Infrastructură cu chei publice (*PKI*)

Prof. Dr. Adrian Atanasiu

March 1, 2018

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 și a elimina o serie de atacuri, cum ar fi *man-in-the-middle*.

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 și 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 certifica o cheie publică (unică) pentru fiecare utilizator.

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 și 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 certifica o cheie publică (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 pentru orice bloc de identificare.

Definiție

Un bloc de identificare este o structură asociată unui utilizator.

Un certificat se poate elibera pentru orice bloc de identificare.

Definiție

Un bloc de identificare este o structură asociată unui utilizator.

El poate conține – printre altele – numărul serial al PC-ului, numărul de telefon, numărului rețelei, numărul de certificare, data de expirare a certificatului, unele autorizații.

Un certificat se poate elibera pentru orice bloc de identificare.

Definiție

Un bloc de identificare este o structură asociată unui utilizator.

El poate conține – printre altele – numărul serial al PC-ului, numărul de telefon, numărului rețelei, numărul de certificare, data de expirare a certificatului, unele autorizații.
Certificatul este creat pentru legarea cheii publice cu blocul de identificare.

Un certificat se poate elibera pentru orice bloc de identificare.

Definiție

Un bloc de identificare este o structură asociată unui utilizator.

El poate conține – printre altele – numărul serial al PC-ului, numărul de telefon, numărului rețelei, numărul de certificare, data de expirare a certificatului, unele autorizații.
Certificatul este creat pentru legarea cheii publice cu blocul de identificare.
Atunci când doi utilizatori stabilesc între ei o comunicare, ei își trimit unul altuia certificatele și fiecare validează identitatea partenerului.

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

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

Obiectivele pe care trebuie să le îndeplinească un sistem atunci când apelează la *CA*:

- Să fie capabil să estimeze scopul și valoarea certificatelor și a procesului de certificare;
- Să înțeleagă durata unui certificat și cum gestionează un *PKI* acest certificat pe durata valabilității lui;

Obiectivele pe care trebuie să le îndeplinească un sistem atunci când apelează la *CA*:

- Să fie capabil să estimeze scopul și valoarea certificatelor și a procesului de certificare;
- Să înțeleagă durata unui certificat și cum gestionează un *PKI* acest certificat pe durata valabilității lui;
- Să fie capabil să aleagă un serviciu de încredere adecvat pentru eliberarea certificatului.

$X.509$

$X.509$ este cel răspândit format de certificat utilizat pentru o structură *PKI*.
Poate fi întâlnit în protocoale *SSL*, *IPsec*, *PGP*, *S/MIME*, *SET* etc.

$X.509$

$X.509$ este cel răspândit format de certificat utilizat pentru o structură *PKI*.
Poate fi întâlnit în protocoale *SSL*, *IPsec*, *PGP*, *S/MIME*, *SET* etc.

$X.509$ este definit prin standardul *RFC* 4325.

Componența unui certificat $X.509$

- Version: Versiunea certificatului.

Componența unui certificat $X.509$

- Version: Versiunea certificatului.
- Certificate serial number (maxim 20 octeți): întreg pozitiv (unic) asignat de *CA* fiecărui certificat.

Componența unui certificat $X.509$

- Version: Versiunea certificatului.
- Certificate serial number (maxim 20 octeți): 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*.

Componența unui certificat $X.509$

- Version: Versiunea certificatului.
- Certificate serial number (maxim 20 octeți): 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 și a eliberat certificatul.

Componența unui certificat $X.509$

- Version: Versiunea certificatului.
- Certificate serial number (maxim 20 octeți): 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 și a eliberat certificatul.
- Validity period: Intervalul de timp în care *CA* asigură validitatea certificatului. Conține data începerii perioadei de validitate și data expirării validității certificatului.

Componența unui certificat $X.509$

- Version: Versiunea certificatului.
- Certificate serial number (maxim 20 octeți): 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 și a eliberat certificatul.
- Validity period: Intervalul de timp în care *CA* asigură validitatea certificatului. Conține data începerii perioadei de validitate și data expirării validității certificatului.
- Subject name: Identifică entitatea a cărei cheie publică este autentificată.

- Subject public-key information: Prezintă cheia publică împreună cu parametrii publici asociați.

- Subject public-key information: Prezintă cheia publică împreună cu parametrii publici asociați.
- Issuer unique *ID*: Zonă opțională alocată istoricului "Issuer name".

- Subject public-key information: Prezintă cheia publică împreună cu parametrii publici asociați.
- Issuer unique *ID*: Zonă opțională alocată istoricului "Issuer name".
- Subject unique *ID*: Zonă opțională alocată istoricului "Subject name".

- Subject public-key information: Prezintă cheia publică împreună cu parametrii publici asociați.
- Issuer unique *ID*: Zonă opțională alocată istoricului "Issuer name".
- Subject unique *ID*: Zonă opțională alocată istoricului "Subject name".
- Extensions: Apare numai dacă versiunea certificatelor este 3. Extensiile pentru certificatele $X.509$ $v3$ oferă metode de asociere de atribute suplimentare referitoare la utilizatori și chei publice pentru gestionarea unei ierarhii de certificare, cum ar fi *CA Key Identifier* și *Subject Key Identifier*.

Informația de pe un certificat $X.509$ este semnată folosind cheia secretă a lui CA.

Informația de pe un certificat $X.509$ este semnată folosind cheia secretă a lui CA.

Perechea **(Semnătură, conținut)** este apoi:

1. concatenată;

Informația de pe un certificat $X.509$ este semnată folosind cheia secretă a lui *CA*.

Perechea **(Semnătură, conținut)** este apoi:

1. concatenată;
2. scrisă cu ajutorul unui sistem de notație sintactică – *Abstract Syntax One*;

Informația de pe un certificat $X.509$ este semnată folosind cheia secretă a lui *CA*.

Perechea **(Semnătură, conținut)** este apoi:

1. concatenată;
2. scrisă cu ajutorul unui sistem de notație sintactică – *Abstract Syntax One*;
3. transformată în date binare cu un sistem de codificare specific: *DER* (Distinguish Encoding Rules);

Informația de pe un certificat $X.509$ este semnată folosind cheia secretă a lui *CA*.

Perechea **(Semnătură, conținut)** este apoi:

1. concatenată;
2. scrisă cu ajutorul unui sistem de notație sintactică – *Abstract Syntax One*;
3. transformată în date binare cu un sistem de codificare specific: *DER* (Distinguish Encoding Rules);
4. convertită în caractere ASCII cu *base*64.

Certificare *RSA*

CA generează parametrii *RSA*: $p_{ca}, q_{ca}, Priv_{ca}$ și Pub_{ca}.
Face publice valorile Pub_{ca} și N_{ca} (unde $N_{ca} = p_{ca} \cdot q_{ca}$).

Certificare *RSA*

CA generează parametrii *RSA*: $p_{ca}, q_{ca}, Priv_{ca}$ și Pub_{ca}.
Face publice valorile Pub_{ca} și N_{ca} (unde $N_{ca} = p_{ca} \cdot q_{ca}$).

Pentru un utilizator A (*Alice*) din rețea trebuie ca:

$$l(N_{ca}) > l(ID_A) + l(Pub_A)$$

Certificare *RSA*

CA generează parametrii *RSA*: $p_{ca}, q_{ca}, Priv_{ca}$ și Pub_{ca}.
Face publice valorile Pub_{ca} și N_{ca} (unde $N_{ca} = p_{ca} \cdot q_{ca}$).

Pentru un utilizator A (*Alice*) din rețea trebuie ca:

$$l(N_{ca}) > l(ID_A) + l(Pub_A)$$

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

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

Certificare *RSA*

CA generează parametrii *RSA*: $p_{ca}, q_{ca}, Priv_{ca}$ și Pub_{ca}.
Face publice valorile Pub_{ca} și N_{ca} (unde $N_{ca} = p_{ca} \cdot q_{ca}$).

Pentru un utilizator A (*Alice*) din rețea trebuie ca:

$$l(N_{ca}) > l(ID_A) + l(Pub_A)$$

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

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

La primirea certificatului, *Alice* îl verifică calculând

$$ID_A \| Pub_A = \left[(ID_A \| Pub_A)^{Priv_{ca}} \ (mod\ N_{ca}) \right]^{Pub_{ca}} \ (mod\ N_{ca})$$

Când *Alice* dorește să stabilească o comunicare securizată cu *Bob*, îi va trimite acestuia certificatul C_A;

Când *Alice* dorește să stabilească o comunicare securizată cu *Bob*, îi va trimite acestuia certificatul C_A;*Bob* – având acces la Pub_{ca} și N_{ca} – va determina $ID_A \| Pub_A$, după care va compara rezultatul cu primele două valori concatenate din C_A.

Când *Alice* dorește să stabilească o comunicare securizată cu *Bob*, îi va trimite acestuia certificatul C_A;*Bob* – având acces la Pub_{ca} și N_{ca} – va determina $ID_A \| Pub_A$, după care va compara rezultatul cu primele două valori concatenate din C_A.

Această variantă asigură doar autenticitatea și integritatea cheii publice.
Dacă vrem să avem și confidențialitate, se renunță 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 secvența binară concatenată.

Certificare Cylink ($SEEK$)

Fie a un număr aleator și p număr prim tare.
CA generează cheile Diffie - Hellman $(Pub_{ca}, Priv_{ca})$ astfel ca

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

Certificare Cylink (*SEEK*)

Fie a un număr aleator și p număr prim tare.
CA generează cheile Diffie - Hellman $(Pub_{ca}, Priv_{ca})$ astfel ca

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

CA calculează certificatul cheii publice lui *Alice* după algoritmul:

1. Calculează $M_A = Pub_A^{ID_A} \ (mod\ p)$

Certificare Cylink (*SEEK*)

Fie a un număr aleator și p număr prim tare.
CA generează cheile Diffie - Hellman $(Pub_{ca}, Priv_{ca})$ astfel ca

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

CA calculează certificatul cheii publice lui *Alice* după algoritmul:

1. Calculează $M_A = Pub_A^{ID_A} \ (mod\ p)$
2. Generează aleator R_A și calculează $C_{caA} = a^{R_A} \ (mod\ p)$

Certificare Cylink (*SEEK*)

Fie a un număr aleator și p număr prim tare.
CA generează cheile Diffie - Hellman $(Pub_{ca}, Priv_{ca})$ astfel ca

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

CA calculează certificatul cheii publice lui *Alice* după algoritmul:

1. Calculează $M_A = Pub_A^{ID_A} \ (mod \ p)$
2. Generează aleator R_A și calculează $C_{caA} = a^{R_A} \ (mod \ p)$
3. Calculează V_A din

$$M_A = [Priv_{ca} \cdot C_{caA} + R_A \cdot V_A] \ (mod \ (p-1))$$

Certificare Cylink (*SEEK*)

Fie a un număr aleator și p număr prim tare.
CA generează cheile Diffie - Hellman $(Pub_{ca}, Priv_{ca})$ astfel ca

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

CA calculează certificatul cheii publice lui *Alice* după algoritmul:

1. Calculează $M_A = Pub_A^{ID_A} \ (mod\ p)$
2. Generează aleator R_A și calculează $C_{caA} = a^{R_A} \ (mod\ p)$
3. Calculează V_A din

$$M_A = [Priv_{ca} \cdot C_{caA} + R_A \cdot V_A] \ (mod\ (p-1))$$

4. Trimite lui *Alice* certificatul $C_A = (C_{caA}, V_A)$.

Alice verifică dacă certificatul a fost eliberat de *CA* astfel:

1. Calculează $M_A = Pub_A^{ID_A} \ (mod \ p)$

Alice verifică dacă certificatul a fost eliberat de *CA* astfel:

1. Calculează $M_A = Pub_A^{ID_A} \ (mod \ p)$
2. Calculează

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

Alice verifică dacă certificatul a fost eliberat de *CA* astfel:

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* dorește să stabilească o sesiune de comunicare cu *Bob*, îi trimite quadruplul

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

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

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

Cum ambele părți dispun de a, p și Pub_{ca}, *Bob* poate autentifica certificatul lui *Alice* calculând

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

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

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

Cum ambele părți dispun de a, p și Pub_{ca}, *Bob* poate autentifica certificatul lui *Alice* calculând

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

și verificând egalitatea $S_B = a^{M_A}\ (mod\ p)$.

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

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

Cum ambele părți dispun de a, p și Pub_{ca}, *Bob* poate autentifica certificatul lui *Alice* calculând

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

și verificând egalitatea $S_B = a^{M_A}\ (mod\ p)$.

Prin acest protocol, *Bob* nu poate deduce identificatorul lui *Alice*.

CyLink bazat pe *ElGamal*

Valorile a și p sunt comune tututor utilizatorilor și autorității de certificare.
a este un număr generat aleator, iar p este un număr prim tare.

CyLink bazat pe *ElGamal*

Valorile a și p sunt comune tututor utilizatorilor și autorității de certificare.
a este un număr generat aleator, iar p este un număr prim tare.
CA generează $(Pub_{ca}, Priv_{ca})$ cu

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

Dacă *Alice* vrea un certificat pentru mesajul M_A, CA:

Dacă *Alice* vrea un certificat pentru mesajul M_A, CA:

1. Generează aleator un număr secret R_{caA};

Dacă *Alice* vrea un certificat pentru mesajul M_A, CA:

1. Generează aleator un număr secret R_{caA};
2. Calculează valoarea publică $V_{caA} = a^{R_{caA}} \ (mod \ p)$;

Dacă *Alice* vrea un certificat pentru mesajul M_A, CA:

1. Generează aleator un număr secret R_{caA};
2. Calculează valoarea publică $V_{caA} = a^{R_{caA}} \ (mod \ p)$;
3. Aplică o funcție de dispersie criptografică

$$H_{caA} = h(M_A \| V_{caA})$$

Dacă *Alice* vrea un certificat pentru mesajul M_A, CA:

1. Generează aleator un număr secret R_{caA};
2. Calculează valoarea publică $V_{caA} = a^{R_{caA}} \ (mod \ p)$;
3. Aplică o funcție de dispersie criptografică

$$H_{caA} = h(M_A \| V_{caA})$$

4. Calculează semnătura sa digitală

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

Dacă *Alice* vrea un certificat pentru mesajul M_A, CA:

1. Generează aleator un număr secret R_{caA};
2. Calculează valoarea publică $V_{caA} = a^{R_{caA}} \ (mod \ p)$;
3. Aplică o funcție 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:

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

Servicii *PKI*

O infrastructură cu chei publice asigură următoarele servicii:

Servicii *PKI*

O infrastructură cu chei publice asigură următoarele servicii:

- Urmărește perioada de valabilitate a cheii și certifică acest lucru;

Servicii *PKI*

O infrastructură cu chei publice asigură următoarele servicii:

- Urmărește perioada de valabilitate a cheii și certifică acest lucru;
- Pentru o cheie certificată, asigură servicii de *back-up* și *recovery*;

Servicii *PKI*

O infrastructură cu chei publice asigură următoarele servicii:

- Urmărește perioada de valabilitate a cheii și certifică acest lucru;
- Pentru o cheie certificată, asigură servicii de *back-up* și *recovery*;
- Up-datează automat perechile de chei și certificatele lor;

Servicii *PKI*

O infrastructură cu chei publice asigură următoarele servicii:

- Urmărește perioada de valabilitate a cheii și certifică acest lucru;
- Pentru o cheie certificată, asigură servicii de *back-up* și *recovery*;
- Up-datează automat perechile de chei și certificatele lor;
- Gestionează un istoric al cheilor certificate;

Servicii *PKI*

O infrastructură cu chei publice asigură următoarele servicii:

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

Sunt 4 entități implicate în managementul unui *PKI*:

1. Utilizatorul *PKI*, numit și *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 opțională a unui *PKI*);
4. Un site unde sunt depuse toate certificatele (*repository site*).

Obligații

Un utilizator *PKI* are următoarele obligații:

Obligații

Un utilizator *PKI* are următoarele obligații:

- Să asigure o reprezentare corectă a datelor sale în cadrul certificatului;

Obligații

Un utilizator *PKI* are următoarele obligații:

- Să asigure o reprezentare corectă a datelor sale în cadrul certificatului;
- Să asigure protecție cheii sale private;

Obligații

Un utilizator *PKI* are următoarele obligații:

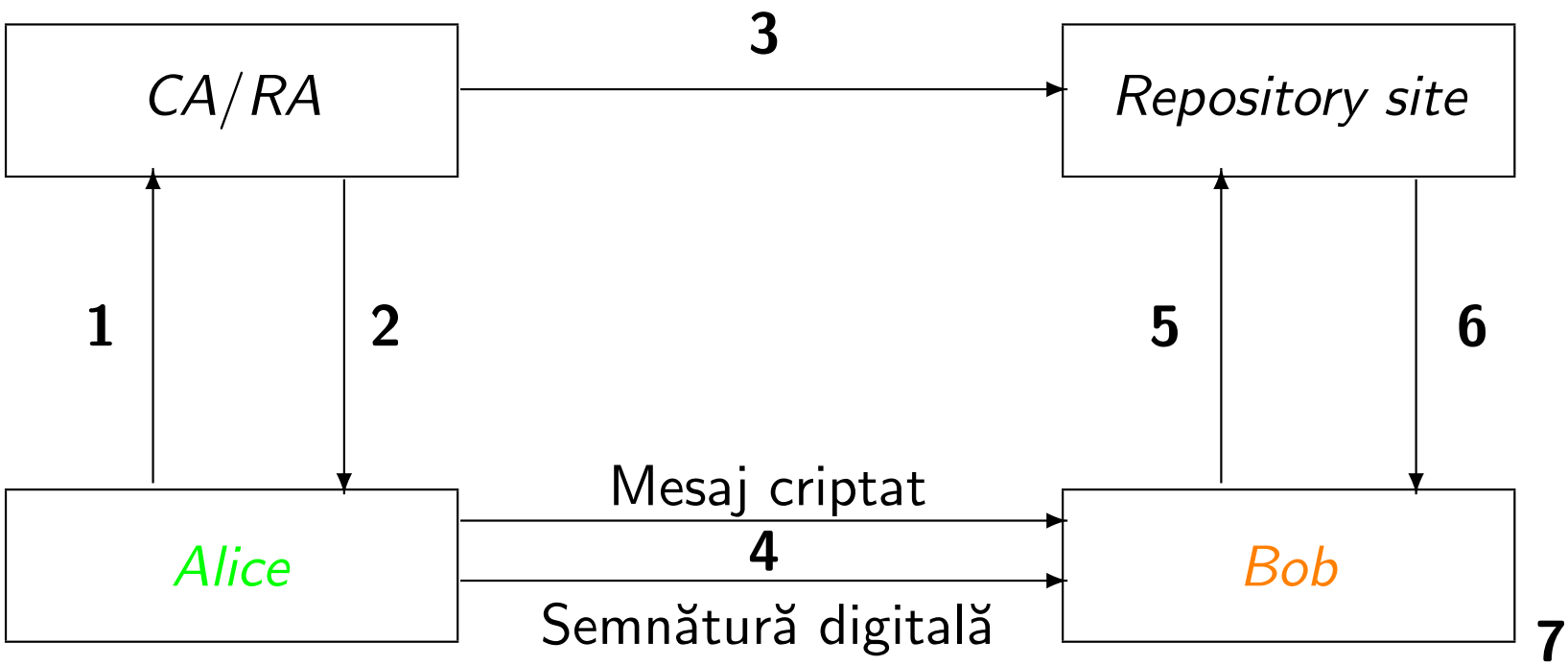
- Să asigure o reprezentare corectă a datelor sale în cadrul certificatului;
- Să asigure protecție cheii sale private;
- Să introducă restricții de acces la cheia sa privată și la utilizarea certificatului;

Obligații

Un utilizator *PKI* are următoarele obligații:

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

Înregistrare și Autentificare



Autoritatea de certificare

Identitatea și cheia publică a unui utilizator *PKI* este autentficată (semnată) de o autoritate de certificare.
Termenul *CA* se referă la entitatea scrisă în zona "*Issuer name*" a unui certificat.

Autoritatea de certificare

Identitatea și cheia publică a unui utilizator *PKI* este autentficată (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:

- certificat pentru un utilizator,

Autoritatea de certificare

Identitatea și cheia publică a unui utilizator *PKI* este autentficată (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:

- certificat pentru un utilizator,
- certificat pentru alt *CA* (*CA* - certificat),

Autoritatea de certificare

Identitatea și cheia publică a unui utilizator *PKI* este autentficată (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:

- certificat pentru un utilizator,
- certificat pentru alt *CA* (*CA* - certificat),
- *cross* - certificat (un proces de autentificare trecând prin diverse domenii de securitate).

Autoritatea de certificare

Identitatea și cheia publică a unui utilizator *PKI* este autentficată (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:

- certificat pentru un utilizator,
- certificat pentru alt *CA* (*CA* - certificat),
- *cross* - certificat (un proces de autentificare trecând prin diverse domenii de securitate).

Definiție

Un "domeniu de securitate" este un domeniu logic în care un CA emite și gestionează certificate.

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

În general, un utilizator *PKI* este certificat de un *CA*, iar un *CA* este certificat de alt *CA*.
Se construiește astfel o rețea 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* aparține aceleiași organizații ca și utilizatorul pe care îl certifică.

Contacte utilizator – *CA*

Contactele dintre un utilizator și *CA* sunt legate exclusiv de operația de certificare.

Contacte utilizator – *CA*

Contactele dintre un utilizator și *CA* sunt legate exclusiv de operația de certificare.
Apar în două situații:

- Când este solicitat un certificat (inițializare),

Contacte utilizator – *CA*

Contactele dintre un utilizator și *CA* sunt legate exclusiv de operația de certificare.
Apar în două situații:

- Când este solicitat un certificat (inițializare),
- La eliberarea unui certificat.

Inițializare

Alice solicită un certificat de la *CA* sau *RA*.

Inițializare

Alice solicită un certificat de la *CA* sau *RA*.

- **Înregistrare**: *CA* (sau *RA*) stabilește și verifică identitatea lui *Alice* ca solicitator de certificat.

Inițializare

Alice solicită un certificat de la *CA* sau *RA*.

- **Înregistrare**: *CA* (sau *RA*) stabilește și verifică identitatea lui *Alice* ca solicitator de certificat.
- **Generarea cheilor**: Este generată perechea *(cheie publică, cheie privată)*. Generarea poate fi efectuată de *Alice*, *CA*, *RA* sau un *TTP* (Random-Key-Generator).

Inițializare

Alice solicită un certificat de la *CA* sau *RA*.

- **Înregistrare**: *CA* (sau *RA*) stabilește și verifică identitatea lui *Alice* ca solicitator de certificat.
- **Generarea cheilor**: Este generată perechea *(cheie publică, cheie privată)*. Generarea poate fi efectuată de *Alice*, *CA*, *RA* sau un *TTP* (Random-Key-Generator).
  Dacă este folosită la o acțiune de non-repudiere, cheia este generată obligatoriu de *Alice*.
- **Crearea certificatului**: *CA* generează un certificat asociat cheii construite anterior.

Continuare

- **Distribuția certificatului și cheii publice**: *CA* trimite lui *Alice* certificatul și perechea de chei.

Continuare

- **Distribuția certificatului și cheii publice**: *CA* trimite lui *Alice* certificatul și perechea de chei.
- **Diseminarea certificatului**: *CA* trimite certificatul lui *Alice* și la un repository site (*RS*).

Continuare

- **Distribuția certificatului și cheii publice**: *CA* trimite lui *Alice* certificatul și perechea de chei.
- **Diseminarea certificatului**: *CA* trimite certificatul lui *Alice* și la un repository site (*RS*).
- **Păstrarea cheii**: Opțional, *CA* poate trimite cheia (pentru backup) unui *TTP*.

Recuperare

Când *Alice* pierde cheia sa privată, sau mediul în care aceasta este stocată a fost corupt, este nevoie de o recuperare a cheii.

Recuperare

Când *Alice* pierde cheia sa privată, sau mediul în care aceasta este stocată a fost corupt, este nevoie de o recuperare a cheii.
O cheie publică este folosită:

- pentru criptare (*chei de criptare*);

Recuperare

Când *Alice* pierde cheia sa privată, sau mediul în care aceasta este stocată a fost corupt, este nevoie de o recuperare a cheii.
O cheie publică este folosită:

- pentru criptare (*chei de criptare*);
- pentru semnare de mesaje și verificarea de certificate (*chei de semnătură*).

Recuperare

Când *Alice* pierde cheia sa privată, sau mediul în care aceasta este stocată a fost corupt, este nevoie de o recuperare a cheii.
O cheie publică este folosită:

- pentru criptare (*chei de criptare*);
- pentru semnare de mesaje și 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 aplicat la cheile de semnătură, deoarece va încălca proprietatea de non-repudiere a cheii (*Alice* este singura entitate care controlează cheia privată).

Recuperare

Când *Alice* pierde cheia sa privată, sau mediul în care aceasta este stocată a fost corupt, este nevoie de o recuperare a cheii.
O cheie publică este folosită:

- pentru criptare (*chei de criptare*);
- pentru semnare de mesaje și 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 aplicat la cheile de semnătură, deoarece va încălca proprietatea de non-repudiere a cheii (*Alice* este singura entitate care controlează cheia privată).
În acest caz se generează o pereche nouă de chei.

Actualizare

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, însoțită de un nou certificat.

Reînnoire și actualizare

Un certificat este eliberat pentru o perioadă fixată de timp.
După expirare, el trebuie reînnoit sau actualizat.

Reînnoire și actualizare

Un certificat este eliberat pentru o perioadă fixată de timp.
După expirare, el trebuie reînnoit sau actualizat.

- **Reînnoire**: eliberarea unui nou certificat pentru aceeași cheie și aceleași date de identificare ale utilizatorului.
- **Actualizare**: eliberarea unui certificat pentru o nouă pereche de chei și/sau o modificare de date de identificare ale End User-ului.

Revocare

Este un proces de invalidare a unui certificat înainte de expirarea sa.

Revocare

Este un proces de invalidare a unui certificat înainte de expirarea sa.
Este inițiat de o persoană autorizată, care atenționează CA asupra unei situații anormale, care impune revocarea certificatului.

Revocare

Este un proces de invalidare a unui certificat înainte de expirarea sa.
Este inițiat de o persoană autorizată, care atenționează *CA* asupra unei situații anormale, care impune revocarea certificatului.

Deoarece prezența unui certificat nu menționează dacă acesta este revocat sau nu, apare necesitatea de a păstra într-o zonă (sigură, dar accesibilă oricărui solicitant valid) o listă cu toate certificatele revocate.

Contacte *CA* – *RS*

Sunt două tipuri de contacte:

- Legat de *diseminarea certificatelor*,

Contacte *CA* – *RS*

Sunt două tipuri de contacte:

- Legat de *diseminarea certificatelor*,
- Referitor la *lista de revocare*.

Contacte *CA* – *RS*

Sunt două tipuri de contacte:

- Legat de *diseminarea certificatelor*,
- Referitor la *lista de revocare*.

Odată cu eliberarea/revocarea unui certificat, *CA* diseminează această informație spre repository site (*RS*), pentru publicarea sa.

Contacte *CA* – *RS*

Sunt două tipuri de contacte:

- Legat de *diseminarea certificatelor*,
- Referitor la *lista de revocare*.

Odată cu eliberarea/revocarea unui certificat, *CA* diseminează această informație spre repository site (*RS*), pentru publicarea sa.
La crearea unui certificat, *RS* îl publică conform unui protocol *LDAP* (Light Weight Directory Access Protocol).
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*.

Contacte *CA* – *RS*

Sunt două tipuri de contacte:

- Legat de *diseminarea certificatelor*,
- Referitor la *lista de revocare*.

Odată cu eliberarea/revocarea unui certificat, *CA* diseminează această informație spre repository site (*RS*), pentru publicarea sa.
La crearea unui certificat, *RS* îl publică conform unui protocol *LDAP* (Light Weight Directory Access Protocol).
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*.
Revocarea este inclusă de *CA* în *CRL* (Certificate Revocation List) și diseminată cu ajutorul *RS*.

Contacte *CA* - *RA*

RA poate avea diferite funcții; două sunt însă obligatorii:

Contacte *CA* - *RA*

RA poate avea diferite funcții; două sunt însă obligatorii:

- În prima fază *RA* este un tampon între utilizator și unitatea de certificare.

Contacte *CA* - *RA*

RA poate avea diferite funcții; două sunt însă obligatorii:

- În prima fază *RA* este un tampon între utilizator și unitatea de certificare.
  Alice trimite spre *RA* cererea de înregistrare.

Contacte *CA* - *RA*

RA poate avea diferite funcții; două sunt însă obligatorii:

- În prima fază *RA* este un tampon între utilizator și unitatea de certificare.
  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*.

Contacte *CA* - *RA*

RA poate avea diferite funcții; două sunt însă obligatorii:

- În prima fază *RA* este un tampon între utilizator și unitatea de certificare.
  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.

Contacte *CA* - *RA*

RA poate avea diferite funcții; două sunt însă obligatorii:

- În prima fază *RA* este un tampon între utilizator și unitatea de certificare.
  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* (informație primită de la *CA*).

Contacte Utilizator – *RS*

Două tipuri de contacte:

Contacte Utilizator – *RS*

Două tipuri de contacte:

- **Găsirea certificatului**: *Alice* contactează *RA* pentru aflarea certificatului lui *Bob*, necesar:

Contacte Utilizator – *RS*

Două tipuri de contacte:

- **Găsirea certificatului**: *Alice* contactează *RA* pentru aflarea certificatului lui *Bob*, necesar:
  - Găsirii cheii publice a lui *Bob* – pentru a cripta un mesaj adresat acestuia.

Contacte Utilizator – *RS*

Două tipuri de contacte:

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

Contacte Utilizator – *RS*

Două tipuri de contacte:

- **Găsirea certificatului**: *Alice* contactează *RA* pentru aflarea certificatului lui *Bob*, necesar:
  - Găsirii cheii publice a lui *Bob* – pentru a cripta un mesaj adresat acestuia.
  - Verificării unei semnături digitale primite de la *Bob*.
- **Validarea certificatului**: După ce *Alice* a găsit certificatul lui *Bob*, ea îl validează.

Contacte Utilizator – *RS*

Două tipuri de contacte:

- **Găsirea certificatului**: *Alice* contactează *RA* pentru aflarea certificatului lui *Bob*, necesar:
  - Găsirii cheii publice a lui *Bob* – pentru a cripta un mesaj adresat acestuia.
  - Verificării unei semnături digitale primite de la *Bob*.
- **Validarea certificatului**: După ce *Alice* a găsit certificatul lui *Bob*, ea îl validează.Validarea include următoarele verificări:
  - Certificatul a fost eliberat de un *CA* de încredere (se verifică autenticitatea).

Contacte Utilizator – *RS*

Două tipuri de contacte:

- **Găsirea certificatului**: *Alice* contactează *RA* pentru aflarea certificatului lui *Bob*, necesar:
  - Găsirii cheii publice a lui *Bob* – pentru a cripta un mesaj adresat acestuia.
  - Verificării unei semnături digitale primite de la *Bob*.
- **Validarea certificatului**: După ce *Alice* a găsit certificatul lui *Bob*, ea îl validează.Validarea include următoarele verificări:
  - Certificatul a fost eliberat de un *CA* de încredere (se verifică autenticitatea).
  - Certificatul nu a fost modificat (se verifică integritatea).

Contacte Utilizator – *RS*

Două tipuri de contacte:

- **Găsirea certificatului**: *Alice* contactează *RA* pentru aflarea certificatului lui *Bob*, necesar:
  - Găsirii cheii publice a lui *Bob* – pentru a cripta un mesaj adresat acestuia.
  - Verificării unei semnături digitale primite de la *Bob*.
- **Validarea certificatului**: După ce *Alice* a găsit certificatul lui *Bob*, ea îl validează.Validarea 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.

Contacte Utilizator – *RS*

Două tipuri de contacte:

- **Găsirea certificatului**: *Alice* contactează *RA* pentru aflarea certificatului lui *Bob*, necesar:
  - Găsirii cheii publice a lui *Bob* – pentru a cripta un mesaj adresat acestuia.
  - Verificării unei semnături digitale primite de la *Bob*.
- **Validarea certificatului**: După ce *Alice* a găsit certificatul lui *Bob*, ea îl validează.Validarea 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*).

Contacte *CA* – *CA*

Dacă *Alice* are certificatul eliberat de CA_1, iar certificatul lui *Bob* este eliberat de CA_2 (cu $CA_1 \neq CA_2$). Fiecare are încredere numai în autoritatea care le-a eliberat certificatul.
În plus, chiar dacă unul din ei dorește să îl certifice pe celălalt, acest lucru nu ar fi posibil deoarece domeniile de certificare sunt diferite.

Contacte CA – CA

Dacă *Alice* are certificatul eliberat de CA_1, iar certificatul lui *Bob* este eliberat de CA_2 (cu $CA_1 \neq CA_2$). Fiecare are încredere numai în autoritatea care le-a eliberat certificatul.
În plus, chiar dacă unul din ei dorește să îl certifice pe celălalt, acest lucru nu ar fi posibil deoarece domeniile de certificare sunt diferite.

Certificare încrucișată: CA_1 certifică pe CA_2, extinzând încrederea lui *Alice* și asupra certificatelor emise de CA_2 (în particular, al lui *Bob*).

Contacte CA – CA

Dacă *Alice* are certificatul eliberat de CA_1, iar certificatul lui *Bob* este eliberat de CA_2 (cu $CA_1 \neq CA_2$). Fiecare are încredere numai în autoritatea care le-a eliberat certificatul.
În plus, chiar dacă unul din ei dorește să îl certifice pe celălalt, acest lucru nu ar fi posibil deoarece domeniile de certificare sunt diferite.

Certificare încrucișată: CA_1 certifică pe CA_2, extinzând încrederea lui *Alice* și asupra certificatelor emise de CA_2 (în particular, al lui *Bob*).
Procesul poate fi rafinat: domeniul lui CA_1 se poate extinde asupra tuturor certificatelor emise de CA_2 sau doar pentru anumite certificate.

Lista de revocare

În formatul $X.509$, *CA* emite periodic o structură semnată numită *CRL* (Certificate Revocation List).
CRL este o listă cu ștampilă de timp, emisă și semnată de un *CA* și făcută publică printr-un *RS*.

Lista de revocare

În formatul $X.509$, CA emite periodic o structură semnată numită CRL (Certificate Revocation List).
CRL este o listă cu ștampilă de timp, emisă și semnată de un CA și făcută publică printr-un RS.
Fiecare certificat revocat este identificat prin numărul său serial.

Lista de revocare

În formatul $X.509$, CA emite periodic o structură semnată numită CRL (Certificate Revocation List).
CRL este o listă cu ștampilă de timp, emisă și semnată de un CA și făcută publică printr-un RS.
Fiecare certificat revocat este identificat prin numărul său serial.

Când se verifică un certificat, înafară de semnătura și perioada sa de valabilitate, se accesează și CRL-ul curent, verificând dacă aici este menționat și numărul serial al certificatului.

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

- **Versiune**.

Componentele unei liste de revocare

- **Versiune**.
- **Semnătură**: conține *ID*-ul algoritmului folosit de *CA* pentru a semna *CRL*-ul.

Componentele unei liste de revocare

- **Versiune**.
- **Semnătură**: conține *ID*-ul algoritmului folosit de *CA* pentru a semna *CRL*-ul.
- **Nume emitent**: Identifică *CA*-ul care emite și semnează.

Componentele unei liste de revocare

- **Versiune**.
- **Semnătură**: conține *ID*-ul algoritmului folosit de *CA* pentru a semna *CRL*-ul.
- **Nume emitent**: Identifică *CA*-ul care emite și semnează.
- **Actualizare**: Indică data emiterii. Informația se poate codifica în două moduri: *UTCTime* sau *Timp generalizat*.

Componentele unei liste de revocare

- **Versiune**.
- **Semnătură**: conține *ID*-ul algoritmului folosit de *CA* pentru a semna *CRL*-ul.
- **Nume emitent**: Identifică *CA*-ul care emite și semnează.
- **Actualizare**: Indică data emiterii. Informația se poate codifica în două moduri: *UTCTime* sau *Timp generalizat*.
- **Următoarea actualizare**: Indică data emiterii următorului *CRL*. Poate apare înainte de data indicată, dar niciodată ulterior datei.

Componentele unei liste de revocare

- **Versiune**.
- **Semnătură**: conține *ID*-ul algoritmului folosit de *CA* pentru a semna *CRL*-ul.
- **Nume emitent**: Identifică *CA*-ul care emite și semnează.
- **Actualizare**: Indică data emiterii. Informația se poate codifica în două moduri: *UTCTime* sau *Timp generalizat*.
- **Următoarea actualizare**: Indică data emiterii următorului *CRL*. Poate apare înainte de data indicată, dar niciodată ulterior datei.
- **Certificatele revocate**: Listează numerele seriale ale certificatelor revocate în intervalul dintre apariția *CRL*-ului anterior și cel actual. Pentru fiecare certificat trebuie menționată și data revocării.

Locațiile care conțin aceste componente ale *CRL*-ului sunt apoi:

Locațiile care conțin aceste componente ale *CRL*-ului sunt apoi:

1. concatenate;

Locațiile care conțin aceste componente ale *CRL*-ului sunt apoi:

1. concatenate;
2. scrise în format standard bazat pe *Abstract Syntax One*;

Locațiile care conțin aceste componente ale *CRL*-ului sunt apoi:

1. concatenate;
2. scrise în format standard bazat pe *Abstract Syntax One*;
3. convertite în binar folosind sistemul de codificare *DER* (*Distinguish Encoding Rules*);

Locațiile care conțin aceste componente ale *CRL*-ului sunt apoi:

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

TTP

Autoritățile de certificare acționează ca agenți de încredere pentru validarea identității și a cheilor publice a utilizatorilor.
Conceptul se numește "*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.

TTP

Autoritățile de certificare acționează ca agenți de încredere pentru validarea identității și a cheilor publice a utilizatorilor.
Conceptul se numește "*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*".

Dificultăți

- *Alice* și *Bob* vor să intre în legătură, dar certificatele lor sunt emise de *AC*-uri cu domenii de securitate distincte.

Dificultăți

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

Dificultăți

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

Structuri formate din mai multe autorități de certificare; se numesc "*modele de încredere*".

Dificultăți

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

Structuri formate din mai multe autorități de certificare; se numesc "*modele de încredere*".

Sunt utilizate mai multe tipuri de modele de încredere: ierarhice, mesh, Web, și centrate pe utilizator.

Model ierarhic

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 (entități care nu sunt abilitate să emită certificate).

Toate entitățile au încredere în *root CA*.

Model ierarhic

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 (entități care nu sunt abilitate să emită certificate).

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

Model ierarhic

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 (entități care nu sunt abilitate să emită certificate).

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

Avantaj: simplitate și ușurință de implementare.

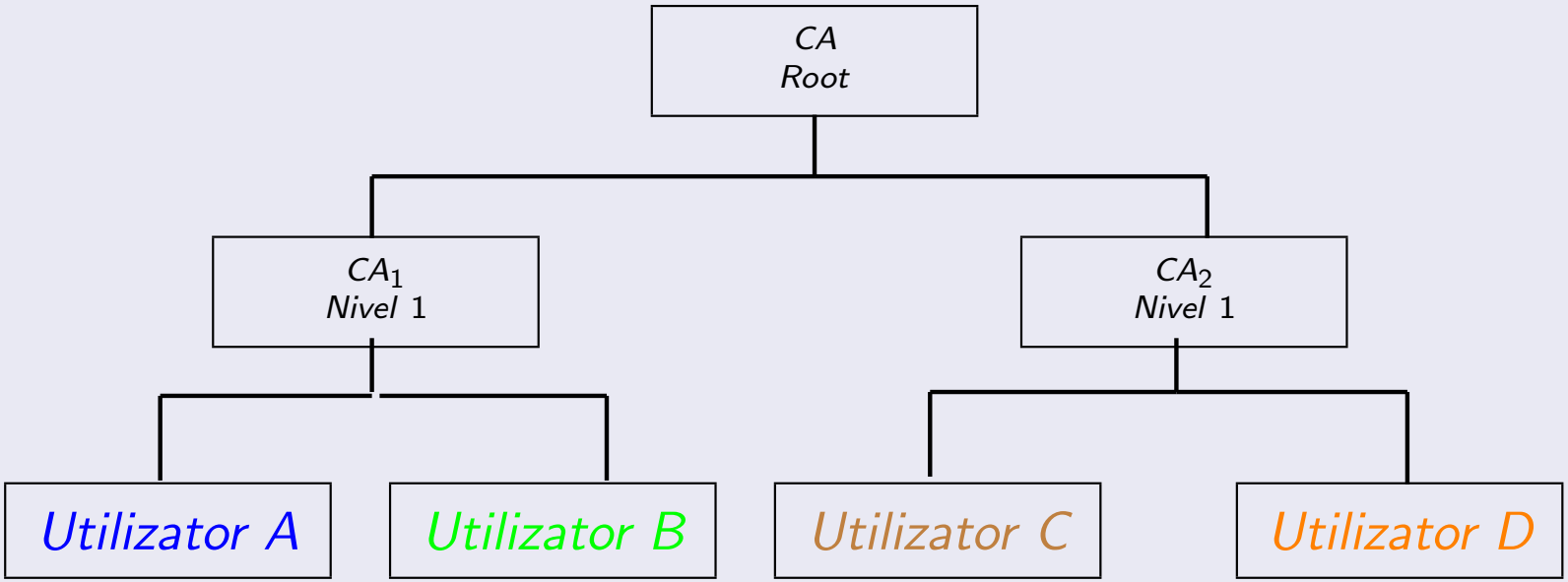
Model ierarhic

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 (entități care nu sunt abilitate să emită certificate).

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

Avantaj: simplitate și ușurință de implementare.
Dezavantaj: nu permite certificări încrucișate.

Exemplu


CA
Root
CA_1
Nivel 1
CA_2
Nivel 1
Utilizator A
Utilizator B
Utilizator C
Utilizator D

Modelul "mesh"

Este bazat pe structura ierarhică și facilitează certificările încrucișate.

Dacă sunt mai multe structuri ierarhice, toate autoritățile de certificare aflate pe poziția de rădăcină se autorizează reciproc prin certificare încrucișată.

Modelul "mesh"

Este bazat pe structura ierarhică și facilitează certificările încrucișate.

Dacă sunt mai multe structuri ierarhice, toate autoritățile de certificare aflate pe poziția de rădăcină se autorizează reciproc prin certificare încrucișată.

Această inter-autorizare are loc înainte de începerea emiterii de certificate către alte entități.

Modelul Web

Pentru un număr mare de utilizatori, modelul de încredere cel mai utilizat este de tip listă, numit și *Web model*.
Fiecare browser Internet acționează ca un *Root CA* virtual, deoarece utilizatorii au încredere în *CA*-ul instalat în softul browserului.

Modelul Web

Pentru un număr mare de utilizatori, modelul de încredere cel mai utilizat este de tip listă, numit și *Web model*.
Fiecare browser Internet acționează ca un *Root CA* virtual, deoarece utilizatorii au încredere în *CA*-ul instalat în softul browserului.

Browserele sunt distribuite în mod normal împreună cu un set inițial de certificate; la acestea utilizatorii pot adăuga sau elimina certificate.

Modelul Web

Pentru un număr mare de utilizatori, modelul de încredere cel mai utilizat este de tip listă, numit și *Web model*.
Fiecare browser Internet acționează ca un *Root CA* virtual, deoarece utilizatorii au încredere în *CA*-ul instalat în softul browserului.

Browserele sunt distribuite în mod normal împreună cu un set inițial de certificate; la acestea utilizatorii pot adăuga sau elimina certificate.
Browserele pot utiliza certificatele pre-instalate pentru a semna, verifica, cripta sau decripta mesajele de e-mail scrise în *S/MIME* și de a stabili sesiuni *TLS* sau *SSL*.

Pentru un utilizator obișnuit, gestionarea numeroaselor certificate instalate în browser constituie o problemă extrem de dificilă.

Exemplu

Browserele Firefox și Microsoft Explorer sunt distribuite împreună cu aproximativ 100 *chei publice pre-instalate, fiecare cheie fiind însoțită de un certificat.*

Pentru un utilizator obișnuit, gestionarea numeroaselor certificate instalate în browser constituie o problemă extrem de dificilă.

Exemplu

Browserele Firefox și Microsoft Explorer sunt distribuite împreună cu aproximativ 100 *chei publice pre-instalate, fiecare cheie fiind însoțită de un certificat.*

În modelul Web nu există o modalitate practică de a revoca certificate.
Astfel, dacă Firefox sau Microsoft instalează din greșeală un *CA* care nu este de încredere, nu există nici o modalitate de a-i revoca certificatul din milioanele browsere aflate în uz.

Model centrat pe utilizator

PGP folosește un model de încredere centrat pe utilizator (User Centric Model).
Orice utilizator *PGP* poate acționa ca o autoritate de certificare și să valideze certificatul cheii publice a altui utilizator *PGP*.

Model centrat pe utilizator

PGP folosește un model de încredere centrat pe utilizator (User Centric Model).
Orice utilizator *PGP* poate acționa ca o autoritate de certificare și să valideze certificatul cheii publice a altui utilizator *PGP*.

Totuși, un certificat eliberat de *Alice* – care acționează ca un *CA* – poate să nu fie valid pentru alt utilizator, pentru că acesta știe că *Alice* nu este de încredere ca autoritate de certificare.

Model centrat pe utilizator

PGP folosește un model de încredere centrat pe utilizator (User Centric Model).
Orice utilizator *PGP* poate acționa ca o autoritate de certificare și să valideze certificatul cheii publice a altui utilizator *PGP*.

Totuși, un certificat eliberat de *Alice* – care acționează ca un *CA* – poate să nu fie valid pentru alt utilizator, pentru că acesta știe că *Alice* nu este de încredere ca autoritate de certificare.

Fiecare utilizator este direct responsabil în a decide ce certificate acceptă și ce certificate respinge.

Algoritmi de semnătură

Certificatele și listele de revocare pot fi semnate teoretic cu orice algoritm de semnătură cu cheie publică.

Algoritmi de semnătură

Certificatele și listele de revocare pot fi semnate teoretic cu orice algoritm de semnătură cu cheie publică.
Algoritmul folosit este totdeauna însoțit de o funcție de dispersie criptografică.
Aceasta produce o amprentă a datelor care trebuie semnate, formatată apoi corespunzător algoritmului de semnătură.

Algoritmi de semnătură

Certificatele și listele de revocare pot fi semnate teoretic cu orice algoritm de semnătură cu cheie publică.
Algoritmul folosit este totdeauna însoțit de o funcție de dispersie criptografică.
Aceasta produce o amprentă a datelor care trebuie semnate, formatată apoi corespunzător algoritmului de semnătură.
După generarea semnăturii, valoarea obținută este codificată cu $ASN.1$ sub forma unui șir de biți și inclusă în certificat.

Algoritmi de semnătură

Certificatele și listele de revocare pot fi semnate teoretic cu orice algoritm de semnătură cu cheie publică.
Algoritmul folosit este totdeauna însoțit de o funcție de dispersie criptografică.
Aceasta produce o amprentă a datelor care trebuie semnate, formatată apoi corespunzător algoritmului de semnătură.
După generarea semnăturii, valoarea obținută este codificată cu $ASN.1$ sub forma unui șir de biți și inclusă în certificat.

Algoritmi recomandați: $DSA/SHA1$.
Alți algoritmi: $HMAC/SHA1$, $RSA/MD5$, $ECDSA/ECDH$

Algoritmi de criptare

1. **Algoritmi cu cheie publică**: Utilizați pentru criptarea cheilor private transportate de mesajele *PKI*.
   Algoritm recomandat: Diffie - Hellman.
   Alți algoritmi: *RSA*, *ECDH*.

Algoritmi de criptare

1. **Algoritmi cu cheie publică**: Utilizați pentru criptarea cheilor private transportate de mesajele *PKI*.
   Algoritm recomandat: Diffie - Hellman.
   Alți algoritmi: *RSA*, *ECDH*.
2. **Algoritmi simetrici**:
   Pot fi utilizați dacă cheia de criptare a fost transmisă prin canal securizat.
   Algoritm recomandat: 3*DES* (în mod *CBC*)
   Alți algoritmi: *RC*5, *Cast* 128.

Standarde *PKCS*

PKCS (Public-Key Cryptography Standards) reprezintă un set de documente sub forma unor specificații, create de Laboratoarele *RSA* în colaborare cu diferiți dezvoltatori din domeniul securității la nivel mondial, în scopul accelerării implementării criptografiei bazate pe chei publice în domeniul privat.

Au fost publicate începând cu 1991, după discuții purtate de un mic grup de adepți ai tehnologiei cu chei publice existent la acea vreme.

PCKS#1

Versiunea existentă: 2.1;

Nume: Standard *RSA*;

Definește proprietățile matematice și formatul de chei publice ale standardului *RSA*, precum și algoritmii de bază pentru schemele de padding *RSA* din domeniul criptării, decriptării și verificării semnăturii digitale.

PKCS#2

Nu mai este disponibil.
Acoperea criptarea *RSA* a amprentelor; acum este înglobat în *PKCS*#1.

PKCS#3

Versiunea existentă: 1.4;
Nume: Standardul Diffie-Hellman.
Protocolul de criptare pentru canale nesigure.

PKCS#4

Nu mai este disponibil.
Acoperea sintaxa cheilor *RSA*; acum este înglobat în *PKCS*#1.

PKCS#5

Versiunea existentă: 2.1;
Nume: Standard de criptare pentru implementarea parolelor.

PKCS#6

Versiunea existentă: 1.5;

Standardul sintaxei certificatelor extinse.

Definește specificațiile extensiilor certificatelor de tip $X.509$ versiunea 3.

PKCS#6

Versiunea existentă: 1.5;

Standardul sintaxei certificatelor extinse.

Definește specificațiile extensiilor certificatelor de tip $X.509$ versiunea 3.

PKCS#7

Versiunea existentă: 1.5;

Nume: Standardul sintaxei mesajelor criptografice.

Standard de semnare și criptare mesaje în interiorul unui *PKI*.

Reprezintă – printre altele – baza pentru *S/MIME*.

PKCS#8

Versiunea existentă: 1.2;
Nume: Sintaxa cheii private.

PKCS#9

Versiunea existentă: 2.0;

Nume: Tipuri de atribute.

Definește tipuri de atribute necesare pentru definirea standardelor *PKCS*#6, *PKCS*#7, *PKCS*#8 și *PKCS*#10.

PKCS#10

Versiunea existentă: 1.7;
Nume: Standardul cererii de emitere certificat.
Cererea către o Autoritate de Certificare pentru certificarea unei chei publice.

PKCS#11

Versiunea existentă: 2.40;
Nume: Interfața Tokenurilor criptografice.
Interfața tokenurilor criptografice (despre *API*-uri generice de acces la dispozitive criptografice).

PKCS#12

Versiunea existentă: 1.1;
Nume: Sintaxa *PIE*.
Definește formatul fizic de fișier necesar stocării cheilor private alături de certificatele publice pereche, protejate de o parolă criptată cu chei simetrice.

PKCS#13

Nume: Criptografia curbelor eliptice. – abandonat.

PKCS#14

Nume: Generarea numerelor pseudo-aleatoare. – abandonat.

PKCS#15

Versiunea existentă: 1.1;
Nume: Formatul informației stocate pe un token criptografic.
Definește un standard prin care diferiți utilizatori de dispozitive token se pot conecta la acestea prin intermediul unor aplicații, independent de driverele definite prin Cryptoki *API* (din *PKCS*#11).

Sfârșit