să aleagă un serviciu de încredere adecvat pentru eliberarea certificatului.

1.2 Formatul de certificat $X.509$

$X.509$ este cel răspândit format de certificat utilizat pentru o structură PKI. Poate fi întâlnit în protocoale SSL, $IPsec$, $S/MIME$, SET, PGP.
Standardul RFC 4325 listează următoarele zone cuprinse într-un certificat $X.509$ (Santesson & Housley, 2005):

- **Version**: Versiunea certificatului.
- **Certificate serial number** (maxim 20 octeţi): întreg pozitiv (unic) asignat de CA fiecărui certificat.
- **Signature algorithm identifier**: Identificatorul algoritmului folosit de CA pentru semnătura digitală a certificatului. Algoritmul folosit este RSA sau DSA.
- CA **issuer name**: Identifică autoritatea de certificare care a semnat şi a eliberat certificatul.
- **Validity period**: Intervalul de timp în care CA asigură validitatea certificatului. Câmpul respectiv este o sevenţă de două date: data începerii perioadei de validitate şi data expirării validităţii certificatului.
- **Subject name**: Identifică entitatea a cărei cheie publică este autentificată.
- **Subject public-key information**: Prezintă cheia publică împreună cu parametrii publici asociaţi. Dacă este necesar, este depus şi identificatorul sistemului de criptare utilizat (de exemplu: $RSA, DSA, Diffie - Hellman$).
- **Issuer unique** ID: Zonă opţională alocată numelor CA apărute în "*CA Issuer name*" (pentru o eventuală reutilizare).
- **Subject unique ID**: Zonă opţională care permite utilizarea ulterioară a numelor din "*Subject name*".
- **Extensions**: Acest câmp apare numai dacă versiunea certificatelor este 3.
  Extensiile pentru certificatele $X.509$ $v3$ oferă metode de asociere de atribute suplimentare referitoare la utilizatori şi chei publice pentru gestionarea unei ierarhii de certificare, cum ar fi *CA Key Identifier* şi *Subject Key Identifier*.

Informaţia scrisă pe aceste câmpuri este semnată folosind cheia secretă a lui CA. Aceste două elemente (semnătura şi conţinutul zonelor) sunt apoi

1. concatenate;
2. scrise cu ajutorul unui sistem de notaţie sintactică numit *Abstract Syntax One*[1];
3. transformate în date binare folosind sistemul de codificare DER (*Distinguish Encoding Rules*);
4. convertite în caractere ASCII folosind codificarea *base*64.

Observaţia 1.1. *DER este un sistem de codificare a cărui sintaxă este specificată în standardul $X.690$. Codificarea DER utilizată de $X.509$ este un standard internaţional rezultat din codificarea BER (Basic Encoding Rules). De fapt, DER diferă de BER doar prin faptul că ignoră opţiunile expeditorului (care în cazul unor certificate nu sunt necesare).*

Scopul principal este oferirea unei codificări unice, asigurând astfel CA că structura de date pe care o semnează va produce o reprezentare serială unică. Multe lucrări consideră DER ca o formă canonică pentru BER (numită de aceea şi CAR –Canonical Encoding Rules).

De exemplu, în BER o valoare booleană de Adevăr poate fi codificată în 255 moduri diferite (în timp ce Fals este codificat prin valoarea 0), iar în DER există un singur mod de a codifica valoarea logică Adevăr.

Pentru detalii privind $ASN.1$ şi DER se poate folosi [5].

1.3 Variante de certificare

1.3.1 Certificare RSA

Pentru o astfel de certificare, autoritatea CA generează proprii săi parametri RSA: p_{ca}, q_{ca}, $Priv_{ca}$ (exponentul de criptare) şi Pub_{ca} (exponentul de decriptare).

CA face publice valorile Pub_{ca} şi N_{ca} (unde $N_{ca} = p_{ca} \cdot q_{ca}$) pentru toţi utilizatorii din reţea.

Dacă notăm $l(\alpha)$ lungimea secvenţei binare α, va trebui ca pentru orice utilizator X din reţea,

$$l(N_{ca}) > l(ID_X) + l(Pub_X)$$

CA autentifică cheia publică Pub_A şi identificatorul ID_A ale lui *Alice*, generând certificatul public

[1] Abstract Syntax Notation One ($ASN.1$) este un standard şi o notaţie definită prin reguli formale, utilizată în reţelele de calculatoare şi în telecomunicaţii. Descrie structuri de date utilizate pentru reprezentarea, codificarea, transmiterea şi – în final – decodificarea datelor.

$$C_A = (ID_A, Pub_A, (ID_A \| Pub_A)^{Priv_{ca}} \ (mod\ N_{ca})) \quad (1)$$

La primirea certificatului, *Alice* îl verifică calculând

$$ID_A \| Pub_A = (ID_A \| Pub_A)^{Pub_{ca}} \ (mod\ N_{ca}) \quad (2)$$

Când *Alice* doreşte să stabilească o comunicare securizată cu alt utilizator din reţea, ea îi va trimite acestuia certificatul C_A; destinatarul *Bob* – având acces la Pub_{ca} şi N_{ca} – va determina $ID_A \| Pub_A$ calculând (2), după care va compara rezultatul cu primele două valori concatenate din C_A.

Această variantă asigură autenticitatea şi integritatea cheii publice. Dacă vrem să avem şi confidenţialitate, se renunţă la primele două componente ale certificatului. În acest caz însă trebuie să existe o modalitate clară care să separe ID_A de Pub_A din secvenţa binară concatenată.

1.3.2 Certificare Cylink (SEEK)

Protocolul de certificare a fost prezentat în 1987 ([2]).
Fie a un număr aleator şi p un număr prim tare ($p = 2q + 1$ cu q prim).

Autoritatea de certificare CA generează propriile sale chei $Pub_{ca}, Priv_{ca}$, în conformitate cu protocolul Diffie - Hellman:

$$Pub_{ca} = a^{Priv_{ca}} \ (mod\ p)$$

CA autentifică cheia publică şi ID lui *Alice* calculând certificatul după algoritmul:

1. Calculează $M_A = Pub_A^{ID_A} \ (mod\ p)$
2. Generează aleator R_A şi calculează $C_{caA} = a^{R_A} \ (mod\ p)$
3. Calculează V_A din ecuaţia $M_A = [Priv_{ca} \cdot C_{caA} + R_A \cdot V_A] \ (mod\ (p-1))$
4. Trimite lui *Alice* certificatul $C_A = (C_{caA}, V_A)$.

Alice verifică faptul că certificatul a fost eliberat de autoritatea CA folosind algoritmul:

1. Calculează $M_A = Pub_A^{ID_A} \ (mod\ p)$
2. Calculează $S_A = \left(Pub_{ca}^{C_{caA}} \ (mod\ p)\right) \cdot \left(C_{caA}^{V_A} \ (mod\ p)\right) \ (mod\ p)$
3. Dacă $S_A = a^{M_A}$, atunci certificatul C_A este valid.

Când *Alice* doreşte să stabilească o sesiune de comunicare cu *Bob*, îi trimite quadruplul

$$(C_{caA},\ Pub_A,\ V_A,\ M_A)$$

Cum ambele părţi dispun de $a,\ p$ şi Pub_{ca}, *Bob* poate autentifica certificatul lui *Alice* calculând

$$S_B = \left(Pub_{ca}^{C_{caA}}\ mod\ p\right) \cdot \left(C_{caA}^{V_A}\ mod\ p\right)\ (mod\ p)$$

şi verificând congruenţa $a^{M_A} \equiv S_B\ (mod\ p)$.

De remarcat că prin acest protocol, *Bob* nu poate deduce identificatorul lui *Alice*.

1.3.3 Certificare *CyLink* bazată pe algoritmul *ElGamal*

Într-o astfel de certificare, valorile a şi p sunt comune tututor utilizatorilor şi autorităţii de certificare. a este un număr generat aleator, iar p este un număr prim tare.

CA generează perechea de chei $(Pub_{ca}, Priv_{ca})$ care verifică relaţia

$$Pub_{ca} = a^{Priv_{ca}}\ (mod\ p)$$

La solicitarea lui *Alice* de a obţine un certificat pentru mesajul M_A, unitatea de certificare:

1. Generează aleator un număr secret R_{caA}
2. Calculează valoarea publică $V_{caA} = a^{R_{caA}}\ (mod\ p)$;
3. Aplică o funcţie de dispersie criptografică: $H_{caA} = h(M_A, V_{caA})$;
4. Calculează semnătura sa digitală
$$S_{caA} = (R_{caA} + H_{caA} \cdot Priv_{ca})\ (mod\ p)$$
5. Trimite lui *Alice* tripletul $(S_{caA}, H_{caA}, V_{caA})$.

La primire, *Alice* verifică certificatul pe baza relaţiei

$$a^{S_{caA}} \equiv V_{caA} \cdot Pub_A^{H_{caA}}\ (mod\ p)$$

Recapitulând: dacă *Alice* şi *Bob* doresc să stabilească un contact bazat pe un certificat eliberat de CA:

- Informaţiile deţinute de

  Alice : $Priv_A, Pub_A, M_A, S_{caA}, V_{caA}, Pub_{ca}, h$
  Bob : $Priv_B, Pub_B, M_B, S_{caB}, V_{caB}, Pub_{ca}, h$

- Amândoi schimbă simultan între ei următoarele informaţii:

  $Alice \longrightarrow Bob: \quad Pub_A, M_A, S_{caA}, V_{caA}$
  $Bob \longrightarrow Alice: \quad Pub_B, M_B, S_{caB}, V_{caB}$

- Fiecare stabileşte autenticitatea certificatului celuilalt calculând şi verificând:

  $Alice: \quad H_{caB} = h(M_B, V_{caB}), \quad a^{S_{caB}} \equiv V_{caB} \cdot Pub_B^{H_{caB}} \ (mod\ p)$
  $Bob: \quad H_{caA} = h(M_A, V_{caA}), \quad a^{S_{caA}} \equiv V_{caA} \cdot Pub_A^{H_{caA}} \ (mod\ p)$

Dacă congruenţele sunt verificate de ambele părţi, atunci *Alice* şi *Bob* ştiu că certificatele sunt eliberate de autoritatea de certificare CA, iar partenerii sunt autentificaţi.

1.3.4 Variantă a protocolului de certificare *ElGamal*

Diferenţa faţa de varianta anterioară constă în modul de calcul al semnăturii S şi în tipul de verificare al semnăturii.

Fie p un număr prim cu proprietatea că $p-1$ are cel puţin un factor prim mare; fie g un generator al lui Z_p.

Autoritatea de certificare CA generează perechea de chei $(Pub_{ca}, Priv_{ca})$ care verifică relaţia

$$Pub_{ca} = g^{Priv_{ca}} \ (mod\ p)$$

Pentru a instala certificatul generat de CA pentru computerul lui *Alice*, se generează întâi un identificator ID_A (acesta poate consta din IP calculatorului, CNP lui *Alice*, numărul ei de telefon etc).
La solicitarea lui *Alice* de a obţine un certificat, unitatea de certificare:

1. Generează aleator un număr secret R_{caA};
2. Calculează valoarea publică $V_{caA} = g^{R_{caA}} \ (mod\ p)$;
3. Aplică o funcţie de dispersie criptografică: $H_{caA} = h(ID_A, Pub_A, V_{caA})$;
4. Calculează semnătura sa digitală

$$S_{caA} = (R_{caA} \cdot H_{caA} + V_{caA} \cdot Priv_{ca}) \ (mod\ (p-1))$$

5. Trimite lui *Alice* perechea (S_{caA}, V_{caA}).

Deci, dacă *Alice* şi *Bob* doresc să stabilească un contact bazat pe un certificat eliberat de CA:

- Informaţiile deţinute de

$Alice:\ \ Priv_A, Pub_A, ID_A, S_{caA}, V_{caA}, Pub_{ca}, h$
$Bob:\ \ Priv_B, Pub_B, ID_B, S_{caB}, V_{caB}, Pub_{ca}, h$

- Amândoi schimbă simultan între ei următoarele informaţii:

  $Alice \longrightarrow Bob:\ \ ID_A, Pub_A, S_{caA}, V_{caA}$
  $Bob \longrightarrow Alice:\ \ ID_B, Pub_B, S_{caB}, V_{caB}$

- Fiecare stabileşte autenticitatea certificatului celuilalt calculând şi verificând:

  $Alice:\ \ H_{caB} = h(ID_B, Pub_B, V_{caB}),\ \ g^{S_{caB}} \equiv (V_{caB})^{H_{caB}} \cdot (Pub_{ca})^{V_{caB}}\ \ (mod\ p)$
  $Bob:\ \ H_{caA} = h(ID_A, Pub_A, V_{caA}),\ \ g^{S_{caA}} \equiv (V_{caA})^{H_{caA}} \cdot (Pub_{ca})^{H_{caA}}\ \ (mod\ p)$

Dacă congruenţele sunt verificate de ambele părţi, atunci *Alice* şi *Bob* ştiu că certificatele sunt eliberate de autoritatea de certificare CA, iar partenerii sunt autentificaţi.

Observaţia 1.2. *Consistenţa congruenţei de verificare este bazată pe secvenţa de calcule (efectuate modulo p) pentru un utilizator X:*
$g^{S_{caX}} = g^{(R_{caX} \cdot H_X + V_{caX} \cdot Priv_{ca})\ (mod\ (p-1))}\ (mod\ p)$
Deoarece $g^{x\ mod\ (p-1)} \equiv g^x\ (mod\ p)$*, vom avea în continuare:*
$g^{S_{caX}} = g^{R_{caX} \cdot H_X} \cdot g^{V_{caX} \cdot Priv_{ca}} = \left(g^{R_{caX}}\right)^{H_X} \cdot \left(g^{Priv_{ca}}\right)^{V_{caX}} = (V_{caX})^{H_X} \cdot (Pub_{ca})^{V_{caX}}.$

Această variantă de certificare este mai robustă decât versiunea anterioară, deoarece:

- Factorul V_{caX} este inclus atât la bază cât şi ca exponent;
- Cheia publică a autorităţii apare explicit în procedura de verificare.

1.4 Managementul unui PKI

În general, o infrastructură cu chei publice asigură următoarele servicii:

- Urmăreşte perioada de valabilitate a cheii şi certifică acest lucru;
- Pentru o cheie certificată, asigură servicii de *back-up* şi *recovery*;
- Up-datează automat perechile de chei şi certificatele lor;
- Gestionează un istoric al cheilor certificate;
- Este capabilă să efectueze certificări încrucişate.

Conform standardului RFC 4210, sunt 4 entităţi implicate în managementul unui PKI:

1. Utilizatorul PKI, numit şi *end-entity* sau *end-user*: entitatea nominalizată în câmpul "Subject name" a unui certificat;

2. Autoritatea de certificare CA: entitatea nominalizată în câmpul "Issuer name" a unui certificat;

3. O autoritate de înregistrare RA (componentă opţională a unui PKI);

4. Un site RS unde sunt depuse toate certificatele (*repository site*).

1.4.1 Utilizatorii PKI

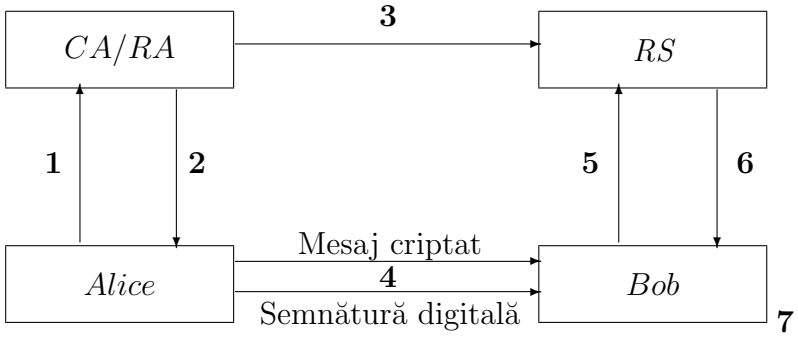
Un utilizator PKI (End-Entity) este un beneficiar al unui certificat PKI. El poate fi o persoană, un computer, o aplicaţie (de exemplu pentru securitatea IP).

Utilizatorul are nevoie de un acces local sigur la anumită informaţie (cel puţin la propriul nume, la cheia sa privată, la numele unui CA în care are încredere, precum şi la cheia publică a acestui CA).

Conform cu RFC 3647, un utilizator PKI are următoarele obligaţii:

- Să asigure o reprezentare corectă a datelor sale în cadrul certificatului;
- Să asigure protecţie cheii sale private;
- Să introducă restricţii de acces la cheia sa privată şi la utilizarea certificatului;
- Să notifice orice compromitere a cheii sale private.

Procesul de înregistrare şi autentificare are loc după schema următoare:



1. *Alice* se înregistrează la autoritatea de certificare CA şi aplică pentru obţinerea unui certificat;

2. CA verifică identitatea lui *Alice* şi eliberează un certificat;

3. CA publică certificatul pe un site dedicat RS (*Repository Site*);

4. *Alice* trimite lui *Bob* mesajul său criptat împreună cu certificatul. Mesajul este semnat cu cheia privată a lui *Alice* (se asigură astfel autenticitatea şi integritatea mesajului, precum şi non-repudierea);

5. După primirea mesajului, *Bob* accesează *RS* şi verifică autenticitatea certificatului lui *Alice*;

6. Site-ul dă lui *Bob* starea actuală a certificatului lui *Alice*;

7. *Bob* verifică integritatea mesajului folosind cheia publică a lui *Alice*.

1.4.2 Autoritatea de certificare

Într-un sistem cu cheie publică, secretul acestei chei nu este obligatoriu; este însă necesară o autentificare a cheii publice, pentru a garanta integritatea şi autenticitatea ei.

Identitatea şi cheia publică a unui utilizator PKI este autentificată (semnată) de o autoritate de certificare.

Termenul CA se referă la entitatea scrisă în zona "*Issuer name*" a unui certificat. Un CA poate emite diverse tipuri de certificate, cum ar fi: certificat pentru un utilizator, pentru alt CA (CA - certificat), sau *cross* - certificat (un proces de autentificare trecând prin diverse domenii de securitate).

Un *domeniu de securitate* este un domeniu logic în care un CA emite şi gestionează certificate.

În general, un utilizator PKI este certificat de un CA, iar un CA este certificat de alt CA. Se construieşte astfel o reţea arborescentă de certificare, care are ca rădăcină un *root - CA*.

Nu este obligatoriu ca o unitate de certificare să fie "*a treia parte*"; frecvent, CA aparţine aceleiaşi organizaţii ca şi utilizatorul pe care îl certifică.

1.4.3 Contactele dintre utilizator şi unitatea de certificare

Contactele dintre un utilizator şi CA sunt legate exclusiv de operaţia de certificare. Ele apar în două situaţii: când este solicitat un certificat (iniţializare) şi la eliberarea unui certificat.

Iniţializarea

Alice solicită un certificat de la CA sau RA. Rezultatul acţiunii este eliberarea unui certificat pentru o cheie publică a lui *Alice*, care este trmis lui *Alice* şi/sau publicat pe un repository site.
Componentele unei faze de iniţializare sunt:

1. **Înregistrare**: CA (sau RA) stabileşte şi verifică identitatea lui *Alice* ca solicitator de certificat.

2. **Generarea cheilor**: Este generată perechea *(cheie publică, cheie privată)*. Generarea poate fi efectuată de *Alice*, CA, RA sau chiar o a *treia parte de încredere*

(TTP). Dacă cheia generată este folosită la o acţiune de non-repudiere, atunci ea este generată obligatoriu de *Alice*.

3. **Crearea certificatului**: CA generează un certificat asociat cheii construite anterior. Numai CA poate construi un astfel de certificat.

4. **Distribuţia certificatului şi cheii publice**: CA trimite lui *Alice* certificatul şi perechea de chei (dacă acestea nu au fost generate de *Alice*).

5. **Diseminarea certificatului**: CA trimite certificatul lui *Alice* şi la un RS.

6. **Păstrarea cheii**: Opţional, CA poate trimite cheia (pentru backup) unui TTP.

Recuperarea şi actualizarea cheii

Dacă *Alice* pierde cheia sa privată, sau mediul în care aceasta este stocată a fost corupt, este nevoie de o recuperare a cheii, deci de un nou contact între utilizator şi unitatea de certificare.

O cheie publică este folosită:

- pentru criptare (*chei de criptare*);

- pentru semnare de mesaje şi verificarea de certificate (*chei de semnătură*).

Procesul de recuperare a cheii este utilizat numai pentru cheile de criptare; aici, organismul abilitat va recalcula cheia privată.
Acest proces nu poate fi similar pentru cheile de semnătură, deoarece va încălca proprietatea de non-repudiere a cheii (*Alice* este singura entitate care controlează cheia privată). Singurul remediu în acest caz este generarea unei noi perechi de chei.

Procesul de actualizare a cheii se referă la o updatare periodică a unei perechi de chei – când aceasta este înlocuită cu o nouă pereche de chei, însoţită de un nou certificat.

Reînnoirea şi actualizarea unui certificat

Un certificat este eliberat pentru o perioadă strict determinată de timp. După expirare, el trebuie reînnoit sau actualizat.
Reînnoire înseamnă eliberarea unui nou certificat, pentru aceeaşi cheie şi aceleaşi date de identificare ale utilizatorului.
Actualizare înseamnă eliberarea unui certificat pentru o nouă pereche de chei şi/sau o modificare de date de identificare ale lui *Alice*.

Cererea de revocare a unui certificat

Este un proces de invalidare a unui certificat înainte de expirarea sa. Această acţiune este iniţiată de o persoană autorizată, care atenţionează CA asupra unei situaţii anormale, care impune revocarea certificatului.

Deoarece prezenţa unui certificat nu menţionează dacă acesta este revocat sau nu, apare necesitatea de a păstra într-o zonă – sigură, dar accesibilă oricui – o listă cu toate certificatele revocate.

Contacte între unitatea de certificare şi RS (repository site)

Sunt două tipuri de contacte: cel legat de *diseminarea certificatelor* şi cel referitor la *lista de revocare*. Odată cu eliberarea sau revocarea unui certificat, CA este obligată să trans,ită această informaţie spre RS, pentru publicarea sa.

La crearea unui certificat, RS îl publică conform unui protocol special $LDAP$ (Light Weight Directory Access Protocol) menţionat în standardul $RFC\ 4510 - 19$.

Cererea de revocare a unui certificat poate apare când cheia privată a lui *Alice* este compromisă sau când *Alice* nu mai face parte din domeniul de securitate al CA (de exemplu când ea părăseşte organizaţia). Revocarea este inclusă de CA în CRL (Certificate Revocation List) şi diseminată cu ajutorul RS.

1.4.4 Contacte între unitatea de certificare şi cea de înregistrare

RA poate avea diferite funcţii; două sunt însă obligatorii:

- În prima fază RA este un tampon între utilizator şi unitatea de certificare.
  Astfel, *Alice* trimite spre RA cererea de înregistrare. RA verifică datele de identificare ale lui *Alice*, după care – dacă acestea sunt corecte – trimite această cerere spre CA. CA răspunde cu rezultatele înregistrării, rezultate pe care RA le retrimite spre *Alice*. Pe baza acestor rezultate, *Alice* trimite ulterior spre CA o cerere de certificare.
- RA publică certificatele eliberate de CA (informaţie primită de la CA).

1.4.5 Contacte între utilizator şi RS

Între cele două entităţi au loc două tipuri de contacte:

- **Găsirea certificatului**: *Alice* contactează RA pentru aflarea certificatului lui *Bob*, care îi este necesar pentru:
  - Găsirea cheii publice a lui *Bob* – pentru a cripta un mesaj adresat acestuia.
  - Verificarea unei semnături digitale primite de la *Bob*.

- **Validarea certificatului**: După ce *Alice* a găsit certificatul lui *Bob*, ea trebuie să îl valideze. Validarea unui certificat include următoarele verificări:
  - Certificatul a fost eliberat de un CA de încredere (se verifică autenticitatea).
  - Certificatul nu a fost modificat (se verifică integritatea).
  - Certificatul nu a expirat.
  - Certificatul nu a fost revocat (se verifică CRL).

1.4.6 Contacte între două autorităţi de certificare

Este posibil ca cei doi utilizaotri care doresc să intre în contact să aibă certificatele eliberate de autorităţi distincte. Astfel, *Alice* are certificatul eliberat de CA_1, iar certificatul lui *Bob* este eliberat de CA_2. Fiecare din ei are încredere numai în autoritatea care i-a eliberat certificatul. În plus, chiar dacă unul din ei doreşte să îl certifice pe celălalt, acest lucru nu ar fi posibil deoarece domeniile de certificare sunt diferite.

Mecanismul folosit pentru a rezolva acest impediment este numit *certificare încrucişată*: CA_1 îl certifică pe CA_2, extinzând încrederea lui *Alice* şi asupra certificatelor emise de CA_2 (în particular, al lui *Bob*).

Procesul poate fi rafinat, în sensul că domeniul lui CA_1 se poate extinde asupra tuturor certificatelor emise de CA_2 sau doar pentru anumite certificate (care aparţin unei singure organizaţii, care sunt într-un anumit grup etc).

1.5 Formatul unei liste de revocare

Standardul RFC 4325 ("Internet $X.509$ Public Key Infrastructure Certificate and Revocation List (CRL) Profile") descrie detaliat formatul şi semantica unei liste de revocare pentru PKI.

În formatul $X.509$, un CA emite periodic o structură semnată numită CRL (Certificate Revocation List). În esenţă, un CRL este o listă cu ştampilă de timp, emisă şi semnată de un CA şi făcută publică printr-un site RS.
Fiecare certificat revocat este identificat prin numărul său serial. Când se verifică un certificat, înafară de semnătura şi perioada sa de valabilitate, se accesează şi CRL-ul curent, verificând dacă aici este menţionat numărul serial al certificatului.

De menţionat că numai CA-ul care emite un certificat poate să îl revoce. Din acest motiv, datele cu care lucrează o unitate de certificare (algoritmul de semnătură, cheile etc) trebuie securizate prin protocoale suplimentare (deoarece compromiterea unui CA conduce automat la revocarea tuturor certificatelor emise de acesta).

Componentele unei liste de revocare sunt:

- **Versiune**: $X.509$ admite în prezent două versiuni. Pentru această componentă este rezervată locaţie numai dacă se lucrează cu versiunea 2.

- **Semnătură**: conţine ID-ul algoritmului folosit de CA pentru a semna CRL-ul.
- **Nume emitent**: Identifică CA-ul care emite şi semnează CRL-ul.
- **Actualizare**: Indică data emiterii. Informaţia se poate codifica în două moduri: $UTCTime$[2] sau *Timp generalizat*[3].
- **Următoarea actualizare**: Indică data emiterii următorului CRL. Acesta poate apare înainte de data indicată, dar în nici un caz nu va apare ulterior datei.
- **Certificatele revocate**: Listează numerele seriale ale certificatelor revocate în intervalul dintre apariţia CRL-ului anterior şi cel actual. Pentru fiecare certificat trebuie menţionată şi data revocării.
  Dacă nu există certificate revocate, această locaţie **trebuie** să lipsească.

În faza următoare, locaţiile care conţin aceste componente ale CRL-ului sunt:

1. concatenate;
2. scrise în format standard bazat pe *Abstract Syntax One* ([4]);
3. convertite în binar folosind sistemul de codificare DER (*Distinguish Encoding Rules*) (pentru detalii a se vedea [6]);
4. transformate în caractere ASCII cu ajutorul codificării *base*64.

1.6 Modele de încredere

Autortăţile de certificare acţionează ca agenţi de încredere pentru validarea identităţii şi a cheilor publice a utilizatorilor. Acest concept este numit "*a treia parte de încredere*" (TTP – thirst-third party): un utilizator are încredere într-un certificat emis de un CA cât timp el are încredere în CA-ul respectiv.
Altfel spus, "*Alice are încredere în Bob*" înseamnă de fapt "*Alice are încredere în CA-ul care semnează certificatul lui Bob*".

Aici apar diverse dificultăţi, cum ar fi:

- *Alice* şi *Bob* vor să intre în legătură, dar certificatele lor sunt emise de AC-uri cu domenii de securitate distincte.
- Pentru a-şi legitima încrederea, un CA trebuie să fie certificat la rândul său de alt CA.

[2] *Coordinated Universal Time*: un compromis între timpul dat de meridianul 0 (numit Timp Universal) şi timpul fizic determinat cu cea mai mare precizie (Timpul Atomic Internaţional - TAI); a se vedea *http://ro.wikipedia.org/wiki/Ora_universală_coordonată*.

[3] Reprezentare sub forma standard $YYYYMMDDHHMMSS$.

Din aceste motive, apar diverse structuri formate din mai multe autorităţi de certificare; aceste structuri se numesc "*modele de încredere*". În prezent sunt utilizate mai multe tipuri de modele de încredere: ierarhice, mesh, Web, şi centrate pe utilizator.

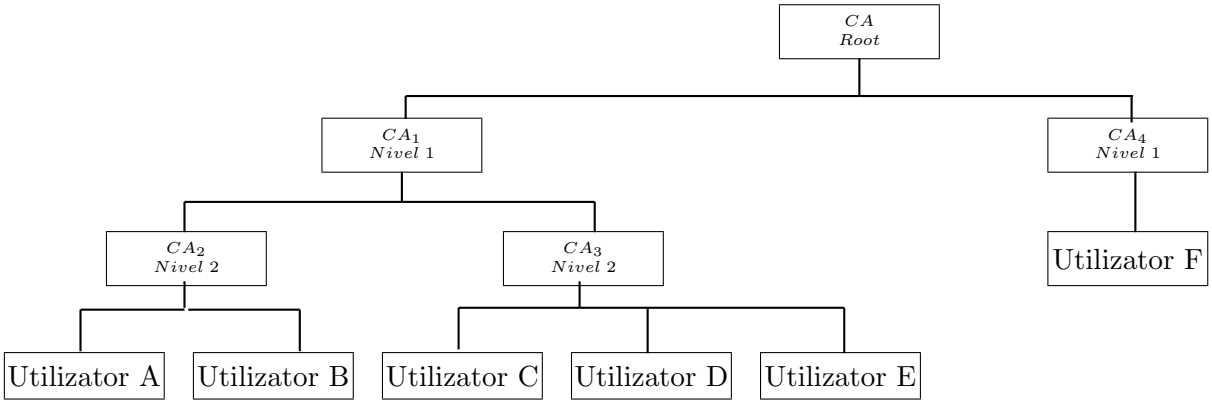
1.6.1 Model de încredere ierarhic

Într-un astfel de model există o autoritate de certificare (*Root CA*) considerată apriori sigură; celelalte CA-uri sunt organizate într-o structură arborescentă ale cărei noduri terminale sunt utilizatorii (entităţi care nu sunt abilitate să emită certificate).

Toate entităţile au încredere în *root CA*. Înafară de el, fiecare entitate dispune de (cel puţin) un certificat eliberat de un CA situat pe drumul (unic) de la el spre *root CA*.

Avantajul acestui model este simplitatea sa şi uşurinţa de implementare. Dezavantajul este acela că nu permite certificări încrucişate între CA-uri diferite.

Un exemplu de model ierarhic este:



Certificatul lui A este emis de CA_2, iar al lui F este emis de CA_4. Dacă cei doi vor să stabilească o legătură, iar A (de exemplu) nu are încredere în CA_4, el va trebui să găsească alt CA în care să aibă încredere. Singurul care poate fi folosit aici este *Root CA*.

Dacă contactul trebuie făcut între utilizatorii A şi D, ei pot folosi ca autoritate de certificare comună pe CA_1.

1.6.2 Model de încredere "mesh"

Este un model de încredere bazat pe structura ierarhică, care facilitează certificările încrucişate.

Dacă sunt mai multe structuri ierarhice, toate autorităţile de certificare aflate pe poziţia de rădăcină se autorizează reciproc prin certificare încrucişată. Această inter-autorizare are loc înainte de începerea emiterii de certificate către alte entităţi.

1.6.3 Model de încredere Web

Pentru un număr mare de utilizatori, modelul de încredere cel mai utilizat este cel de tip listă, numit şi *Web model*.

Fiecare browser Internet acţionează ca un *Root CA* virtual, deoarece utilizatorii au încredere în CA-ul instalat în softul browserului. Browserele sunt distribuite împreună cu un set iniţial de certificate; la acestea utilizatorii pot adăuga certificate noi sau pot elimina certificate. Browserele pot utiliza certificatele pre-instalate pentru a semna, verifica, cripta sau decripta mesajele de e-mail scrise în $S/MIME$ şi de a stabili sesiuni TLS sau SSL. De asemenea, ele sunt abilitate să verifice semnăturile în cazul codurilor semnate.

Pentru un utilizator obişnuit, gestionarea numeroaselor certificate instalate în browser-constituie o problemă extrem de dificilă[4]; mai mult, nu există nici o modalitate practică de a preveni o modificare neautorizată (din partea utilizatorilor sau celor care au acces la staţiile de lucru) a listei de certificate.

În acest tip de model de încredere nu există o modalitate practică de a revoca certificate. Astfel, dacă Firefox sau Microsoft (cei doi lideri actuali pe piaţa browserelor Internet) instalează din greşeală un CA care nu este de încredere, nu există nici o modalitate de a-i revoca certificatul din milioanele browsere aflate în uz.

1.6.4 Model de încredere centrat pe utilizator

Protocolul de poştă electronică PGP foloseşte un model de încredere centrat pe utilizator (User Centric Model). Orice utilizator PGP poate acţiona ca o autoritate de certificare şi să valideze certificatul cheii publice a altui utilizator PGP.

Totuşi, un certificat eliberat de *Alice* – care acţionează ca un CA – poate să nu fie valid pentru alt utilizator, pentru că acesta ştie că *Alice* nu este de încredere ca autoritate de certificare. Fiecare utilizator este direct responsabil în a decide ce certificate acceptă şi ce certificate respinge.

1.7 Algoritmi de criptare acceptaţi în PKI

Standardul RFC 4210 stabileşte algoritmii criptografici, funcţiile de dispersie şi semnăturile digitale care pot fi folosite pentru a semna certificatele şi listele de revocare.

1. **Algoritmii de semnătură**:

   Certificatele şi listele de revocare pot fi semnate teoretic cu orice protocol de semnătură cu cheie publică. Algoritmul folosit este totdeauna însoţit de o funcţie de dispersie criptografică. Aceasta produce o amprentă a datelor care trebuie semnate.

[4]De exemplu, browserele Firefox şi Microsoft Explorer sunt distribuite împreună cu aproximativ 100 chei publice pre-instalate, fiecare cheie fiind însoţită de un certificat.

Amprenta este apoi formatată corespunzător algoritmului de semnătură care va fi folosit.
După generarea semnăturii, valoarea obţinută este codificată cu $ASN.1$ sub forma unui şir de biţi şi inclusă în certificat.

Algoritmi recomandaţi: $DSA/SHA1$.

Alţi algoritmi: $HMAC/SHA1$, $RSA/MD5$, $ECDSA/ECDH$

2. **Algoritmi de criptare**:

   (a) **Algoritmi cu cheie publică**: Utilizaţi pentru criptarea cheilor privatre transportate de mesajele PKI.
   Algoritm recomandat: Diffie - Hellman[5].
   Alţi algoritmi: RSA, $ECDH$.

   (b) **Algoritmi simetrici**:
   Pot fi utilizaţi dacă cheia de criptare a fost transmisă prin canal securizat.
   Algoritm recomandat: $3DES$ (în mod CBC)
   Alţi algoritmi: $RC5$, $Cast$ 128.

[5]Aloritmul de criptare Diffie - Hellman acceptat de $X.509$ este definit în ANSI X.9.42, publicat în 2003.

Bibliografie

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

[2] D.B. Newman, J.K. Omura, R.L. Pickholtz - *Public key management for network security* IEEE Network Magazine, 1(2), 1987, pp. 12-13

[3] Simmons, G. J. - *Contemporary Cryptology*, IEEE Press, 1992.

[4] `http://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One`

[5] `http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf`

[6] `http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf`