TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Teadmussüsteemide õppetool

ID-kaardi baastarkvara teekide automatiseeritud regressioonitestimise raamistiku edasiarendus

magistritöö

Üliõpilane: Ksenija Dvinskihh
Üliõpilaskood: 111707IABM
Juhendaja: Jaak Tepandi

Tallinn
2014

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

ID-kaardi baastarkvara teekide automatiseeritud regressioonitestimise raamistiku edasiarendus

Lõputöö eesmärk on korrastada ID-kaardi tarkvara teekide automatiseeritud regressioonitestimine. Lõputöö kirjutamise hetkel on olemas automatiseeritud testide raamistik, kuid see oli ehitatud 5 aastat tagasi ning nüüdseks vajab uuendamist. 5 aasta jooksul on testimise skoopi lisandunud mitmed testid. See on tekitanud olukorra, kus ei ole täpset ettekujutust, millised teekide funktsionaalsused on testitavad olemasolevas testimise raamistikus ja millised mitte. Automaattestimise raamistik on kirjutatud *Ruby* programmeerimiskeeles ning kasutatakse *RSpec* testimise vahendit. Nii *Ruby* kui ka *RSpec* versioon vajab uuendmist ning testide kood refaktoreerimist.

Selleks, et kohandada testimise skoopi on läbi viidud põhjalik teekide funktsionaalsuse ja ristkasutuse kaardistus. Ekspertide abiga on teostatud teekide funktsionaalsuse riskianalüüs ning paika pandud teekide prioriteedid. Analüüsi põhjal valmis uus testimise skoop.

Lõputöö raames on kaardistatud automatiseeritud raamistiku arhitektuur ning kohandatud see vastavalt 5 aasta jooksul muutunud vajadustele.

Lõputöö raames on kohandatud testide koodi ja selle üldist struktuuri. Testide koodis on uuendatud kasutatavate vahendite versioonid ning läbi viidud koodi refaktoreerimine. Vastavalt uuele testimise skoobile on kohandatud olemasolevad ja loodud uued testid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 94 leheküljel, 5 peatükki, 18 joonist, 11 tabelit.

Abstract

Modernisation of automated regression testing framework of ID-card software libraries.

The aim of this master's thesis is to organize automated regression testing of ID-card software libraries. At the moment of writing this thesis, there was an automated framework for library testing which was built 5 years ago and needed to be updated. During these 5 years several new tests were added to the testing scope. This situation has led to the problem, that there is no overall picture of the scope, what is actually tested and which libraries functionalities are not included in the testing scope. Automated testing framework for library testing is written in Ruby using RSpec testing tool. Versions of Ruby and RSpec are outdated and there is a need of updating them. Also testing code has to be refactored.

Thorough mapping of libraries functionalities and cross-usage is done to reorganize testing scope. Risk analysis of libraries functionalities is done in cooperation with experts and libraries priorities are set. On the basis of risk analysis and libraries priorities, new testing scope is set.

The structure of current automated test framework is mapped and updated according to changed requirements and needs.

Also testing code and its structure is updated. Versions of tools used in automated testing framework are updated and code refactoring is done. According to changed testing scope old tests are updated and new tests are created.

The thesis is in estonian and contains 94 pages of text, 5 chapters, 18 figures, 11 tables., etc.

Lühendite ja mõistete sõnastik

CNG	***CNG*** Windows operatsioonisüsteemi jaoks loodud tööriist. Windowsi keskkonnas on võimalik ID-kaardiga konteinerit allkirjastada kasutades CNG (minidraiver) moodulit.
ECC	***Elliptic Curve Cryptography*** Elliptiliste kõverate krüptograafia on lähenemine avalikule krüptograafilisele võtmele, kasutades elliptiliste kõverate omadusi.
Hard-coded	**Hard-coded** Tarkvaras teostatud nii, et lõppkasutaja ei saa muuta.
Jenkins	***Jenkins*** Avatud lähtekoodiga vahend, mis pakub pideva integratsiooni teenust tarkvara arendusele.
no_zip	**no_zip** Faili krüpteerimisel sisendfaili ei pakita kokku enne krüpteerimist. Funktsionaalsus on võimalik sisse lülitada konfiguratsiooni parameetri abil.
OCSP	***Online Certificate Status Protocol (OCSP)*** OCSP kujutab endast lihtsat klient-server süsteemi, kus OCSP klient saadab OCSP responderile (serverile) päringu mingi sertifikaadi kohta ning responder annab selle sertifikaadi kohta kinnituse, mis sisaldab selle sertifikaadi (mitte)kehtivust ja kinnituse andmise aega. Responderi poolt antud vastus on digitaalselt signeeritud. [1]
Pakend	***Wrapper*** Andmestruktuur, mis võimaldab teist programmi pakendada nii, et mõlemad saavad uuemas süsteemis eksisteerida.

PKCS#11	***PKCS#11*** Avaliku võtme krüptograafia standard, mis määrab platvormist sõltumatu API krüptograafiliste lubakaartide tarvis, reguleerib suhtlust ID-kaardi ja teegi vahel.
PKCS#12	***PKCS#12*** Avaliku võtme krüptograafia standard, mis määrab arhiveeritud failide formaati. Kasutakse mitme krüptograafilise objekti hoidmiseks ühes arhiveeritud failis.
Plugin	**Plugin** Tarkvara komponent, mis lisab spetsiifilise funktsonaalsuse tarkvarale.
Reliis	***Release*** Tarkvara väljaanne avalikuks kasutamiseks.
Repositoorium	***Repository*** Andmete hoiustamiseks ja korduvkasutuseks kasutatav andmebaas.
RSpec	***RSpec*** Testimise vahend *Ruby* programmeerimiskeele jaoks. [2]
Sisendvoog	***Stream*** Sisendfaili sisse lugemise viis. Fail loetakse sisse sisendvoona.
SVN	***Subversion*** Versioonihalduse süsteem. Antud töös kasutatakse testikoodi hoidmiseks ja versioniseerimiseks.
Teek	***Library*** Programmeerimises nimetatakse teegiks valmiskompileeritud alamprogramme, mida programm saab kasutada. DigiDoc teekide abil saad luua DigiDoc-ühilduvaid rakendusi.

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Lõputöö projektiks on valitud ID-kaardi baastarkvara projekti teegid ja nende regressioonitestimine. Lõputöö raames vaadatakse ID-kaardi tarkvara raames arendatavaid teeke ning nende funktsionaalsust. Samuti analüüsitakse praegu[1] olemasolevat testimise raamistiku, mis oli loodud spetsiaalselt teekide testimiseks.

Lõputöö eesmärk on korrastada ID-kaardi baastarkvara teekide regressioonitestimist. Selleks on tarvis kaardistada iga ID-kaardi tarkvara raames arendatava teegi funktsionaalsus ning analüüsida, kui suur osa on praegu testitud. Selline kaardistus aitab aru saada, milline osa funktsionaalsusest ei ole testidega kaetud.

Selleks, et koostada testimise skoop selliselt, mis vastaks hetkevajadustele on tarvilik teostada riskianalüüs ning ekspertide abiga hinnata riskide mõju ja teekide prioriteeti. Selle tulemusena peab valmima uus kohandatud testimise skoop.

Uue skoobi realiseerimisel on tarvilik üle vaadata automatiseeritud raamistiku struktuur ning koodi struktuur, vajadusel kohandada nii raamistikku kui ka koodi. Lisaks on tarvilik uuendada testide koodis kasutatavaid vahendeid, näiteks programmerimiskeele versiooni ning testimise vahendi versiooni.

1.1 Taust ja probleem

ID-kaart on Eesti kodaniku ja Eestis püsivalt elava Euroopa Liidu kodaniku isikut tõendavaks dokumendiks. ID-kaart võimaldab kasutada riigi ja erasektori e-teenuseid ning anda digitaalseid allkirju. ID-kaardiga on võimalik end tuvastada elektroonilises keskkonnas. Näiteks on ID-kaart lihtsaim, mugavaim ja turvalisim võimalus internetipankade ja teiste e-teenuste kasutamiseks. [3]

ID-kaardi elektrooniliseks kasutamiseks on tarvis omada ID-tarkvara. ID-tarkvara võimaldab isikul ID-kaardiga tõendada oma isikut elektrooniliselt - näiteks kasutades pankade, riigi või

[1] Praeguse all peetakse silmas lõputöö kirjutamise hetkel kehtivaid ja olemasolevaid asjaolusid.

erinevate ettevõtete teenuseid veebi kaudu, samuti on võimalik digitaalselt allkirjastada ja krüpteerida dokumente. [4]

ID-kaardi tarkvara projekt koosneb mitmest alamosast:

- *Desktop* tarkvara
- Veebilehitseja *pluginad*
- Tarkvara teegid
- ID-kaardi draiverid

Käesolev lõputöö keskendub ID-kaardi tarkvara ühele osale – teekidele, teekide testimise skoobile, selle automatiseeritud testimise raamistikule ning tulemuste raporteerimisele.

ID-kaardi teekide testimiseks on juba olemas automatiseeritud testide raamistik. See raamistik on kirjutatud programmeerimiskeeles *Ruby*, kasutades *RSpec* testimise vahendit. Antud raamistik oli kirjutatud hinnanguliselt 5 aastat tagasi ning vajab kaasajastamist ja kasutatavate vahendite uuendamist. Automatiseeritud testides praegu kasutatav *Ruby* versioon on 1.8.7 ja *RSpec* versioon on 1.0. Hetkel kõige uuem *RSpec* versioon on 2.14 ja selles versioonis lisandunud funktsionaalsused aitaksid kaasa ID-kaardi teekide automatiseeritud testimisele. Uuemate *Ruby* ja *RSpec*'i versioonide kasutamine lahendaks autotestide raamistikus mitmeid probleeme, näiteks uuemas *Ruby*'s on tehtud lihtsamaks UTF-8 kodeeringu kasutamine [5], mis ID-kaardi teekide testimise vaatenurgast on oluline muudatus. Mitmed funktsionaalsused, mis on toetatud *RSpec* 1.0's ning kasutatud ka ID-kaardi teekide autotestimisel on aga *RSpec* 2.14s ümber nimetatud või koguni ära kaotatud [6], seetõttu vajab kood üle vaatamist ja uue versiooniga vastavusse viimist. Lisaks on *RSpec*'i kasutamise stiil veidi muutunud, seega olemasolevat koodi on tarvilik ümber kirjutada vastavalt uuele standardile.

Algsele automatiseeritud testimise raamistiku koodile on ajaga lisandunud uusi koodijuppe, testimaks teekide uusi funktsionaalsusi. Lisaks on mõned teekide funktsionaalsused ära kaotatud, aga testikood on alles. Selline olukord on viinud asjaoluni, et tänases koodis esineb koodi dubleerimist ning leidub koodi ridu või isegi terveid meetodeid ja testifaile, mida nüüdseks enam ei kasutata. Seega vajab kood üle vaatamist ja refaktoreerimist. Lisaks kogu testiraamistiku arhitektuur vajab üle vaatamist, kuna testiraamistiku ajaloo jooksul olid

tekitatud mõned testimiseks vajalikud funktsionaalsused, mida nüüd saab muuta paremini kasutatavaks ja sõltumatumaks testiandmetest.

Teekide funktsionaalsus on laienenud ning mõned funktsionaalsused on teekidest välja võetud. Aja jooksul on tekitatud juurde palju teste teekide funktsionaalsuste kontrollimiseks. Praeguseks seistakse silmitsi aga olukorraga, kus täpselt ei teata, milline kood on testidega kaetud ning mida üleüldse teekide funktsionaalsusest ei testita. Raamistiku ümber ehitamisel on tarvis kaardistada testitav skoop ning vajadusel ja võimalusel optimeerida see.

Igal teegil on oma funktsionaalsuste nimekiri, mida teek toetab. Lisaks peavad teegid olema ristkasutatavad omavahel, selline ristkasutus peab samuti olema testitud. Peale teekide ristkasutuse peavad teegid veel toetama mitmeid aspekte, näiteks on kasutusel erinevad kaardid ja kaardilugejad, kusjuures igal kaardilugejal on oma draiver ja neid komponente, mida tuleb arvestada teegi testimisel, on veel. Selline suur hulk komponente, mida tuleb testimisel arvestada on loonud olukorra, et praegused automatiseeritud testid on kohmakad ning testide käitamise aeg on pikk (näiteks kõige pikemate dekrüpteerimise testide jooksmise aeg ulatub 5-6 tunnini). Nii testide struktuuri kui ka skoobi üle vaatamine võimaldaks muuta testid modulaarsemateks, iseseisvamateks ja lühemateks.

Koodi refaktoreerimine ja skoobi optimeerimine aitaks kaasa teekide testimiseprotsessi parendamisele. Skoobi kaardistamine ja optimeerimine annaks parema ülevaate testitavatest aspektidest ning testiraamistiku arhitektuuri ümber disainimine ja kohandamine aitaks anda paremat ja kiiremat tagasisidet nii arendajale kui ka projektijuhile.

Üheks testiraamistiku osaks on ka raporteerimine. Praegu genereeritakse suhteliselt detailne raport testitulemustest, kuid puudub projektijuhile suunatud informatsioon kogu tarkvara üldisest seisukorrast. Lisaks seoses sellega, et testid on suured ja kohmakad, siis ka testide raport on suur ja kohmakas ning arendajal kulub palju aega tulemuse leidmiseks ja analüüsiks.

1.2 Ülesande püstitus

Lõputöö eesmärk on korrastada ID-kaardi tarkvara teekide testimist.

Teekide testide korrastamiseks on tarvis kõigepealt kaardistada tarkvara osad ning tarkvara mõjutatavad vahendid. Seejärel on tarvilik kaardistada teekide funktsionaalsuste skoop ning kaardistada olemasolev testidega kaetud skoop. Selline kaardistus aitab mõista, milline osa

teekide funktsionaalsusest ja teekide vahelisest ning operatsioonisüsteemide vahelisest ristkasutusest on praeguses raamistikus kaetud testidega ning milline mitte.

Teekide skoobi kaardistuse ja praeguses raamistikus testidega kaetud skoobi alusel on tarvilik üle vaadata testimise skoop ning vajadusel korrigeerida see. Skoobi korrigeerimisel on planeeritud kasutada riskipõhist hindamist ning eksperthinnangut. Testide struktuuri genereerimisel arvestatakse sellega, et testid oleksid mõistlikult lühikesed ning võimalikult iseseisvad ja sõltumatud. Testide eesmärk on anda kiiret ja adekvaatset tagasisidet igale projektis osalevale rollile - kõigepealt arendajatele ning testijatele, aga lisaks ka testijuhtidele ja projektijuhtidele.

Vastavalt korrigeeritud skoobile võib olla tarvilik ümber ehitada testimise raamistik. Selleks on tarvilik kaardistada praeguse raamistiku üldine arhitektuur ning kaaluda selle ümber disainimist.

Testide kood vajab kasutatavate vahendite uuendamist ning kood vajab refaktoreerimist. Testide kood on vaja muuta korduvkasutatavaks ning võimalikult sõltumatuks testiandmetest. Mõned testides kasutatavad väärtused ja vahendid, mis on praegu testide koodis nii öelda „*hard-coded*“ on vaja muuta selliseks, et neid vahendeid ja andmeid oleks võimalik hallata koodist väljas pool. Kui koodist väljaspool ei ole võimalik mõningaid andmed muuta, siis peab seda saama teha võimalikult lihtsalt ning koodis vaid ühes kohas. Selline lähenemine tagaks testide ja testikoodi kergema hallatavuse.

1.3 Metoodika

Skoobi genereerimisel ja optimeerimisel lähtutakse *SMART* kriteeriumitest [7], lisaks kasutatakse riskipõhist ja eksperthinnangul põhinevat lähenemist.

SMART kriteerium on spetsiifiline lähenemine eesmärkide seadmisele. Iga täht sõnas tähendab erinevat lähenemise vaatenurka eesmärkide seadmisele ning kõik kokku loovad juhendi eesmärkide defineerimisele. *SMART* kriteeriumit võib tõlgendada erinevat moodi lähtudes valdkonnast ja tegevusalast. Käesoleva töö mahus rakendame *SMART* kriteeriumit testimise skoobi defineerimisele.

S - Specific ehk konkreetne. Konkreetset eesmärki on palju lihtsam saavutada, kui üldist. Konkreetne eesmärk peab vastama järgmistele küsimustele: kes?, mis?, kus?, millal?, miks? kuidas?.

Testide automatiseerimise mõttes on tarvilik kõigepealt defineerida, mida hakatakse automatiseerima ja miks? Tuleb fikseerida, milliseid vahendeid automatiseerimiseks kasutatakse. Automaattestide skoobi genereerimisel lähtutakse põhimõttest, et testid, mida hakatakse automatiseerima, on võimalikult konkreetselt defineeritud ning on üheselt arusaadavad.

M- Measurable ehk mõõdetav. Eesmärk peab olema mõõdetav ning eesmärgi defineerimisel peab olema defineeritud ka see, kuidas hakatakse hindama, kas eesmärk on täidetud või mitte.

Skoobi defineerimisel võib seda tõlgendada nii - skoop peab olema mõõdetav ehk peab olema üheselt arusaadav ja selge, milline osa tarkvarast hakkab olema kaetud testidega ja milline mitte. Millised funktsionaalsused hakkavad olema kaetud automaattestidega ja millised mitte.

A - Attainable ehk saavutatav. Eesmärk peab olema defineeritud nii, et see oleks saavutatav ehk hindamisel oleks võimalik selgeks teha, kas eesmärk on saavutatud või mitte.

Skoobi defineerimise mõttes võib tõlgendada seda järgnevalt: On ilmselge, et ei ole mõttekas testida kõike. See tähendab seda, et ei ole mõttekas testida kõiki kombinatsioone, mida on võimalik välja mõelda. Skoobi defineerimisel tuleb arvestada saavutatavust andmete ja tarkvara funktsioonide kombinatsioonide välja töötamisel, et nende kombinatsioonide ja andmetega oleks testimine nii tarkvaraliselt kui ka tarkvara mõjutavate tegurite poolest võimalik.

R - Relevant ehk asjakohane. Eesmärk peab väljendama suunda, mille suunas on tahe ning võimalus liikuda. See peab väljendama progressi millegi suunas. Eesmärk saab olla samal ajal nii kõrge kui ka realistlik.

Skoobi defineerimisel on oluline aru saada, et osa tarkvara funktsionaalsusest või ristkasutusest jääb katmata automaattestidega. Oluline on valida üks suund, mille alusel valitakse funktsionaalsused või ristkasutuse osad, mida hakatakse katma automatiseeritud testidega.

T - Timely ehk ajakohane. Eesmärgi defineerimisel tuleb paika panna ka ajapiirang, millal selle eesmärgini tahetakse jõuda. Abstraktse ajapiiranguga eesmärgid tihtipeale jäävad saavutamata, kuna sellega ei ole nii öelda “kiiret” ning ajaga kaotavad need eesmärgid oma tähtsuse.

Skoobi defineerimiseks võib seda tõlgendada nii:

Arvestada tuleb testimisele ja testide automatiseerimisele planeeritud ajaga ja ressurssidega ning defineerida testimise skoop selliselt, et selle saavutamine oleks etteantud ajapiirides ja kättesaadava ressursi poolest üleüldse võimalik. Testide skoobi defineerimisel tuleb samuti arvestada ka hetkel või testimise ajal kasutatavate tehnoloogiatega. Näiteks skoobis defineeritud operatsioonisüsteemid ei oleks vananenud või kasutusest ära võetud.

Automatiseeritud testide skoobi defineerimisel kasutatakse ka senini kasutusel olevat eksperthinnangu meetodit ning kombineeritakse seda riskipõhise meetodiga.

Testitava süsteemi või tarkvara riskide identifitseerimise ja analüüsimisega on võimalik koondada testimine kõige kriitilisematele kohtadele süsteemis või tarkvaras. Selline lähenemine aitab õigesti prioritiseerida riske arvestades testimisel kasutatavate ressursside ja etteantud ajaga. (Automaattestimise puhul ressursside all mõeldakse nii rahalisi ressursse kui ka testimisel kasutatavat masinateparki ja nende võimalusi). Lisaks aitab riski-põhine lähenemine leida mõistliku kaetuse testidega, lähtuvalt konkreetsest riskist. Kuna õige riskide hindamine, prioritiseerimine ning mõistliku piiri leidmine riski testidega kaetuse osas nõuab kogemust testimises üleüldiselt kui ka kogemust just selle konkreetse tarkvara testimisel ja arendamisel, siis kombineeritakse riski-põhist lähenemist ekspertide hinnangutega. [8]

Riskipõhise lähenemise korral on mitmeid eeliseid:

1. Riskipõhise lähenemise korral on võimalik defineerida, millal lõpetada testimine
2. Nii testimise skoopi kui ka teste endid on võimalik lühemaks teha ning keskenduda kõige kriitilisematele kohtadele.
3. Riskipõhise lähenemise korral on defineeritud vähem teste, kuid need on spetsiifilisemad ja detailsemad.
4. Riskipõhise lähenemise korral teostatakse detailne riskianalüüs ning sellest lähtuvalt on testid ja testimise skoop rohkem keskendunud kõrgema riskiga tarkvara või süsteemi osadele.
5. Probleemsed kohad on tuvastatud varakult ning nendega tegelemine saab alata varasemas staadiumis.

6. Valitakse strateegiad testimise läbiviimiseks ja testitavate objektide valikuks.

7. Üleüldiselt testimise eesmärgid ja strateegiad keskenduvad tarkvara või süsteemi probleemsetele kohtadele.

8. Tarkvara või süsteemi iga risk on pideva järelevalve all ning selle alusel saab hinnata tarkvara kvaliteeti ning projekti staatust.

Tuleb arvestada, et isegi riskipõhise lähenemise rakendamisel, jääb alati üles oht, et mingisugused riskid jäävad tuvastamata. Sellisel juhul ei arvestata neid tuvastamata jäänud riske ka testimisel ja tarkvara seisu ja kvaliteedi hindamisel. See võib viia tarkvara seisukorra valesti hindamiseni. Selline olukord tõestab veelkord, kui oluline on pidev ja põhjalik riskide tuvastus ja analüüs.

Samuti võib probleemiks olla olukord, kus riskid ja nende prioriteedid on hinnatud väga subjektiivselt. Mistõttu ekspertide valimine, kes osalevad riskide identifitseerimisel ja hindamise teostamisel, on väga oluline. Lisaks riskianalüüsi tegemisel on oluline tuvastatud riskide võimalikult täpne kirjeldamine, et ei tekiks probleeme testide ja riskide sidumisega ning kriitilised riskid saaksid võimalikult suure kaetuse testidega.

Testimise jaotamine

Kuna komponente, mida antud projektis tuleb testida, ning komponente, mida tuleb arvestada tarkvara testimisel, on palju, siis on otsustatud jaotada testid suitsu- ja regressioonitestideks. [9]

Suitsutestimine on kiire ja nii-öelda “must” testimine. Suitsutestimise käigus kontrollitakse tarkvara või süsteemi osa põhifunktsioonide korrektset toimimist. Suitsutestimise alla kuuluvad testid on kiired ning keskenduvad vaid põhifunktsioonidele. Tihtipeale kasutatakse suitsutestimist kontrollimaks, et uus muudatus või tarkvara uus osa ei rikkunud ära kõikide teiste tarkvara osade funktsionaalsust.

Regressioonitestimine võimaldab järjekindlat ja korratavat tarkvara valideerimist. Selline testimine kindlustab, et eelmise regressioonitestimise ajal leitud vead on parandatud ning hooldusprotsessi käigus tehtud tegevused ei mõjutanud tarkvara kvaliteeti. Kui suitsutestimine on kiire ning keskendutakse põhifunktsioonidele, siis regressioonitestimise all mõeldakse väga põhjalikku ja detailset tarkvara testimist.

1.4 Ülevaade tööst

Lõputöö peatükis 2 on kaardistatud ID-kaardi tarkvara ning tarkvara mõjutavad osad, mida tuleb testimisel arvestada. Samuti on kaardistatud ID-kaardi tarkvara teekide funktsionaalsus ning olemasolevas raamistikus testidega kaetud skoop.

Peatükis 3 on teostatud teekide funktsionaalsuse riskianalüüs ning selle alusel on koostöös ekspertidega koostatud uus testimise skoop.

Peatükis 4 on kaardistatud olemasolev automatiseeritud testimise raamistik ning kirjeldatud uus testimise raamistiku struktuur. Vastavalt uuele raamistiku struktuurile on kohandatud testide koodi struktuuri, lisatud uued testid vastavalt uuele testimise skoobile ning kood on refaktoreeritud.

2. Tarkvara kaardistus

2.1 Tarkvara üldise struktuuri kaardistus

2.1.1 Tarkvara struktuur

ID-kaardi tarkvara on võimalik alla laadida id.ee lehelt ning installeerimise tulemusena paigaldatakse arvutisse mitme osaline ID-tarkvara, millest 4 osa sellest tarkvarast on tavakasutajale nähtavad ja kasutatavad. Need on:

1. Haldusvahend, mille abil on võimalik kontrollida ID-kaardi töötamist elektrooniliselt ning sertifikaatide kehtivust, muuta ja lahti blokeerida pin-koode.

2. Digidoc 3 klient, mille abil on võimalik allkirjastada dokumente, kontrollida allkirjastatud dokumentide allkirjade kehtivust ning avada ja salvestada allkirjastatud konteinerites olevaid dokumente.

3. DigiDoc 3 krüpto, mille abil on võimalik allkirjastatud dokumente salastada ehk krüpteerida dokumentide turvaliseks edastamiseks ning muuta dokumente uuesti kõigile loetavaks ehk dekrüpteerida.

4. Veebilehitsejate pluginad, mis võimaldavad allkirjastamist brauseris.

Haldusvahend, DigiDoc 3 klient, DigiDoc 3 krüpto ja veebilehitsejate pluginad on ID baastarkvara projekti nii öelda *desktop* lahendus. Selle tarkvara osa sihtgrupiks on inimene, kes omab Eesti ID-kaarti ning soovib kasutada seda elektrooniliselt.

Tegelikkuses on ID-kaardi baastarkvara palju keerukam, kui tavakasutajale nähtav *desktop* lahendus. Lisaks neljale tavakasutajale nähtavatele tarkvara osale paigaldatakse arvutisse veel tarkvara komponendid, mis toetavad *desktop* lahenduse tööd ning mis aitavad kasutada ID-kaarti elektrooniliselt veebibrauseris.

Desktop lahenduse korrektse töö tagamiseks paigaldatakse arvutisse abiteegid - cdigidoc ja libdigidocpp teegid, ning draiverid - opensc, minidraiver ja csp draiver.

Lisaks üheks tarkvara osaks on teegid. Teegid võimaldavad arendada teenuseid, mis kasutavad ID-kaarti. ID-kaardi projekti raames arendatakse mitmeid teeke: multiplatvormne C-teek, CPP

teek, *JDigiDoc* teek Java rakenduste tarbeks, COM teek Windowsi platvormil kasutamiseks, *NDigiDoc* teek krüpteerimiseks ja dekrüpteerimiseks.

Kokkuvõtteks võib öelda, et id-kaardi baastarkvara projekt koosneb kolmest osast:

- *Desktop* lahendus
- ID-kaardi tarkvara teegid
- ID-kaardi draiverid

2.1.2 Formaadid

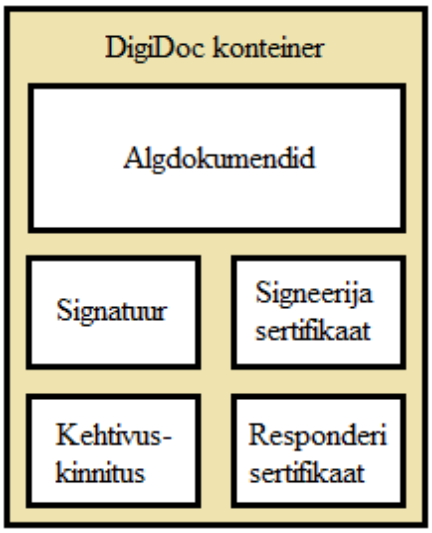
ID-kaardi tarkvara abil on võimalik luua mitut DigiDoc konteineri tüüpi. Allkirjastamisel on võimalik luua konteinereid, mis on DDOC või BDOC formaadis.

- DDOC ehk allkirjastatud konteiner, mille laiendiks on .ddoc. „Digitaalallkirjastatud failide formaat baseerub ETSI TS 101 903 standardil, mida kutsutakse XML Advanced Electronic Signatures (XAdES). Dokumendivormingu (.ddoc) versioon määrab üheselt DigiDoc faili struktuuri. Uus failiformaat võetakse kasutusele, kui tulenevalt rahvusvahelistest standarditest või ühilduvusest muude digitaalallkirja süsteemidega on vajalik DigiDoc dokumendistruktuuri muutmine. DigiDoc-i failiformaadi versioon muutub väga harva.

  DigiDoc failivormingu esimene versioon 1.0 kandis nime SK-XML, järgmised versioonid (1.1 ja edasi) kannavad nime DIGIDOC-XML. Versioon ja nimi on näha dokumendi päises näiteks järgmiselt: <SignedDoc format="DIGIDOC-XML" version="1.1">“ [10]. Praegusel hetkel on kasutusel DigiDoc-XML 1.3 Digidoc formaat, mis on ka viimane ametlik Digidoc formaat.“

  DDOC konteineri puhul kinnitatakse digiallkiri OCSP teenuse abil, kus kasutatakse OCSP standardi protokolli standardilaiendust nimega „nonss“ (nonce). OCSP kinnituse täpsem kirjeldus on leitav dokumendis „SK ajatempliteenuse ajatembelduspõhimõtted“ versioon 2.0 [1]. Lõputöös on see allkirja kehtivuse kontrollimise viis tinglikult tähistatud terminiga „time-mark“.

  Joonis 1 kujutab DigiDoc konteineri struktuuri.



Joonis 1. DigiDoc konteineri struktuur. [10]

- BDOC ehk allkirjastatud konteiner. BDOC versioon 1.0 baseerub EVS 821:2009 standardil [11]. Seda digiallkirja vormingu versiooni ei soovitatud kasutada. „BDOC 2.0 on uus digitaalallkirja vorming, mis on loodud selleks, et asendada Eesti-spetsiifilist DDOC (DigiDoc) digitaalallkirja vormingut. BDOC 2.x põhieesmärk ongi olla ühilduv rahvusvaheliste ETSI standarditega“. [10] Praegu avalikusele kättesaadav digiallkirja vormingu versioon on 2.1, BDOC 2.0 ei jõudnud avalikku kasutusse.

  BDOC allkirjastamisel kasutatakse „*time-mark*“ profiili, see tähendab, et allkirjastaja sertifikaadi kehtivus lisatakse allkirjale ning kehtivuse tõend saadakse OCSP vastusena. BDOC „*time-mark*“ vastab XAdES LT taseme nõuetele.

Krüpteerimisel on võimalik luua konteinerit formaadis CDOC.

- „CDOC on laiend, mida kasutatakse krüpteeritud DigiDoc vormingus failide eristamiseks. Krüpteeritud DigiDoc failivorming (ENCDOC-XML) põhineb rahvusvahelisel standardil XML-ENC.

  CDOC fail sisaldab krüpteeritud kujul andmefaili (XML dokument või muu binaarfail (MS Word, Excel, PDF, RFT jne)), vastuvõtja sertifikaati, krüpteeritud kujul võtit andmefaili lahtikrüpteerimiseks (nn. transpordivõti) ja muid mittekohustuslikke meta-andmeid. Andmefail krüpteeritakse AES krüpteerimisalgoritmiga, kasutades 128-bitist võtit. Ühele krüpteeritud failile on võimalik lisada mitu vastuvõtjat (võimalikku

dekrüpteerijat) – selleks lisatakse CDOC faili kõikide vastuvõtjate sertifikaadid ja iga vastuvõtja jaoks transpordivõti andmefaili lahtikrüpteerimiseks.“ [10]

2.1.3 Mida tuleb arvestada tarkvara testimisel

ID-kaardi elektroonilisel kasutamisel on tarvis omada ID-kaarti ennast ja ID-kaardi lugejat.

Järgnevalt on kirjeldatud osad, mis on otseselt seotud ID-kaardi kasutamisega elektrooniliselt, mis ei kuulu ID-kaardi tarkvara alla, kuid nendega tuleb arvestada ID-kaardi tarkvara arendamisel ja testimisel.

10 - aastase ID-kaardi kasutusajaloo jooksul on välja antud väga palju erinevaid ID-kaartide versioone ning organisatsiooni jaoks mõeldud sertifikaate, mida võib grupeerida järgnevalt:

- enne 2011 aastat väljastatud ID-kaardid. Võib öelda, et siia kuulub kaks põlvkonda kaarte. Esimesed, mis olid välja antud aastatel 2002-2007 aastatel 10 aastase kehtivusega ning 2007-2011 aastatel välja antud kaardid, mis on 5 aastase kehtivusega. ID-kaardi tarkvara kontekstis on need kaardid käitumuslikult sarnased.

- 2011 aastal väljastatud ID-kaardid, mida nimetatakse uue põlvkonna kaartideks (ei ole enam elektrooniliselt kasutatavad). Peale kaardi tarkvara uuendamist Politsei- ja Piirivalveametis kvalifitseeruvad 2012 aastast väljastatud kaartideks.

- 2012 aastast väljastatavad ID-kaardid uuenenud kaardi tarkvaraga.

- Digi-ID ehk digitaalne dokument, mida on võimalik kasutada elektrooniliselt isikutuvastamiseks ning digiallkirjastamiseks.

- Mobiil-ID on ID-kaardil põhinev isikusamasuse tõendamise ning digitaalse allkirjastamise lahendus mobiiltelefonis.

Lisaks avalikkudele ja laialt levinud ID-kaartidele peab ID-kaardi tarkvara toetama allkirjastamiseks mõeldud sertifikaate, mille tüübid on toodud välja järgnevalt:

- Softsert – faili kujul kasutada olev allkirjastamiseks mõeldud sertifikaat. Sellega ei saa anda Digitaalallkirja seaduse [12] mõttes kehtivaid allkirju, vaid süsteemi sisemiseks kasutamiseks nii nimetatud tehnilisi allkirju. Seda sertifikaati kasutatakse süsteemi

sisemiselt tõendamaks, et allkirjastatud fail on tulnud asutuse infosüsteemist ning faili ei ole vahepeal muudetud.

- Elliptiliste kõveratega softsert (edaspidi kõveratega softsert)– sertifikaat, mis kasutab elliptiliste kõverate krüptoalgoritmi (ECC [13]). Uue põlvkonna digiallkirjastamise lahendus, kus SHA-2 perekonna asemel kasutatakse elliptilisi kõveraid [14]. Praegusel hetkel igapäevaselt kõveratega softserti ei kasutata, kuid testitakse tarkvara valmidust sellele. Testimise jaoks on loodud spetsiaalne faili kujul sertifikaat, mis kasutab elliptiliste kõverate krüptoalgoritmi.

- Digitaalne tempel – ettevõtetele mõeldud digitaalse templi sertifikaat, mida väljastatakse Sertifitseerimiskeskuse poolt Aladdin'i krüptopulgal (edaspidi lihtsalt krüptopulk).

ID-kaardi tarkvara korrektseks töötamiseks elektrooniliselt on tarvis omada kaardilugejat. ID tarkvara seisukohalt testitud lugejate nimekiri on kätte saadav ID.ee lehel [15]. Kaardilugejate puhul on olulised ka draiverid, millega seda kaardilugejat kasutatakse.

Kuna ID-kaardi kasutamise sihtgrupp on suur, siis on tarvis tagada ID tarkvara korrektset töötamist mitmel operatsioonisüsteemil. ID-kaardi tarkvara peab toimima kõikidel enamlevinud operatsioonisüsteemidel nagu [16]:

- Windows 7 SP1 32bit ja 64bit,
- Windows 8 32bit ja 64bit,
- Windows 8.1 32bit ja 64bit,
- Mac OS X 10.7, 10.8, 10.9
- Ubuntu 12.04 32bit ja 64bit, Ubuntu 13.10 32bit ja 64bit,
- Windows Server 2008 R2*3,
- Windows Server 2012

Eelnevalt oli operatsioonisüsteemide skoobis lisaks eelmainitutele veel Windows XP ning Windows Vista. Lisaks oli testimise skoobis olnud Fedora ning OpenSuse. Erinevate põhjuste tõttu on need operatsioonisüsteemid testimise skoobist välja võetud.

On olemas mitmeid veebiteenuseid, kus on võimalik kasutada ID-kaarti, seega on tarvilik toetada ka brausereid. Erinevatel operatsioonisüsteemidel võivad nad olla veidi erinevad, seega toetatud brauserid operatsioonisüsteemide lõikes on järgmised:

- Windows: Internet Explorer, Mozilla Firefox, Google Chrome
- Mac OSX: Mozilla Firefox, Safari, Google Chrome
- Linux: Mozilla Firefox, Google Chrome

Kokkuvõtteks osad, mis otseselt ei ole ID-kaardi tarkvara osad, kuid mida on vaja arvestada tarkvara arendamisel ja testimisel on järgmised:

- Kaardid / asutuse sertifikaadid
- Lugejad ja draiverid
- Operatsioonisüsteemid
- Brauserid

2.1.4 Olemasolev teekide testimise raamistik ja kasutatavad vahendid

Teekide testimiseks kasutatakse *Ruby*'s kirjutatud teste *RSpec* vahendi abil. Need testid praguseks jooksevad iga öö pärast arenduse paki koostamist. Testide eesmärk on kontrollida teekide funktsionaalsuste korrektse toimimine, lisaks testitakse teekide vahelist ristkasutust ning operatsioonisüsteemide vahelist ristkasutust. Teekide automaattestidega verifitseeritakse teekide vastavust nõuetele. Kuna põhiliselt töötavad teegid failidega, siis testides arvestatakse ka erinevaid faile, näiteks erimärkidega faile ning erinevas kodeeringus faile. Lisaks sellele automaattestidega kontrollitakse ka tarkvara korrektset töötamist tarkvara eelmiste versioonidega loodud failidega ja ka uue tarkvara versiooniga loodud failide korrektset töötamist eelmiste tarkvara versioonidega. Seega automaattestidega kaetud skoop on väga suur. Kuna ajalooliselt on automaatteste pidevalt arendatud, on nüüdseks testide kood muutunud väga pikaks. Teekide testide jooksutamisele kuluv aeg on pikk ja struktuuriliselt on testid

muutunud kohmakateks ja raskesti edasi arendatavateks. Üks test võib aega võtta 5-6 tundi ning üks test võib sisaldada väga palju erinevaid tegevusi. Testidega kaetud teekide skoop ja osa, mida testid tegelikult ei kata on suhteliselt hägune, kuna algsetesse testidesse oli lisatud palju uut funktsionaalsust ning nüüdseks enam ei teata, mis teegi funktsionaalsust testitakse ja mida tegelikult ei testita. Lisaks testiraamistik oli ehitatud umbes 5 aastat tagasi ning nii *Ruby* versioon, kui ka raamistikus kasutatav *RSpec*'i versioon on vananenud ning vajavad uuendamist. Teekide kaetus testidega vajab kaardistamist ja on tarvilik välja selgitada kohad, mis ei ole testidega kaetud, selleks et oleks võimalik realistlikult hinnata riske. Lisaks ekspertide hinnangul on tarvilik kogu testimise struktuur ümber korraldada. Näiteks kõik testid hetkel kasutavad erimärkidega faile kas faili nimes või faili sisus. Erinevate failide käsitlemist võib võtta kui eraldiseisvat testi ning muuta seeläbi üldist teegi funktsionaalsust testitavat automatiseeritud testi kiiremaks.

Automaattestimise skoop oli esialgselt koostatud eksperthinnangu alusel ning seejärel arendati testimise skoopi vajaduspõhiselt ning ilma konkreetse plaanita. Tihti võeti uus test skoopi selle pärast, et oli leitud tarkvaras viga ning selleks, et see viga ei korduks, oli otsustatud lisada see test või oodatavate tulemuste kontroll automaattestide skoopi. Praeguseks on seetõttu tekkinud palju oodatavate tulemuste kontrolle või isegi iseseisvaid teste, mis on loodud veaolukorra vältimiseks, kuigi üldises pildis see olukord ei ole enam piisavalt oluline või selle vea taas tekkimise tõenäosus on väga väike.

2.2 Teegid

ID-kaardi tarkvara DigiDoc klient 3-s toimub allkirjastamine tehniliselt kasutades teeke. Tarkvara teekide abil on võimalik allkirjastada ning krüpteerida faile. Failide allkirjastamisel või krüpteerimisel tekitatakse kõigepealt konteiner, kuhu lisatakse fail või failid ning seejärel on võimalik konteineris olevaid faile allkirjastada või krüpteerida.

Järgnevalt on kirjeldatud täpsemalt ID-kaardi tarkvara teeke, nende funktsionaalsust, võimalusi ja kitsendusi.

ID-kaardi projekti raames arendatakse mitut teeki, nendeks on [17]:

- multiplatvormne C-teek – „DigiDoc C-teek võimaldab koostada, allkirjastada ja verifitseerida DigiDoc faile.

Windowsi platvormil on võimalik allkirjastamiseks kasutada CryptoAPI-t ja PKCS#11 moodulit, teistel platvormidel ainult PKCS#11 moodulit. Lisaks digiallkirjastamisele pakub DigiDoc C-teek ka krüpteerimist ja dekrüpteerimist vastavalt XML-ENC standardile.“ [18] Seega C-teegi poolt toetatud DigiDoc failivormingud on DDOC, CDOC.

- CPP teek (*Libdigidocpp*)- „Alates ID tarkvara versioonist 3.8 lisandub arendajatele ja integraatoritele kasutamiseks mõeldud teekide hulka uus multiplatvormne libdigidocpp teek. Antud teek on mõeldud asendama alates 2003. aastast kasutusel olnud libdigidoc C-teeki ja Windowsi COM teeki, mille iseseisva kasutamise tugi kaob 2015. aastal. *Libdigidocpp* teek on olnud ID-tarkvaras kasutusel alates 2010. aastast, kuid see ei ole seni olnud mõeldud integratsiooniks kolmandate osapoolte rakenduste ja infosüsteemidega“ [19]. *Libdigidocpp* teegi poolt toetatud faili vormingud on BDOC ja DDOC.

- *JDigiDoc* teek on Java rakenduste tarbeks. „Antud teek pakub funktsionaalsust DIGIDOC-XML 1.3 ja BDOC 2.1 formaadis digitaalselt allkirjastatud failide loomiseks, lugemiseks, allkirjastamiseks, kehtivuskinnituse hankimiseks ja allkirjade ning kehtivuskinnituste kontrolliks. Lisaks digiallkirjastamisele pakub *JDigiDoc* ka krüpteerimist ja dekrüpteerimist vastavalt XML-ENC standardile.“ Seega *JDigiDoc* teek toetab DDOC, BDOC ja CDOC faili vorminguid.

- COM teek on Windowsi platvormil kasutamiseks. „DigiDoc COM on DigiDoc C-teegil baseeruv lisakiht, mis võimaldab lihtsalt Windowsi platvormi rakendustesse DigiDoc funktsionaalsust integreerida. DigiDoc COM teeki kasutades on võimalik ise allkirjastamise ja digitaalallkirjade verifitseerimise funktsionaalsust realiseerida näiteks Visual Basicus. COM teek, nagu ka DigiDoc C-teek on ühilduv .NET platvormiga.“ [20] DigiDoc COM teegi abil on võimalik allkirjastada DDOC konteinereid.

- *NDigiDoc* teek on krüpteerimiseks ja dekrüpteerimiseks. „*NDigiDoc* on .NET raamistikul põhinev tarkvarateek, mis pakub andmefailide krüpteerimise ja dekrüpteerimise funktsionaalsust vastavalt XML-ENC standardile ning võimaldab mainitud funktsionaalsust lisada Windowsi platvormil realiseeritud rakendustesse.

> Krüpteerimisoperatsioonid on toetatud ainult tarkvarapõhiste PKCS#12 sertifikaatidega, krüpteeritud failid luuakse ENCDOC-XML failivormingus.“ [21] *Ndigidoc* teegi abil on võimalik luua CDOC failivormingus konteinereid.

Funktsionaalsused, mida iga teek toetab, on mõnevõrra erinevad. Kõige laialdasema funktsionaalsusega on C-teek ja *JDigiDoc* teek.

Nagu oli eelnevalt mainitud, on teekidega võimalik teostada faili allkirjastamist. Allkirjastamise läbiviimiseks kasutatakse järgnevaid abifunktsioone:

- DDOC või BDOC failivormingus konteineri loomine.
- Andmefaili(de) lisamine konteinerisse.
- Konteineri allkirjastamine, kusjuures allkirjastamisel teostatakse OCSP päring.

Failide krüpteerimine koosneb samuti mitmest tegevusest. Lisaks on võimalik faili krüpteerimist teostada mitut moodi. Esiteks on võimalik fail enne krüpteerimist paigaldada konteinerisse. DigiDoc Krüpto 3 kasutab just seda meetodit faili krüpteerimisel. Teiseks on võimalik fail krüpteerida nii, et seda ei paigaldata eraldi konteinerisse. Vaikimisi enne krüpteerimist pakitakse algne fail kõigepealt kokku ning seejärel alles krüpteeritakse, kuid on alles jäetud võimalus faili kokku pakkimine enne krüpteerimist välja lülitada.

Automaattestimise raamistiku testides kasutatakse vana ID-kaarti ning aastal 2012 väljastatud ID-kaarti, kuid testiskoobi seisukohalt ei tehta nendel kaartidel vahet. Põhjus seisneb selles, et teegi jaoks mängib rolli see, mis moodulit kasutatakse. Antud juhul kõik ID-kaardid kasutavad sama PKCS#11 moodulit. Erinevate ID-kaartide toimimise eest vastutavad draiverid ning draiverite testimine teostatakse teekide testimisest eraldi. Automaattestimise raamistikus eeldatakse, et draiver, mis suhtleb kaardiga, on toimiv. ID-kaartide seisukohalt on testides katmata Digi-ID ning krüptopulk, kuid nende puhul samuti kasutatakse PKCS#11 moodulit. Automaattestimise raamistikus tehakse vahet softserdil ning kõveratega softserdil, kuigi mõlemat kasutatakse PKCS#12 mooduliga. Eristamise põhjus on selles, et softserdi puhul kasutatakse SHA-2 perekonna algoritme ning kõveratega softserdi puhul kasutatakse elliptilisi kõveraid. Elliptiliste kõveratega krüptograafia rakendamine on uue põlvkonna digiallkirjastamise lahendus, kus SHA-2 perekonna asemel kasutatakse elliptilisi kõveraid. Praegusel hetkel igapäevaselt kõveratega softserti ei kasutata, kuid testitakse tarkvara valmidust sellele.

Lisaks veel mängib suurt rolli teegi konfiguratsioonifail, kus määratakse funktsionaalsuse kasutamise viis ning testis kasutatav allkirjastamise moodul (PKCS#11, PKCS#12).

2.2.1 Konfiguratsioonid

Teegi kasutamine ja käitumine on otseselt seotud teegi konfiguratsioonifailiga. Teegi vaikimisi konfiguratsioonifailist on võimalik luua mitu konfiguratsioonifaili koopiat väikeste muudatustega.

Kõik võimalikud konfiguratsiooni muudatused on järgmised:

Tabel 1. Konfigureeritavad parameetrid

	Konfiguratsiooni muudatus	**Selgitus**
1.	Allkirjastamise implementatsioon DIGIDOC_SIGN_IMPL=ee.sk.digidoc.factory.Pkcs12SignatureFactory	Konfigureeritav parameeter näitab, millist moodulit on vaja testis kasutada ning kust lugeda sertifikaat. Näiteks ID-kaardi kasutamisel tuleb sertifikaat lugeda otse kaardilt, selleks kasutatakse PKCS#11 moodulit. Softserdi puhul aga on tarvilik, et sertifikaat loetakse failist ning selleks kasutatakse PKCS#12 moodulit.
2.	Konteineri kokku pakkimine DENC_COMPRESS_MODE=1	Konteinerit krüpteerides pakitakse kõigepealt andmefail kokku. Konfiguratsiooni parameeter võimaldab andmefaili kokku pakkimise välja lülitada. Edaspidi joonistes kasutatakse selle konfiguratsiooni tähistamiseks lühendit *no_zip*
3.	Allkirjastamisel kasutatav algoritm DIGIDOC_DIGEST_TYPE=SHA-1	Vaikimisi allkirjastamise algoritmiks kasutatakse SHA-256. Kuid on võimalik kasutada ka teisi allkirjade räsi – SHA-1, SHA-224, SHA-512

	DIGIDOC_DIGEST_TYPE=SHA-512 DIGIDOC_DIGEST_TYPE=SHA-224	
4.	OCSP päringu ja vastuse signatuuri räsi binaarse väärtuse ning räsialgoritmi identifikaatori kontroll. CHECK_OCSP_NONCE=1	OCSP päringu ja vastuse signatuuri räsi ning räsialgoritmi identifikaatori kontroll ühendab sertifikaadi kehtivuse kontrolli ja ajatembelduse teenuse. Selle tulemusena ei ole ajatempel vajalik.

Konfiguratsiooni parameetrid lisavad teegile kasutamise võimalusi. See tähendab, et neid konfiguratsioone tuleb samuti arvestada testiskoobi loomisel ja teekide funktsionaalsuste skoobi kaardistamisel.

2.2.2 JDigiDoc

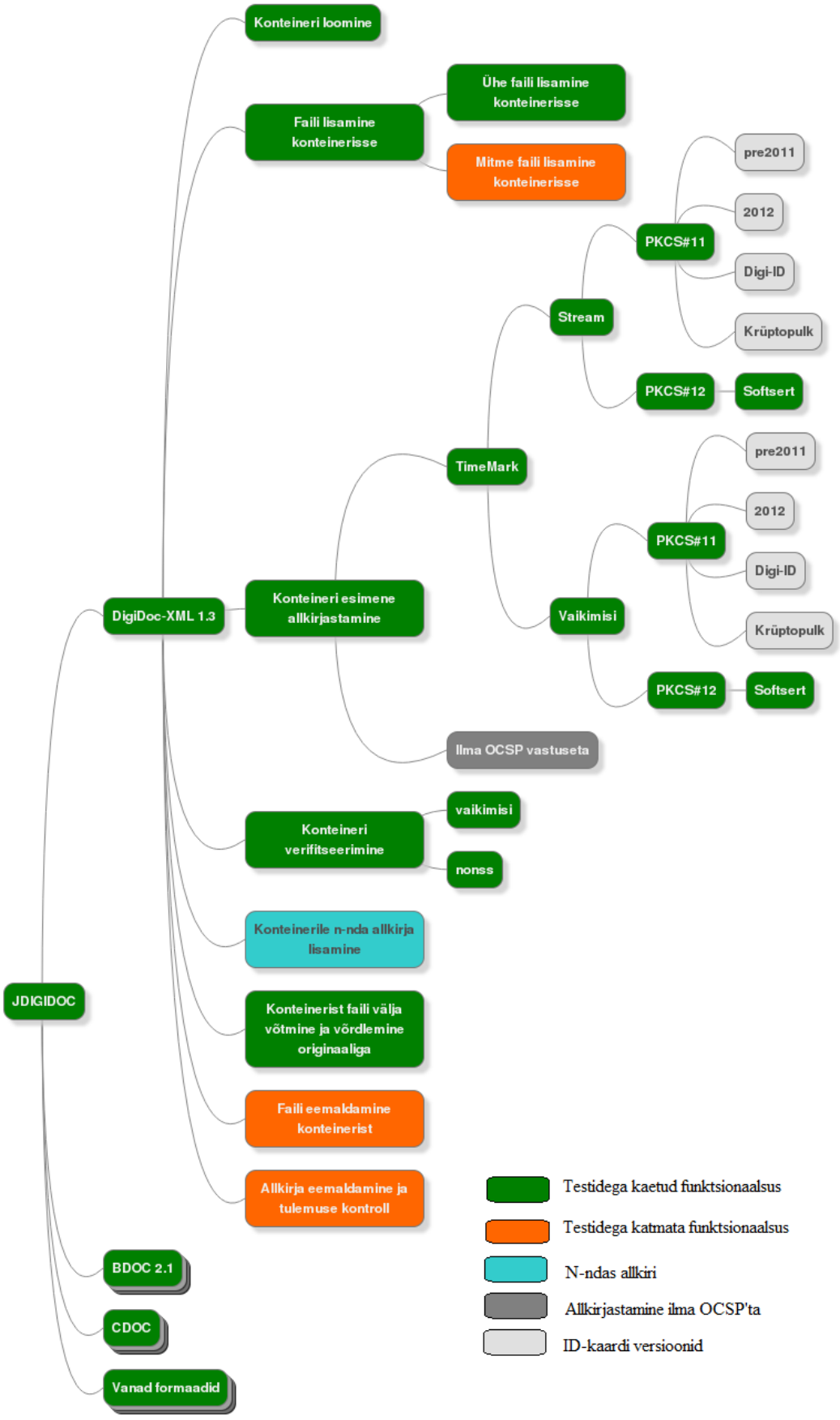
JDigiDoc [22] teek on loodud Java rakenduste tarbeks. Käesolevas peatükis on kirjeldatud *JDigiDoci* funktsionaalsuse skoop. Arvestatud on nii teegi toetatud funktsionaalsust, konteineri tüüpe, konfigureeritavaid parameetreid ja kasutatavaid mooduleid.

Teegi funktsionaalsuse skoobi analüüsiga paralleelselt on teostatud automaattestimise raamistikus olevate testide ja testi skoobi analüüs. Peatükis on esitatud teegi funktsionaalsuse ja testidega kaetud skoobi võrdlus. Analüüsi tulemusena tuuakse välja kitsaskohad, mis ei ole praeguses kasutusel olevas testiraamistiku skoobis.

Järgnevalt on esitatud teegi skoobi joonised. Joonisel esitatakse, milliseid funktsionaalsusi teek toetab ning millised funktsionaalsuste ja konfigureeritavate parameetrite kombinatsioonid on võimalikud. Joonisel tuuakse värvidega välja teegi funktsionaalsused, mis on praegu kasutusel olevas automaattestimise raamistikus kaetud testidega ja millised ei ole. Lisaks on joonisel kujutatud teegi funktsionaalsused, mida on plaanis lisada teegile ning mida tuleks arvestada uue raamistiku disainimisel.

DigiDoc-XML 1.3

Jdigidoc funktsionaalsus DIGIDOC-XML 1.3 raames:



Joonis 2. *JDigiDoc* Digidoc-XML 1.3 funktsionaalsus

Joonis 2 näitab, et *jdigidoc*'i teek DIGIDOC-XML 1.3 formaadis toetab järgmiseid tegevusi:

1. Konteineri loomine
2. Failide lisamine konteinerisse
3. Konteineri allkirjastamine
4. Konteineri verifitseerimine
5. Failide konteinerist välja võtmine
6. Allkirja eemaldamine

DDOC konteineri puhul kinnitatakse digiallkiri OCSP teenuse abil, kus kasutatakse OCSP standardi protokolli standardilaiendust nimega „nonss“ (nonce). OCSP kinnituse täpsem kirjeldus on leitav dokumendis „SK ajatempliteenuse ajatembelduspõhimõtted“ versioon 2.0 [1]. Joonisel on see tähistatud terminiga „*time-mark*“.

Jdigidoc teek toetab lisaks ka allkirjastamist ilma OCSP vastuseta, kuid sellist konteinerit käsitletakse kui mittevaliidset konteinerit ning lõputöö raames käsitletakse eraldi failide peatükis 2.2.7. Joonisel on see osa märgitud tumehalliks.

Konteinerit on võimalik allkirjastada mitut moodi. Erinevus seisneb faili sisse lugemise viisis. Ühel juhul loetakse andmefail korraga mällu (vaikimisi), teisel juhul loetakse fail sisse sisendvoona (joonistel kasutatakse terminit „*stream*“).

Allkirjastatud konteinerit on võimalik verifitseerida kasutades vaikimisi konfiguratsiooni või kasutades lisakontrolli nimega „nonss“ (nonce). Kasutades „nonss“ verifitseerimist kontrollitakse OCSP päringu ja vastuse puhul ka allkirja räsi ning räsialgoritmi identifikaatorit. Need väärtused peavad nii OCSP päringus kui ka vastuses olema samad.

Joonisel on rohelisega märgitud funktsionaalsused, mis on automatiseeritud raamistikus kaetud testidega ning oranžiga on märgitud need funktsionaalsused, mis ei ole testidega kaetud.

JDigiDoc'i DigiDoc-XML 1.3 formaadi funktsionaalsuste analüüs ja automatiseeritud testiraamistiku testide analüüs näitas, et praegusel hetkel DigiDoc-XML 1.3 formaadis testitakse järgmiseid teegi funktsionaalsusi:

1. Konteineri loomine
2. Konteinerisse ühe faili lisamine
3. Konteineri allkirjastamine nii vaikimisi faili sisse lugemisega, kui ka faili sisse lugemisega *streamina* kasutades nii PKCS#11 kui ka PKCS#12 moodulit.
4. Konteineri verifitseerimine
5. Konteinerist faili välja võtmine ja võrdlemine originaaliga

Automatiseeritud testiraamistiku praegusel hetkel DigiDoc-XML 1.3 formaadis ei testita järgmiseid teegi funktsionaalsusi:

- Mitme faili konteinerisse lisamise ja allkirjastamise funktsionaalsus
- Faili konteinerist eemaldamise funktsionaalsus
- Allkirja eemaldamise funktsionaalsus

Pärast kõiki neid tegevusi konteiner peab jääma valiidseks.

Konteinerile n-nda allkirja lisamine on joonisel märgitud siniseks. N-nda allkirja lisamisel tuleb arvestada nii allkirjastamise mooduli kasutamise järjekorraga, kui ka allkirjastamisel faili sisse lugemise viisiga. Järgnevalt on esitatud

Tabel 2 allkirja lisamise ristkasutusest nii ühe teegi raames, kui ka etteruttavalt on esitatud allkirjastamise ristkasutus teiste ühilduvate teekidega.

Tabelis on näidatud võimalik ristkasutus juhul, kui esimene allkiri oli loodud softserdiga kasutades PKCS#12 moodulit, kuid testimise skoobis sellist võimalust ei arvestada, kuna softserdiga loodud allkiri ei ole Digitaalallkirja seaduse mõttes kehtiv. Lisaks võib eksperthinnangu alusel öelda, et sellisele olukorrale puuduvad reaalsed kasutusjuhud, kus pärast softserdiga allkirjastamist lisatakse teine allkiri. Tabelis on see osa märgitud halliks.

Rohelisega on esitatud ristkasutuse osad, mis on kaetud testidega praeguses automaattestimise raamistikus ja oranžiga need osad, mis ei ole praeguses raamistikus testidega kaetud.

Tabel 2. *JDigiDoc* allkirjastatud DDOCi ristkasutus

<table>
<tr><td colspan="6">Esimine allkiri JDIGIDOC</td></tr>
<tr><td colspan="6">JDIGIDOC teise allkirja lisamine</td></tr>
<tr><td colspan="3" rowspan="2">JDIGIDOC esimene allkiri</td><td colspan="2">Vaikimisi</td><td colspan="2">Stream</td></tr>
<tr><td>PKCS#11
ID-kaart</td><td>PKCS#12
Softsert</td><td>PKCS#11
ID-kaart</td><td>PKCS#12
Softsert</td></tr>
<tr><td rowspan="4">Teine allkiri JDIGIDOC</td><td>Vaikimisi</td><td>PKCS#11
ID-kaart</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>PKCS#12
Softsert</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Stream</td><td>PKCS#11
ID-kaart</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>PKCS#12
Softsert</td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="6">CDIGIDOC teise allkirja lisamine</td></tr>
<tr><td rowspan="4">Teine allkiri CDIGIDOC</td><td>Vaikimisi</td><td>PKCS#11
ID-kaart</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>PKCS#12
Softsert</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Mälus</td><td>PKCS#11
ID-kaart</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>PKCS#12
Softsert</td><td></td><td></td><td></td><td></td></tr>
</table>

LIBDIGIDOCPP teise allkirja lisamine						
JDIGIDOC esimene allkiri			Vaikimisi		*Stream*	
			PKCS#11 ID-kaart	PKCS#12 Softsert	PKCS#11 ID-kaart	PKCS#12 Softsert
Teine allkiri *LIBDIGIDOCPP*	PKCS#12	Softsert				
	PKCS#11	ID-kaart				
	CNG	ID-kaart				
DIGIDOC-COM teise allkirja lisamine						
Teine allkiri *DIGIDOC-COM*	Vaikimisi	PKCS#11 ID-kaart				

Tabel 2 näitab, et praeguses automaattestimise raamistikus testitakse ainult järgmisi osi:

1. *JDigiDoc* vaikimisi PKCS#11 mooduliga allkirjastatud konteinerile teise allkirja lisamine *JDigiDoc* teegiga kasutades PKCS#11 moodulit ja vaikimisi parameetreid.

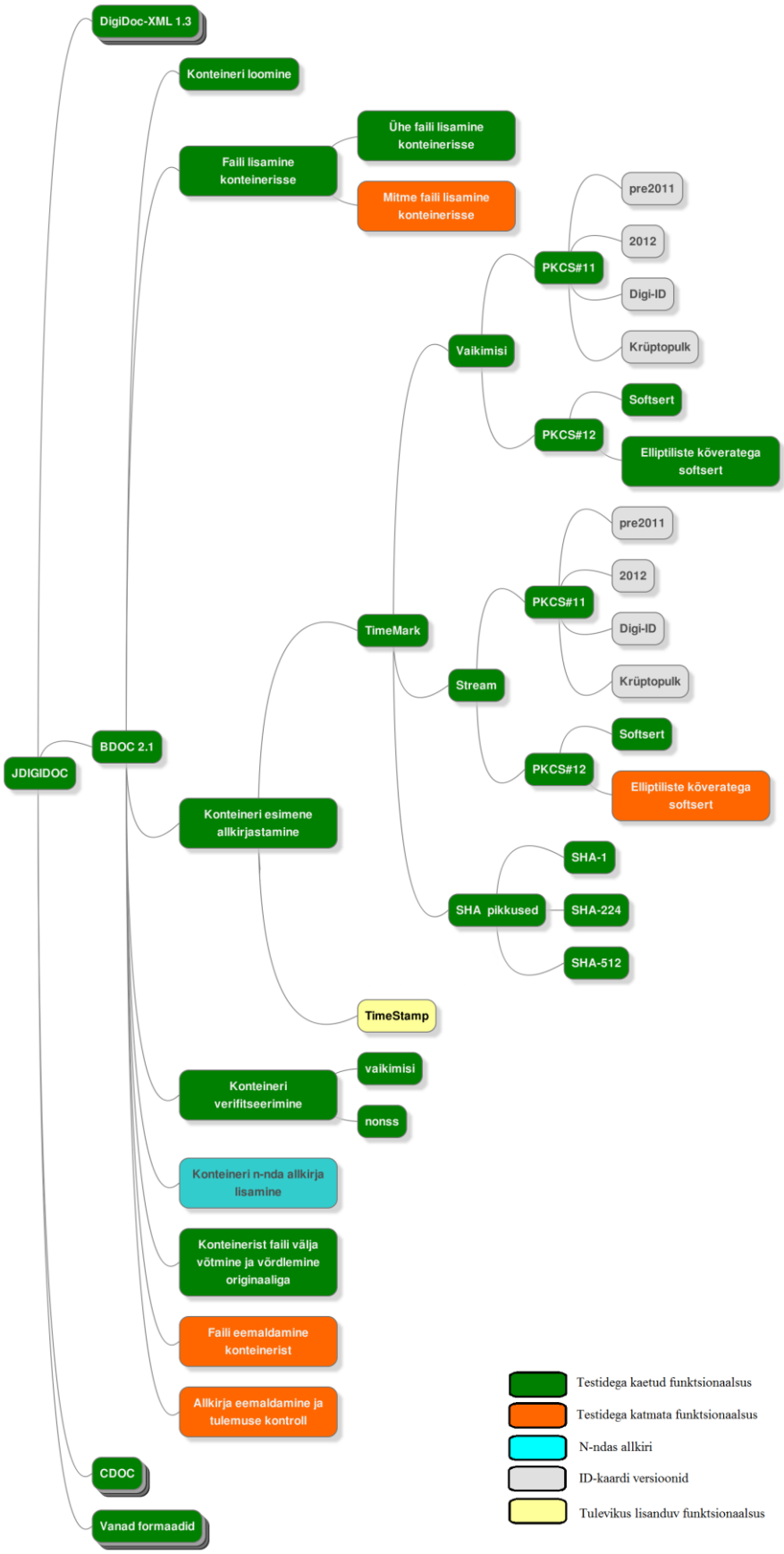
2. *JDigiDoc* vaikimisi PKCS#11 mooduliga allkirjastatud konteinerile teise allkirja lisamine *CDigiDoc* teegiga kasutades PKCS#11 moodulit ja vaikimisi parameetreid.

3. *JDigiDoc* faili *streamina* sisse lugemisega PKCS#11 mooduliga allkirjastatud konteinerile teise allkirja lisamine *CDigiDoc* teegiga kasutades PKCS#11 moodulit ja vaikimisi parameetreid.

Ülejäänud ristkasutuse võimalused on praeguses automatiseeritud raamistikus katmata.

BDOC 2.1

JDigiDoc funktsionaalsused BDOC 2.1 formaadi puhul toob välja Joonis 3:



Joonis 3. *JDigiDoc* BDOC 2.1 funktsionaalsus

BDOC 2.1 formaadi poole pealt on *JDigiDoc*'i funktsionaalsus sarnane võrreldes DigiDoc-XML 1.3-ga. Lisaks on BDOC konteinerit võimalik *JDigiDoc*'i abil luua ja allkirjastada kasutades kõveratega softserti.

BDOC „*time-mark*" korral allkirjastaja sertifikaadi kehtivus lisatakse allkirjale ning kehtivuse tõend saadakse OCSP vastusena. BDOC „*time-mark*" vastab XAdES LT taseme nõuetele.

Tulevikus on plaaneeritud implementeerida ka „*time-stamp*" profiili allkirjastamise tõendi hankimiseks. See osa on joonisel esitatud kollasega.

Uue profiili funktsionaalsust tuleb arvestada testide skoobi genereerimisel ja testide projekteerimisel. See osa on märgitud joonisel oranžiga.

BDOC'i puhul on nii testitud, kui ka testimata funktsionaalsused samad, mis ka Digidoc-XML 1.3 puhul. Need on märgitud joonisel samuti oranžiga. Lisaks aga *CDigiDoc* teegi funktsionaalsuse skoobi analüüs ja testidega kaetud funktsionaalsuse analüüsi tulemus näitas, et allkirja lisamisel kasutades andmefaili sisse lugemisel *stream* meetodit ei testita allkirja lisamist kõveratega softserdiga.

Joonisel on eraldi märgitud konteineri allkirjastamine, kus allkirja räsi on võimalik konfigureerida. Vaikimisi kasutatakse SHA-256 räsi algoritmi, kuid *JDigiDoc* teegi puhul on võimalik allkirjastada kasutades SHA-1, SHA-224, SHA-512 algoritme.

Nagu ka DDOC puhul, on võimalik konteiner verifitseerida vaikimisi konfiguratsioonidega kui ka kasutades lisakontrolli „nonss".

Järgnevalt on esitatud tabel võimalustest, kuidas saab *JDigiDoc* teegi poolt loodud BDOC konteinerile lisada teise allkirja nii *JDigiDoc* teegiga kui ka *libdigidocpp* teegiga. (See funktsionaalsus on joonisel märgitud siniseks.) Tabel 3-s on rohelisega näidatud osad, mis on praeguses testiraamistikus kaetud testidega ning oranžiga esitatakse osad, mis ei ole praeguses automatiseeritud raamistikus testidega kaetud.

Tabelis on näidatud võimalik ristkasutus juhul, kui esimene allkiri oli loodud softserdiga kasutades PKCS#12 moodulit, kuid testimise skoobis sellist võimalust ei arvestada, kuna softserdiga loodud allkiri ei ole Digitaalallkirja seaduse mõttes kehtiv. Tabelis on see märgitud halliks.

Tabel 3. *JDigiDoc* allkirjastatud BDOCi ristkasutus

Esimene allkiri *JDIGIDOC*									
JDIGIDOC teise allkirja lisamine									
JDIGIDOC esimene allkiri			Vaikimisi			*Stream*			
			PKCS#11 ID-kaart	PKCS# 12 Softsert	PKCS#12 Kõveratega softsert	PKCS#11 ID-kaart	PKCS# 12 Softsert	PKCS#12 Kõveratega softsert
Teine allkiri *JDIGIDOC*	Vaikimisi	PKCS#11 ID-kaart						
		Softsert						
		Kõveratega softsert						
	Stream	PKCS#11 ID-kaart						
		Softsert						
		Kõveratega softsert						

LIBDIGIDOCPP teise allkirja lisamine									
JDIGIDOC esimene allkiri			Vaikimisi			*Stream*			
			PKCS#11 ID-kaart	PKCS# 12 Softsert	PKCS#12 Kõveratega softsert	PKCS#11 ID-kaart	PKCS# 12 Softsert	PKCS#12 Kõveratega softsert
Teine allkiri *LIBDIGIDOCPP*	PKCS #12	Softsert						
		Kõveratega softsert						
	PKCS #11	ID-kaart						
	CNG	ID-kaart						

Tabel 3 näitab, et praeguses automatiseeritud raamistikus on kaetud järgmised allkirjastamise ristkasutuse võimalused:

1. *JDigiDoc* teegiga vaikimisi parameetritega PKCS#11 allkirjastatud BDOC failile lisatakse *JDigiDoc* teegiga vaikimisi parameetritega teine allkiri PKCS#11 mooduliga.

2. *JDigiDoc* teegiga vaikimisi parameetritega PKCS#12 moodulit kasutades kõveratega softserdiga allkirjastatud BDOC failile lisatakse *JDigiDoc* teegiga vaikimisi parameetritega teine allkiri ID-kaardiga (PKCS#11 moodul).

3. *JDigiDoc* teegiga faili *streamina* sisse lugemisega ID-kaardiga (PKCS#11 moodul) allkirjastatud BDOC failile lisatakse *JDigiDoc* teegiga vaikimisi parameetritega teine allkiri ID-kaardiga(PKCS#11 moodul).

4. *JDigiDoc* teegiga vaikimisi parameetritega ID-kaardiga(PKCS#11 moodul) allkirjastatud BDOC failile lisatakse *libdigidocpp* teegiga vaikimisi parameetritega teine allkiri ID-kaardiga(PKCS#11 moodul).

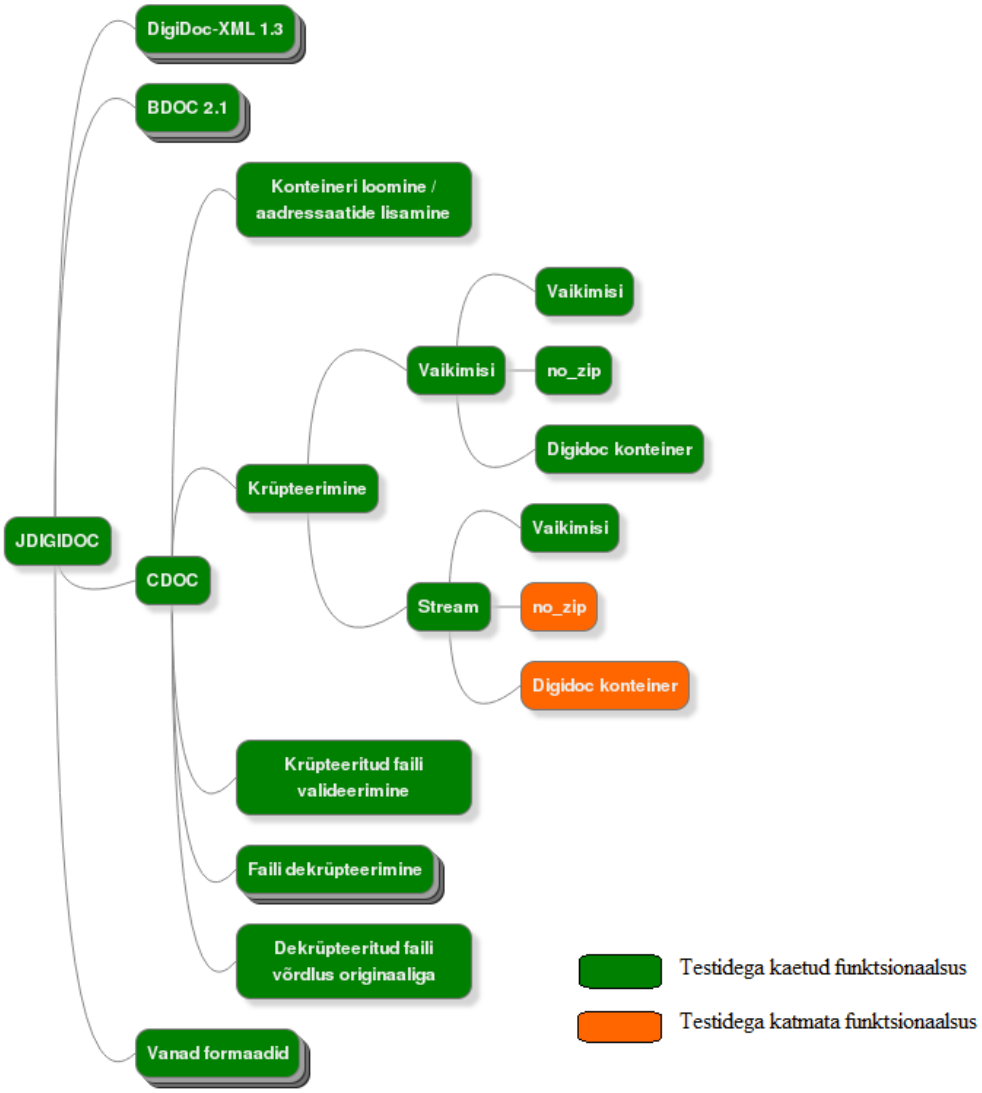
5. *JDigiDoc* teegiga vaikimisi parameetritega PKCS#12 moodulit kasutades kõveratega softserdiga allkirjastatud BDOC failile lisatakse *libdigidocpp* teegiga vaikimisi parameetritega teine allkiri ID-kaardiga(PKCS#11 moodul).

6. *JDigiDoc* teegiga faili *streamina* sisse lugemisega ID-kaardiga (PKCS#11 moodul) allkirjastatud BDOC failile lisatakse *libdigidocpp* teegiga vaikimisi parameetritega teine allkiri ID-kaardiga(PKCS#11 moodul).

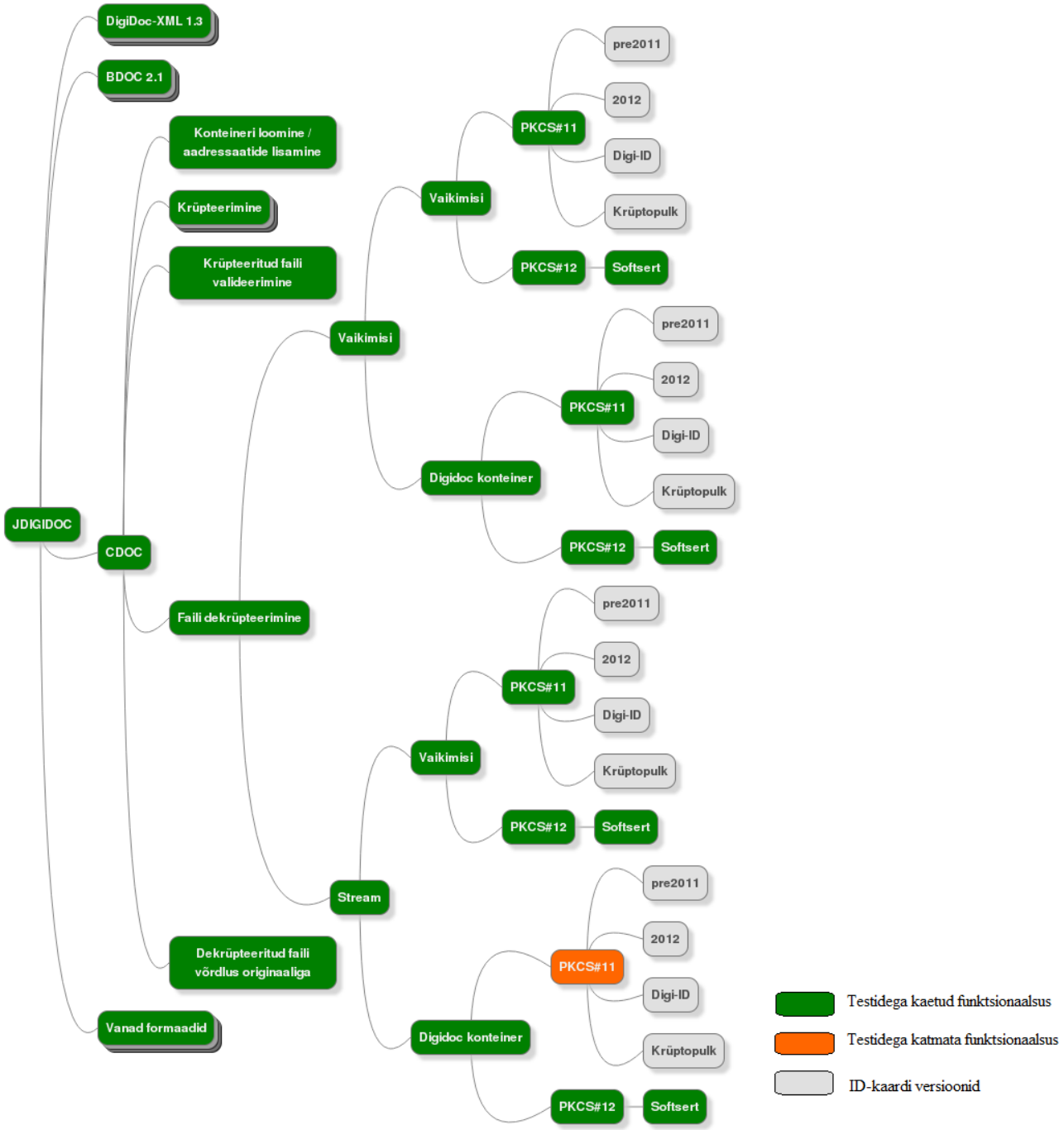
Ülejäänud ristkasutuse võimalused on praeguses raamistikus automaattestidega katmata.

CDOC

JDigiDoc funktsionaalsus CDOC raames on esitatud järgnevatel joonistel:



Joonis 4. *JDigiDoc* CDOC üldine funktsionaalsus



Joonis 5. *JDigiDoc* CDOC dekrüpteerimise funktsionaalsus

CDOC formaadi puhul toetab *JDigiDoc* järgmiseid tegevusi:

1. Konteineri loomist ja konteinerisse adressaatide lisamist
2. Konteineri krüpteerimist – nagu ka ülal toodud joonisel on näha, krüpteerimise võimalusi on mitmeid.

3. Konteineri dekrüpteerimist kasutades erinevaid dekrüpteerimise viise ja ID-kaarte või asutuse sertifikaate, millega konteiner dekrüpteeritakse.

4. Konteineri verifitseerimist

5. Faili välja võtmist konteinerist

Nagu eelnevalt on mainitud, krüpteerimise viise on *JDigiDoc*'is mitmeid. Neid jaotatakse vaikimisi krüpteerimiseks ja *streamina* krüpteerimiseks. Lisaks on mõlemal viisil võimalik krüpteerida:

1. Vaikimisi parameetritega (vaikimisi konfiguratsiooniga pakitakse algne andmefail enne krüpteerimist kokku)

2. Konteineri loomisega – sellisel juhul algne andmefail paigutatakse uude DigiDoc konteinerisse ning seejärel krüpteeritakse. Selline krüpteerimise viis on ühilduv teiste DigiDoc tarkvara komponentidega (joonisel on tähistatud „*konteiner*“ terminiga)

3. Ilma andmefaili eelneva kokku pakkimiseta (joonisel on tähistatud terminiga „*no_zip*“)

CDOC failivorminduse testidega kaetud funktsionaalsused:

1. Konteineri loomine / adressaatide lisamine

2. Konteineri krüpteerimine faili sisselugemisega korraga mällu. Joonisel toodud kõik võimalikud krüpteerimise viisid.

3. Konteineri krüpteerimine *streamina ning* vaikimisi parameetritega.

4. Konteineri krüpteerimine vaikimisi faili sisselugemise viisiga vaikimisi parameetritega nii PKCS#11 kui ka PKCS#12 moodulit kasutades ning konteineri dekrüpteerimine PKCS#11, PKCS#12 moodulit kasutades.

5. Konteineri dekrüpteerimine *streamina* vaikimisi parameetritega nii PKCS#11 kui ka PKCS#12 moodulit kasutades ning konteineri dekrüpteerimine PKCS#12 moodulit kasutades.

CDOC krüpteerimise ja dekrüpteerimise funktsionaalsuse ja testide skoobi analüüsi tulemus näitab, et testimise skoobist on välja jäänud järgmised funktsionaalsused:

- Krüpteerimine kasutades *stream* meetodit ning ilma sisendfaili kokku pakkimiseta enne krüpteerimist (*no_zip*)

- Krüpteerimine kasutades *stream* meetodit ning uue DigiDoc konteineri loomist (*konteiner*)

- Dekrüpteerimine PKCS#11 mooduliga kasutades *stream* meetodit juhul, kui krüpteeritud andmefail on enne krüpteerimist paigaldatud DigiDoc konteinerisse

Lähtuvalt mitmest sisendfaili krüpteerimise viisist, on võimalik sisendfail dekrüpteerida kasutades samu meetodeid. Lisaks dekrüpteerimisel tuleb veel arvestada erinevate moodulite kasutamisega, millega krüpteeritud konteiner dekrüpteeritakse. Krüpteerimisel ei oma moodulite kasutamine tähtsust, kuna krüpteerimisel lisatakse failidele adressaadid ning ei pöörduta ID-kaardi ega muude sertifikaatide poole.

Konteineri dekrüpteerimise viis sõltub sellest, kuidas oli teostatud krüpteerimine. Näiteks juhul, kui sisendfail on enne krüpteerimist paigaldatud DigiDoc konteinerisse, siis on võimalik see krüpteeritud fail dekrüpteerida ainult kasutades vastavat dekrüpteerimise viisi. Ühe teegi raames igale krüpteerimisviisile võib vastata mitu dekrüpteerimise viisi sõltuvalt dekrüpteerimise parameetritest või kasutatavatest moodulitest. Järgnevalt on esitatud krüpteerimismeetodite ja dekrüpteerimismeetodite maatriks (vt. Tabel 4).

Etteruttavalt on tabelis esitatud ka teiste teekide dekrüpteerimise viisid – *CDigiDoc* ja *NDigiDoc* teekide dekrüpteerimise viisid. Tabel näitab, milliste dekrüpteerimisviiside abil ja milliseid mooduleid kasutades on võimalik dekrüpteerida fail, mis oli krüpteeritud *JDigiDoc* teegi abil kas vaikimisi parameetritega, ilma andmefaili lisanduva kokku pakkimiseta enne krüpteerimist, andmefaili paigaldamisega Digidoc konteinerisse enne krüpteerimist ning kõigi nende kolme viisi kasutades faili sisse lugemisega *streamina*.

Tabelis on sümboliga „+“ märgitud osad, kus vastava krüpteerimisviisi korral on dekrüpteerimine võimalik ning „-“ tähendab, et antud krüpteerimise korral ei ole võimalik seda faili dekrüpteerida kasutades tabelis märgitud dekrüpteerimise viisi. Tabelis on rohelisega märgitud alad, mis on kaetud praegu automaattestimise raamistikus olevate testidega ning oranžiga on märgitud osad, mis testimise raamistikus on katmata.

Tabel 4. *JDigiDoc* krüpteerimise ristkasutus

				Krüpteerimine					
JDIGIDOC									
Faili sisse lugemise viis				Vaikimisi			Stream		
				Vaikimisi	no_zip	Digidoc konteiner	Vaikimisi	no_zip	Digidoc konteiner
JDIGIDOC dekrüpteerimine	Vaikimisi	PKCS#11 ID-kaart	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
		PKCS#12 softsert	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
	Stream	PKCS#11 ID-kaart	Default	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
		PKCS#12 softsert	Default	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+

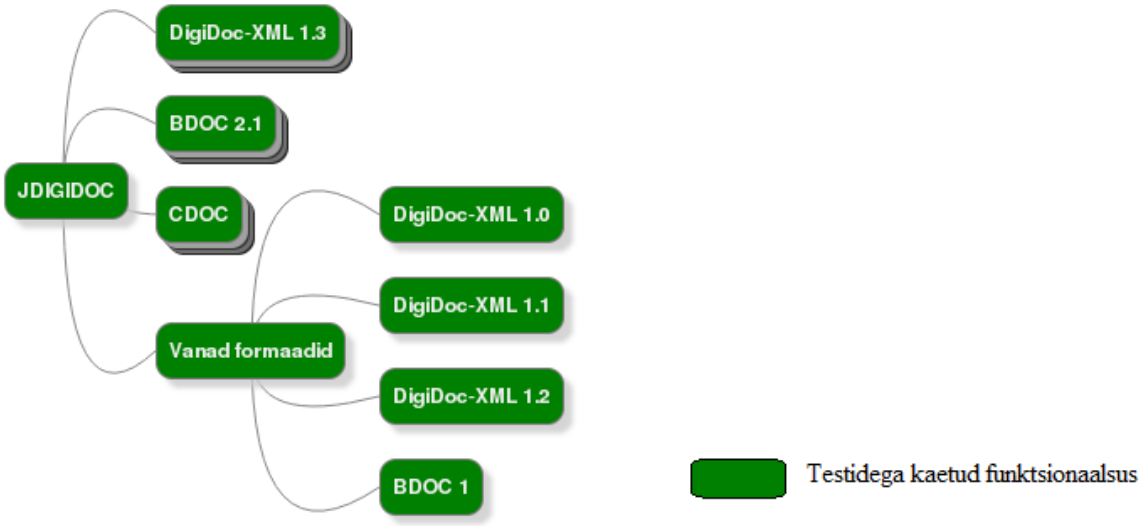
CDIGIDOC									
Faili sisse lugemise viis				Vaikimisi			Stream		
				Vaikimisi	no_zip	Digidoc konteiner	Vaikimisi	no_zip	Digidoc konteiner
CDIGIDOC dekrüpteerimine	Vaikimisi	PKCS#11 ID-kaart	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
		PKCS#12 softsert	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
	Mälus	PKCS#11 ID-kaart	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
		PKCS#12 softsert	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
NDIGIDOC									
	Vaikimis	PKCS#12 softsert	Vaikimisi	+	+	-	+	+	-

Nagu tabelist näha võib, on ühel krüpteeritud konteineril mitu võimalikku viisi, kuidas teda dekrüpteerida ühe teegi raames. Lisaks on võimalik *JDigiDoc* teegi abil krüpteeritud konteinerit dekrüpteerida kasutades *CDigiDoc* ning *NDigiDoc* teegi dekrüpteerimisviise. See asjaolu tagab krüpteerimisfunktsionaalsuse ristkasutuse teekide vahel.

Tabeli koostamisel selgus, et mitmed dekrüpteerimise viisid on välja jäänud testimise skoobist. Seega uue testiraamistiku skoobi koostamisel tuleb arvestada nende võimalustega ning kaaluda nende funktsionaalsuste lisamist testimise skoopi.

Vanad formaadid

Lisaks DigiDoc-XML 1.3, BDOC 2.1 ja CDOC'ile, peab *JDigiDoc*'i teek toetama ka vanu konteineri formaate.



Joonis 6. *JDigiDoc* vanad formaadid

Teegi funktsionaalsuste hulka ei kuulu vanades formaatides konteinerite tekitamine, kuid teek peab suutma verifitseerida varasemate versioonidega konteinereid. Vastavalt konteineri versioonile peab teek kas teatama kasutajale, et vastav konteiner on vanas formaadis ning soovitama allkirjastada konteineris olevad dokumendid uuesti, luues uue konteineri ning kasutades uuemat failivormingut.

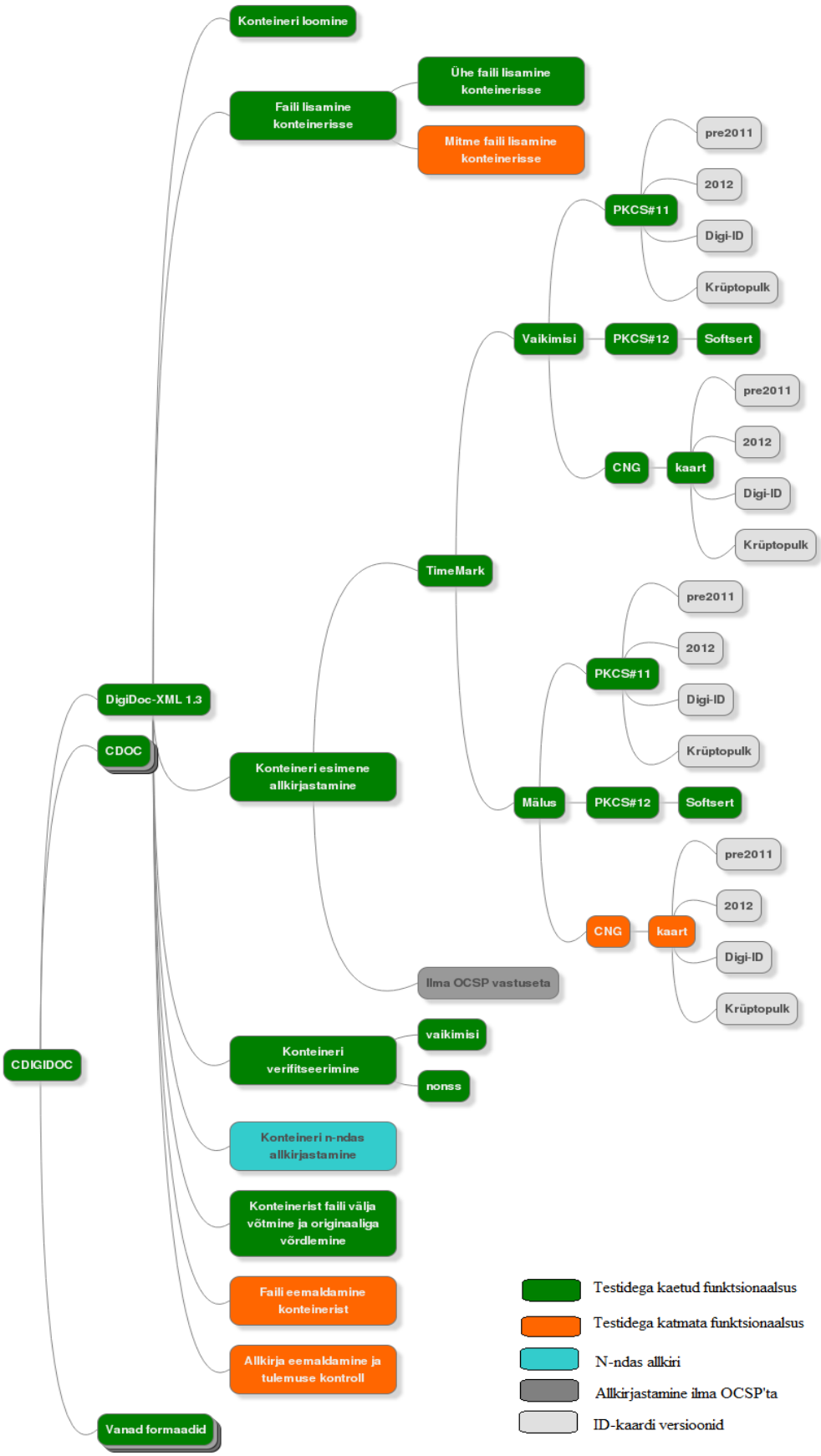
Vanade konteineri formaatide versioonidega loodud failide toetamist käsitletakse eraldi peatükis 2.2.7.

2.2.3 CDigiDoc

CDigiDoc'i [23] teek toetab DigiDoc-XML ja CDOC konteineri formaate. Funktsionaalsuse poole pealt on *CDigiDoc* teek sarnane *JDigiDoc*'ile.

DigiDoc-XML 1.3

DigiDoc-XML 1.3 raames on *CDigiDoc* teegi funktsionaalsuse skoop järgmine:



Joonis 7. *CDigiDoc* Digidoc-XML 1.3 funktsionaalsus

*CDigiDoc'*i puhul on võimalik allkirjastada konteinerit kasutades vaikimisi parameetreid ning lisaks on Windowsi keskkonnas võimalik konteinerit allkirjastada ID-kaardiga kasutades *CNG* (minidraiver) moodulit.

Faili on võimalik sisse lugeda kahte moodi – vaikimisi, mis tähendab, et käsu täitmisel võidakse vahefail ajutiselt maha kirjutada kettale, ning mälus, mis tähendab seda, et andmed loetakse mälu puhvrisse ning ajutisi faile ei tekitata.

Konteinerit on võimalik verifitseerida lisaks vaikimisi verifitseerimisele ka kasutades lisakontrolli „nonss".

CDigiDoc teegi testidega kaetud funktsionaalsused DDOC faili vormingus on järgmised:

1. Konteineri loomine
2. Loodud konteinerisse ühe faili lisamine
3. Konteineri allkirjastamine vaikimisi faili sisse lugemise viisiga kasutades nii PKCS#11, PKCS#12 kui ka CNG moodulit
4. Konteineri allkirjastamine faili korraga sisse lugemisega mällu kasutades nii PKCS#11 ja PKCS#12 moodulit
5. Konteineri verifitseerimine
6. Konteinerist faili välja võtmine ja võrdlus originaaliga

Nagu ka *JDigiDoc*'i puhul, on *CDigiDoc*'i DigiDoc-XML 1.3's testimata jäänud funktsionaalsused järgmised:

- Mitme faili lisamine konteinerisse ja selle konteineri allkirjastamine
- Faili eemaldamine konteinerist
- Allkirja eemaldamine
- Kasutades CNG moodulit automatiseeritud testidega ei testita allkirjastamist ID-kaardiga faili mällu sisse lugemisega.

Nagu ka *JDigiDoc*'i puhul, *CDigiDoc* 'i abil on võimalik luua konteinereid, mis ei sisalda OCSP vastust, kuid selliseid konteinereid käsitletakse kui mittevaliidseid ning käsitletakse eraldi peatükis 2.2.7.

N-da pealkirja puhul on allpool toodud ristkasutuse tabel, kus algfail oli loodud *CDigiDoc* teegiga ning teine allkiri lisatakse *JDigiDoc*, *CDigiDoc*, *libdigidocpp* ja *DigiDoc-COM* teekidega.
Tabelis (vt. Tabel 5) on rohelisega märgitud alad, mis on praeguses automatiseeritud raamistikus kaetud testidega ja oranžiga on märgitud alad, millele ei ole praeguses automatiseeritud raamistikus teste.

Tabelis on näidatud võimalik ristkasutus juhul, kui esimene allkiri oli loodud softserdiga kasutades PKCS#12 moodulit, kuid testimise skoobis sellist võimalust ei arvestata, kuna softserdiga loodud allkiri ei ole Digitaalallkirja seaduse mõttes kehtiv. Tabelis on see märgitud halliks.

Tabel 5. *CDigiDoc* allkirjastatud DDOCi ristkasutus

Esimine allkiri *CDIGIDOC*								
JDIGIDOC teise allkirja lisamine								
CDIGIDOC esimene allkiri			Vaikimisi			Mälus		
			PKCS#11 ID-kaart	PKCS#12 softsert	CNG ID-kaart	PKCS#11 ID-kaart	PKCS#12 softsert	CNG ID-kaart
Teine allkiri *JDIGIDOC*	Vaikimisi	PKCS#11 ID-kaart						
		PKCS#12 softsert						
	Stream	PKCS#11 ID-kaart						
		PKCS#12 softsert						

CDIGIDOC teise allkirja lisamine									
CDIGIDOC esimene allkiri			Vaikimisi			Mälus			
			PKCS#11 ID-kaart	PKCS#12 softsert	CNG ID-kaart	PKCS#11 ID-kaart	PKCS#12 softsert	CNG ID-kaart
2. allkiri *CDIGIDOC*	Vaikimisi	PKCS#11 ID-kaart						
		PKCS#12 softsert						
	Mälus	PKCS#11 ID-kaart						
		PKCS#12 softsert						

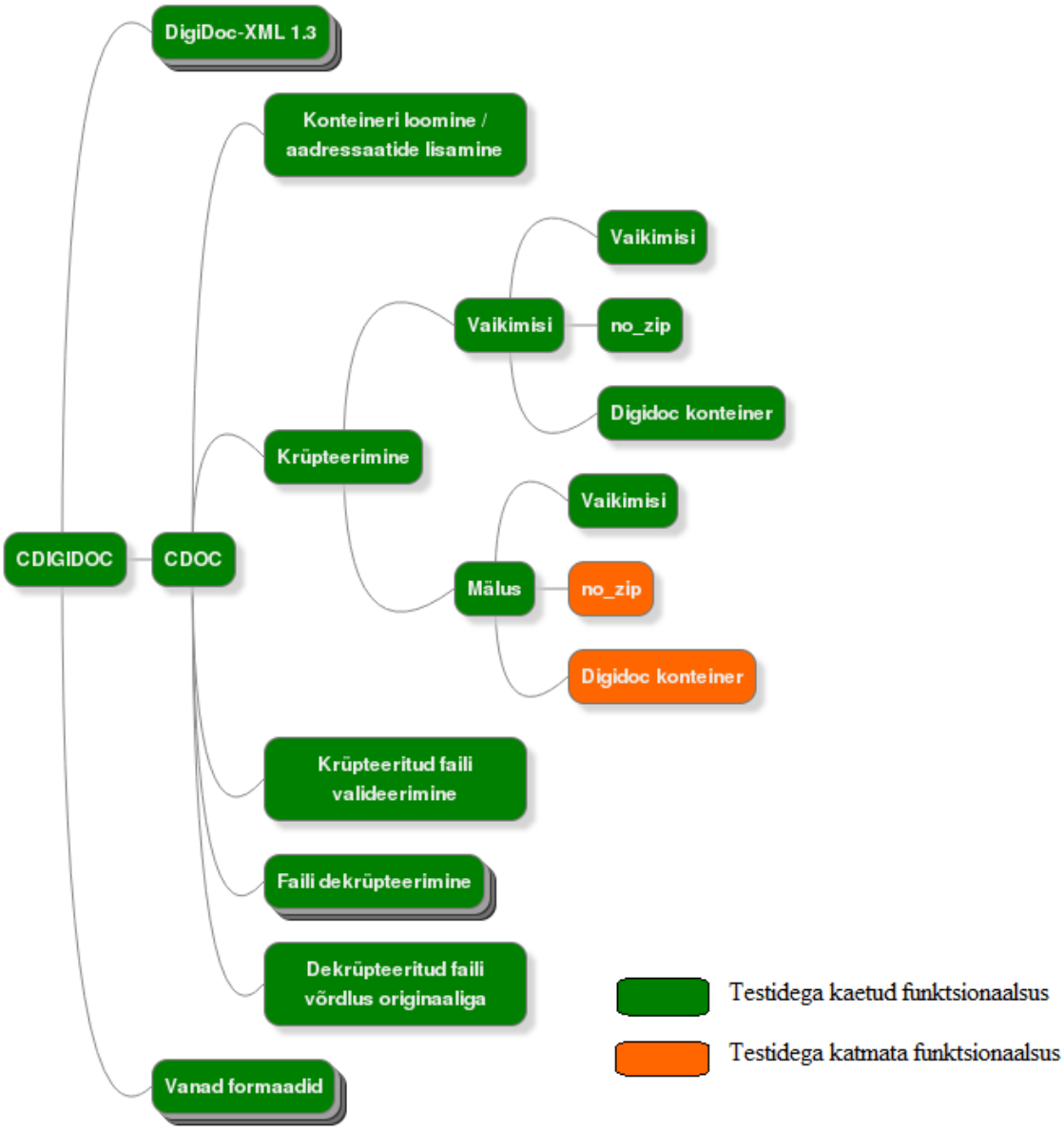
LIBDIGIDOCPP teise allkirja lisamine									
CDIGIDOC esimene allkiri			Vaikimisi			Mälus			
			PKCS#11 ID-kaart	PKCS#12 softsert	CNG ID-kaart	PKCS#11 ID-kaart	PKCS#12 softsert	CNG ID-kaart
2. allkiri *LIBDIGIDOCPP*	PKCS#12	Softsert						
	PKCS#11	ID-kaart						
	CNG	ID-kaart						

DIGIDOC-COM teise allkirja lisamine									
CDIGIDOC esimene allkiri			Vaikimisi			Mälus			
			PKCS#11 ID-kaart	PKCS#12 softsert	CNG ID-kaart	PKCS#11 ID-kaart	PKCS#12 softsert	CNG ID-kaart
DIGIDOC-COM	Vaikimisi	PKCS#11 ID-kaart						

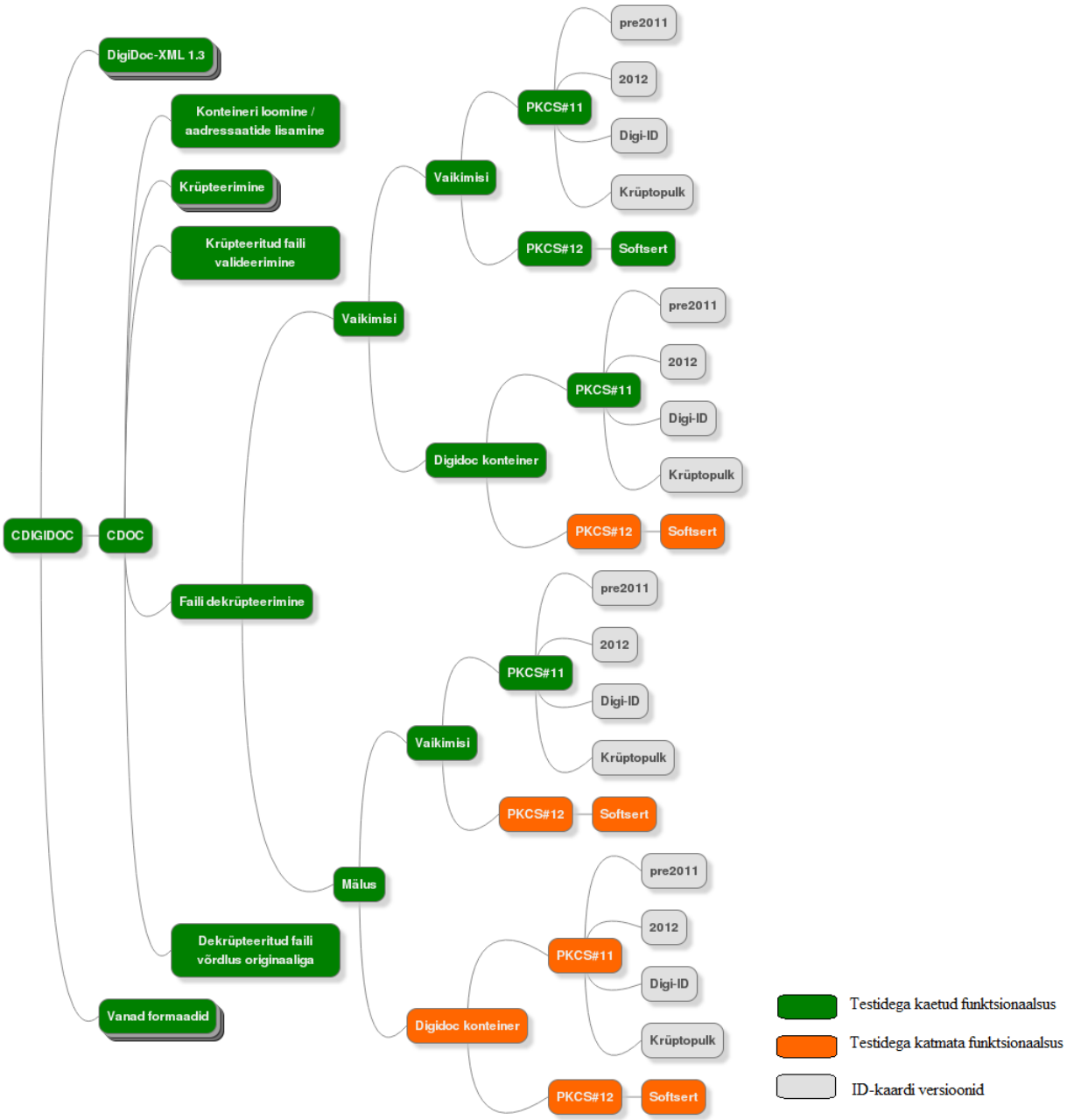
CDigiDoc allkirjastatud konteineri puhul lisatakse teine allkiri ainult *JDigiDoc* teegiga ID-kaardiga (PKCS#11 moodul) kasutades vaikimisi parameetreid ning *CDigiDoc* teegiga ID-kaardiga(PKCS#11 moodul) kasutades vaikimisi parameetreid.

CDOC

CDOC puhul *CDigiDoc*'i funktsionaalsust näitab Joonis 8. *CDigiDoc* CDOC üldine funktsionaalsus ning täpsemat dekrüpteerimise võimaluste funktsionaalsust näitab Joonis 9. *CDigiDoc* CDOC dekrüpteerimise funktsionaalsus:



Joonis 8. ***CDigiDoc*** **CDOC üldine funktsionaalsus**



Joonis 9. *CDigiDoc* CDOC dekrüpteerimise funktsionaalsus

CDigiDoc teegil on CDOC failivormingu puhul testidega kaetud järgmised funktsionaalsused:

1. Konteineri loomine / adressaatide lisamine
2. Faili krüpteerimine vaikimisi faili sisse lugemisega kõikide võimalikkude krüpteerimisviisidega
3. Faili krüpteerimine mälus faili hoidmisega vaikimisi krüpteerimise viisiga.

4. Faili dekrüpteerimine vaikimisi faili sisse lugemisega vaikimisi dekrüpteerimisviisiga

5. Faili dekrüpteerimine vaikimisi faili sisse lugemisega kasutades PKCS#11 moodulit, kus fail oli enne krüpteerimist paigaldatud konteinerisse

6. Faili dekrüpteerimine mälus hoidmisega vaikimisi dekrüpteerimisviisiga kasutades PKCS#11 moodulit (ID-kaart)

7. Dekrüpteeritud faili võrdlus originaaliga

CDigiDoc teegil on CDOC konteineri puhul testiskoobist väljas olnud järgmised funktsionaalsused:

- Krüpteerimine kasutades faili hoidmist mälus ning ilma andmefaili kokku pakkimiseta enne krüpteerimist (*no_zip*)

- Krüpteerimine kasutades faili hoidmist mälus ning uue DigiDoc konteineri loomist

- Dekrüpteerimine softserdiga (PKCS#12 moodul) juhul, kui krüpteeritud andmefail on enne krüpteerimist paigaldatud DigiDoc konteinerisse.

- Dekrüpteerimine softserdiga (PKCS#12 moodul) kasutades faili hoidmist mälus vaikimisi parameetritega

- Dekrüpteerimine ID-kaardiga (PKCS#11 moodul) kasutades faili hoidmist mälus juhul, kui krüpteeritud andmefail on enne krüpteerimist paigaldatud DigiDoc konteinerisse

- Dekrüpteerimine softserdiga (PKCS#12 moodul) kasutades faili hoidmist mälus juhul, kui krüpteeritud andmefail on enne krüpteerimist paigaldatud DigiDoc konteinerisse

Järgnevalt on esitatud tabel *CDigiDoc*'i võimalikest krüpteerimisviisidest ning *CDigiDoc'*i, *NDigiDoc*'i ja *JDigiDoc'*i dekrüpteerimise viisidest (vt. Tabel 6). Tabeli alguses on esitatud *JDigiDoc* dekrüpteerimise viisid, seejärel on *CDigiDoc* dekrüpteerimise viisid ning tabeli lõpus on *NDigiDoc* dekrüpteerimise viis. Selline tabel näitab krüpteerimiste ja dekrüpteerimiste ristkasutust nii teegi siseselt kui ka ristkasutust teekide vahel.

Tabelis on sümboliga „+“ märgitud osad, kus vastava krüpteerimisviisi korral on dekrüpteerimine võimalik ning „-“ tähendab, et antud krüpteerimise korral ei ole võimalik seda

faili dekrüpteerida kasutades tabelis märgitud dekrüpteerimisviisi. Tabelis on rohelisega märgitud alad, mis on kaetud praegu automaattestimise raamistikus olevate testidega ning oranžiga on märgitud osad, mis testimise raamistikus ei ole kaetud.

Tabel 6. *CDigiDoc* krüpteerimise ristkasutus

				Krüpteerimine *CDIGIDOC*					
JDIGIDOC dekrüpteerimine									
Faili sisse lugemise viis				Vaikimisi			Mälus		
				Vaikimisi	no_zip	Digidoc konteiner	Vaikimisi	no_zip	Digidoc konteiner
JDIGIDOC dekrüpteerimine	Vaikimisi	PKCS#11 ID-kaart	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
		PKCS#12 Softsert	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
	Stream	PKCS#11 ID-kaart	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
		PKCS#12 Softsert	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+

CDIGIDOC dekrüpteerimine									
Faili sisse lugemise viis				Vaikimisi			Mälus		
				Vaikimisi	no_zip	Digidoc konteiner	Vaikimisi	no_zip	Digidoc konteiner
CDIGIDOC dekrüpteerimine	Vaikimisi	PKCS#11 ID-kaart	Vaikimisi	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
		PKCS#12 Softsert	Default	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
	Mälus	PKCS#11 ID-kaart	Default	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
		PKCS#12 Softsert	Default	+	+	-	+	+	-
			Digidoc konteiner	-	-	+	-	-	+
NDIGIDOC dekrüpteerimine									
	Vaikimisi	PKCS#12 Softsert	Vaikimisi	+	+	-	+	+	-

Tabel 6 näitab, et mitmed dekrüpteerimise viisid on jäänud täielikult katmata ning mõned on kaetud osaliselt.

Nii juba testidega kaetud funktsionaalsust kui ka katmata jäänud funktsionaalsust tuleb analüüsida ning kaaluda uue raamistiku skoopi lisamist või skoobist välja jätmist. Juba testimise skoobis olev funktsionaalsus ja testide sisu vajab üle vaatamist, mis on vajalik selleks, et leida testides ülekattuvus, kui seda esineb. Samuti on tarvis skoobis olev funktsionaalsus üle vaadata, et tuvastada väiksema prioriteediga funktsionaalsused.

Vanad formaadid

Lisaks DigiDoc-XML 1.3 ja CDOC'ile, peab *CDigiDoc* teek suutma verifitseerida ka vanu konteineri formaate (vt. Joonis 10).

Joonis 10. *CDigiDoc* Vanad formaadid

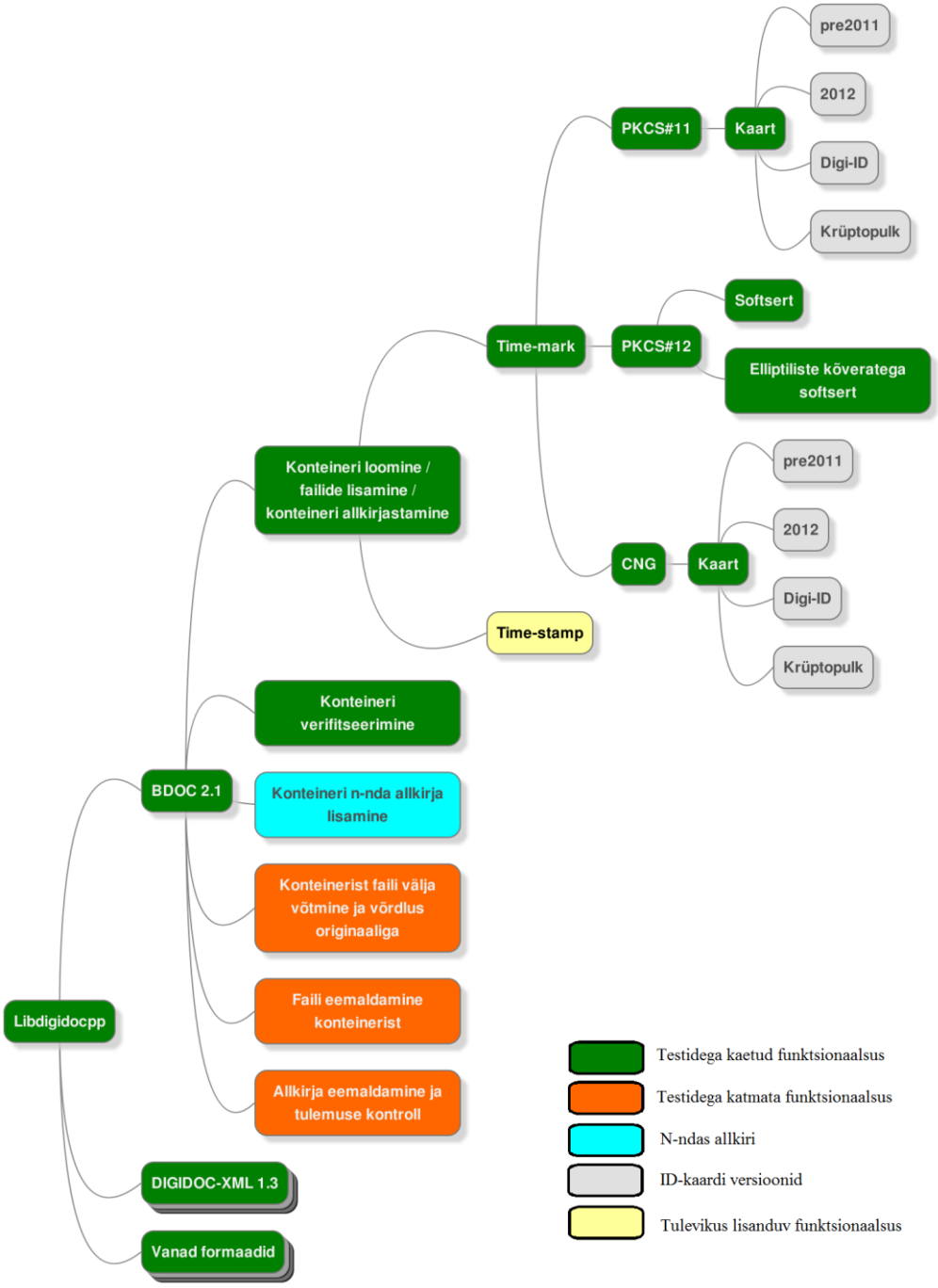
Vanade konteineri formaatide versioonidega loodud failide toetamist käsitletakse eraldi peatükis 2.2.7

2.2.4 Libdigidocpp

Libdigidocpp [24] toetab BDOC konteineri formaati ning mõningal määral ka Digidoc-XML 1.3 konteineri formaati. Digidoc-XML 1.3 puhul on *libdigidocpp* teegis olemas pakend (*wrapper*), mis kasutab tegelikult *CDigiDoc* teegi funktsioone. *Libdigidocpp* teek on tuntud ka *digidoc-tool* nime all.

BDOC 2.1

BDOC 2.1 raames *libdigidocpp* teegi funktsionaalsuse skoopi näitab Joonis 11. *Libdigidocpp* BDOC funktsionaalsus:



Joonis 11. *Libdigidocpp* BDOC funktsionaalsus

Libdigidocpp allkirja profiilina kasutatakse vaikimisi „*time-mark*“ profiili. Tulevikus on plaanis lisada ka „*time-stamp*“ profiili kasutamise võimalus.

Libdigidocpp allkirjastamisel on võimalik kasutada mitut allkirjaloomise moodulit - PKCS#11, PKCS#12 ja CNG. PKCS#11 moodulit kasutatakse allkirjastamisel ID-kaardiga Linux ja OSX keskkondades. *CNG* (minidraiver) on kasutusel ID-kaardiga allkirjastamisel Windows

keskkonnas. PKCS#12 moodulit kasutatakse allkirjastamisel softserdiga ning kõveratega softserdiga (ECC)

Libdigidocpp allkirjastamise viis erineb mõnevõrra *JDigiDoc* ja *CDigiDoc* teekide allkirjastamise viisidest. *Libdigidocpp* puhul on võimalik luua konteiner, lisada sinna sisendfailid ning allkirjastada see konteiner ühe tegevusena.

Libdigidocpp funktsionaalsuse koha pealt on praeguses automatiseeritud raamistikus kaetud testidega teegi järgmine funktsionaalsus:

1. Konteineri loomine, failide lisamine konteinerisse, konteineri allkirjastamine kasutades kõiki võimalikke mooduleid – PKCS#11, PKCS#12, CNG
2. Konteineri verifitseerimine

Libdigidocpp teegi puhul on testimise skoobist välja jäänud järgmised funktsionaalsused:

1. Konteinerist faili välja võtmine ja võrdlus originaaliga
2. Faili eemaldamine konteinerist
3. Allkirja eemaldamine ja tulemuse kontroll

Libdigidocpp teegi abil loodud BDOC konteinerit on võimalik teistkordselt allkirjastada järgmiste võimaluste ja teekide abil (Tabel 7. *Libdigidocpp* allkirjastatud BDOCi ristkasutus):

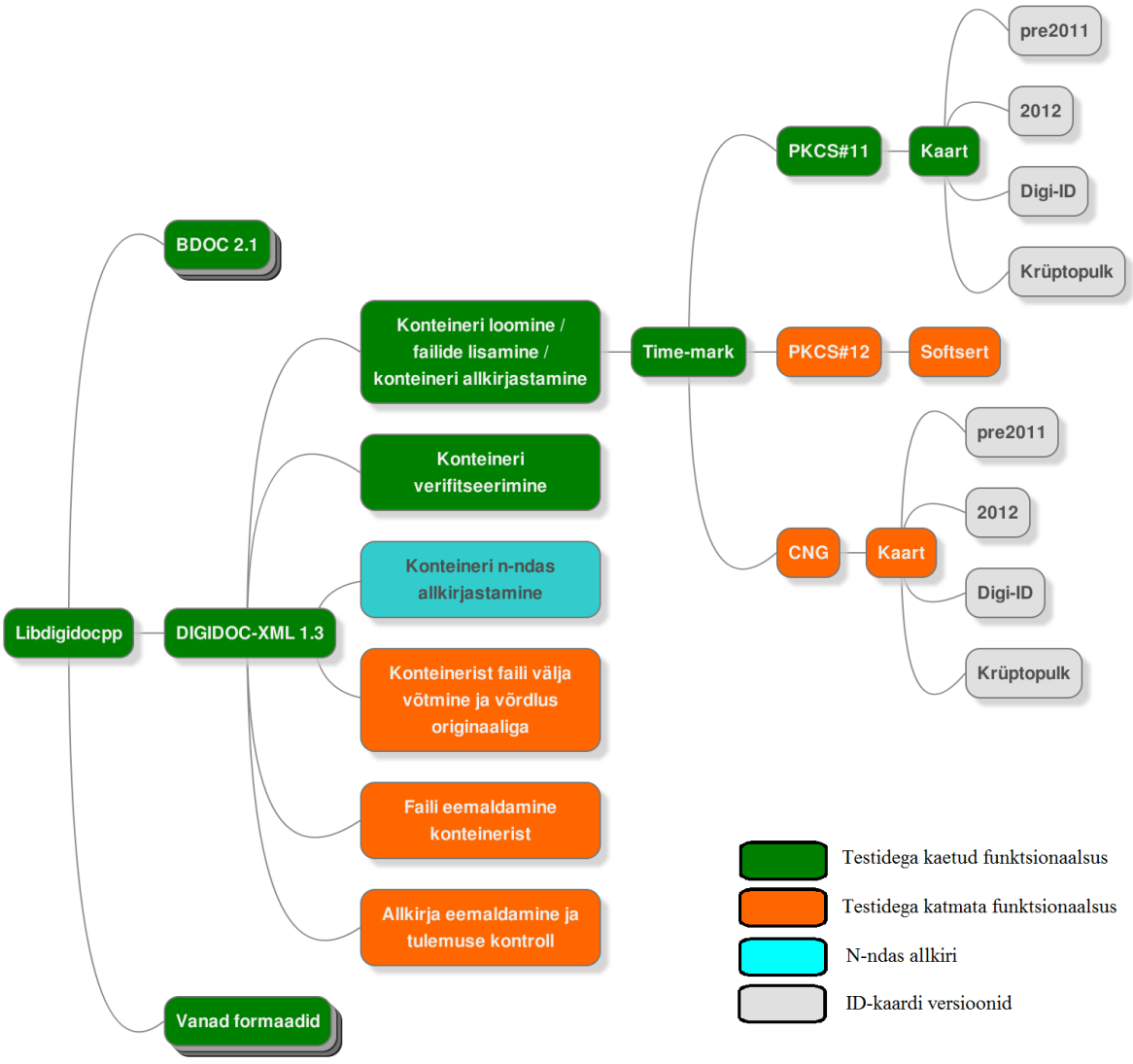
Tabel 7. *Libdigidocpp* allkirjastatud BDOCi ristkasutus

LIBDIGIDOCPP esimene allkiri			PKCS#12		PKCS#11	CNG
			Softsert	Kõveratega softsert	ID-kaart	ID-kaart
Teine allkiri *JDIGIDOC*	Vaikimisi	PKCS#11 ID-kaart				
		PKCS#12 Softsert				
	Stream	PKCS#11 ID-kaart				
		PKCS#12 Softsert				
LIBDIGIDOCPP esimene allkiri			PKCS#12		PKCS#11	*CNG*
			Softsert	Kõveratega softsert	ID-kaart	ID-kaart
Teine allkiri *LIBDIGIDOCPP*	PKCS#12	Softsert				
		Kõveratega softsert				
	PKCS#11	ID-kaart				
	CNG	ID-kaart				

Libdigidocpp PKCS#11 moodulit kasutades ID-kaardiga allkirjastatud konteinerit allkirjastatakse teistkordselt vaikimisi parameetritega *JDigiDoc* teegi ja *libdigidocpp* teegi abil ID-kaardiga (PKCS#11 moodul).

Digidoc-XML 1.3

Digidoc-XML 1.3 puhul on *libdigidocpp* teegil funktsionaalsuste nimekiri sama, mis BDOC 2.1 puhul. *Libdigidocpp* teegi puhul on Digidoc-XML 1.3 konteineri formaat alternatiiviks BDOC formaadile. Digidoc-XML 1.3 faili vormingu puhul kasutatakse pakendit (*wrapper*), mis kasutab *CDigiDoc* teeki, seega ka selle testimine on jäänud tahaplaanile ning testimise skoobis on ainult allkirja lisamine ID-kaardiga „*time-mark*“ profiiliga, nagu näitab Joonis 12. *Libdigidocpp* Digidoc-XML 1.3 funktsionaalsus.



Joonis 12. *Libdigidocpp* Digidoc-XML 1.3 funktsionaalsus

Digidoc-XML 1.3 faili vormingu puhul testitakse:

1. Konteineri loomist, failide lisamist konteinerisse ning konteineri allkirjastamist kasutades PKCS#11 moodulit.
2. Konteineri verifitseerimist

Testimise skoobist väljas on järgmised funktsionaalsused:

1. Konteineri loomine, failide lisamine konteinerisse ning konteineri allkirjastamine kasutades PKCS#12 ja CNG moodulit.
2. Konteinerist faili välja võtmine ja võrdlus originaaliga
3. Faili eemaldamine konteinerist
4. Allkirja eemaldamine ja tulemuse kontroll

Kui DDOC konteiner on *libdigidocpp* teegiga loodud ja allkirjastatud, siis seda on võimalik teistkordselt allkirjastada järgnevate võimalustega (Tabel 8. *Libdigidocpp* allkirjastatud DDOCi ristkasutus):

Tabel 8. *Libdigidocpp* allkirjastatud DDOCi ristkasutus

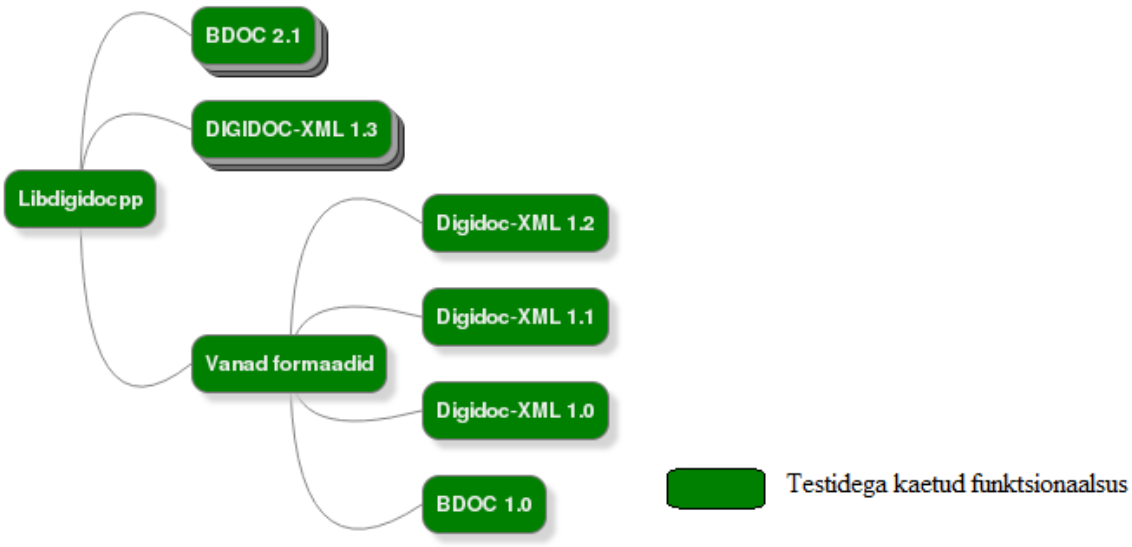
Esimene allkiri *LIBDIGIDOCPP*					
JDIGIDOC teise allkirja lisamine					
LIBDIGIDOCPP esimene allkiri			PKCS#12	*CNG*	PKCS#11
			Softsert	ID-kaart	ID-kaart
Teine allkiri *JDIGIDOC*	Vaikimisi	PKCS#11 ID-kaart			
		PKCS#12Softsert			
	Stream	PKCS#11 ID-kaart			
		PKCS#12Softsert			

LIBDIGIDOCPP esimene allkiri			PKCS#12	*CNG*	PKCS#11
Teine allkiri *CDIGIDOC*			Softsert	ID-kaart	ID-kaart
	Vaikimisi	PKCS#11 ID-kaart			
		PKCS#12Softsert			
	Mälus	PKCS#11 ID-kaart			
		PKCS#12Softsert			
LIBDIGIDOCPP teise allkirja lisamine					
LIBDIGIDOCPP esimene allkiri			PKCS#12	*CNG*	PKCS#11
			Softsert	ID-kaart	ID-kaart
Teine allkiri *LIBDIGIDOCPP*	PKCS#12	Softsert			
	PKCS#11	ID-kaart			
	CNG	ID-kaart			
DIGIDOC-COM teise allkirja lisamine					
DIGIDOC-COM	Vaikimisi	PKCS#11 ID-kaart			

Kuna *libdigidocpp* teegis on DIGIDOC-XML 1.3 konteineri loomine alternatiivlahendus BDOC'ile, siis ka DIGIDOC-XML 1.3 konteineri ristkasutust ei ole automaattestimise raamistikus testitud.

Vanad formaadid

Libdigidocpp teek peab mõningal määral toetama ka vanas formaadis konteinereid:



Joonis 13. *Libdigidocpp* vanad formaadid

Vanadest formaatidest *libdigidocpp* toetab Digidoc-XML 1.2, 1.1, 1.0 konteinerite formaate. Nende konteinerite puhul peab teek oskama avada neid ning konteineritest peab olema võimalik failid välja võtta. Uute konteinerite loomine eelmainitud versioonidega ning olemasolevate konteinerite allkirjastamine ei ole teegi poolt toetatud.

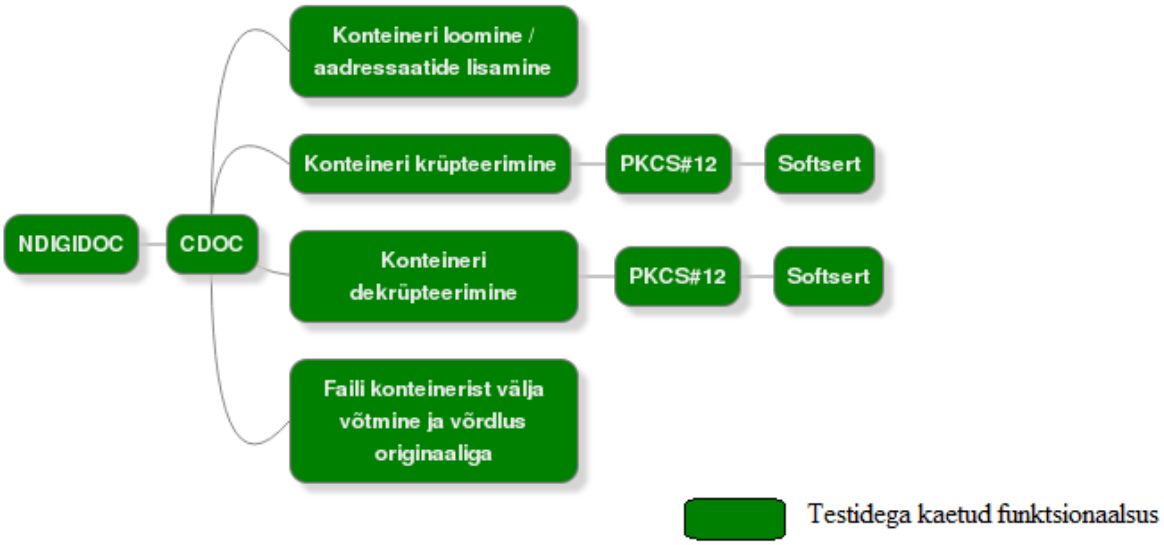
BDOC 1.0 formaadis konteineri käsitlemine ei ole teegi poolt toetatud.

2.2.5 NDigiDoc

NDigiDoc [25] teek on loodud krüpteerimiseks ja dekrüpteerimiseks, mis tähendab seda, et *NDigiDoc* toetab ainult CDOC faili vormingut. *NDigiDoc* teek on mõeldud kasutamiseks Windowsi platvormil. Krüpteerimine *NDigiDoc* teegiga on võimalik vaid tarkvarapõhiste sertifikaatidega.

CDOC

*NDigiDoc'*i poolt toetatud funktsionaalsused on näitab Joonis 14:



Joonis 14. *NDigiDoc* CDOC funktsionaalsus

NDigiDoc teegi funktsionaalsus on praeguses testiraamistikus kaetud testidega, need funktsionaalsused on järgmised:

1. Konteineri loomine, adressaatide lisamine.
2. Konteineri krüpteerimine kasutades PKCS#12 moodulit.
3. Konteineri verifitseerimine.
4. Konteineri dekrüpteerimine kasutades PKCS#12 moodulit.
5. Faili konteinerist välja võtmine ja võrdlus originaaliga.

NDigiDoc teegi poolt krüpteeritud konteinerit ristkasutuse mõttes peab olema võimalik dekrüpteerida *NDigiDoc* teegi endaga ning ka teiste teekidega, mis toetavad CDOC formaate.

Tabel 9 näitab *NDigiDoc* krüpteerimise ristkasutust. Tabelis on sümboliga „+“ märgitud osad, kus vastava krüpteerimisviisi korral on dekrüpteerimine võimalik ning „-“ tähendab, et antud krüpteerimise korral ei ole võimalik seda faili dekrüpteerida kasutades tabelis märgitud dekrüpteerimisviisi. Tabelis on rohelisega märgitud alad, mis on kaetud praegu automaattestimise raamistikus olevate testidega ning oranžiga on märgitud osad, mis testimise raamistikus ei ole kaetud.

Tabel 9. *NDigiDoc* krüpteerimise ristkasutus

<table>
<tr><td colspan="6">Krüpteerimine NDIGIDOC</td></tr>
<tr><td colspan="5"></td><td>Vaikimisi</td></tr>
<tr><td colspan="6">JDIGIDOC</td></tr>
<tr><td rowspan="8">JDIGIDOC dekrüpteerimine</td><td rowspan="4">Vaikimisi</td><td rowspan="2">PKCS#11
ID-kaart</td><td>Vaikimisi</td><td>+</td></tr>
<tr><td>Digidoc konteiner</td><td>-</td></tr>
<tr><td rowspan="2">PKCS#12
Softsert</td><td>Vaikimisi</td><td>+</td></tr>
<tr><td>Digidoc konteiner</td><td>-</td></tr>
<tr><td rowspan="4">Stream</td><td rowspan="2">PKCS#11
ID-kaart</td><td>Vaikimisi</td><td>+</td></tr>
<tr><td>Digidoc konteiner</td><td>-</td></tr>
<tr><td rowspan="2">PKCS#12
Softsert</td><td>Vaikimisi</td><td>+</td></tr>
<tr><td>Digidoc konteiner</td><td>-</td></tr>
</table>

CDIGIDOC				
CDIGIDOC dekrüpteerimine				Vaikimisi
	Vaikimisi	PKCS#11 ID-kaart	Vaikimisi	+
			Digidoc konteiner	-
		PKCS#12 Softsert	Vaikimisi	+
			Digidoc konteiner	-
	Mälus	PKCS#11 ID-kaart	Vaikimisi	+
			Digidoc konteiner	-
		PKCS#12 Softsert	Vaikimisi	+
			Digidoc konteiner	-
NDIGIDOC dekrüpteerimine				
	Default	Softsert	Vaikimisi	+

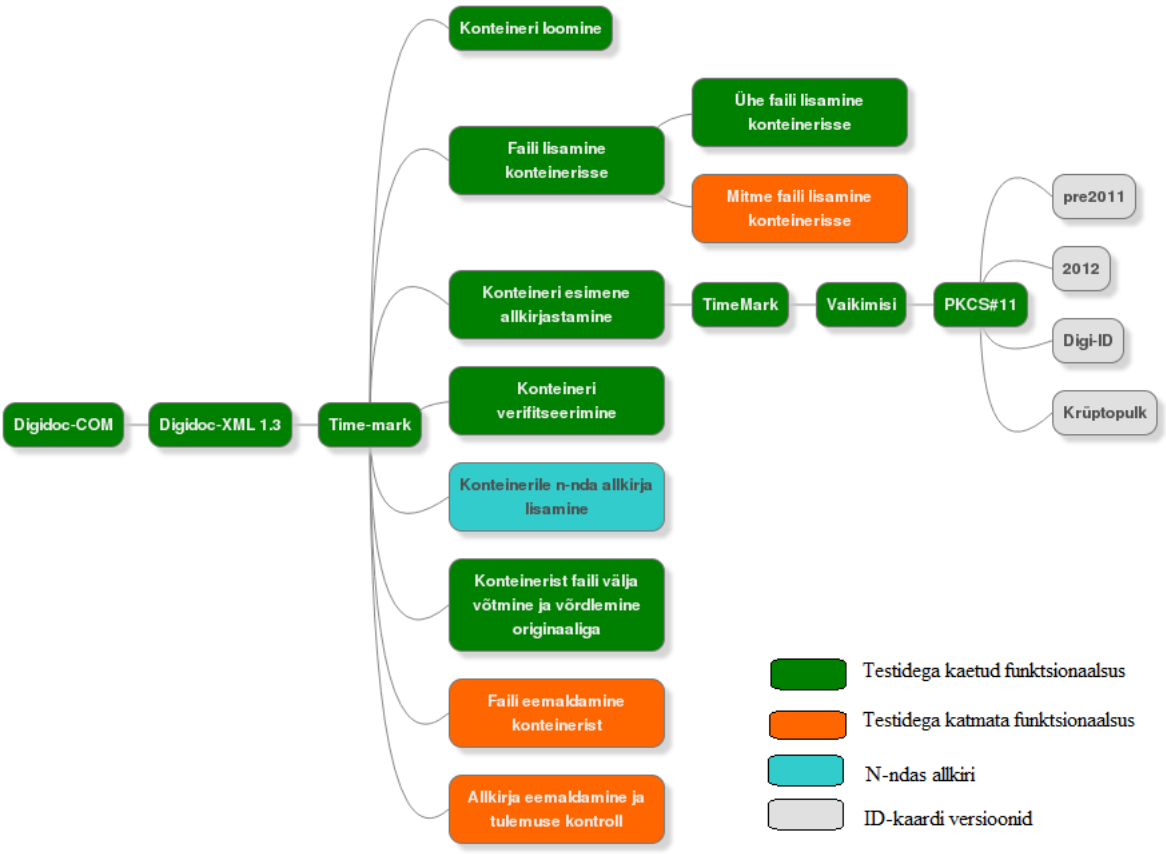
Tabel näitab, et *NDigiDoc*'iga krüpteeritud konteinerit praeguses testiraamistiku skoobis ei dekrüpteerita *CDigiDoc* teegi abil faili mällu sisse lugemisega.

2.2.6 DigiDoc-COM

DigiDoc-COM [26] teek toetab Digidoc-XML 1.3 konteinerite loomist. *DigiDoc-COM* teek on kirjutatud Windowsi platvormi jaoks.

Digidoc-XML 1.3

DigiDoc-COM teegi funktsionaalsus on sama, mis teiste teekide puhul, mis toetavad Digidoc-XML 1.3 konteineri formaati. Järgnevalt esitab Joonis 15 *DigiDoc-COM* teegi funktsionaalsuste nimekirja ja testidega kaetud skoopi olemasolevas raamistikus. Samuti on joonisel kajastatud osad, mis ei ole raamistikus testidega kaetud.



Joonis 15. *DigiDoc-COM* Digidoc-XML 1.3 funktsionaalsus

Testidega katmata jäänud funktsionaalsus on järgmine:

1. Mitme faili lisamine konteinerisse
2. Faili eemaldamine konteinerist.
3. Allkirja eemaldamine konteinerist ja tulemuse kontroll

Kuna 2015 aastal on plaanis ära lõpetada *DigiDoc-COM* teegi tugi, siis ka teegi testimine on väiksema prioriteediga võrreldes teiste teekide ja nende funktsionaalsustega.

Tabel 10. *DigiDoc-COM* allkirjastatud DDOCi ristkasutus

DIGIDOC-COM esimene allkiri			PKCS#11 ID-kaart
Teine allkiri *JDIGIDOC*	Vaikimisi	PKCS#11 ID-kaart	
		PKCS#12 softsert	
	Stream	PKCS#11 ID-kaart	
		PKCS#12 softsert	
Teine allkiri *CDIGIDOC*	Vaikimisi	PKCS#11 ID-kaart	
		PKCS#12 softsert	
	Mälus	PKCS#11 ID-kaart	
		PKCS#12 softsert	
Teine allkiri *LIBDIGIDOCPP*	PKCS#12	Softsert	
	PKCS#11	ID-kaart	
	CNG	ID-kaart	
Teine allkiri *DIGIDOC-COM*	Vaikimisi	PKCS#11 ID-kaart	

DigiDoc-COM ID-kaardiga allkirjastatud konteinerit allkirjastatakse teistkordselt ID-kaardiga vaikimisi parameetritega *JDigiDoc*, *CDigiDoc* ja *DigiDoc-COM* teekide abil, nagu näitab Tabel 10. *DigiDoc-COM* allkirjastatud DDOCi ristkasutus.

2.2.7 Failid

Teegi funktsionaalsust ei ole võimalik kasutada ilma andmefailita. Siinkohal on andmefaili all mõeldud igasugust faili, mida tahetakse allkirjastada või krüpteerida. Kuna igasuguseid failide

formaate ja kodeeringuid on palju, siis teegi funktsionaalsuse testimisel tuleb arvestada ka erinevate kodeeringutega, failide formaatidega ja ka failide suurustega.

Eraldi käsitletakse veel veaolukordi failides. Need on failid, mille töötlemisel peab teek andma konkreetse vea. Vigaste failide hulka kuulub ka ilma OCSP vastuseta allkirjastatud konteiner. Veaolukordade tuvastamiseks olid sihilikult loodud failid, mis sisaldavad konkreetset viga. Näiteks failid, mis ei vasta kehtivale allkirjastamise standardile. Failide ja vastavate vigade ja veakoodide analüüs toimub pidevalt. Vigaste failide ja veaolukordade analüüs ei kuulu lõputöö skoopi.

Lisaks käsitletakse veel faile, mida teek peab töötlemisel tunnistama valiidseteks, kuid peab andma hoiatuse. Valiidsete failide alla kuuluvad failid, mis on loodud vanade tarkvara versioonidega või kasutades aegunud algoritme. Need failid on valiidsed, kuid nende käsitlemine ja nendega edasi tegutsemine ei ole enam turvaline aegunud algoritmide kasutamise tõttu või mingisugusel muul põhjusel ei ole enam tarkvara poolt toetatud. Teek peab sellised failid tuvastama, hoiatama kasutajat sellest, et antud faili allkirjastamine ei ole soovitatav. Selliste failide analüüs ja kogumine ei kuulu lõputöö skoopi.

Vanades faili vormingu formaatides loodud failide käsitlemine kuulub samuti valiidsete failide käsitlemise hulka. Vanades formaatides konteinerite puhul peab teek tuvastama, et konteiner on vanas formaadis ning hoiatama sellest kasutajat. Selliste failidega edasi tegutsemine ei tohi olla teegi poolt lubatud, kuid konteineris olevaid faile peab olema võimalik konteinerist välja võtta ning allkirjastada luues uut konteinerit kehtiva faili vorminguga.

2.2.8 Operatsioonisüsteemide vaheline ristkasutus

ID-kaardi tarkvara teekide abil saab luua DigiDoc-ühilduvaid rakendusi ning teegid on esmaselt mõeldud arendajatele kasutamiseks. Kuna arenduskeskkonnad ja platvormid, mille peal ja mille jaoks uusi arendusi tehakse on erinevad, siis peab olema tagatud teekide korrektne töö ka erinevatel enamlevinud platvormidel ja operatsioonisüsteemidel.

Tarkvara teegi testimisel on oluline tagada teegi korrektne toimimine kõikides toetatud operatsioonisüsteemides. Lisaks sellele ühe teegi poolt loodud konteinerid peavad olema käideldavad ka teisel operatsioonisüsteemil. See tähendab, et näiteks Windows 7'l *JDigiDoc* teegi abil allkirjastatud konteiner peab olema avatav *JDigiDoc* 'i teegi abil Ubuntus ning lisaks peab olema võimalik selle konteineriga edasi tööd teha. Näiteks seda veelkord allkirjastada

ja/või krüpteerida. Samuti see sama konteiner peab olema töödeldav Ubuntus ka teise teekide abil.

Lisaks allkirjastamisele tuleb tagada ka dekrüpteerimise funktsionaalsuse korrektne toimimine erinevates operatsioonisüsteemides. Näiteks Windows 7'l *JDigiDoc* teegiga krüpteeritud fail peab olema dekrüpteeritav Ubuntus.

3. Testimise skoop

3.1 Riski analüüs

Lõputöö skoop sisaldab ainult teekide funktsionaalsust, seega antud lõputöös ei käsitleta turvalisuse ja käideldavuse riske ning keskendutakse ainult funktsionaalsuse ja ühilduvusega seotud riskidele. Samuti ei käsitleta aja ja rahaliste ressurssidega seotuid riske. Lõputöös lähtutakse teekide teadaolevast ja eelpool analüüsis välja toodud funktsionaalsusest.

Riskide identifitseerimiseks olid läbi viidud intervjuud ekspertidega, antud kontekstis ekspertide all on mõeldud ID-kaardi baastarkvara arenduse projektijuhti ning testijuhti.

Iga riski korral hinnatakse selle majanduslikku mõju teegi lõikes ning arvestatakse teegi prioriteeti. Riskide maandamise arvestamisel arvutatakse iga teegi prioriteedi ja riski mõju korrutis. Riski prioriteeti ja mõju hinnatakse järgmise skaala järgi:

1 – Madal 2 – Keskmine 3 – Kõrge 4 – Väga kõrge

Nagu riskide identifitseerimine oli riskide prioriteedi ja mõju hindamine teostatud koostöös ekspertidega.

Tabel 11. Riskide hindamine

		JDigiDoc		*CDigiDoc*		*libdigidoc pp*		*NDigiDoc*		*Digidoc-com*	
	Risk	**prioriteet**	**mõju**	**prioriteet**	**mõju**	**prioriteet**	**mõju**	**prioriteet**	**mõju**	**prioriteet**	**mõju**
	Funktsionaalsuse riskid										
R1	Vaikimisi faili sisse lugemine	4	4	4	3	3	2	4	2	4	2
R2	Stream faili sisse lugemine / faili sisse lugemine mällu	3	4	2	2	-	-	-	-	-	-
R3	Allkirjastamine DDOC	3	4	3	4	2	1	-	-	3	1
R4	Allkirjastamine BDOC	4	3	-	-	3	4	-	-	-	-
R5	Krüpteerimine CDOC	4	4	3	3	-	-	1	2	-	-
R6	Faili välja võtmine allkirjastatud konteinerist	2	4	2	3	2	2	-	-	1	1
R7	Allkirja eemaldamine	2	4	2	3	2	2	-	-	2	1
R8	Fail ei dekrüpteeru	4	4	4	4	-	-	4	1	-	-
R9	Teekide vaheline ristkasutus ei tööta	4	4	4	4	4	4	4	4	4	4
R10	Teek ei tunnista vigaseks vigast konteinerit	4	4	4	4	4	2	2	1	4	1
R11	Teek tunnistab vigaseks korrektse konteineri	4	4	4	3	4	2	-	-	3	1
R12	Algne fail on pärast allkirjastamist muutunud	4	4	4	3	4	2	-	-	4	1

R13	Algne fail on pärast krüpteerimist muutunud	4	4	4	3	-	-	4	1	-	-
	Andmetega seotud riskid										
	Läheandmete käitlemine										
R14	Teek ei suuda töödelda faili kodeeringut	3	4	3	3	3	2	2	1	2	1
R15	Teek ei suuda töödelda teatud suuruses faili	2	4	2	2	2	1	1	1	1	1
R16	Teek ei suuda töödelda faili, mis sisaldab pilti	1	2	1	1	1	1	1	1	1	1
R17	Teek ei suuda töödelda teatud formaadis faili (pdf, docx, txt, zip, 7z jne)	3	3	3	2	3	1	2	1	2	1
R18	Teek ei suuda töödelda allkirjastaja andmeid	2	3	2	2	2	1	-	-	1	1
	BDOC, DDOC, CDOC failivormingus failide käitlemine										
R19	Väljundfaili allkiri on kehtetu DDOC	4	4	4	3	4	2	-	-	3	2
R20	Väljundfaili allkiri on kehtetu BDOC	4	3	-	-	4	2	-	-	-	-
R21	Väljundfaili konteiner on vigane	4	4	4	3	4	2	3	1	3	2
R22	Krüpteeritud konteineri adressaatide nimekiri on vigane	4	4	4	4	-	-	4	1	-	-
	Liidesed										

R23	Teek ei tööta operatsioonisüsteemil	4	3	4	2	4	1	3	1	2	1
R24	Teegi funktsionaalsus ei tööta kasutades PKCS#11 moodulit	3	2	2	1	2	1	-	-	2	1
R25	Teegi funktsionaalsus ei tööta kasutades PKCS#12 moodulit.	4	4	4	3	4	2	4	1	-	-
R26	Teegi funktsionaalsust ei ole võimalik kasutada PKCS#11 ja/ või PKCS#12 mooduliga pärast PKCS#11 mooduli kasutamist	3	3	3	2	3	1	-	-	2	1
R27	Teegi funktsionaalsust ei ole võimalik kasutada PKCS#11 ja/ või PKCS#12 mooduliga pärast PKCS#12 mooduli kasutamist	3	3	3	2	3	1	-	-	-	-
	Ühilduvus										
R28	Teek ei suuda töödelda vanades teekide versioonides loodud faile	3	4	4	3	3	2	3	1	3	2
R29	Uues teegi versioonis loodud failid ei ühildu teegi vanade versioonidega	4	4	4	3	4	2	3	2	3	1
R30	Ühel operatsioonisüsteemil loodud väljundfail ei tööta teisel operatsioonisüsteemil sama teegiga	4	4	4	3	4	2	3	2	3	1
R31	Ühel operatsioonisüsteemil loodud väljundfail ei tööta teisel	4	4	4	3	4	2	3	2	3	1

	operatsioonisüsteemil teise teegiga										
	Keskkonna riskid										
R32	Keskkonnas PKCS#11 moodulid uusima Javaga ei tööta	2	4	-	-	-	-	-	-	-	-

näitab teekide funktsionaalsustega seotud riske. Tuleb märkida, et tabel ei sisalda kõiki võimalikke riske ning see kindlasti vajab edasist arendamist ja täiendavat riskide tuvastamist ja prioriteetide/mõju hindamist. Iga riski juures on hinnatud riski prioriteeti teegi seisukohalt ning selle riski mõju hetke olukorrale. Näiteks uus BDOC 2.1 formaat on kõrge prioriteediga, kuna BDOC 2 on mõeldud asendama Eesti spetsiifilist DDOC formaati, kuid selle mõju hetkeolukorrale on madalam, kui DDOC'i puhul, kuna see on uus formaat ning selle kasutamise ei ole veel populaarne.

Tabelis on oranžiga märgitud kohad, kus risk on kõige suurem ning nende riskikohtadega on tarvilik tegeleda kõigepealt. Oranžiga tähistatud riskide kriteeriumiks on riski prioriteedi ja riski mõju korrutis, mis peab olema vähemalt 12 (määratud ekspertide poolt).

Kõige väiksema prioriteediga on *DigiDoc-COM* teek ning *NDigiDoc* teek. DigiDoc-COM teek on väikse prioriteediga seetõttu, et toode on oma aja ära elanud ja moraalselt vananenud. See on edasise arendamise lõpetamise põhjus.. On aga oluline, et *DigiDoc-COM* oleks ühilduv teiste teekidega ehk *DigiDoc-COM* teegi abil allkirjastatud failid oleksid korrektselt töötavad teiste teekidega.

NDigiDoc teek on samuti väiksema prioriteediga, sest see on kasutusel vähestes integratsioonides ning pakub ainult krüpteerimise-dekrüpteerimise funktsionaalsust. Selle teegiga on võimalik vaid krüpteerida ja dekrüpteerida faile ning seda ainult kasutades softserti.

Riskide maandamiseks on loodud automatiseeritud testid.

3.2 Riskide maandamine

Eksperthinnangul on testide loomise põhimõte asjakohane ja mõistlik ning sobib ka edasiarendatavasse raamistikku. Muudetud on vaid lähenemine sisendfailidesse. Algselt sisendfailideks kasutati erikodeeringus ja eriformaatides sisendfaile, nüüd on koostöös ekspertidega otsustatud moodustada erikodeeringus ja eriformaatides failide testimiseks eraldi test. Lisaks on otsustatud moodustada eraldi test erisuurustega failidele. Kuna algses raamistikus on katmata *JDigiDoc* ja *CDigiDoc* puhul mitme faili allkirjastamine, siis on otsustatud lisada uude raamistikku ka see test.

Testide jaotus toimub teegi, konteineri tüübi ja funktsionaalsuse järgi. Ühe testi raames kaetakse võimalik ristkasutus teiste teekidega. Testid on praeguses testiraamistikus loodud järgnevalt kirjeldatud põhimõtete järgi.

Funktsionaalsuse testid:

- Esimene allkirjastamine (BDOC / DDOC, kõik võimalikud allkirjastamise viisid): (Kaetavad riskid:R1, R2, R3, R4, R6, R9, R11, R12, R18, R19, R20, R21, R23, R24, R25, R32)
    - Konteineri loomine
    - Faili lisamine konteinerisse
    - Konteineri allkirjastamine (PKCS#11 moodul – ID-kaart / PKCS#12 moodul - softsert / kõveratega softsert)
    - Konteineri verifitseerimine (sama teek, teised ühilduvad teegid)
    - Faili välja võtmine konteinerist ning väljavõetud faili võrdlemise kontroll originaaliga
- Esimene allkirjastamine – erinevad SHA pikkused, vaikimisi allkirjastamine. (Kaetavad riskid:R1, R4, R6, R9, R11, R12, R18, R20, R21, R23, R24, R32)
- Krüpteerimine: (Kaetavad riskid:R1, R2, R5, R8, R9, R11, R13, R21, R22, R23, R24, R25, R32)Adressaatide lisamine ja faili krüpteerimine

    - Faili dekrüpteerimine (sama teek, teised ühilduvad teegid, kõik dekrüpteerimise viisid / PKCS#11 moodul - ID-kaart, PKCS#12 moodul - softsert)
    - Faili välja võtmine konteinerist ning võrdlus originaalfailiga
- Teine allkirjastamine. Testi mõte on kasutada ühel operatsioonisüsteemil genereeritud ja allkirjastatud faile (üks teek) ning allkirjastada need teisel operatsioonisüsteemil (üks teek). Teekide ristkasutuse kombinatsioonide valikul lähtutakse kahe teegi (teek, millega on loodud esimene allkiri ning teek, millega hakatakse teine allkiri looma) prioriteedi korrutisest. (Kaetavad riskid:R1, R3, R4, R9, R11, R19, R20, R21, R23, R24, R25, R26, R27, R30, R31, R32)
    - Konteinerile teise allkirja lisamine
    - Konteineri verifitseerimine (sama teek, teised ühilduvad teegid)
- Konteineri dekrüpteerimine. Testi mõte on võtta ühes operatsioonisüsteemis ühe teegiga krüpteeritud fail ning dekrüpteerida see teises operatsioonisüsteemis (kõikide võimalike teekide ja dekrüpteerimisviisidega). (Kaetavad riskid:R1, R8, R9, R11, R21, R22, R23, R24, R25, R30, R31, R32)
    - Krüpteeritud faili dekrüpteerimine (kõik ühilduvad teegid, kõik võimalikud dekrüpteerimisviisid, PKCS#11 moodul – ID-kaart, PKCS#12 moodul - softsert)

Sisendfailidega seotud testid

Testides kasutatavad mõisted allkirjastamine ja krüpteerimine vastavad eelnevalt kirjeldatud tegevuste nimekirjale. Antud testides kasutatakse vaikimisi ja *streamina* / mällu faili sisse lugemisega allkirjastamist ja vaikimisi krüpteerimist, kuna teegi funktsionaalsus teiste krüpteerimis/dekrüpteerimisviisidega on juba kaetud eelmainitud krüpteerimise testides. Failide analüüs ei kuulu lõputöö skoopi:

- Erikodeeringus ja eriformaatides failide allkirjastamine (vaikimisi ja *streamina* / mällu faili sisse lugemine) ning võrdlus originaaliga, failide krüpteerimine (Kaetavad riskid: R1, R2, R3, R4, R5, R8, R9, R11, R12, R13, R14, R16, R17, R19, R20, R21, R22, R23, R25)

- Erisuurustes ja eriformaatides failide allkirjastamine ning võrdlus originaaliga, failide krüpteerimine (Kaetavad riskid: R1, R2, R3, R4, R5, R8, R9, R11, R12, R13, R15, R16, R17, R19, R20, R21, R22, R23, R25)

- Mitme faili allkirjastamine ühes konteineris, allkirja eemaldamine, failide krüpteerimine (Kaetavad riskid: R1, R2, R3, R4, R5, R7, R8, R9, R11, R12, R13, R16, R17, R19, R20, R21, R22, R23, R25)

- Eelnevate teegi versioonidega loodud failide verifitseerimine ja allkirja lisamine (Kaetavad riskid: R1, R3, R4, R28)

- Veahalduse kontrolliks loodud vigaste failide verifitseerimine (Kaetavad riskid: R10)

Ühilduvuse testid:

- Vanas teegi versioonis genereeritud failide verifitseerimine uue teegi versiooniga. Siin kohal test sõltub testimise skoobist. Võimalik, et skoopi kuulub mitu vana teegi versiooni, siis test viiakse läbi iga testi skoobis oleva teegi versiooniga loodud failidega. (Kaetavad riskid: R1, R3, R4, R28)

- Uue teegi versioonis loodud failide verifitseerimine vana teegi versiooniga. Siin kohal test sõltub hetkel testitava teegi jooksvast testimise skoobist. Võimalik, et skoopi kuulub mitu vana teegi versiooni, siis test viiakse läbi iga testi skoobis oleva teegi versiooniga. (Kaetavad riskid: R1, R3, R4, R29)

Praeguses automatiseeritud raamistikus on kasutusel iga teegi jaoks eraldi eelnevalt valmis kirjutatud konfiguratsioonifail. Selline lähenemine aga ei taga konfiguratsioonifaili uuendamist muutuste korral. Selleks, et maandada konfiguratsioonifailiga seotud riske, teostatakse muudatuste analüüs igas konfiguratsioonifailis ning lisatakse testiraamistikku funktsionaalsus, kus testi alguses võetakse teegi installeerimisega kaasa tulnud konfiguratsioonifail ning kohandatakse see vastavalt testi vajadustele.

Eelneva riskianalüüsi põhjal ja testide kirjelduse põhjal on peatükis 3.4 välja toodud näide skoobi defineerimisest *JDigiDoc* teegi puhul. Valitud on *JDigiDoc* seetõttu, et selle prioriteet on eksperthinnangul kõige suurem võrreldes teiste teekidega.

3.3 Testide jaotus

Projektis testitavaid teeke ning nende funktsionaalsusi on palju. Selleks, et testimine ja testi tulemused oleksid ülevaatlikumad jaotatakse skoop suitsutestideks, regressioonitestideks ning põhjalikumateks regressioonitestideks. Põhjalikumatesse regressioonitestidesse kuuluvad testid, mida on mõistlik läbida enne teegi või teekide reliisi.

Testitavad funktsionaalsused on otsustatud jaotada järgmise põhimõtte järgi:

1. Suitsutestide alla kuuluvad testid, mis testivad teegi põhifunktsionaalsust.
2. Regressioonitestide alla kuuluvad testid, mille ülesandeks on testida teekide ja operatsioonisüsteemide vahelist ristkasutust. Siia kuuluvad ka testid, mille ülesandeks on kontrollida erinevate andmefailide käsitlust.
3. Põhjalikumatesse regressioonitestidesse kuuluvad testid, mille ülesandeks on kontrollida teegi ühilduvust eelmiste teekide versioonidega.

Testide jaotamise põhimõtte järgi kuuluvad järgmised testid suitsutestide alla:

1. Esimene allkirjastamine Digidoc-XML 1.3 (kõik teegid vastavalt skoobile)
2. Esimene allkirjastamine BDOC 2.1 (kõik teegid vastavalt skoobile)
3. Krüpteerimine (kõik teegid vastavalt skoobile)

Regressioonitestide alla kuuluvad:

1. Teise allkirja lisamine (kõik kombinatsioonid vastavalt skoobile)
2. Dekrüpteerimine (vastavalt skoobile)
3. Erikodeeringus failide allkirjastamine
4. Erinevate suurustega failide allkirjastamine
5. Vigaste failide verifitseerimine
6. Valiidsete failide verifitseerimine ja allkirjastamine

Põhjalikumatesse regressioonitestidesse kuuluvad testid:

1. Vanades tarkvara versioonides loodud failide verifitseerimine ja allkirjastamine (vastavalt skoobile)

2. Uues versioonis loodud failide verifitseerimine ja allkirjastamine vanade versioonidega (vastavalt skoobile)

3.4 *JDigiDoc* testid

Järgnevalt on kirjeldatud testide skoop *JDigiDoc* teegi raames. Analoogselt luuakse testid teistele teekidele vastavalt nende prioriteedi ja mõjule riski analüüsis.

Nimekiri *JDigiDoc* teegi testidega katmiseks on järgmine:

Suitsutestid (Kõik skoobis olevad operatsioonisüsteemid):

- *JDigiDoc* ühe faili allkirjastamine vaikimisi parameetritega DDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert). Verifitseerimine *JDigiDoc*, *CDigiDoc*, *libdigidocpp*, *DigiDoc-COM* (Windowsi operatsioonisüsteemis) teekidega.

- *JDigiDoc* ühe faili allkirjastamine vaikimisi parameetritega BDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert, kõveratega softsert). Verifitseerimine *JDigiDoc*, *libdigidocpp* teekidega.

- *JDigiDoc* ühe faili allkirjastamine *streamina* DDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert). Verifitseerimine *JDigiDoc*, *CDigiDoc*, *libdigidocpp*, *DigiDoc-COM* (windowsil) teekidega.

- *JDigiDoc* ühe faili allkirjastamine *streamina* BDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert, kõveratega softsert). Verifitseerimine *JDigiDoc*, *libdigidocpp* teekidega.

- *JDigiDoc* ühe faili krüpteerimine vaikimisi parameetritega CDOC. Dekrüpteerimine *JDigiDoc* (vaikimisi PKCS#11, PKCS#12, *stream* PKCS#11, PKCS#12), *CDigiDoc* (vaikimisi PKCS#11, PKCS#12, mälus PKCS#11, PKCS#12), *NDigiDoc* (vaikimisi Windowsil)

- *JDigiDoc* ühe faili krüpteerimine ilma sisendfaili kokku pakkimiseta CDOC. Dekrüpteerimine *JDigiDoc* (vaikimisi PKCS#11, PKCS#12, *stream* PKCS#11,

PKCS#12), *JDigiDoc* (vaikimisi PKCS#11, PKCS#12, mälus PKCS#11, PKCS#12), *NDigiDoc* (vaikimisi Windowsil)

- *JDigiDoc* ühe faili krüpteerimine faili panemisega DigiDoc konteinerisse CDOC. Dekrüpteerimine *JDigiDoc* (vaikimisi DigiDoc konteineri dekrüpteerimine PKCS#11, PKCS#12, *stream* DigiDoc konteineri dekrüpteerimine PKCS#11, PKCS#12), *CDigiDoc* (vaikimisi DigiDoc konteineri dekrüpteerimine PKCS#11, PKCS#12, mälus DigiDoc konteineri dekrüpteerimine PKCS#11, PKCS#12)

- *JDigiDoc* ühe faili krüpteerimine *streamina* vaikimisi parameetritega CDOC. Dekrüpteerimine *JDigiDoc* (vaikimisi PKCS#11, PKCS#12, *stream* PKCS#11, PKCS#12), *CDigiDoc* (vaikimisi PKCS#11, PKCS#12, mälus PKCS#11, PKCS#12), *NDigiDoc* (vaikimisi Windowsil)

- *JDigiDoc* ühe faili krüpteerimine *streamina* ilma sisendfaili kokku pakkimiseta CDOC. Dekrüpteerimine *JDigiDoc* (vaikimisi PKCS#11, PKCS#12, *stream* PKCS#11, PKCS#12), *CDigiDoc* (vaikimisi PKCS#11, PKCS#12, mälus PKCS#11, PKCS#12), *NDigiDoc* (vaikimisi Windowsil)

- *JDigiDoc* ühe faili krüpteerimine *streamina* faili panemisega DigiDoc konteinerisse CDOC. Dekrüpteerimine *JDigiDoc* (vaikimisi DigiDoc konteineri dekrüpteerimine PKCS#11, PKCS#12, *stream* DigiDoc konteineri dekrüpteerimine PKCS#11, PKCS#12), *CDigiDoc* (vaikimisi DigiDoc konteineri dekrüpteerimine PKCS#11, PKCS#12, mälus DigiDoc konteineri dekrüpteerimine PKCS#11, PKCS#12)

Regressioonitestid (Kõik skoobis olevad operatsioonisüsteemid):

Regressioonitestide mooduliks on enamjaolt valitud PKCS#12 moodul, kuna moodulite käsitlemine on juba kaetud suitsutestides ning PKCS#12 mooduli käsitlemine on ajaliselt kiirem. Parem kiirus on tingitud asjaolust, et sertifikaat loetakse failist, mitte ei pöörduta ID-kaardi poole.

PKCS#12 mooduliga softserdiga allkirjastatud failile ei lisata teist allkirja, kuna softserdiga allkirjastatud fail ei ole Digitaalallkirja seaduse mõistes kehtiv ning sellele puuduvad reaalsed kasutuslood.

- Funktsionaalsus

- Allkirja eemaldamine konteinerist ning konteineri verifitseerimine. BDOC ja DDOC.

- Faili eemaldamine allkirjastatud konteinerist ning konteineri verifitseerimine. BDOC, DDOC.

- Erifailide käitlemine

  - Erikodeeringus failide esimene allkirjastamine *JDigiDoc* vaikimisi parameetritega DDOC ja BDOC (PKCS#12 - softsert)

  - Erisuurustes failide allkirjastamine *JDigiDoc* vaikimisi parameetritega DDOC ja BDOC (PKCS#12 - softsert)

  - Mitme faili lisamine konteinerisse ja allkirjastamine *JDigiDoc* vaikimisi parameetritega DDOC ja BDOC (PKCS#12 - softsert)

  - Erikodeeringus failide krüpteerimine *JDigiDoc* vaikimisi parameetritega. Dekrüpteerimine *JDigiDoc* (vaikimisi PKCS#11, PKCS#12, *stream* PKCS#11, PKCS#12), *CDigiDoc* (vaikimisi PKCS#11, PKCS#12, mälus PKCS#11, PKCS#12), *NDigiDoc* (vaikimisi Windowsil)

  - Erisuurustes failide krüpteerimine *JDigiDoc* vaikimisi parameetritega, dekrüpteerimine *JDigiDoc* (vaikimisi PKCS#11, PKCS#12, *stream* PKCS#11, PKCS#12), *CDigiDoc* (vaikimisi PKCS#11, PKCS#12, mälus PKCS#11, PKCS#12), *NDigiDoc* (vaikimisi Windowsil)

  - Veahalduse test. DDOC - *JDigiDoc* ja *CDigiDoc* failide verifitseerimine ja veaolukorra kontroll. BDOC - *JDigiDoc* ja *libdigidocpp* failide verifitseerimine ja veaolukorra kontroll.

  - Valiidsete failide test. DDOC - *JDigiDoc*, *CDigiDoc* valiidsete failide hoiatusolukordade kontroll. BDOC - *JDigiDoc*, *libdigidocpp* valiidsete failide hoiatusolukordade kontroll.

- Teise allkirja lisamine. Teise allkirja lisamise testimisel allkirjastatakse ainult need failid, millele on esimene allkiri lisatud ID-kaardiga või kõveratega softserdiga. Põhjus

seisneb selles, et softserdiga allkirjastatud fail ei ole Digitaalallkirja seaduse mõistes kehtiv ning sellisele olukorrale puuduvad reaalsed kasutusjuhud.

Kuna teisel allkirjastamisel ei oma olulist rolli see, kuidas oli teostatud esimene allkirjastamine (faili sisse lugemise poolest), siis teise allkirja lisamisel kasutatakse faile, mis olid loodud vaikimisi sisse lugemisega. Testid viiakse läbi kõigil skoobis olevatel operatsioonisüsteemidel ning sisendfailideks kasutatakse faile, mis olid loodud kõikidel skoobis olevatel operatsioonisüsteemidel – see tagab operatsioonisüsteemide vahelise ristkasutuse.

Teise allkirja lisamisel jäetakse skoobist välja DDOC konteinerile teise allkirja lisamine *libdigidocpp* teegiga, kuna sellel on väike prioriteet.

- *JDigiDoc* vaikimisi parameetritega allkirjastatud faili *JDigiDoc* teegiga teise allkirja lisamine vaikimisi parameetritega DDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert). Verifitseerimine *JDigiDoc*, *CDigiDoc*, *libdigidocpp*, *DigiDoc-COM* (windowsil) teekidega.

- *JDigiDoc* vaikimisi parameetritega allkirjastatud (ID-kaart, kõveratega softsert) faili *JDigiDoc* teegiga teise allkirja lisamine vaikimisi parameetritega BDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert, kõveratega softsert). Verifitseerimine *JDigiDoc*, *libdigidocpp* teekidega.

- *JDigiDoc* vaikimisi parameetritega allkirjastatud faili *CDigiDoc* teegiga teise allkirja lisamine vaikimisi parameetritega DDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert). Verifitseerimine *JDigiDoc*, *CDigiDoc*, *libdigidocpp*, *DigiDoc-COM* (windowsil) teekidega.

- *JDigiDoc* vaikimisi parameetritega allkirjastatud faili *CDigiDoc* teegiga teise allkirja lisamine mällu faili sisse lugemisega DDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert). Verifitseerimine *JDigiDoc*, C*DigiDoc*, *libdigidocpp*, *DigiDoc-COM* (windowsil) teekidega.

- (Windows platvormil) *JDigiDoc* vaikimisi parameetritega allkirjastatud faili *DigiDoc-COM* teegiga teise allkirja lisamine vaikimisi parameetritega DDOC (PKCS#11 moodul – ID-kaart). Verifitseerimine *JDigiDoc*, *CDigiDoc*, *libdigidocpp*, *DigiDoc-COM* teekidega.

- *JDigiDoc* vaikimisi parameetritega allkirjastatud (ID-kaart, kõveratega softsert) faili *libdigidocpp* teegiga teise allkirja lisamine vaikimisi parameetritega BDOC (PKCS#11 moodul – ID-kaart, PKCS#12 moodul – softsert, CNG moodul – ID-kaart). Verifitseerimine *JDigiDoc*, *libdigidocpp* teekidega.

- Krüpteeritud konteineri dekrüpteerimine

  Dekrüpteerimisel samuti ei tehta vahet, kas krüpteeritud konteiner oli loodud *streamina* või vaikimisi:

  - *JDigiDoc* vaikimisi krüpteeritud faili dekrüpteerimine *JDigiDoc* vaikimisi faili sisse lugemisega vaikimisi parameetritega PKCS#11, PKCS#12, *stream* faili sisse lugemisega PKCS#11, PKCS#12, *CDigiDoc* vaikimisi sisse lugemisega vaikimisi parameetritega dekrüpteerimine PKCS#11, PKCS#12, mällu faili sisse lugemisega vaikimisi parameetritega dekrüpteerimine PKCS#11, PKCS#12

  - *JDigiDoc* ilma sisendfaili kokku pakkimiseta krüpteeritud faili dekrüpteerimine *JDigiDoc* vaikimisi faili sisse lugemisega vaikimisi parameetritega PKCS#11, PKCS#12, *stream* faili sisse lugemisega PKCS#11, PKCS#12, *CDigiDoc* vaikimisi sisse lugemisega vaikimisi parameetritega dekrüpteerimine PKCS#11, PKCS#12, mällu faili sisse lugemisega vaikimisi parameetritega dekrüpteerimine PKCS#11, PKCS#12

  - *JDigiDoc* DigiDoc konteinerisse paigaldatud krüpteeritud faili dekrüpteerimine *JDigiDoc* vaikimisi faili sisse lugemisega DigiDoc konteineri dekrüpteerimise viisiga PKCS#11, PKCS#12, *streamina* DigiDoc konteineri dekrüpteerimise viisiga PKCS#11, PKCS#12, *CDigiDoc* vaikimisi sisse lugemisega DigiDoc konteineri dekrüpteerimise viisiga dekrüpteerimine PKCS#11, PKCS#12, mällu faili sisse lugemisega DigiDoc konteineri dekrüpteerimise viisiga dekrüpteerimine PKCS#11, PKCS#12

Detailsed regressioonitestid (kõik skoobis olevad operatsioonisüsteemid):

- Vanades versioonides loodud failide verifitseerimine *JDigiDoc*'iga ning nendele teise allkirja lisamine vaikimisi parameetritega kasutades PKCS#11 moodulit – ID-kaart. (kõik skoobis olevate *JDigiDoc* versioonidega loodud failid)

- Uue *JDigiDoc* versiooniga loodud failide verifitseerimine vanade *JDigiDoc* versioonidega ning teise allkirja lisamine vaikimisi parameetritega kasutades PKCS#11 moodulit – ID-kaart. (kõik skoobis olevad *JDigiDoc* versioonid)

- *JDigiDoc* vaikimisi parameetritega allkirjastamine BDOC PKCS#11 moodulit kasutades (ID-kaart), kus allkirja räsi algoritm on SHA-1, SHA-224 ja SHA-512.

4. Raamistik

4.1 Raamistiku struktuur

Antud projektis kasutatakse hübriid raamistikku, mis sisaldab endas kombinatsiooni andmete põhisest ja võtmesõna põhisest lähenemisest.

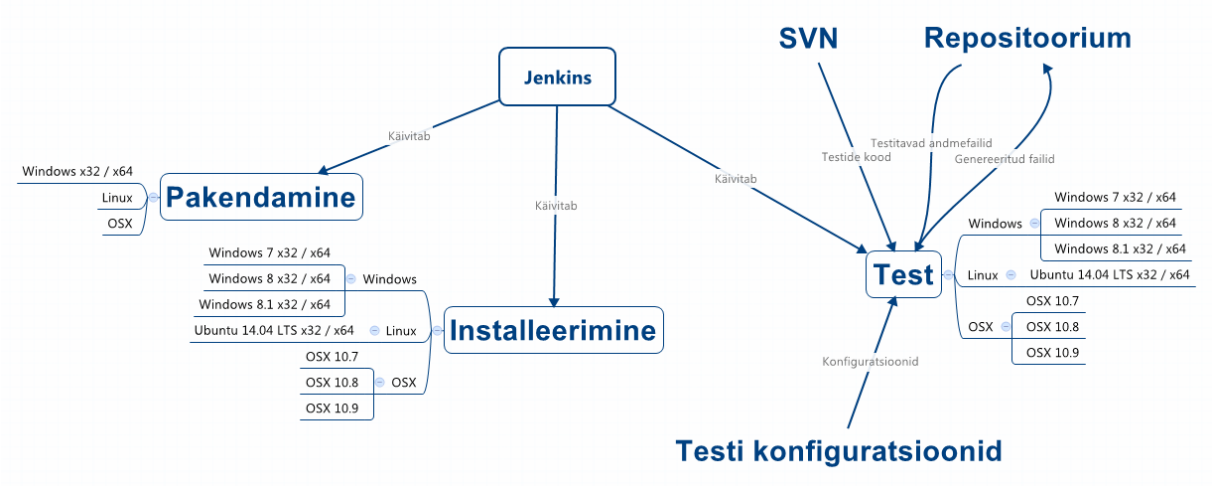
Hübriid raamistik peab olema disainitud nii, et see oleks taaskasutatav ja lihtsalt skaleeruv. Samuti testide raamistikus kirjutatav kood peab olema kergesti arusaadav ning lihtsasti hallatav. Testiraamistik peab olema teatud määral konfigureeritav väliselt, ehk ilma koodi muutmata.

Andmete põhisest lähenemisest on võetud mitmed tunnusjooned, näiteks see, et teste on võimalik käivitada erinevate andmetega. Andmeid on võimalik hoida testidest eraldi, mis tagab andmete parema ja lihtsama hallatavuse. See võimaldab uuendada ja hallata testiandmeid testikoodist sõltumatult.

Võtmesõnal põhineval lähenemisel on samuti omad tunnusjooned, mida hübriid testiraamistikus kasutatakse, näiteks see, et testides tehtavad tegevused on nimetatud kindla võtmesõnaga ning seetõttu uute testide genereerimine võtmesõnade alusel on lihtsam ning ei vaja nii palju (või siis üldse mitte) programmeerimisoskust. [27]

4.1.1 Olemasoleva raamistiku struktuur

Automatiseeritud raamistiku üldine struktuur on järgmine:



Joonis 16. Raamistiku üldine struktuur

Nagu Joonis 16 näitab, tarkvara kood (ka tarkvara teekide pakid) pakitakse kokku *Jenkins* töövahendi ülesande abil, mis aitab tagada pidevat integratsiooni. Tarkvara pakid genereeritakse erinevate platvormide tarvis. Kui tarkvara kood on kokku pakitud, siis käivitatakse automaatselt järgmised *Jenkins*'i ülesanded.

Jenkins'i nii öelda sisenditeks on masinad, millele on paigaldatud erinevad operatsioonisüsteemid.

Kõigepealt installitakse värske tarkvara testimise skoobis olevatele masinatele. Seejärel käivituvad testid. Igal testil on võimalik määrata oma konfiguratsioonid testide läbi viimiseks. Testiülesanded on üles ehitatud nii, et üks ja sama test käivitatakse erinevatel masinatel, seega ka erinevatel operatsioonisüsteemidel.

Testiülesande alguses sünkroniseeritakse testide kood *SVN*'is oleva koodiga ning testis kasutatavad sisendfailid sünkroniseeritakse repositooriumis olevate failidega. Seejärel käivitatakse testid.

Uue operatsioonisüsteemi lisandumisel skoopi on tarvis vaid omada masin selle operatsioonisüsteemiga ning lisada see testiülesande alla.

Teekide ristkasutuse tagamiseks allkirjastatud ja krüpteeritud konteinerid salvestatakse. Testide puhul, kus väljundiks on allkirjastatud või krüpteeritud fail ning mis on järgmise testi sisendfailiks, laaditakse testi käigus genereeritud failid testi lõpus repositooriumi.

Selleks, et hajutada sisendfailidega seotud riske, on igal operatsioonisüsteemil oma sisendfailid. Lisaks nagu eelnevalt oli mainitud, on olemas vigaste failide kogumid ja vanade valiidsete failide kogumid, mis on kõikidele operatsioonisüsteemidele samad ning hoitakse operatsioonisüsteemide failidest sõltumatult. Vanades tarkvara versioonides loodud failide ühilduvuse tagamiseks uute tarkvara versioonidega salvestatakse failid samuti repositooriumisse, kuid vastava operatsioonisüsteemi alla, kus need olid loodud.

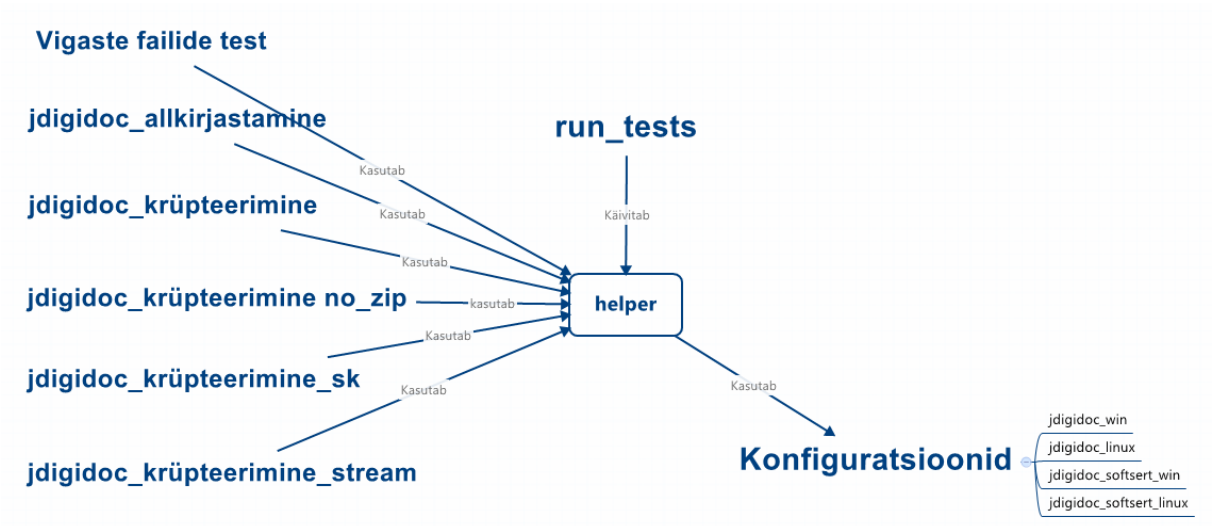
Andmete ja testifailide hoidmiseks on kausta struktuur järgmine (faili struktuur on toodud näitena ning Windows XP alla kuuluv struktuur on sama ka teiste operatsioonisüsteemide puhul):

- Vigased failid
- Valiidsed konteinerid
- Windows XP

    - 2_7_10 (siin on failid, mis olid loodud tarkvaraga 2_7_10)
    - 3_2
    - 3_4
    - 3_5
    - 3_5_1
    - 3_6
    - 3_7
    - 3_7_1
    - 3_7_2
    - algandmed
- Windows 7
- Windows 8
- Windows 8.1
- Windows Vista
- Ubuntu
- OSX 10.7
- OSX 10.8
- OSX 10.9

Jenkins testiülesande käivitumisel kõigepealt käivitatakse üldine test, mis tegeleb failide sünkroniseerimise ja keskkonna ette valmistamisega. Seejärel käivitub konkreetne test.

Järgnevalt on esitatud Joonis 17, mis illustreerib raamistiku koodi struktuuri. Joonis ei näita täielikku koodi struktuuri ning on mõeldud vaid ettekujutuse saamiseks.



Joonis 17. Koodi struktuuri näide

Testide kood on jaotatud teegi järgi ning funktsionaalsuse järgi. Näiteks *JDigiDoc* teegi abil on võimalik allkirjastada faile ning krüpteerida faile. Lisaks on nii öelda erandlikud testi failid, mis ei sõltu otseselt teegist, näiteks vigaste failide test.

Allkirjastamise testide juures on võimalik seadistada konteineri tüüpi ja selle versiooni – näiteks Digidoc-XML 1.3 või BDOC 2.1. Lisaks on võimalik konfigureerida allkirjastamise viisi ning määrata, kas testis kasutatakse ID-kaarti või asutuse sertifikaati – näiteks kasutades *stream* faili sisse lugemise viisi ning allkirjastades softserdiga. Neid parameetreid on võimalik seadistada testikoodist väljas poolt ning seda tehakse *jenkins*'i testiülesande juures.

Krüpteerimine on praeguses raamistikus jaotatud krüpteerimisvõimaluste järgi. Näiteks, sisendfaili krüpteerimiseks paigaldamisega DigiDoc konteineisse (ülal pool toodud joonisel märgitud „*-sk"*) ja vaikimisi krüpteerimise jaoks on genereeritud eraldi testifailid.

Kõik testid kasutavad üldisi meetodeid, mis on defineeritud eraldi klassis „helper". Praeguses koodis esineb olukord, kus mitme testi jaoks on kasutatud samu muutujaid ning nende defineerimine käib igas testis eraldi. Seega näiteks testis kasutatavate andmefailide asukoha muutmisel tuleb igas testis käsitsi see asukoht ära muuta.

Praeguses automatiseeritud testiraamistikus on kasutatud spetsiaalseid teegi jaoks valmis kirjutatud konfiguratsioonifaile, mida testi käigus kasutati. Iga platvormi jaoks on koostatud eraldi konfiguratsioonifail, lisaks on konfiguratsioonifailid näiteks ID-kaardi ja softserdi kasutamiseks, kusjuures ID-kaardi kasutamist loetakse vaikimisi tegevuseks. Sellega aga ilmnes probleem, et konfiguratsioonifaili muutmisel arendaja poolt, testides kasutatud konfiguratsioonifail ei muutunud. Seda tuli muuta ja uuendada käsitsi. Kuna arendaja alati ei teatanud konfiguratsioonifailide muutustest ning testijatel ise pidevalt jälgida muudatusi on raske, siis esines olukordi, kus testide tulemused näitasid ebaadekvaatset tulemust.

4.1.2 Loodava raamistiku struktuur

Raamistiku üldine struktuur jääb samaks nagu on kirjeldatud peatükis 4.1.1. Siiani vaikimisi testifailideks on kasutatud erikodeeringus ja erimärkidega faile ning ühel operatsioonisüsteemil viidi läbi teste mitme andmefailiga (tavaliselt 1-3). Kuna aga allkirjastamise viise on palju ja veel rohkem on krüpteerimise viise, siis failide arv, mida ristkasutusel oli tarvis ära testida, kasvas eksponentsiaalselt. Selline olukord aeglustas testide läbimise protsessi ning kõikide failide läbi testimine ei tundunud mõistlik.

Uues raamistikus on otsustatud kombineerida erinevat faili suurust ja kodeeringut ning vaikimisi testida ühte faili ühel operatsioonisüsteemil, kuid varieerida failid operatsioonisüsteemide vahel. Põhjalikum nii erimärkide, erikodeeringute kui ka faili suuruste

testid on otsustatud testida iseseisvalt eraldi testidena. See võimaldab optimeerida ressursside kasutust. Esiteks, teegi väikeste muudatuste korral on võimalik läbi jooksutada väikseid ja kiireid teste, kindlustamaks teegi korrektse toimimise pärast muudatust. Lisaks on võimalus hajutada riske ning tuvastada kiiremini veaolukordi. Näiteks veaolukorra ilmnemisel enne ei olnud võimalik koheselt aru saada, kas probleem on teegi funktsionaalsuses või erifailide käsitlemises. Uues raamistikus on võimalik selliseid olukordi eristada.

Lisaks sellele, eraldi testidena käsitletakse vanu faile, ehk verifitseeritakse teegi eelmiste versioonidega loodud failid teegi uue versiooniga.

Kausta struktuur jääb enamjaolt samaks, mis oli. Lisanduvad ainult kaustad, kus operatsioonisüsteemis sõltumatult hakatakse hoidma erikodeeringus ning erisuurustega faile vastavate testide läbiviimiseks.

Kontrollimaks teegi vastavust standarditele testitakse teegi abil ka vigaseid faile. Vigaste failide komplekt oli loodud võimalike vigade analüüsimisel ning analüüsi tulemusena tekkisid failid, mida teek peab tunnistama vigasteks ning olid fikseeritud vea olukorrad ja veakoodid, mille alusel peab teek tunnistama failid vigasteks.

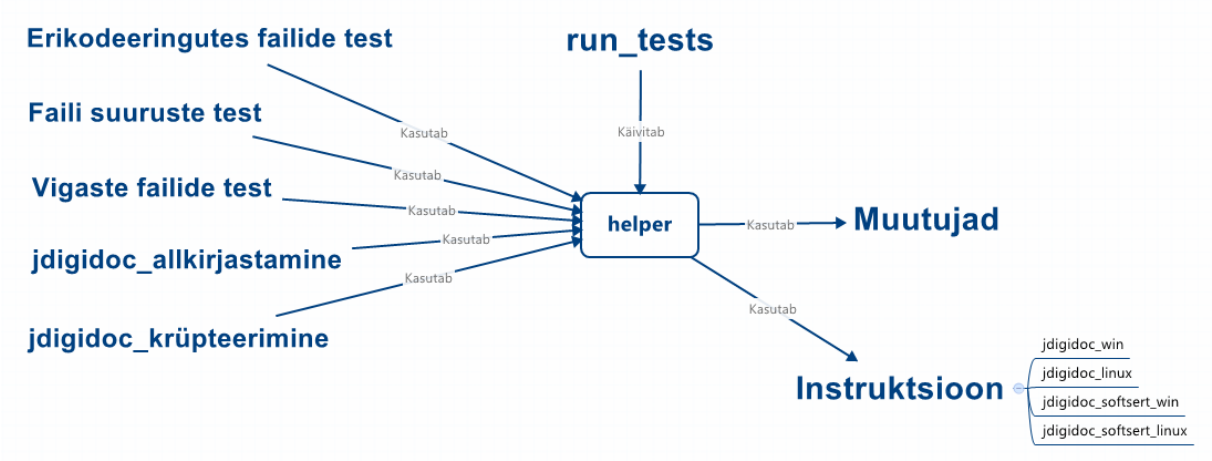
Uues raamistikus on planeeritud realiseerida funktsionaalsus, kus testides alati kasutatakse värsket konfiguratsioonifaili ning enne iga testi algust kohandatakse vaikimisi konfiguratsioonifail vastavalt testi vajadustele. Selleks, et teada saada, millised muudatused on tarvilikud kindla testi läbi viimiseks, kasutatakse instruktsioonifaili, kuhu on kirjeldatud kõik kindla testi jaoks tarvilikud konfiguratsiooni muudatused. See muudatus aitab ära hoida vana konfiguratsioonifaili kasutamist.

Teekide jaotus testides jääb samaks, kuid krüpteerimise funktsionaalsuse jaoks genereeritakse üks testifail, nagu on allkirjastamisel, ning erinevate krüpteerimisviiside ja kombinatsioonide kasutamist saab määrata *jenkins* testülesande kaudu.

Testides kasutatavad muutujad viiakse testikoodi seest välja ning genereeritakse selle jaoks eraldi fail. Nii näiteks testifailide asukoha muutumisel saab muuta seda koodis vaid ühes kohas.

Testikoodi uuenduste käigus refaktoreeritakse kood ning viiakse koodi sisse muudatused, mis on seotud *Ruby* ja *RSpec*'i versioonide uuendustega.

Järgnevalt on esitatud Joonis 18, mis kujutab raamistiku testi koodi uut struktuuri. Joonis ei näita täielikult koodi struktuuri, vaid annab ülevaate muudatustest.



Joonis 18. Raamistiku uus koodi struktuur

4.2 Testide koodi optimeerimine

Vastavalt kohandatud raamistiku ja koodi struktuurile on hakatud koodi optimeerima. Testides on üle mindud *Ruby* 1.9 ning *RSpec* 2 peale. Kood on ümber tehtud vastavalt uuele koodi struktuurile. On loodud eraldi muutujate failid, mida testid kasutavad. On realiseeritud funktsionaalsus, kus iga testi alguses võetakse teegi konfiguratsioonifail ning kohandatakse see vastavalt testi nõudmistele kasutades selleks vastavat instruktsiooni faili. Testide kood on kohandatud vastavalt uuele skoobile. Vastavalt skoobile on loodud *Jenkins*'is ülesanded testide käivitamiseks.

5. Kokkuvõte

Antud töö eesmärk oli korrastada ID-kaardi tarkvara teekide automatiseeritud regressioonitestimist. Selleks, et seda eesmärki saavutada oli kaardistatud tarkvara struktuur ja tarkvara mõjutavad tegurid, mida on tarvilik testimisel arvestada. Seejärel oli tarvilik selgeks teha, mis osa tarkvarast moodustavad teegid ning mitu ja milliseid teeke ID-kaardi tarkvara raames arendatakse. Töö suurima osa moodustas teekide funktsionaalsuste kaardistus ning automatiseeritud testide analüüs. Selle tulemusena selgusid teekide funktsionaalsused ja ristkasutuse kombinatsioonid, mis olid automatiseeritud testidega kaetud ning millised olid testidega katmata.

Selleks, et hinnata, milliseid teekide funktsionaalsusi ja ristkasutuse kombinatsioone on tarvilik testida, oli läbi viidud teekide funktsionaalsusele fokuseeritud riskianalüüs ning eksperthinnangu alusel oli moodustatud testimise skoop.

Lõputöö raames oli kaardistatud automatiseeritud raamistiku ning testide koodi struktuur. Automatiseeritud testimise raamistiku struktuur on töö käigus kohandatud ning testide koodis kasutatavad vahendid on uuendatud. Lisaks on testide koodi struktuur optimeeritud ning kood refaktoreeritud.

Töö käigus läbiviidud riskianalüüs ning koodi refaktoreerimine oli ühekordne tegevus. Selleks, et automatiseeritud raamistik ning testitav skoop vastaks ka edaspidi vajadustele ning nõudmistele, on tarvilik pidevalt jälgida teekide funktsionaalsusi ja võimalusi, analüüsida riske ning koodi poole pealt jälgida vahendite kasutust. Sellisel juhul ei teki uuesti olukorda, kus ei ole selget nägemust testitavatest ning mitte testitavatest funktsionaalsustest.

Summary

The aim of this thesis was to organize automated regression testing of ID-card software libraries. In order to accomplish this goal, software structure and possible factors that influence the software were analysed. After that, it was necessary to make sure how many and which libraries were developed in ID-card software. The majority of the thesis consists of mapping the functionalities of different ID-card software libraries and automated tests analysis. As a result, the functionalities of libraries and their cross-usage combinations which were covered in testing scope were identified.

In order to assess which libraries functionalities and their cross-usage combinations are optimal for testing, risk analysis was done. The risk analysis was focused on libraries' functionalities. The testing scope was formed on the basis of expert opinion and risk analysis.

As a part of this thesis, automated test framework and tests code structure was mapped. Automated tests framework structure was adjusted and tools that are used in test codes were updated. In addition, test code structure was optimized and code refactored.

Risk analysis and code refactoring were one time performed actions. In order to make sure that automated test framework and testing scope would meet the needs and requirements, actions have to be performed regularly. It is necessary to monitor the functionalities of libraries and possibilities. Also, analysing risks and monitoring the usage of tools is required to avoid the situation, where it is unclear which functionalities are tested and which are not.

Kasutatud kirjandus

[1] “Sertifitseerimiskeskus. SK ajatempliteenuse ajatembelduspõhimõtted.,” [WWW] https://sk.ee/upload/files/sk_atp_et-2_0.pdf. [10 05 2014].

[2] “RSpec.info,” [WWW] http://rspec.info/. [10 05 2014].

[3] “ID- kaart ja Digi-ID. Sinu isikutunnistus.,” [WWW] http://id.ee/index.php?id=30056 [10 05 2014].

[4] “ID-tarkvara.,” [WWW] https://installer.id.ee/ [10 05 2014].

[5] “Ruby 1.9.1.0 News.,” [WWW] http://svn.ruby-lang.org/repos/ruby/tags/v1_9_1_0/NEWS [10 05 2014].

[6] “RSpec Core 2.14 Changelog.,” [WWW] https://www.relishapp.com/rspec/rspec-core/v/2-14/docs/changelog. [10 05 2014].

[7] G. Donohue, “Creating S.M.A.R.T Goals – Top Achievement.,” [WWW] http://topachievement.com/smart.html.

[8] “The concept of risk-based testing and its advantages and disadvantages. ICT Standard.,” [WWW] https://www.ictstandard.org/article/2011-10-25/concept-risk-based-testing-and-its-advantages-and-disadvantages. [10 05 2014].

[9] “Software Testing Types.,” [WWW] http://www.aptest.com/testtypes.html. [10 05 2014].

[10] “Digidoc failivormingud: DDOC, BDOC, CDOC.,” [WWW] http://id.ee/index.php?id=30289. [10 05 2014].

[11] “Eesti Standardikeskus. BDOC. Digitaalallkirja vorming.,” [WWW] http://www.evs.ee/tooted/evs-821-2009. [10 05 2014].

[12] “Riigi teataja. Digitaalallkirja seadus.,” [WWW] https://www.riigiteataja.ee/akt/694375. [10 05 2014].

[13] “Elliptic Curve Cryptography (ECC).,” [WWW] https://www.certicom.com/index.php/ecc. [10 05 2014].

[14] “Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring. Cybernetica.,” [WWW] http://www.id.ee/public/kruptograafiliste_algoritmide_elutsukli_uuring_II.pdf. [10 05 2014]

[15] “ID-kaardi tarkvara poolt toetatud kaardilugejad.,” [WWW] http://www.id.ee/?id=36180 . [10 05 2014].

[16] “ID-tarkvara versioonides toetatud op. süsteemid.,” [WWW] http://id.ee/?id=36138 . [10 05 2014].

[17] “DigiDoc teegid - Java teek - jdigidoc (ver 3.8.1.709).,” [WWW] http://www.id.ee/index.php?id=30393 . [10 05 2014].

[18] “DigiDoc teegid - c-teek - libdigidoc (ver 3.8.0.1135).,” [WWW] http://www.id.ee/index.php?id=30391 . [10 05 2014].

[19] “DigiDoc teegid - c++ teek - libdigidocpp (ver 3.8.0.1209).,” [WWW] http://www.id.ee/index.php?id=36483. [10 05 2014].

[20] “DigiDoc teegid - COM teek (ver 3.7.2.1115).,” [WWW] http://www.id.ee/index.php?id=30392 . [10 05 2014].

[21] “DigiDoc teegid - NDigiDoc teek.,” [WWW] http://www.id.ee/index.php?id=35856. [10 05 2014].

[22] “JDigiDoc Programmer’s Guide. Library Version: 3.8.1,” 27 02 2014. [WWW] http://www.id.ee/public/SK-JDD-PRG-GUIDE.pdf. [10 05 2014].

[23] “CDigiDoc Programmer’s Guide. Library Version: 3.8,” 11 12 2013. [WWW] http://id.ee/public/SK-CDD-PRG-GUIDE.pdf. [10 05 2014].

[24] “Libdigidocpp Programmer’s Guide. Library Version: 3.8,” 9 12 2013. [WWW] http://id.ee/public/SK-CPP-PRG-GUIDE.pdf. [10 05 2014].

[25] J. Ilves, “NDigiDoc Programmers Guide.,” [WWW] http://id.ee/public/NDigiDoc.pdf. [10 05 2014].

[26] “DigiDoc COM Programmer’s Guide. Library Version: 3.7,” 22 01 2013. [WWW] http://id.ee/public/SK-COM-PRG-GUIDE.pdf. [10 05 2014].

[27] G. Ghanakota, “TESTING FRAMEWORKS,” [WWW] http://www.cs.colorado.edu/~kena/classes/5828/s12/presentation-materials/ghanakotagayatri.pdf. [10 05 2014].