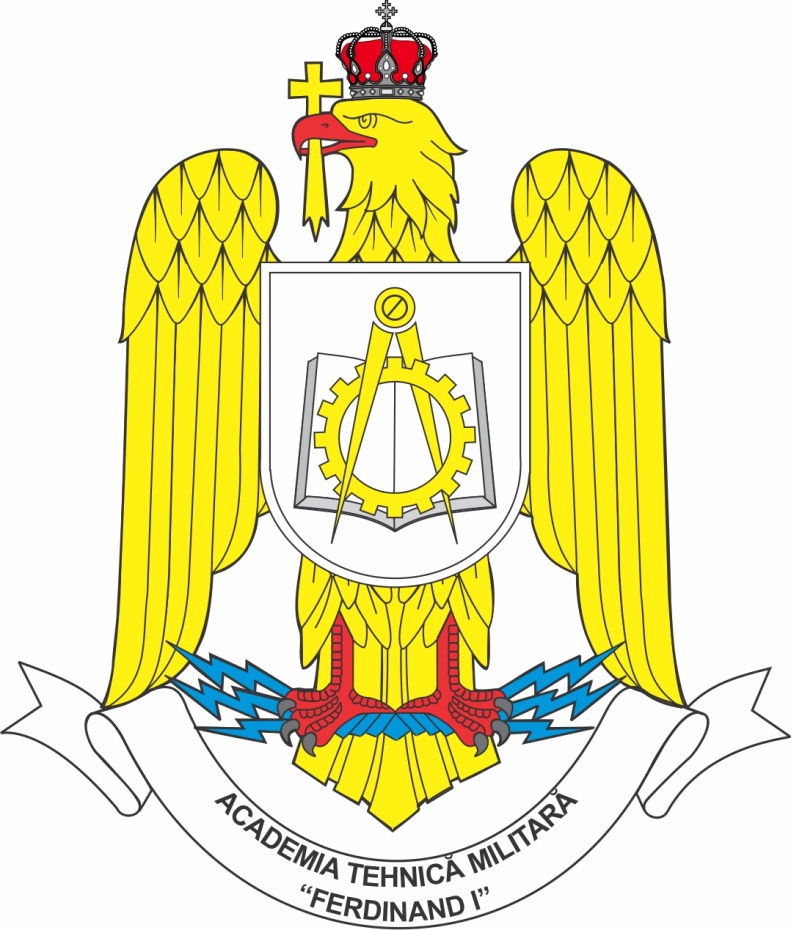
**ROMÂNIA**

**MINISTERUL APĂRĂRII NAŢIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**FACULTATEA DE SISTEME INFORMATICE ŞI SECURITATE CIBERNETICĂ**

**Specializarea: Securitatea Tehnologiei Informației**



**SERVICIU WEB DE SEMNĂTURĂ DIGITALĂ ÎN**

**CLOUD ÎN CONFORMITATE CU CLOUD**

**SIGNATURE CONSORTIUM STANDARD**

CONDUCĂTOR ŞTIINŢIFIC:

**Cpt. conf. univ. dr. ing. Mihai-Lică PURA**

MASTERAND:

**Slt. ing. Robert-George SIMION**

Conţine \_\_\_\_\_\_\_\_\_\_\_ file Inventariat sub nr. \_\_\_\_\_\_

Poziţia din indicator: \_\_\_\_

Termen de păstrare: \_\_\_\_\_

**BUCUREŞTI**

**2020**

Cuprins

[**Listă de figuri** 3](#_Toc11859307)

[**1.** **State of the Art** 4](#_Toc11859308)

[**1.1.** **Criptografia și semnatura electronica in cloud. Tendințe. Beneficii și dezavantaje.** 4](#_Toc11859309)

[**1.2.** **Solutii existente pe piață.** 5](#_Toc11859310)

[**1.3.** **Stadiul actual** 6](#_Toc11859311)

[**1.3.** **Standardul CSC. Arhitecturi și protocoale pentru aplicații de semnătură la distanță** 11](#_Toc11859312)

[**2.** **Design** 13](#_Toc11859313)

[**2.1.** **Arhitectura generală** 13](#_Toc11859314)

[**2.2.** **Descrierea funcționalităților componentelor** 13](#_Toc11859315)

[**2.3.** **Diagrame de utilizare (use-case)** 13](#_Toc11859316)

[**2.4.** **Diagrame de functionalitate (diagrame de activitati/diagrame de secventa)** 13](#_Toc11859317)

[**Bibliografie** 14](#_Toc11859318)

[**Lucrări de specialitate** 14](#_Toc11859319)

[**Online** 14](#_Toc11859320)

[**Standarde, regulamente și directive** 14](#_Toc11859321)

**Listă de figuri**

[Fig. 1 Framework-ul de standarde eIDAS: Standarde publicate [5] 8](#_Toc11872678)

1. **State of the Art**
   1. **Criptografia și semnatura electronica in cloud. Tendințe. Beneficii și dezavantaje.**

Cloud computing este un model de calcul distribuit pe scară largă care integrează un set de resurse abstracte, virtualizate, scalabile dinamic cum ar fi puterea de calcul, memoria, platformele și serviciile. Utilizatorii pot accesa aceste resurse pe Internet folosind terminale, în special cele mobile.

În acest sens, reglementările recente din diferite regiuni ale lumii - cum ar fi eIDAS în Uniunea Europeană - au introdus conceptul de semnături electronice create folosind un "dispozitiv de creare a semnăturilor la distanță", ceea ce înseamnă că dispozitivul de semnătură nu mai este unul personal, sub controlul fizic al utilizatorului, ci mai degrabă este înlocuit de servicii cloud oferite și gestionate de un furnizor de servicii de încredere.

Unul dintre cele mai mari beneficii ale utilizării unei soluții la distanță este faptul că poate fi rulat de aproape oriunde există o conexiune la Internet, indiferent de dispozitivul utilizatorului, dar să fie utilizată o autentificare puternică. Semnătura la distanță poate fi integrată perfect cu dispozitive mobile, cu browsere web sau cu aplicații pentru PC-uri. În loc să se furnizeze o cheie privată de semnare pe o cartelă inteligentă sau un token USB, cheia este generată și stocată central, rămânând sub controlul exclusiv al semnatarului său. Deoarece nu este nevoie de hardware dedicat pentru semnături pe partea utilizatorului, semnătura în cloud este eficientă din punct de vedere al costurilor pentru implementările pe scară largă.

Studiul realizat de Florian Reimair [2] arată o tendință recentă a produselor spre a oferi doar părți/module ale soluțiilor all-in-one pe care le vedem astăzi, datorită creșterii accelerate a numărului de utilizatori, a cerințelor auditorilor și a costurilor de certificare. El precizează că protocoalele și framework-urile viitoare vor contribui la interconectarea produselor într-o manieră modulară, cauzând colapsul soluțiilor all-in-one.

În raportul celor de la Arthur D. Little [3], se compară cazurile de utilizare pentru care soluțiile de semnătură digitală în cloud sunt de așteptat să acapareze o mare parte din piață în viitor: clienții pot vizualiza contractul, declarația fiscală, importul sau exportul de taxe, etc pe orice dispozitiv - fie PC, tabletă sau telefon – pot obține sms-uri pentru autentificare și apoi semnătura fără nicio nevoie de hardware sau software special.

Pe de altă parte, pentru ca o soluție de semnătură digitală în cloud să fie recunoscută și certificată, ea trebuie să îndeplinească unele condiții. Pentru a putea furniza semnături calificate la distanță, conforme cu eIDAS, cu o puternică non-repudiere în cadrul unei instanțe judecătorești, este necesar să se aducă dovada că toate cheile de semnătură centralizate au rămas întotdeauna sub controlul exclusiv al utilizatorului-proprietar, conform specificațiilor prevăzute în anexa 2, secțiunea 1 din eIDAS: Cheia privată poate fi utilizată numai de către proprietar. În plus, un alt dezavantaj este costul și durata foarte mare pentru procesul de certificare.

**Unde ne aflăm în procesul de standardizare pentru semnătura electronică la distanță?**

* Principala provocare este procesul prin care se asigură că semnatarul remote este aceeași persoană cu cea care a fost înscrisă de CA/RA (Autoritatea de certificare, Autoritatea de înregistrare).
* De asemenea, eIDAS încă nu recunoaște crearea de semnături la distanță ca serviciu independent de încredere.
* În plus, propunerea de a elabora un standard pentru serviciul independent de încredere pentru crearea de semnături a fost respinsă
  1. **Solutii existente pe piață.**

Nevoile actuale ale pieței au determinat producătorii să implementeze soluții in-house pentru semnătura electronică la distanță. To deliver eIDAS-compliant remote qualified signatures, with strong non-repudiation in a court of law, requires proof that the centrally-held signing keys always remained under the sole control of the owning user. Exemple precum Bit4ID, Intarsys, DigitalSign au creat produse proprii, certificate, care să asigure acest lucru și care înglobează procesul de înregistrare, autentificare și de semnare electronică.

Bit4ID se centrează astfel pe întreaga stivă PKI de la produse hardware certificate până la emiterea de certificare și verificarea semnăturii elecronice într-un singur ecosistem. DigitalSign pune bazele înregistrării electronice prin identificarea video desktop și mobile a clienților. ComSignTrust™ Cloud de la Digital Signing Solution, Cryptomathic Signer, Cloud-ID de la Infocert, Intarsys au dezvoltat soluții de semnătură în cloud proprii, cu specificații particulare și interoperabile intern. Există și exemple precum grupul Bluecube-Ascertia care își externalizează servicile de semnătură în cloud-ul prin Microsoft Azure.

Tendințele merg din ce în ce mai mult înspre separarea verificării identității și serviciile de semnătură digitală. Adobe și Intesi Group au pionierat sisteme compatibile cu identificări electronice terțe, eID-uri, precum BankID.

Printre problemele pe care le-au întâlnit pe parcursul anilor de experiență se enumeră capacitatea modulelor HSM de reținere a cheilor, dificultatea acreditării dispozitivelor hardware și software, cerințele de auditare aflate în continuă schimbare, păstrarea datelor pe termen lung. Aproape la toți, însă, se remarcă absența publicării standardelor și a regulamentelor privind semnătura electronică în cloud.

* 1. **Stadiul actual**
     1. **eIDAS & Comisia Europeană**

Regulamentul (UE) nr. 910/2014 privind identificarea electronică și serviciile de încredere pentru tranzacțiile electronice pe piața internă (eIDAS) adoptat de Parlamentul European la 23 iulie (și de și de abrogare a Directivei 1999/93/CE ) este un model pentru a oferi un mediu de reglementare previzibil pentru a permite o funcționare sigură și fără probleme a interacțiunilor dintre întreprinderi, cetățeni și autoritățile publice. eIDAS înseamnă o securitate sporită și mai multă conveniență pentru orice activitate online, cum ar fi depunerea declarațiilor fiscale, înscrierea într-o universitate străină, deschiderea la distanță a unui cont bancar, înființarea unei afaceri într-un alt stat membru, autentificarea pentru plățile prin internet, licitație, etc.

eIDAS tratează două domenii principale:

* **Identificarea electronică** asigură că persoanele și întreprinderile pot utiliza scheme de identificare electronică (eID) ale propriei națiuni pentru a accesa serviciile publice din alte țări UE care au disponibile servicii de identificare electronică;
* **Serviciile de încredere** creează o piață internă europeană a sistemului eTS (electronic Trust Services) - și anume semnăturile electronice, sigiliile electronice, ștampila de timp, serviciul de livrare electronică și autentificarea site-urilor - asigurându-se că acestea vor funcționa transfrontalier și vor avea același statut juridic ca cele tradiționale. Numai prin asigurarea certitudinii cu privire la validitatea legală a tuturor acestor servicii, întreprinderile și cetățenii vor folosi serviciile digitale ca mod natural de interacțiune.

Față de legislația anterioară care trata, la nivel european, numai semnătura electronică, iar la nivel național semnătura electronică și marcarea temporală noua reglementare definește o gamă mult mai largă de servicii de încredere. Astfel, conform Regulamentului eIDAS, sunt 9 servicii de încredere calificate:

1. Furnizarea de certificate calificate pentru semnături electronice
2. Furnizarea de certificate calificate pentru sigilii electronice
3. Furnizarea de certificate calificate pentru autentificarea site-urilor web
4. Servicii calificate de păstrare a semnăturilor electronice
5. Servicii calificate de păstrare a sigiliilor electronice
6. Servicii de validare a semnăturilor electronice calificate
7. Servicii de validare a sigiliilor electronice calificate
8. Furnizarea de mărci temporale electronice calificate
9. Servicii de distribuție electronică înregistrată calificate

Toate serviciile de încredere calificate sunt recunoscute ca având aceeași valoare în oricare stat membru UE. Fiecare stat membru înscrie într-o listă de încredere comună la nivelul UE prestatorii de servicii de încredere calificați și serviciile de încredere calificate prestate de aceștia. Această listă este cunoscută sub denumirea de Trust List și este utilizată, de exemplu, pentru a verifica automat dacă un document este semnat cu un certificat calificat.

Comisia a elaborat o legislație secundară și a delegat organisme cu responsabilități pentru punerea în practică a Regulamentului eIDAS:

• Organismele de standardizare **ETSI** (The European Telecommunications Standards Institute) și **CEN** (The European Committee for Standardization) lucrează la standarde tehnice de implementare [5]. Nu toate sunt obligatorii, dar aplicarea unui standard demonstrează alinierea serviciilor prestate cu Regulamentul eIDAS.

• **ENISA** (The European Union Agency for Network and Information Security) a creat ghiduri de implementare a Regulamentului eIDAS pentru toți cei implicați în această activitate: organismele de auditare, prestatorii de servicii de încredere și organismele de supraveghere, responsabile în fiecare stat membru de implementarea Regulamentului eIDAS [6].

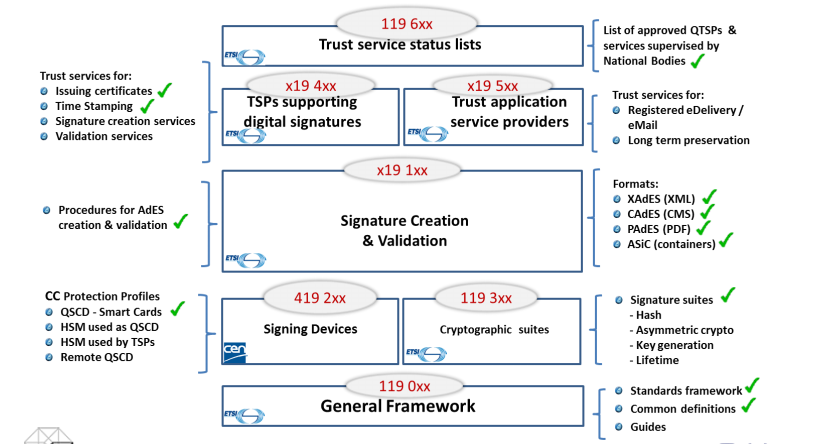


Fig. 1 Framework-ul de standarde eIDAS: Standarde publicate [5]

1. **ETSI**

**ETSI** este o organizație europeană de standardizare. Este singurul organism de standardizare regional, recunoscut, care se ocupă cu telecomunicațiile, radiodifuziunea și alte rețele și servicii de comunicații electronice. ETSI face parte din sigurele trei organisme (CEN, CENELEC și ETSI) care elaborează standarde europene recunoscute.

**ESI TC** (Electronic Signatures and Infrastructures Technical Committee) este responsabilă pentru standardizarea electronică a **semnăturilor** și **infrastructurilor** în cadrul ETSI. Ei lucrează în colaborare cu **CEN TC** pentru a furniza standarde pentru semnăturile digitale.

De asemenea, activitatea lor acoperă crearea și verificarea semnăturilor bazate pe:

* CAdES (semnături digitale CMS),
* XAdES (semnături digitale XML),
* PAdES (semnături digitale PDF),
* ASiC (Associated Signature Container).

Până în momentul actual, ESI a elaborat două standarde:

* **ETSI TS 119 431-1**. Cerințe de politică și de securitate pentru furnizorii de servicii de încredere

Documentul specifică cerințele de politică și de securitate general aplicabile pentru furnizorii de servicii de încredere (TSP) care implementează o componentă de serviciu care operează un dispozitiv de creare a semnăturilor la distanță (SCD).

Pentru crearea de semnături digitale la distanță, datele de creare a semnăturii sunt menținute și gestionate de o terță parte în numele semnatarului. Pentru a garanta că mediul de creare a semnăturilor este fiabil și că datele de creare a semnăturii sunt utilizate sub controlul semnatarului, furnizorul serviciului de semnătură digitală la distanță trebuie să aplice proceduri specifice de gestionare și de securitate administrativă și să utilizeze sisteme și produse de încredere și canale de comunicații electronice.

* **ETSI TS 119 431-2**. Componentele serviciile TSP care sprijină crearea de semnături digitale AdES

Documentul vizează serviciile de încredere, susținând crearea de semnături digitale în conformitate cu cerințele Regulamentului (UE) nr. 910/2014 privind semnăturile electronice și sigiliile electronice (avansate și calificate). Anexa B conține cerințele specifice pentru SSASC (server signing application service component) în contextul Regulamentului (UE) nr. 910/2014, care vizează furnizarea cerințelor de bună practică pentru crearea de semnături electronice avansate și sigilii pe baza certificatelor X.509.

ESI TC lucrează pentru elaborarea unui nou standard:

* **ETSI TS 119 432 Draft Future**. Protocoale pentru crearea semnăturii digitale la distanță.

ETSI TS 119 432 definește protocoalele și interfețele care permit unui client să solicite crearea de semnături digitale AdES definite de ETSI.

Protocolul permite solicitarea creării și returnării rezultatului creării pentru următoarele tipuri de semnături digitale:

* Semnăturile CAdES
* Semnăturile PAdES
* Semnăturile XAdES
* ASiC (Associated Signature Container).
  + 1. **CEN**

CEN este una dintre cele trei organizații europene de standardizare (împreună cu CENELEC și ETSI) care au fost oficial recunoscute de Uniunea Europeană și de Asociația Europeană a Liberului Schimb (AELS) ca fiind responsabile pentru dezvoltarea și definirea standardelor voluntare la nivel european.

În domeniul semnăturilor electronice în cloud, CEN se ocupă cu definirea cerințelor de bună practică pentru modulele criptografice deținute la nivel de server, profilele de protecție pentru QSCD (Qualified Signature Creation Device) la nivel de server și cerințe generale ale SSCD (Secure Signature Creation Device).

CEN a definit standardul TS 419 241 part 1, privind semnătura în cloud. Acesta introduce cerințe de securitate și recomandări generice pentru TSP (Trust Service Providers-Furnizori de servicii de încredere).

* + 1. **CSC**

Cloud Signature Consortium este un grup de organizații academice și din industrie, al cărui obiect este construirea unui nou standard pentru semnăturile digitale bazate pe cloud, care vor sprijini aplicațiile web și mobile și vor respecta cele mai exigente reglementări privind semnătura electronică; un grup internațional de experți din industrie și universități, inclusiv soluții, tehnologie și furnizori de servicii de încredere.

Când semnăturile digitale sunt create într-un dispozitiv, interfețele și funcțiile sunt standardizate, de ex. API-ul folosit de programul de aplicație pentru a accesa bibliotecile de creare a semnăturilor și interfața cu cardul inteligent sau cu un dispozitiv similar (dacă este utilizat un dispozitiv) care ține cheia de semnare. Atunci când semnăturile digitale sunt realizate în cloud, funcțiile necesare pentru a crea o semnătură digitală pot fi distribuite în mai multe instanțe de servicii, fiecare realizând unul sau mai mulți pași în procesul de creare a semnăturii. Cu toate acestea, interfețele dintre astfel de servicii nu au fost standardizate până acum.

Cloud Signature Consortium își propune să elimine această lipsă în standardizare prin definirea designului arhitectural, a protocoalelor de comunicare, a interfețelor de programare a aplicațiilor, a structurilor de date și a cerințelor tehnice necesare pentru a stabili soluții interoperabile pentru semnăturile digitale bazate pe cloud. Deși aceste specificații sunt aplicabile într-o gamă largă de cazuri de utilizare cu cerințe diferite de securitate, îndeplinirea cerințelor impuse de Regulamentul eIDAS al UE [i.1] este abordată în mod special, sprijinind crearea de sisteme electronice "avansate" sau "calificate" semnături și sigilii electronice în cloud.

Standardul CSC este rezultatul contribuțiilor echipei din Technical Working Group al CSC. Acest document conține specificații tehnice care sunt destinate utilizării de către aplicații pentru crearea de semnături digitale în cloud și de o varietate de aplicații care consumă aceste servicii.

1. **Design**

Această secțiune oferă o prezentare generală a arhitecturii sistemului, cu accent pe detalii privind părțile componente ce îl alcătuiesc.

* 1. **Arhitectura generală**

Versiunea curentă a specificației se concentrează pe interfața dintre aplicația de semnătură și furnizorul de servicii de semnătură la distanță. Figura următoare prezintă un exemplu al arhitecturii.

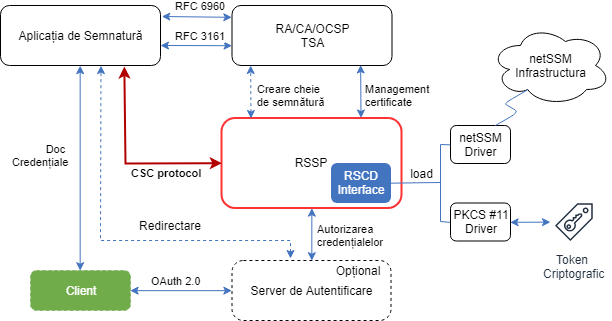
****

Fig. 2 Arhitectura completă a sistemului

Sunt implicate patru entități principale în scenariul de semnătură la distanță.

**Aplicația de semnătură** preia documentul care urmează să fie semnat de la utilizator și, dacă este necesar, certificatele, informațiile de revocare și timpul de la furnizorul de servicii de încredere corespunzător. Acesta solicită furnizorului de servicii de semnătură la distanță să creeze semnătura valorii hash.

**RSSP** (Remote Signature Signing Provider- furnizorul de servicii de incredere) se conectează la CA pentru schimbul de credențiale (certificate). În unele cazuri, CA poate fi de asemenea inclus în procesul de creare a cheii de semnare.

Autorizarea accesului la servicii sau autentificarea poate fi efectuată fie prin intermediul aplicației de semnătură, fie prin utilizarea unei redirecționări către un server extern de autorizare **OAuth 2.0**. De asemenea, serviciul de autorizare poate face parte din RSSP.

* 1. **Descrierea funcționalităților componentelor**

RSSP va putea încărca un **driver** **RSCD**, sub forma unui **plugin**, care implementează o interfață **simplă** predefinită (un **API**), similar procesului de încărcare a unei librării PKCS#11. Acest modul va administra comunicarea cu orice fel de soluție de criptare/semnătură, precum un token criptografic fizic sau virtual, ori enclava netSSM ce folosește tehnologia Intel SGX pentru protecția cheilor.

De ce o interfață simplă și nu una bine cunoscută precum PKCS#11? Un beneficiu este atragerea dezvoltatorilor în a implementa o soluție în cel mai scurt timp. Un dezavantaj îl reprezintă procesul de învățare a interfeței, însă simplitatea ei și o documentație clară l-ar anihila. De ce nu un PKCS#11? Raspunsul este că un astfel de driver ar fi destul de complex (o mașină de stare sau orice fel de modalitate de a ține evidența funțiilor apelate) și ar necesita un timp de implementare și de testare mult mai mare decât timpul destinat creerii unei interfețe simple.

Se vor implementa funcționalitățile de autorizare a serviciului și a semnăturii la nivelul unui **modul de autentificare**, care poate fi apoi externalizat într-un serviciu extern sau înglobat direct în RSSP.

* 1. **Diagrame de utilizare (use-case)**

**Procesul de înregistrare.**

**Scenariul 1**. CA implicat în generarea cheilor de semnare.

Clientul se adreseza RA pentru crearea unui certificat dedicat semnării documentelor. CA generează perechea de chei criptografice, împreună cu certificatul aferent semnat. Obiectele sunt transmise criptat, offline, sau prin alte cai dedicate, RSSP-ului spre stocare.

**Scenariul 2**. RSSP este implicat direct în generarea cheilor de semnare.

Clientul se adreseza RA pentru crearea unui certificat dedicat semnării documentelor. RA contactează RSSP pentru generarea unui CSR asociat unor perechi de chei criptografice. RSSP oferă CSR CA-ului spre semnare, rezultând certificatul clientului. Acesta din urmă este ulterior stocat de RSSP, împreună cu perechea de chei asociată.

**Semnarea documentelor.**

Este inițiată de către aplicația de semnătură și realizată în mod exclusiv de către RSSP. Aplicarea semnăturii de timp se realizează de către aplicația de semnătură.

**Verificarea certificatelor.**

Este realizată în mod obligatoriu de către aplicația de semnătură prin intermediul OCSP sau manual, prin verificarea informațiilor de revocare. La intervale regulate, RSSP actualizează starea certificatelor deținute sau la fiecare cerere de certificat a clienților.

**Verificarea semnăturii documentelor.**

Este realizată numai de către aplicația de semnătură electronică.

* 1. **Diagrame de functionalitate (diagrame de activitati/diagrame de secventa)**

1. **Autorizari**
   1. **Autorizarea la serviciu**

Autentificarea presupune prezentarea unor credentiale corecte si obtinerea unui token de acces. Pe de alta parte, autorizarea presupune prezentarea unui token valabil, finalizandu-se astfel cu accesul la serviciu.

Inainte de toate, accesul la serviciu CSC trebuie autentificat. Autentificarea poate fi aflata sub controlul aplicatiei client sau direct sub controlul serviciului CSC. In primul caz, aplicatia client gestioneaza intern autentificarea utilziatorului cu mentiunea ca trebuie efectuata o corespondenta manuala intre identificatorul utilziatorului in domeniul propriu si cel din domeniul serviciului CSC. Cu alte cuvinte, solutia client va trebui sa gaseasca un mijloc de autentificare la serviciu.

Daca autentificarea are loc direct la serviciul CSC, atunci aplicatia client va trebui sa colecteze credentialele utilizatorului si sa se autentifice cu acestea la serviciu. In practica insa, aplicatia client va redirecta utilizatorul sa se autentifice direct la serviciul de semnatura, pentru a degreva implementarea unei interactiuni prealabile intre aplicatia client si serviciul de semnatura.

**In vederea autorizarii la serviciu, exista 2 metode definite de standard si care se exclud una pe cealalta** (nu se folosesc amandoua una dupa alta sau in acelasi timp):

* **Auth/login**, in care utilizatorul va da credentialele aplicatiei client care le va trimite mai departe catre serviciul CSC. Clientul face tot in locul utilizatorului si ii cunoaste credentialele, PIN-ul.
* **Oauth2/autorize si Oauth2/token**, unde aplicatia client nu va colecta datele utilizatorului, ci acesta se va autentifica intr-o pagina separata de incredere. Pentru alte detalii, vezi cap. 3.1

De mentionat faptul ca serviciul **Oauth2** se foloseste atunci cand o aplicatie client nu este „de incredere” pentru a evita trimiterea directa a credentialelor de acces catre aceasta. In acest sens, se vor folosi doar anumite moduri Oauth care permit „bypass-ul” aplicatiei client. Bineinteles, serverul Oauth trebuie sa fie de incredere.

* 1. **Autorizarea utilizarii cheilor de semnare**

Pentru a permite semnarea unui document, utilizatorul trebuie sa dovedeasca posesia cheii de semnatura prin autorizarea utilizarii credentialelor.

Acest tip de autorizare poate fi sub diferite forme si variatii in functie de implementare sau politici de regularizare. De exemplu, in cazul adoptarii conformitatii SCAL2 „sole control assurance levels” [[1]](#footnote-1), codul SAD necesar autorizarii semnaturii trebuie sa fie legat de documentul/documentele ce trebuie semnate si trebuie sa se foloseasca o autorizare cu factori multiplii (cel putin dublu factor).

**In acest sens, sunt definite 3 tipuri de autorizare a utilizarii credentialelor:**

* Autorizare implicita
* Autorizare explicita
* Autorizare Oauth2

**Autorizarea implicita** inseamna ca serviciul CSC va gestiona singur acest proces, prin colectarea separata a datelor de autorizare fata de aplicatia client, prin diferite metode. De exemplu, se poate folosi autentificarea video sau date biometrice colectate de un dispozitiv de incredere. Utilizatorul nu va interactiona deloc cu aplicatia client in vederea autentificarii. De obicei, se foloseste in aces caz o schema de autorizare Oauth 2.0.

**Autorizarea explicita** presupune ca aplicatia de semnatura sa lase inresponsabilitatea aplicatiei client colectarea tuturor credentialelor si a datelor necesare autentificarii multiple precum PIN/OTP. De exemplu, se poate specifica un PIN static asociat cu cheia de semnare dar si un trimiterea unui OTP gestionat de catre aplicatia client. Clientul face totul in locul utilizatorului si ii cunoaste credentialele, PIN-ul s.a.

**Autorizarea OAuth2** presupune existenta unui server de autorizare de incredere care sa colecteze credentialele si datele utilizatorului printr-o interfata web. Serverul Oauth2 va returna un cod de autorizare si un access token aplicatiei client. Astfel, se degreveaza din responsabilitatea serviciului CSC partea de autorizare a credentialelor.

* 1. **Autorizarea OAuth 2.0 pentru accesul la serviciu si semnarea electronica**

Autorizarea Oauth 2.0 se poate folosi atat pentru autorizarea la serviciul CSC cat si pentru autorizarea semnaturii. Aceasta autorizare presupune afisarea unei interfete web utilizatorului si redirectarea catre o pagina specificata de aplicatia de semnatura care va contine si un authorisation code / access token atasat, ce vor putea fi folosite in relatia cu serviciul CSC.

In cazul in care se foloseste autorizarea OAuth2, aplicatia client va trebui, in prealabil, sa obtina un client id si un client secret **de la serviciul CSC** pe care sa le inregistreze impreuna cu un URL de redirectare -**redirect\_uri-** la serviciul de autorizare. Aceste 3 variabile sunt specifice doar aplicatiei de semnatura si nu fiecarui utilizator in parte.

* + 1. **Autorizarea Oauth 2.0 la serviciu**

1. **Authorisation Code Flow**

Se va folosi fluxul Oauth2 **Authorisation Code Flow,** atunci cand este necesara permisiunea utilizatorului ca aplicatia client sa foloseasca serviciul CSC in **numele** acestuia. Prin apelul **oauth2/authorize**, serverul Oauth2 va prezenta utilizatorului o interfata web prin intermediul careia acesta se va loga direct la serviciul de semnatura CSC pentru a autoriza aplicatia client (a da drepturi acesteia de a se folosi de identitatea lui). In urma acestui apel, aplicatia client va primi un authorisation code pe care poate sa il schimbe impreuna cu client\_id si client\_secret pe un access token prin intermediul apelului **oauth2/token**. Aplicatia client va tine secret fata de utilizator (care poate fi unul rau intentionat) parametrul **client\_secret** preinregistrat anterior, prin comunicarea separata cu serviciul CSC, fara implicarea utilizatorului.

1. **Client Credentials Grant**

Se poate folosi si fluxul Oauth 2.0 **Client Credentials Grant** atunci cand nu este necesara permisiunea utilizatorului ca aplicatia client sa acceseze serviciul CSC. De regula, se executa functii care nu necesita acordul utilizatorului, insa pot exista cazuri in care aplicatia client sa execute functii fara permisiunea explicita a utilizatorului. Acest tip de grant se foloseste atunci cand exista utilizatori logati in domeniul aplicatiei client si de regula pentru functionalitati care nu tin de semnare electronica.

Aici, comunicarea are loc direct intre aplicatia client si serverul csc/serviciul oauth2, fara ca utilizatorul sa poata vedea vreo comunicare intre acestia. In acest caz, se va folosi direct endpointul **oauth2/token**. Cu alte cuvinte, nu se va autentifica aplicatia client la server, ci numai utilizatorul. (**Detalii de implementare)** Aplicatia implementata nu va folosi acest tip de grant.

* + 1. **Autorizarea OAuth 2.0 la semnare**

Pentru autorizarea credentialelor, se pot folosi un singur tip de autorizatie OAuth2 -**Authorisation Code Flow-**.

Ca si la autorizarea pentru accesul la serviciu, acest tip va folosi **oauth2/authorise** si **oauth2/token** cu specificarea tipului de autorizare „credential”.

**Oauth2/autorize** se foloseste pentru schimbul a unui cod de autorizare in baza unor date, odata pentru autorizarea la serviciu si alta data pentru autorizarea utilizarii cheii de semnatura. La finalul autorizarii, utilizatorul va fi redirectat catre un URL specificat de un parametru **redirect\_uri** preinregistrat anterior, care apartine aplicatiei de semnatura prin intermediul careia va primi codul respectiv. Hash-ul din url trebuie sa fie codat **base-64 url safe.**

**Oauth2/token** se foloseste pentru a obtine un access token prin prezentarea fie a codului de autorizare de mai sus, fie a unui refresh token returnat anterior tot de acest endpoint, fie direct a credentialelor aplicatiei client inregistrare anterior in cazul in care nu este necesara permisiunea utilizatorului

**Restrictii privind accesul la serverul de autorizare.**

Se poate dori restrictionarea accesului la serverul de autorizare pentru anumite entitati. De exemplu, intr-o corporatie unde se foloseste intern un serviciu CSC de semnatura, se poate dori restrictionarea accesului pentru toti utilizatorii care nu sunt logati intern la un sistem de management de identitate. Pentru aceasta, protocolul CSC introduce un parametru -**account\_token**- pentru a putea fi folosit in a dovedi identitatea fiecarui utilizator care se conecteaza la acel sistem de management de identitate. De regula acesta este un JWT[[2]](#footnote-2). Acesta token va fi populat de catre aplicatia client care logheaza fiecare utilizator in parte. Acest token va fi verificat de catre serverul Oauth2.

1. **Implementarea securitatii, autentificarii si autorizarii.**

Aplicatia a fost realizata in 2 pasi, respectiv doua nivele de securitate: **autorizarea explicita** si **autorizarea OAuth 2.0**

* 1. Primul layer de securitate reprezinta implementarea **autorizarii explicite**.

In acest caz, pentru autorizarea accesului la serviciu si, respectiv, la credentiale, s-au implementat endpointurile:

* **auth/login**, in care utilizatorul va da credentialele aplicatiei client care le va trimite mai departe catre serviciul CSC. Clientul are acces la credentialele utilizatorului. Dupa autentificare, aplicatia **client** va folosi access tokenul returnat pentru fiecare request **in locul utilizatorului** (si nu in **numele** lui ca la Oauth2).
* **credentials/authorise:** functie implementata pentru autorizarea accesului la credentialele utilizatorului. Functia poate fi folosita si pentru **autorizarea implicita**, caz in care aceasta actioneaza ca un trigger pentru procesul de obtinere a autorizariei direct intre serviciul CSC si utilizator. Clientul afla PIN-ul uilizatorului. Aici apare un risc si anume ca aplicatia client poate retine aceste date si sa realizeze semnaturi si atunci cand utilizatorul nu permite aceasta. Pentru a combate aceasta posibilitate, se poate introduce un OTP[[3]](#footnote-3) generat dinamic si trimis utilizatorului prin alte cai (de ex. SMS). Practic, aplicatia client semneaza in locul utilizatorului.

**Detalii de implementare.** In vederea acestor functii, la nivel de server, s-au implementat 3 tipuri de strategii Passport:

* 1. **Basic Strategy** pentru returnarea unui token de acces pe baza credentialelor date prin Basic Auth Http Header,
  2. **Custom Strategy** pentru gestionarea refresh\_token din body-ul requestului.
  3. **Bearer Startegy** pentru protejarea tuturor endpointurilor prin verificarea tokenului Bearer din headerul HTTP
  4. Al doilea layer de securitate vine in locul primului si presupune introducerea autorizarii OAuth2 atat la nivelul accesului la serviciu cat si la credentiale, mai exact a folosirii fluxului -**Authorisation Code Flow-**.

In acest caz, pentru autorizarea accesului la serviciu si, respectiv, la credentiale, s-au implementat:

* Pagina de logare a utilizatorului la serverul Oauth2.0
* Functionalitate de delogare
* Strategii de gestionare a sesiunilor si de protectie a endpoint-urilor

**Observatie**. Serviciul **Oauth2** se foloseste atunci cand o aplicatie client nu este „de incredere” pentru a evita trimiterea directa a credentialelor de acces catre aceasta. In acest sens, se vor folosi doar anumite moduri Oauth care permit „bypass-ul” aplicatiei client. Bineinteles, serverul Oauth trebuie sa fie de incredere.

In acest scenariu, **Oauth2/token**

A confidential client SHALL authenticate with the authorization server by applying one of the following means:

Passing a pre-issued client secret as a parameter in the request body as described in Section 2.3.1 of RFC 6749 [11].

Applying a pre-issued client secret within the HTTP Basic authentication scheme as described in Section 2.3.1 of RFC 6749 [11].

Passing a client assertion as defined in section 4.2 of RFC 7521 [14].

Aplicatia client se va autentifica prin schema HTTP Basic Authentication

**Dupa logarea utilizatorului, acesta va da acces clientului sa utilizeze serviciul.**

**Detalii de implementare.** In vederea acestor functii, la nivel de server, s-au implementat 2 tipuri de strategii Passport:

1. **Local Strategy** pentru autentificarea utilizatorilor direct la serviciul de autorizare. Permite pastrarea sesiunilor unui utilizator.
2. **Bearer Startegy** pentru protejarea tuturor endpointurilor prin verificarea tokenului Bearer din headerul HTTP; s-a completat cel de la primul nivel in vederea folosirii si tokenurilor emise de serverul Oauth2.0

**OAuth 2.0 Service Authorisation**

1. Clientul deschide aplicatia client pentru a semna un document.
2. Aplicatia client **deschide un browser** si trimite un request catre **/oauth2/authorize** pentru a cere accesul la servicu. Se specifica tipul de flux Authentication Flow, tipul accesului, identificatorul clientului, un url pentru a primi inapoi un cod de autentificare, **scope**-ul (service sau credential), precum si alti parametrii suplimentari necesari pentru o securitate imbunatatita. La nivelul serverului, se creeaza o structura de date (sesiune) referentiata printr-un ID de sesiune unic stocata la nivelul browserului printr-un cookie.
3. Utilizatorul este redirectat catre pagina /login. Utilizatorul este retinut in acea structura de date.
4. Utilizatorul completeaza si trimite formularul. Nicio verificare nu este efectuata aici in prealabil.
5. Daca sunt corecte credentialele (Local Strategy este responsabil pentru verificarea lor), serverul Oauth2 redirecteaza requestul catre endpointul initial **/oauth2/authorize**, utilizatorul fiind logat de data aceasta. Aceasta redirectare initiaza o asa numita **tranzactie**. Pentru ca tranzactia sa fie completa, utilizatorul trebuie sa se logheze si sa aprobe cererea de autentificare a clientului. Pentru ca acest proces include mai multe mesaje HTTP, **tranzactia** este stocata in acea structura de date impreuna cu clientul serializat si requestul initial de la aplicatia client. **Vom avea deci un utilizator logat si salvat in sesiune, independent de o tranzactie.**
6. Daca totul este in regula, utilizatorului ii este afisat pagina de autorizare a aplicatiei client care atasata id-ul de **tranzactie**, detalii despre aplicatia client si despre utilizator. Id-ul este atasat astfel incat sa stie serverul carei tranzactii ii acorda drepturi de autorizatie.
7. Daca utilizatorul aproba tranzactia, atunci se trimite decizia catre endpoint-ul **/oauth2/authorize/decision**
8. Se genereaza un cod de autorizatie. Acesta este legat de client\_id si redirection\_uri.
9. Mai departe, se apeleaza endpoint-ul inregistrat de aplicatia client -**redirect\_uri**- cu codul de autorizatie generat. Se include si prametrul state trimis initial de client pentru o mai buna securizare.
10. Se va falsifica un server de aplicatie client folosind Postman. Se va copia codul de autorizatie si se va trimite un request catre **oauth2/token**. Se va folosi autentificare prin headerul http Basic Authentication cu username -client\_id- si parola –client\_secret-.
11. Serverul va valida requestul, va sterge codul de autentificare si va genera un token de acces. Validarea codului se face prin verificarea **codului de acces**, a **client\_id** si a **redirection\_uri**. Optional, se poate genera si un **refresh\_token,** folositor pentru ca utilizatorul sa nu se mai logheze. (**Detalii de implementare.** Nu se foloseste aici pentru ca utilizatorul are deja creata o sesiune).

**Observatie:** Data fiind natura implementarii Oauth2.0, nu s-a putut realiza returnarea tipului erorii prin apelarea redirect\_uri (daca aceasta exista) in cazul in care se incearca aflarea unui cod de autorizatie.

**Oauth 2.0 Credential Authorisation**

Totul se intampla ca la Oauth 2.0 Service Authorization insa cu urmatoarele observatii:

* La generarea **codului de autorizare**, se retin parametrii **credentialID** si **hash** pentru a fi luati in calcul mai departe la generarea **SAD**.
* Generarea **SAD** va tine cont de **credentialID**, **hash** si **timestamp**-ul curent,

1. **Scenariu de testare**

Fisierul de configurare config.json are valorile in secunde.

Toate valorile din baza de date ce tin de securitatea aplicatiei le sunt aplicate o functie hash.

Aplicatia este formata dintr-un binar, **csc-server**, aflat in posesia unui administrator de sisteme. Cu ajutorul lui, se poate crea un utilizator

1. Instalarea celor necesare
2. **Testarea binarului**
   1. **Se instaleaza binarul apeland**

npm i csc-server

* 1. **Se pot vedea toate optiunile binarului apeland comanda**

csc-server -l

**Rezultatul:**

'Usage:',

        'csc-server [options]',

        '',

        'Options:',

        '  --createUser    Create a user using the arguments below.',

        '       --user           Username of the user.',

        '       --pass           Password of the user.',

        '       --pin            PIN associated with the generated private key.',

        '',

        '  --createClient  Create an OAuth 2.0 client using the arguments below.',

        '       --name          Name of the client application.',

        '       --id            Client id.',

        '       --secret        Client\_secret.',

        '       --redirectUri   Redirect\_uri.',

        '',

        '  --listen, -l     Start the server',

        '  --port, -p       Port to use [8080]',

        '  --address, -a    Address to use [0.0.0.0]',

        '',

        '  --cert, -c       Path to SSL cert file (default: cert.pem).',

        '  --key, -k        Path to SSL key file (default: key.pem).',

        '  --passphrase     Password to SSL private key (default: 0000).',

        '',

        '  --silent, -s     Suppress log messages from output',

        '  --version, -v    Print the version and exit.',

        '  --help, -h       Print this list and exit.',

        '',

        'Examples',

        '   csc-server -l',

        '   csc-server --createUser --user=username --pass=password --pin=password',

        '   csc-server --createClient --name=name --id=id --secret=secret --redirectUri=redirectUri'

* 1. **Testarea generarii unui utilizator**

 csc-server --createUser --user robert.simion --pass 1 --pin 0000

**Raspuns:**

info: Creating the user ...

info: User robert.simion was successfully created!

info: Generating credentials ...

info: Generating certificate and keys...

info: Certificate and keys successfully generated.

info: Converting certificate...

info: Certificate successfully converted.

info: Importing keys to token...

info: Keys successfully imported to token.

info: Importing certificate to token...

info: Certificate successfully imported to token.

info: Credentials successfully generated!

* 1. **Testarea generarii unui client OAuth 2.0.**

csc-server --createClient --name "Academia Tehnica Militara" --id 1 --secret 1 --redirectUri=”<https://localhost:3000>”

**Raspuns:**

info: Creating the client ...

info: Client "Academia Tehnica Militara" was successfully created!

* 1. **Testarea rularii serverului CSC.**

csc-server -l

**Raspuns:**

info: Starting up csc - server, serving through https

Available on:

https://172.17.252.177:8080

https://172.17.252.177:8080

https://192.168.88.1:8080

https://192.168.88.1:8080

https://192.168.213.1:8080

https://192.168.213.1:8080

https://192.168.100.3:8080

https://192.168.100.3:8080

https://127.0.0.1:8080

https://127.0.0.1:8080

info: Hit CTRL-C to stop the server

1. Testarea serverului CSC

Se va folosi un tool exemplificator, Postman, specific testarii serviciilor de tip API[[4]](#footnote-4). Datorita specificitatii Oauth 2.0, testarea acestuia se va face cu ajutorul unui browser:

Se viziteaza in browser pagina:

<https://localhost:8080/oauth2/authorize?response_type=code&client_id=1&redirect_uri=https://localhost:8080&scope=service&lang=en-US&state=1234567>

Utilizatorul se va loga cu credentialele sale introduce de administrator. El va accepta sau respinge cererea aplicatiei client de a accesa resursele sale. Daca cererea e acceptata, atunci utilizatorul va fi rediretat catre o pagina separata (redirect\_uri) in care se afla in url codul proaspat generat. Acel cod se va copia si introduce in Postman la endpoint-ul **oauth2/token**.

**Bibliografie**

**Lucrări de specialitate**

[1] N. Schaettgen, D. Levy, J. Duvaud-Schelnast, S. Socol, „Paving the Way to a Digital Europe”, Technical report, Digital Signatures, Arthur D. Little – Statele Unite. [Online]. Available: <https://technology.a-sit.at/en/cloud-based-signature-solutions-a-survey/> . [Accesat în 25 ianuarie, 2019].

[2] F. Reimair, „Cloud-based signature solutions: a survey”, Technical report, Secure Information Technology Center – Austria. [Online]. Available: <https://technology.a-sit.at/en/cloud-based-signature-solutions-a-survey/> . [Accesat în 25 ianuarie, 2019].

**Online**

[3] “Regulamentul eIDAS privind semnatura electronică”, Certdigital, Iun. 2017. [Online]. Available: https://www.certdigital.ro/wp- content/uploads/2017/06/Regulamentul-eIDAS-privind-semnatura-electronic%C4%83.pdf. [Accesat în 25 ianuarie, 2019].

[4] “Specialist Task Force 539: Standards for eIDAS trust services including electronic signatures – TSP Component Services for Digital Signature Creation” [Online]. Available: https://portal.etsi.org/STF/stfs/STFHomePages/STF539. [Accesat în 25 ianuarie, 2019].

[5] „CEN & ETSI standards & eIDAS Compliance”, Cryptomathic Thales - Standards Compliance, eIDAS conference

**Standarde, regulamente și directive**

[eIDAS] REGULAMENTUL (UE) NR. 910/2014 AL PARLAMENTULUI EUROPEAN ȘI AL CONSILIULUI din 23 iulie 2014 privind identificarea electronică și serviciile de încredere pentru tranzacțiile electronice pe piața internă și de abrogare a Directivei 1999/93/CE, EUR-Lex. Parlamentul European și Consiliul Uniunii Europene.

[CSC] "Architectures and protocols for remote signature applications, published version 1.0.3.0 (2018-12)”, Cloud Signature Consortium. [Accesat în 25 ianuarie, 2019].

1. CEN EN 419 241-1 [i.5], [↑](#footnote-ref-1)
2. JWT – JSON Web Token [↑](#footnote-ref-2)
3. OTP – One Time Pad [↑](#footnote-ref-3)
4. API = Aplication programming Interface [↑](#footnote-ref-4)