TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika eriala

Tiit Pikma

Interneti teel hääletamise individuaalne kontrollitavus

Bakalaureusetöö (6 EAP)

Juhendajad: Sven Heiberg, MSc
Sven Laur, PhD

Autor: . . . . . " . . . . . " mai 2012

Juhendaja: . . . . . " . . . . . " mai 2012

Juhendaja: . . . . . " . . . . . " mai 2012

Lubada kaitsmisele

Professor: . . . . . " . . . . . " mai 2012

Tartu 2012

Sisukord

Peatükk 1

Sissejuhatus

Eestis on alates 2005. aastal toimunud kohaliku omavalitsuse volikogu valimistest olnud võimalus lisaks paberile ka interneti teel hääletada[1]. Hääleõiguslikud inimesed saavad eelhääletamise perioodil kasutada arvutit ning ID-kaarti või mobiil-ID'd, et teha oma valik.

Viimasel ajal on aga päevakorda tõusnud küsimus interneti teel hääletamise turvalisuse kohta. Elektrooniliselt hääletades kaasneb vajadus usaldada hääletamiseks kasutatavat arvutit, millel võib aga olla spetsiaalne kahjurvara, mis muudab kasutaja häält. Kui selline kahjurvara käivitatakse piisavalt suurel hulgal hääletamiseks kasutatavatest arvutitest, on nii võimalik muuta valimiste tulemusi.

1.1 Individuaalne kontrollitavus

Üks vahend häält muutvate rünnete ehk manipuleerimisrünnete vastu võitlemiseks on lisada hääletamise mudelisse individuaalne kontrollitavus. Kontrollitavus võimaldab veenduda mõnes hääle omaduses, kusjuures erinevad meetodid näitavad erinevaid omadusi.

Popoveniuci et al. järgi peavad täielikult kontrollitavatel valimistel olema kuus omadust[2]:

1. valijale antud sedel on korrektselt vormistatud (*presented ballots are well-formed*),
2. valija on sedeli korrektselt täitnud (*cast ballots are well-formed*),
3. valija antud hääl on sama, mis süsteem vastu võtab ning salvestab (*recorded as cast*),
4. hääled loetakse kokku nii nagu nad on salvestatud (*tallied as recorded*),

5. kolmas ja neljas kontroll rakendatakse samale sedelite hulgale (*consistency*),

6. igale salvestatud sedelile on võimalik rakendada kolmandat kontrolli (*each recorded ballot is subject to the "recorded as cast" check*).

Kõigi nende omaduste olemasolul on süsteem täielikult kontrollitav (*end-to-end verifiable*) ning kui ükski kontrollidest ei ebaõnnestu on ülimalt ebatõenäoline, et valimiste tulemus erineb oluliselt õigest tulemusest.

Käesolev töö keskendub vaid kolmandale kontrollile: hääl, mille valija annab, peab olema sama, mis hääletamise süsteem vastu võtab ning salvestab. See tähendab, et igal hääletanud isikul on võimalus hiljem ise üle kontrollida, kas ta hääl salvestati nii nagu tema seda ette nägi või ei. Keegi teine ei saa ka seda kontrolli tema eest teha, kuna ta on ainus, kes teab, kuidas ta hääletas ehk kas talletatud hääl on õige või ei.

Oluline on märkida, et kolmas omadus üksinda ei taga hääle arvesse minemist: valija häält võib vaatamata korrektselt serverisse talletamisele tulemuse kokkulugemise hetkel ignoreerida. Et kõik talletatud hääled kindlasti ka arvesse läheksid on vaja lisaks neljandat omandust. Olemaks kindel, et kõik hääled jõudsid valijatelt lugemiseni ilma, et ühtegi oleks muudetud, kustutatud või lisatud, on vaja vähemalt kolmandat kuni kuuendat omadust. Töö käigus oletame, et kõik nõuded peale kolmanda on rahuldatud.

Kuna kedagi ei saa sundida oma häält kontrollima, siis pole võimalik kindel olla, et iga viimane kui üks antud häältest on üle kontrollitud. Seda pole aga vaja, kuna piisab väikesest hulgast auditeeritud häältest, et saaks piisavalt suure tõenäosusega olla kindel, et pole muudetud nii palju hääli, et mõjutada valimiste tulemusi. Selles on võimalik veenduda lihtsa kombinatoorikaga: kui on muudetud 100 häält, siis selleks, et 95% tõenäosusega leida üks muudetud häältest, piisab, kui oma häält kontrollib umbes 3% hääletanutest[3]. Ehk näiteks, kui hääletas 140000 inimest ning 100 häält muudeti, siis selle avastamiseks piisab kui 4200 inimest oma häält kontrolliksid. Siin on aga oluline ära märkida, et need numbrid kehtivad ainult siis, kui kontrollitavad hääled on juhuslikud ning ühtlase jaotusega.

Hetkel aga puudub Eestis valijal igasugune võimalus oma häält kontrollida.

1.2 Töö eesmärk

Töö eesmärgiks on tuua interneti teel hääletamise lihtsustatud mudelisse sisse individuaalne kontrollitavus ning uurida, kuidas see mõjutab eelnevalt toiminud ründeid.

Selle võimaldamiseks tuleb esimesena luua mainitud mudel, milles saab valimisi läbi viia, ning rünne, mis valimiste tulemusi muudab. Hiljem lisandub mudelisse kontrollitavus kahe alternatiivse rakenduse näol, millel mõlemal on omad head ja vead. Kui need kaks hääle kontrollimise varianti suudavad tuvastada ründe poolt muudetud häält, siis on individuaalne kontrollitavus mudelisse edukalt lisatud.

Seejärel on vaja uurida uute rakenduste poolt sisse toodud ohtusid ning hinnata, kas need on tõsisemad kui ohud, mis ära hoitakse (ehk manipuleerimisründed). Kui jah, siis tuleb teha järeldused, kas uued ohud on implementatsiooni spetsiifilised või on need üldisemad ning eksisteerivad igasuguse kontrollitavuse korral.

Kui probleem on vaid konkreetses prototüübis, siis on ehk võimalus kuidagi olukorda parandada. Kui oht eksisteerib aga igasuguse kontrollitavuse korral, siis on see olemas ka Eesti mudelis ning tuleks vältida individuaalse kontrollitavuse lisamisest.

Töö tulemused võiksid näidata, kui hästi saab kontrollitavust lisada Eestis kasutusel olevasse interneti teel hääletamise mudelisse. Seejuures tuleb silmas pidada ka probleeme, mis võivad tekkida Eesti mudelis, ent mitte lihtsustatud variandis.

Kogu töö käigus loodud kood on lisas 1.

Peatükk 2

Hääletamise prototüüp

Selleks, et uurida kontrollitavuse mõju hääletamise mudelile, tuli esimese asjana luua viimase lihtsustatud versioon. Loodud prototüüpi saab kasutada, et näha ning teha järeldusi erinevate muudatuste mõjude kohta.

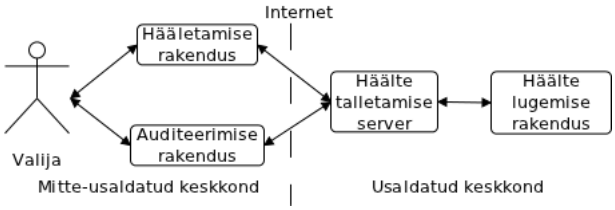
2.1 Interneti teel hääletamise mudel

Prototüüp koosneb algselt kolmest elemendist: hääletamise rakendus, häälte talletamise server ning häälte lugemise rakendus. Nende koostöös saab läbi mängida hääletamise lihtsustatud stsenaariumi, kus valijad kasutavad hääletamise rakendust, et saata oma hääled talletamise serverisse. Igal isikul on võimalus mitu korda hääletada–sellisel juhul jääb viimane valik kehtivaks ning eelnevad ignoreeritakse. Kui hääletamine loetakse lõppenuks, siis kasutatakse häälte lugemise rakendust, et leida valimiste tulemus.

Kui esialgne mudel on olemas, siis tuuakse sisse rünne, millega saab valijate hääli muuta ilma, et nad sellest aru saaksid. Ründe tagajärjel erineb häälte lugemise rakenduse poolt kokku arvutatav tulemus hääletanute tahtest.

Individuaalse kontrollitavuse saavutamiseks lisandub mudelisse ka võimalus oma hääle auditeerimiseks, seda kahe alternatiivse rakenduse–korduvkrüpteeriva ning kandidaate läbi vaatava rakenduse–näol. Tänu viimastele on võimalik tuvastada ründe poolt muudetud hääli ning vastavalt reageerida, muutes mudeli algsest variandist turvalisemaks. Komponentide vahelised seosed on välja toodud joonisel 2.1.

Kuna huviobjektiks on just hääletamise keskkond, siis oletame, et nii serveri-poolne (k.a häälte lugemise rakendus) kui ka kliendiga suhtlemiseks kasutatav ühenduskanal on turvalised. Ilmselgelt võib ka nendes nõrkusi olla, ent need jäävad töö skoobist välja.



Joonis 2.1: Interneti teel hääletamise lihtsustatud mudel

2.1.1 Erinevused Eesti mudelist

Käesolev prototüüp realiseerib lihtsamat varianti Eestis kasutusel olevast hääletamise mudelist. Eesti mudelis kasutatakse valija rakendust, et saata oma valik häälteedastamisserverile, mis viib läbi ka kasutajatuvastuse. Sealt edastatakse valik häältetalletamisserverisse. Lõpuks toimetatakse kõik anonümiseeritud hääled vallasrežiimis häältelugemisrakendusse, kus loetakse kokku tulemus[4].

Lihtsustatud mudelis on välja jäetud häälteedastamisserver. Prototüübis suhtlevad hääletamise ja auditeerimise rakendused otse häälte talletamise serveriga. Samuti ei tööta häälte lugemise rakendus eraldi arvuti peal, vaid suhtleb otse häälte talletamise serveriga. Kuna kasutame eeldust, et serveripoolne on turvaline, siis ei muuda see meie tulemusi ning hoiab mudeli võimalikult lihtsana.

Prototüübi rakendused omavad ka teistsuguseid nimesid: kui Eesti mudelis on kasutusel valija rakendus, häältetalletamisserver ning häältelugemisrakendus, siis nende rolle täitvad komponendid on vastavalt hääletamise rakendus, häälte talletamise server ning häälte lugemise rakendus. Nimed on muudetud loetavuse parandamiseks.

2.1.2 Tehnoloogiad

Häälte talletamise server on kirjutatud Pythonis kasutadeks aluseks SocketServerit[9]. Andmebaasi mootoriks on SQLite3[10, 11] ning XML kujul sõnumeid töödeldakse xml.dom.minidom[12] mooduliga.

Hääletamise ning auditirakendused on kirjutatud Javas ning ainsaks väliseks sõltuvuseks on Bouncy Castle Crypto API[13].

Häälte lugemise rakendus on kirjutatud Pythonis, suhtleb sama SQLite3 andmebaasiga ning kasutab krüptograafiliste operatsioonide jaoks M2Crypto[14] moodulit.

2.2 Rakenduste suhtlus serveriga

Häälte talletamise serveri ning klientrakenduste vahel suhtlemisel kasutatakse lihtsaid omaloodud olekuga protokolle. Kõik sõnumid saadetakse UTF-8 kodeeringus ning neil on eesliiteks nende pikkus baitides, mis on eraldatud sisust komaga.

Näiteks sõna "sõnum" saadetaks kujul `6,sõnum`. Siinkohal tuleb tähele panna, et sisu pikkuseks on 6, kuna "õ" täht võtab kaks baiti.

Sõnumi sisuks on kas XML märgendatud tekst või lühikese sõned. Ka enamus märgendatud tekstist koosneb üksikutest sõnedest, erandiks on kandidaatide nimekiri, mis saadetakse kujul:

```
nimekiri =  "<candidates>",
                { kandidaat },
            "</candidates>";
kandidaat = "<candidate>",
                "<nr>", number, "</nr>",
                "<name>", sõne, "</name>",
            "</candidate>"
```

Näiteks:

```
<candidates>
    <candidate>
        <nr>101</nr>
        <name>Ivar Kask</name>
    </candidate>
    <candidate>
        <nr>102</nr>
        <name>Juhan Tamm</name>
    </candidate>
</candidates>
```

2.3 Häälte talletamise server

Häälte talletamise serveri ülesandeks on salvestada valijate hääli ning anda klientrakendustele tööks vajalikku infot, näiteks kandidaatide nimekirja. Iga serverisse tulnud ühenduse jaoks käivitatakse eraldi lõim. Ühendust käsitleb SocketServer.BaseRequestHandleri alamklass, mis sisaldab kogu funktsionaalsust.

Nagu nimigi reedab, on serveri põhiliseks eesmärgiks hääletamise rakendusest tulnud valikute talletamine andmebaasi vastavate valijate juurde.

Sellega seoses haldab see ka kandidaatide ning võimalike valijate nimekirju. Kui hääletada sooviva inimese isikukoodi pole valijate nimekirjas, siis edasisest suhtlusest keeldutakse. Kui valija leidub nimekirjas, siis saadetakse hääletamise rakendusele kandidaatide nimekiri, et kasutaja saaks kandidaatide seast valida.

Lisaks pakub server auditeerimise funktsionaalsust. Korduvkrüpteerimist tegeva auditirakenduse korral võrdleb server saadetud krüptogrammi andmebaasi talletatuga. Kandidaate läbi vaatavale rakendusele saadab ta uuesti kandidaatide nimekirja ning andmebaasi talletatud hääle. Mõlema rakenduse puhul vaadatakse tuvastatud isiku viimast, kõige uuema kuupäevaga häält.

Kui server saab mõne teate, mida see ei tea või mille jaoks ta on vales olekus (kuna tegu on olekuga protokolliga), saadab ta vastuseks märgendamata sõne `err` ning sulgeb ühenduse. Kuna iga ühendus on eraldi lõimes, siis ühenduse sulgemine seisneb lihtsalt lõime töö lõpetamises.

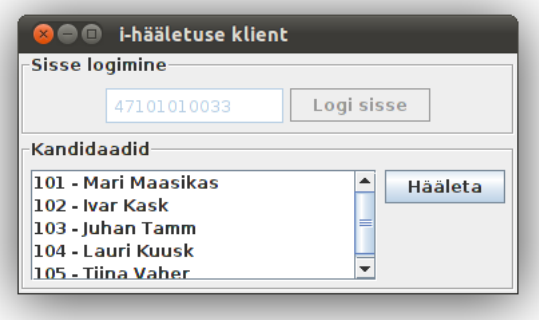
2.4 Hääletamise rakendus

Hääletamise rakenduse esimeseks sammuks on valija tuvastamine. Kasutaja sisestab rakendusse oma isikukoodi ning vajutab nupule "Logi sisse". Serverile saadetakse valija isikukood märgendi `ik` sees, näiteks `20,<ik>12345671234</ik>`. Kui sellise isikukoodiga inimest ei leidu serverisse paigaldatud valijate nimekirjas, siis saadab see vastuseks märgendamata sõne `err` ning sulgeb ühenduse. Rakendus kuvab kasutajale veateate ning lubab uuesti proovida. Kui aga valija leidub nimekirjas, siis saadab server vastuseks kandidaatide nimekirja ning jääb ootama rakenduse järgmist sõnumit.

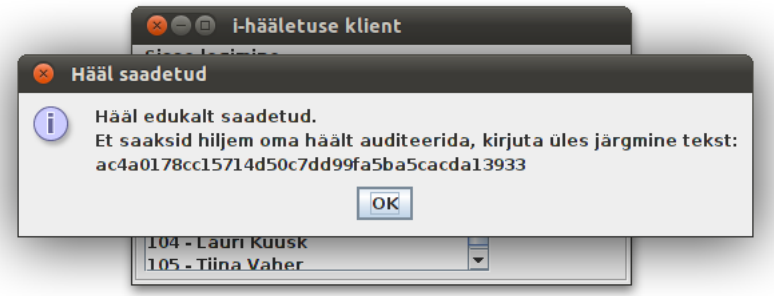
Eelkirjeldatud valija tuvastamine ei asenda turvalist kasutaja autentimist, mida on hääletamisel kindlasti vaja. Käesolevas mudelis aga pole vajadust seda realiseerida, mistõttu eeldame, et turvaline autentimine on olemas.

Rakenduses kuvatakse kõik kandidaadid kujul "number - nimi", mille seast hääletaja valib ühe (joonis 2.2) ning kinnitab vajutades nupule "Hääleta". Vastava kandidaadi number krüpteeritakse RSA-OAEP'ga [15], kodeeritakse Base64 formaati ning saadetakse serverile märgendi `vote` sees. Viimane talletab saadud sedeli oma andmebaasi eelnevalt tuvastatud isiku külge. Kui hääletanud inimesel oli juba andmebaasis sedel, siis on tegu korduvhääletamisega ning olemasolev tehakse kehtetuks.

Viimase asjana saadab server vastuseks märgendamata sõne `ok` ning ühendus suletakse. Rakendus kuvab kasutajale teate, et ta on edukalt hääletanud ning kuvab välja kontrollkoodi, mida on hiljem auditeerimise rakendustes vaja (joonis 2.3). Kui valija ei kavatse oma häält auditeerida, siis ta võib seda



Joonis 2.2: Hääletamise rakendus



Joonis 2.3: Auditeerimiseks kasutatav kontrollkood

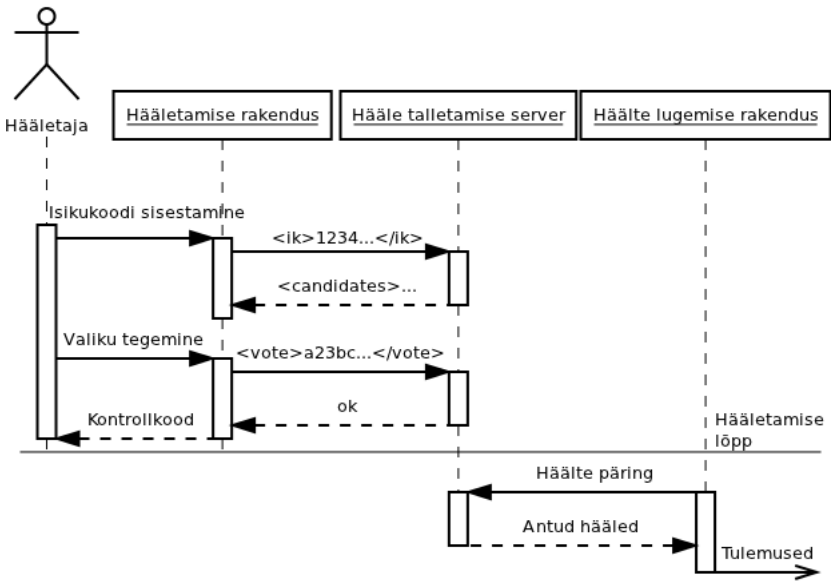
koodi ignoreerida ning lugeda oma hääle antuks.

2.5 Häälte lugemise rakendus

Häälte lugemise rakendus loeb häälte andmebaasist iga valijaga seotud viimase antud hääle, krüpteerib need serveri privaatvõtmega lahti ning loetleb need kokku. Seejärel trükib ta ekraanile kui mitu häält iga kandidaat saanud on.

Siinkohal on oluline märkida, et see rakendus ei vaata häälega seotud valijaid, jättes nad seega anonüümseks. Kuigi käesolevas prototüübis miski otseselt ei takista seda tegemast, on mudeli kirjelduses välja toodud eeldus, et häälte lugemise rakendus on turvaline.

Seni kirjeldatud prototüübi tööd näeb joonisel 2.4.



Joonis 2.4: Hääletamine lihtsustatud mudelis

2.6 Rünne

Järgmisena tuleb mudelisse lisada element, mis muudab valija häält enne serverile saatmist. Kuna valimiste tulemused määravad riigi võimu mitmeks järgnevaks aastaks, siis on manipuleerimisrünnete loomiseks põhjust küllaga. Interneti teel hääletades on paratamatu vajadus usaldada kasutaja arvutit, mis ei pruugi olla kaitstud sellise kahjurvara eest. See oht on olnud põhjuseks mitmete elektrooniliste hääletamissüsteemide, näiteks SERVE, kritiseerimiseks[7, 8]. Eestis on see risk aga tunnistatud aktsepteeritavaks[5].

Ründeks võib olla arvutis aktiivne protsess, mis muudab mälus hoitavaid andmeid hääletamise rakenduse töö käigus, tarkvara paik, mis muudab rakenduse tööd, või muu selline. Eesmärgiks on luua olukord, kus server võtab vastu ning salvestab hääle, mis erineb valija omast.

Edukas rünne peaks muutma valija häält niimoodi, et kasutaja ei saa aru, et seda tehtud on. Rakenduses ei tohiks olla ilmset märki sellest, et serverile saadeti teise kandidaadi number ning ei tohiks olla ka lihtsasti aru saadav, et teine protsess hääletamise rakenduse tööd mõjutas. Vastasel korral saab valija vastavaid osapooli kahjurvarast teavitada ning hiljem turvalisemas keskkonnas uuesti hääletada[1].

Samuti ei tohiks hääle muutmine olla serverilogidest liiga lihtsalt tuvasta-

[1]Eesti mudelis on ka võimalus peale interneti teel hääletamise perioodi lõppu minna veel paberhääletama, mille korral tühistatakse interneti teel antud hääl.

tav–näiteks kui keegi on edukalt hääletanud ning mõni millisekund hiljem uue hääle saatnud, siis on teine hääl tõenäoliselt kahjurvara saadetud ning valimiste läbiviijad avastavad kahjurvara olemasolu.

Lihtsuse huvides kasutatakse prototüübi ründeks tarkvara paika–kuigi aktiivse protsessiga on võimalik enamat korda saata on sellist kahjurvara ka keerulisem luua.

Hääletamise rakenduse terviklust on võrdlemisi lihtne kindlustada, mistõttu ei ole tõenäoline, et rakendust ennast muutvat paika ei avastata. Hääletamise rakendus aga sõltub Bouncy Castle krüptograafia teegist, mistõttu on vajadus ka seda usaldada. Kuna see teek ei ole hääletamise rakenduse loojate kontrolli all, siis saab seda usaldust ära kasutada ning luua teegile paik, mis muudab selle tööd. Paik sisaldub töö lisas 1 koos ülejäänud prototüübi koodiga. Lisasse on kaasa pandud ka juba paigatud versioon teegist.

Kui Bouncy Castle teegil on rakendatud eelnimetatud paik, siis enne häälele OAEP puhvri lisamist kontrollitakse, kas krüpteeritav sõne koosneb täpselt kolmest arvust. Kui jah, siis sõne asendatakse konstandiga 101. Seejärel minnakse OAEP puhvri lisamisega ning RSA krüpteerimisega edasi tavalisel moel. Seega tagastatakse hääl kandidaadi 101 poolt, ent ilma üle kontrollimata pole seda võimalik tuvastada. Seda ei saa aga üle kontrollida, kuna klientrakendusel puudub serveri salajane võti krüptogrammi avamiseks.

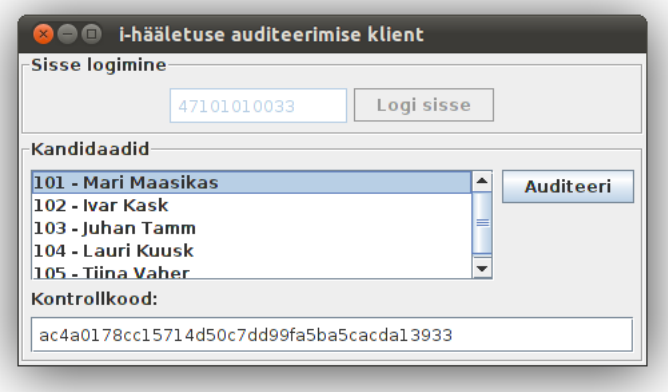
Seega on prototüübi vastu olemas rünne, mille tagajärjel antakse hääl alati kandidaadi 101 poolt. Lisaks jätab see klientrakenduse puutumata ning ei vaja hääletamise ajal jooksvaid protsesse. Muudetud hääl salvestatakse serverisse ning kasutajal puuduvad vahendid selle kontrollimiseks–seega on rikutud kontrollitavuse kolmas nõue (*recorded as cast*).

2.7 Auditeerimise rakendused

Kuna eelmises punktis toodud rünne muudab häält niimoodi, et ei kasutaja ega klientrakendus sellest aru ei saa, siis on vajadus serverisse talletatud häält hiljem auditeerida.

Auditeerimine viiakse läbi peale hääle andmist ning enne hääletamise perioodi lõppu[2]. Kuna serverisse talletatud hääl on krüpteeritud RSA-OAEP'ga ning valijal puudub võti selle lahti tegemiseks, ei saa lihtsalt vaadata, kas kandidaat on sama, kelle poolt hääletati. Siin tuleb mängu hääletamise hetkel saadud kontrollvõti.

[2]Popoveniuci et al. järgi, peaks olema võimalik kontrolli läbi viia ka peale hääletamise lõppu. Vastupidiselt sellele tasuks auditeerimise akent hoopis rohkemgi piirata. Sellest lähemalt peatükis 3.5.4.



Joonis 2.5: Korduvkrüpteeriv rakendus

OAEP algoritm kasutab juhuslikult genereeritud arvu, et andmed enne krüpteerimist ära maskeerida. Kui kasutatav arv ei muutu, ei muutu ka mask. Kontrollvõtmeks ongi hääletamise hetkel genereeritud juhuarv, millega saame sama maski mis hääletadeski. Kuna RSA krüpteerimisel juhuslikkust ei lisa on samade sisendite korral tulemuseks sama väljund. Seega kui võtta sama kandidaadi number mis hääletadeski ning lisada sellele sama mask, siis on tulemuseks ka sama krüptogramm. Tulemuse võrdlemine serverisse salvestatuga on samaväärne krüptogrammi sees olevate sõnede võrdlemisega. Seega on meil võimalus veenduda selles, et serverisse salvestati sama hääl, mis valija andis.

Auditeerimine peaks toimuma hääletamisest erinevas keskkonnas vältimaks olukordi, kus sama kahjurvara ründab nii hääletamise kui ka auditeerimise rakendust.

Konkreetsed auditeerimise rakendused kasutavad korduvkrüpteerimist ning kogu kandidaatide hulga läbivaatust.

2.7.1 Korduvkrüpteerimine

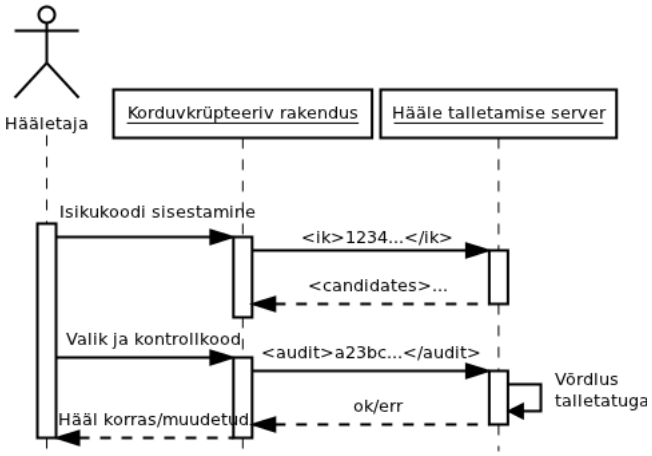
Esimese auditeerimise rakenduse puhul kasutame kontrollimiseks korduvkrüpteerimist. Kasutajaliides näeb välja samasugune nagu hääletamise rakenduse puhul, kuid lisatud on väli kontrollkoodi sisestamiseks (joonis 2.5).

Kõigepealt tuvastab valija ennast isikukoodiga samamoodi nagu hääletamisel. Ehk siis rakendus saadab isikukoodi ning saab vastuseks kas `err` või kandidaatide nimekirja. Esimesel puhul kuvatakse veateade ning palutakse uuesti proovida, teisel nimekiri kandidaatidega.

Seejärel valib kasutaja nimekirjast sama kandidaadi, kelle hääletadeski, ning sisestab kontrollkoodi. Rakendus krüpteerib kandidaadi numbri RSA-OAEP'ga, seekord aga kasutades juhuslikult genereeritud arvu asemel kasutaja sisestatud kontrollkoodi–seega saame sama krüptogrammi, mis hääletadeski. Tulemus kodeeritakse Base64 formaati ning saadetakse serverile märgendi `audit` sees. Server võrdleb saadud krüptogrammi oma andmebaasi salvestatud sedeliga ning saadab vastuseks märgendamata sõne `ok`, kui need kattuvad, või `err`, kui need seda ei tee. Seejärel suletakse ühendus.

Kui rakendus saab vastuseks `ok`, siis kuvatakse kasutajale teade, et valitud kandidaat vastab serverisse talletatule. Vastasel korral kuvatakse hoiatus, et kandidaadid ei kattunud ning palutakse veenduda õige kandidaadi valimises ning kontrollkoodi korrektses sisestamises. Kui kandidaadid endiselt ei kattu, siis tuleks ühendust võtta hääletamist läbiviiva organisatsiooniga.

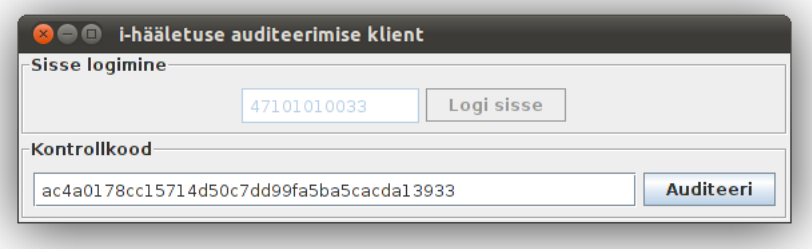
Korduvkrüpteeriva rakenduse sammud on ka näha joonisel 2.6.



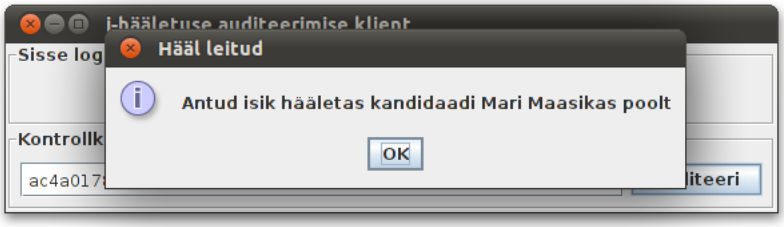
Joonis 2.6: Korduvkrüpteeriva rakenduse sammud

2.7.2 Kandidaatide läbivaatus

Teise auditeerimise rakenduse korral kasutatakse täielikku läbivaatust–*brute-force*'mist. Kasutaja sisestab rakendusse (joonis 2.7) oma isikukoodi, mis saadetakse serverile märgendi `audit` sees. Nagu ennegi, kui sellist valijat nimekirjas ei leidu, saadab server vastu `err` ning sulgeb ühenduse. Kui isikukood aga sobib, saadab server vastuseks selle isikukoodiga seotud sedeli märgendi `audit` sees. Kui sellist sedelit ei leidu, saadetakse `err` ning rakendus kuvab kasutajale teate, et ta pole veel hääletanud.



Joonis 2.7: Kandidaate läbi vaatav rakendus

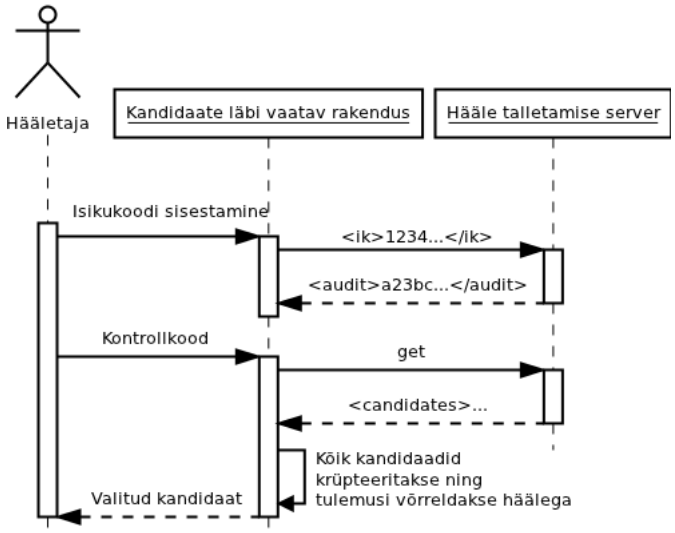


Joonis 2.8: Valija hääle kuvamine

Seejärel sisestab valija rakendusse oma kontrollkoodi. Rakendus saadab serverile märgendamata sõne `get`, millele server saadab vastuseks kandidaatide nimekirja. Siinkohal on oluline tähele panna, et kontrollkoodi serverile ei saadeta. Kõik nimekirjas olevad kandidaadid krüpteeritakse kasutades kontrollkoodi OAEP juhuarvuna ning iga krüptogramm võrreldakse serverist saadud sedeliga. Kui mõni neist kattub, siis kuvatakse välja vastav kandidaat ning öeldakse, et valija hääl on tema poolt (joonis 2.8). Siinkohal saab siis inimene ise oma peaga teha selle kontrolli, kas ta ka hääletas selle kandidaadi poolt või ei.

Kui ükski krüptogramm ei kattu saadud sedeliga, siis öeldakse kasutajale, et kas kontrollkood või serverisse talletatud sedel on vigane. Mõlemal juhul peaks ta uuesti hääletama.

Kandidaate läbi vaatava rakenduse sammud on näha joonisel 2.9.



Joonis 2.9: Kandidaate läbi vaatava rakenduse sammud

Peatükk 3

Kontrollitavuse mõju hääletamise mudelile

Loodud prototüübi põhjal saab uurida individuaalse kontrollitavuse lisamise mõju nii ründele kui ka hääletamisele. Lisaks tuleb ka vaadelda sellega kaasnevaid nüansse nagu näiteks millal üldse lubada auditeerimist ning kuidas see teha võimalikult mugavaks.

3.1 Auditeerimise keskkond

Eduka auditeerimise üheks eeltingimuseks on auditeerimise ja hääletamise keskkondade lahus hoidmine. Konkreetsemalt tähendab see, et auditeerimisrakendus käivitatakse teises seadmes kui hääletamise rakendus.

Nõue on vajalik vältimaks olukordi, kus kahjurvara muudab valikut nii hääletamisel kui ka auditeerimisel. Kuigi pole välistatud, et mõlemal kasutatud seadmel on sama kahjurvara, on tõenäosus selleks oluliselt väiksem. Lisaks saame sellega takistada (või vähemalt teha väga keeruliseks) nende protsesside omavahelist suhtlust. Kuigi prototüübis kasutatud näiteründel seda vaja ei ole, on see oluline mitmetel keerulistematel lahendustel, et teavitada üksteist kasutaja reaalsest valikust. Näiteks on selline funktsionaalsus kindlasti vajalik kui üritada luua rünnet, mida kandidaate läbi vaatav auditeerimise rakendus ei tuvastaks: valijale on vaja kuvada see valik, mida ta hääletades tegi.

Triviaalseim näide erinevatest keskkondadest oleks, kuidas hääletatakse ühe arvuti peal ning auditeeritakse teisel. Et ründeid veel keerulisemaks teha, oleks ka hea, kui need arvutid ei asuks samas kohtvõrgus. Ent pole tingimata vajalik, et mõlemaks seadmeks on just arvuti–sellest täpsemalt peatükis 3.5.1.

3.2 Auditeerimise rakenduste head ja vead

Järgnevalt on kirjeldatud kahe loodud auditeerimise rakendused head ning halvad küljed nende käest saadava lisainfo, nende pihta suunatud rünnete keerukuse, vajaliku usalduse ning veaohtlikkuse seisukohalt.

3.2.1 Korduvkrüpteerimine

Korduvkrüpteeriva rakenduse eeliseks on see, et kontroll toimub serveri poolel, mis tähendab, et kontrollitavat krüptogrammi auditeerijale ei tagastata. Kuigi on vähetõenäoline, et isegi krüptogrammi kätte saades saab ründaja sellega midagi peale hakata, on parem kui tal seda pole.

Siinkohal tekib vajadus usaldada serverit, et see leidis õige vastuse, aga igasugusel läbi serveri käiva auditeerimise puhul on meil niikuinii vajadus usaldada selle pakutud infot.

Korduvkrüpteerimise puuduseks on aga see, et kasutajale kuvatakse lihtsalt "jah" või "ei" vastus ilma detailideta, mis tähendab, et valija peab talle kuvatavat pimesi uskuma. Ekraanil olev info aga ei pruugi olla sama, mis serverilt saadi. Siin võib olla lihtne luua rünnet, mis paneb rakenduse alati "jah" vastust kuvama.

Samuti on siin oht, et kasutaja unustab, kelle poolt ta alguses hääletas, ning võib teisel korral muu kandidaadi valida, mille peale talle kuvatakse teade, et tema hääl on vale.

3.2.2 Kandidaatide läbivaatus

Kandidaate läbi vaatava rakenduse eeliseks on see, et valijale kuvatakse "jah/ei" vastuse asemel mingi kandidaadi nimi. Siinkohal saab valija ise veenduda, kas tegu on sama kandidaadiga, kelle poolt ta oma hääle andis. Lisaks pole vajadust kandidaati uuesti valida, tänu mille saab vältida olukordi, kus teisel korral võetakse nimekirjast keegi teine.

Puuduseks aga on see, et server saadab auditeerijale valijaga seotud krüptogrammi. Nagu enne öeldud, siis on vähetõenäoline, et ründaja sellega midagi teha saab, ent mida vähem infot lekitatakse seda parem.

Juhul kui ründajal on olemas kontrollkood, siis saab ta väikese vaevaga teada, kelle poolt hääletati (sama moodi nagu teeb seda rakendus). Teisalt on korduvkrüpteeriva rakenduse puhul võimalik peale panna piirang, mitu korda mingit häält auditeerida saab. Seega kui ründaja näiteks kolme esimese korraga ära ei arva, kelle poolt hääletati, ei saagi ta seda teada. Kuigi see lekitab infot selle kohta, kelle poolt ei hääletatud, on see siiski parem kui see, et valik saadakse lihtsalt teada.

3.3 Auditeerimise mõju ründele

Mõlema auditeerimise rakenduse eeliseks on, et kui neid käivitada arvutis, millel pole auditeerimisrakendusele suunatud kahjurvara, siis suudavad need tuvastada hääletamisel muudetud häält ning sellest kasutajale teada anda. See lubab valijal vastavatele osapooltele ründest teada anda ning uuesti hääletada või siis peale eelhääletamist osaleda tavalisel paberhääletusel, kus antud häälel on suurem prioriteet kui interneti teel antul.

Prototüübis loodud hääletamise rakenduse ründe eripäraks on, et sama kahjurvara mõjutab nii hääletamise kui auditeerimise rakendusi. Seetõttu on vaja kaaluda ka olukordi, kus kahjurvara on paigaldatud auditeerimise keskkonda.

Kui hääletamiseks kasutatavas keskkonnas pole kahjurvara, ent auditeerimiseks kasutatavas on, siis kuvatakse valijale teade, et hääled ei lange kokku. Kuigi serverisse on talletatud valija õige hääl, soovitatakse tal siiski uuesti hääletada. Sellest ei tulene ka midagi halba, kuna kasutajad võivad korduvhääletada palju tahavad. Siin on ka lisanäide sellest, miks tuleks auditeerimist ning hääletamist hoida erinevates keskkondades: kui valija nüüd hääletaks uuesti auditeerimiseks kasutatud arvuti peal, millele on kahjurvara, siis muudaks ta oma korrektse hääle valeks.

Kui mainitud rünnet on aga läbi viidud nii hääletamisel kui ka auditeerimisel kasutatud keskkondadel, siis rakenduseti tulemused erinevad:

- Korduvkrüpteeriva rakenduse korral jääks rünne tuvastamata, kuna hääletamise keskkonnas loodud krüptogramm langeks kokku sellega, mis luuakse auditeerimise keskkonnas.

- Kandidaate läbi vaatava rakenduse korral väidetaks kasutajale alati, et ta hääletas nimekirjas esimesel kohal oleva kandidaadi poolt–see tähendab, et juhul kui valitigi too kandidaat, siis jääb rünne tuvastamata, vastasel korral see avastatakse.

Väga lihtsa muudatusega on ka võimalik pääseda olukordadest, mil auditeerimise rakendused rünnet ei leia: tuleb lasta kõik kandidaadid sama kontrollkoodiga ära krüpteerida ning omavahel neid võrrelda. Kui vähemalt kaks neist langevad kokku, siis on midagi valesti ning kasutajat teavitatakse, et selles arvutis pole ohutu auditeerida.

Sellise täienduse tulemusena töötaks vaid selline rünne, mis tagastab muudetud hääle krüptogrammi ainult kasutaja esialgse valiku korral. Kui kahjurvara eesmärk on muuta kõik hääled samaks (nt kandidaat 101 poolt), siis on vajalik krüptogramm peale kontrollkoodi sisestamist lihtsasti leitav.

Selleks aga, et tagastada seda vaid valija esialgse kandidaadi puhul, on vaja teada, kelle poolt hääletati. Kuna aga need kaks rakendust töötavad erinevates keskkondades, siis on valiku teada saamine praktikas piisavalt keeruline, et selle riskiga leppida.

Seega on näha, et väheste täiendustega on auditeerimise rakendused võimelised tuvastama rünnet antud hääle vastu ning individuaalne kontrollitavus võimalik.

Kui on soov olla eriti põhjalik, siis pole vaja piirduda ühe auditeerimise meetodiga, vaid võib ka kasutada mõlemat rakendust–sellest aga võib tuleneda muid probleeme, näiteks kasutajate segadus või ründed rakenduste vastu, mida ainult ühe kasutamisel ei esineks. Sellised probleemid jäävad töö skoobist välja ning neid ei käsitleta.

3.4 Uued ohud

Kuna loodi kaks uut rakendust ning hääli talletavale serverile lisati neid toetav funktsionaalsus, siis tuleb ka uurida, millised uued ohud võivad sellega kaasneda.

3.4.1 Hääle lekitamine

Kontrollitavuse ning sellega kaasnevate auditeerimise rakenduste lisamisega tekivad näiliselt mitmed võimalused valija hääle teada saamiseks. Nii kaua aga, kui valija hoiab oma kontrollkoodi saladuses, ei ole tema antud häält võimalik leida.

Korduvkrüpteerimine

Korduvkrüpteeriva rakenduse puhul saadetakse serverile mingi baidijada ning ta vastab, kas see on sama, mis valijal talletatud. Oletame nüüd, et ründaja suudab end mingit moodi serverile kellegi teisena tutvustada–näiteks hoiab kasutaja oma ID-kaarti kogu aeg arvutis ning seal on kahjurvara, mis oskab seda kasutada ja on eelnevalt klaviatuurivajutustelt PIN koodid tuvastanud. Seega on ründajal võimalik serverilt kinnitust küsida igasuguse baidijada kohta. Vaatamata sellele on aga tõenäosus, et ta suudab õige jada leida, kaduvväike: kuigi kandidaatide nimekiri on lõplik, kasutatakse RSA-OAEP krüpteerimist, mis lisab krüptogrammi juhuarvu, mida meie kasutame kontrollkoodina. Ilma viimast teadmata on võimatu sobivat baidijada taasluua.

Kandidaatide läbivaatus

Oletame sarnaselt korduvkrüpteeriva rakendusega, et ründaja suudab end serverile tutvustada kui rünnatav. Sel juhul tagastab server talle lihtsalt rünnatava valijaga seotud krüptogrammi, mis sisaldab antud häält. Sellest aga on ründajale vähe kasu, kuna nii kaua, kui kontrollkood on salajane, ei ole krüptogrammi põhjal võimalik tuvastada, kelle poolt hääl antud on: ilma serveri privaatvõtmeta seda avada ei saa ning ilma kontrollkoodita ei saa ka korduvkrüpteerimist teha, et võrrelda kõikide kandidaatide krüptogrammidega.

3.4.2 Ründed auditeerimise rakenduste vastu

Ründaja võib ka levitada kahjurvara, mis on spetsiaalselt mõeldud auditeerimise rakenduste töö häirimiseks ning kasutajale vale info kuvamiseks.

Korduvkrüpteerimine

Kasutaja vaatenurgast sisestab ta rakendusse oma kontrollkoodi, valib sama kandidaadi, kelle hääletadeski, ning saab vastuse, kas kõik on korras või ei. Siinkohal on väga lihtne luua rünnet, mis tunneb ära hetke, mil kuvatakse "jah/ei" vastus, ning näitab selle peal lihtsalt pilti "jah" vastusest. Seega jääb kasutajale alati mulje, et kõik on korras.

Sellise ründe vastu saab aga lihtsalt, paludes valijatel lisaks oma õigele valikule proovida auditeerida ka mõnda vale valikut. Kui kasutaja teeb neid kahte tegevust juhuslikus järjekorras, siis pole ründel õiget valikut teadmata võimalik tuvastada, kummal korral peab ta "jah" vastuse kuvama. Õige valiku teadmiseks peaks tal aga olema ühendus hääletamise keskkonnaga, mida me juba enne tuvastasime olevat kui piisavalt keeruline.

Kandidaatide läbivaatus

Kandidaate läbi vaatava rakenduse puhul on ründajal oluliselt keerulisem luua süsteem, mis jätab kasutajale auditeerides vale mulje, et ta hääl on korrektselt talletatud. Kuna kasutajale kuvatakse "jah/ei" asemel hoopis valitud kandidaat, siis toimub korrektsuse kontroll kasutaja peas, mitte arvutis–tema ise on ainus, kes teab, kas kuvatud kandidaat on õige.

Siin on ainus võimalus kasutajat petta kuvades talle tema tehtud valikut, mida ründav kahjurvara aga ei tea. Nagu ennegi, peame me ka hääletamise keskkonnas töötavalt kahjurvaralt õige vastuse küsimist piisavalt keeruliseks.

3.4.3 Mõju häälte ostmisele

Nüüd, kus valijal on võimalus kinnitada oma häält, võib jääda mulje, et häälte ostmine on kohe oluliselt lihtsam, mis tegelikult ei ole nii, kuna säilib võimalus korduvhääletamiseks.

Isegi kui valija annab ostjale kõik vajalikud andmed, et viimane saaks hääle talletamist serverisse ise üle kontrollida, ei erine see sellest, kui ostja seisab tal hääletamise hetkel kõrval või laseb omale saata näiteks videolindistuse sellest, kuidas hääletati. Mõlemal juhul teab ostja vaid seda, mis valik tehti mingil kindlal hetkel–alati on võimalus, et müüja hiljem muudab oma hääle teise osapoole teadmata.

Samuti on mõlemal juhul ainsaks võimalikuks variandiks hääle ostmises kindel olla, lasta müüjal hääletada vahetult enne eelhääletamise perioodi lõppu ning võtta neilt ära kõik isikut tõendavad dokumendid kuni hääletamise lõppemiseni. Viimane on vajalik, et nad ei saaks minna paberhääletama. Selline lahendus aga ei skaleeru üldse, kuna pole võimalik kõigi müüjate juures hääletamise lõppedes viibida.

Seega on näha, et kuigi valija mingil hetkel antud hääle kontrollimine võib olla lihtsam, ei muutu kontrollitavuse tõttu häälte ostmine juures midagi ning tegu on sama ebakindla tegevusega kui praegu.

3.4.4 Auditeerimisel eksimine

Individuaalse kontrollitavusega lisandub ka oht, et valijad hakkavad kontrolli läbiviimisel eksima. Kui hääletaja ei mäleta enam oma valitud kandidaati (või väidetakse, et hääletati kellegi teise poolt), siis ütlevad rakendused ka korrektselt talletatud hääle puhul, et valikut on muudetud. Sellises olukorras tuleb kindlasti koheselt pöörduda valimiste läbiviijate poole.

Välja selgitada, kas tegu oli kasutajaveaga, tahtliku eksimisega (valimiste maine ründamiseks) või kahjurvaraga, on läbiviijate kohustus ning organisatsiooniline küsimus, mistõttu seda siin töös ei käsitleta. Minimaalseks nõudeks aga oleks, et valija ei ole sunnitud avaldama, kelle poolt ta hääletas.

3.5 Kaalumist vajavad nüansid

Järgnevalt on loetletud mitmed aspektid, mida saaks töö raames tehtud prototüübi juures muuta või sellele lisada. Need kõik toovad endaga kaasa teatud eeliseid, ent ka puuduseid, ning neid tuleks kaaluda hääletamisele kontrollitavuse lisamisel. Tegu pole kaugeltki ammendava nimekirjaga ning leidub mitmeid alternatiive ja lisavaatenurki, mida võiks silmas pidada.

3.5.1 Kasutatavad seadmed

Eksisteerib range nõue, et hääletamine ning auditeerimine toimuksid kahes erinevas keskkonnas. Küll pole aga sellist nõuet, et mõlemaks kasutatavaks seadmeks oleks arvuti.

Näiteks on mõeldav, et valija annab oma hääle arvutis, ent auditeerimine viiakse läbi nutitelefonis. See tagab mitte ainult selle, et hääletamine ning auditeerimine tehakse kahes erinevas keskkonnas, vaid ka selle, et need keskkonnad on üksteisest erinevad–nutitelefonide operatsioonisüsteemid erinevad oluliselt arvutite omadest (vähemalt kirjutamise hetkel).

Siinkohal on ilmselgeks vastuväiteks, et paljudel inimestel ei ole nutitelefone, mis võimaldaksid sellist auditeerimist–see ei pea aga olema ainus variant. Telefonis auditeerimine võiks olla lisaks arvutile. Samuti saab sellele argumendile vastukaalus tõstatada küsimuse, kas on vähem inimesi, kellel on arvuti ja nutitelefon, kui inimesi, kellel on kaks arvutit.

3.5.2 Kontrollkoodi talletamine ning transport

Üheks oluliseks küsimuseks mudelis kasutatava individuaalse kontrollitavuse juures on, et kuidas kontrollkood hääletamise keskkonnast auditeerimise keskkonda toimetada.

Käsitsi üles kirjutamine või trükkimine

Selle töö jaoks mindi kõige lihtsamat teed pidi, kus valijale kuvatakse krüpteerimisel kasutatud juhuarv kuueteistkümnendsüsteemis. Seejärel kirjutab kasutaja selle lihtsalt paberi ja pastakaga üles ning auditeerides sisestab rakendusse. Lihtne variatsioon sellest oleks üles kirjutamise asemel kood printeriga välja trükkida.

Kuueteistkümnendsüsteem sai valitud, kuna siis on number võrdlemisi lühikene ning veakindel. Üks variant oleks kasutada sama Base64 kodeeringut mis rakenduste suhtluselgi, tänu mille oleks kood lühem, ent siis oleks see ka tõstutundlik, mis võib tekitada vigu koodi üles kirjutamisel.

Irdseadmed

Et üldse eemaldada inimlik eksimus kontrollkoodi transpordi küsimusest, siis on ka võimalik see lihtsalt faili kirjutada ning lasta valijal see irdseadmega teise arvutisse toimetada. Seal loeb auditeerimise rakendus koodi failist sisse ning käsitsi ümberkirjutamise vead on kõrvaldatud.

Siin võib aga probleemiks saada see, et kasutaja otsustab näiteks mälupulga kasutamise asemel see fail iseendale lihtsalt e-posti teel saata. See on aga

erakordselt ohtlik, kuna ühendus postiteenusega ei pruugi olla krüpteeritud ning faili saatmist saab pealt kuulata. Lisaks tekib vajadus usaldada teenuse pakkujat, et see säilitaks postkasti talletatud faili salastatuse.

Nii et kuigi irdmeedia kasutamine toimiks, on siin ohud, et kasutaja lekitab kontrollkoodiga faili enesele teadmata.

QR-koodid ning nutitelefonid

Nagu eespool mainitud, siis võib auditeerimist läbi viia ka nutitelefonil. Kui seda varianti kasutada, siis tekib võimalus väga mugavaks ning kiireks auditeerimiseks.

Kui kasutaja on oma hääle ära andnud, siis kuvatakse talle kontrollkood QR-koodina[16]. Ta võtab nutitelefoni, kus töötab auditeerimise rakendus ja ta on end juba serverile tutvustanud, ning teeb koodist pilti. Sealt seest saab rakendus kontrollvõtme ning ta saab jätkata auditeerimisega nii nagu tavaliselt.

Eriti mugav on see näiteks kandidaate läbi vaatava rakenduse puhul. Ta käivitab rakenduse, logib ennast mobiil-ID'ga sisse, teeb talle kuvatavast koodist pilti ning peale natukest mõtlemist ütleb ta telefon talle, kelle poolt ta hääletas.

Enne sellise lahenduse kasutamist tuleks aga teha põhjalik uurimus nutitelefonide turvalisuse kohta ning võimalike rünnete sellise auditeerimise vastu.

3.5.3 Auditeerija tuvastamine

Hääletamise ajal tuvastatakse valija kas ID-kaardi või mobiil-ID'ga. Sellest pole pääsu, kuna on vaja kindlalt selgeks teha, kellega tegu on.

Küll aga on võimalik sellest nõudest lahti saada auditeerimise ajal: näiteks antakse hääletamise ajal kasutajale lisaks kontrollkoodile ka seansivõti. Hiljem auditeerides saadetakse see serverile selle asemel, et end ID-kaardiga tuvastada. Selle põhjal server teab, kelle häält auditeerida ning kuna võti anti välja vaid kindlalt identifitseeritud isikule, siis saab eeldada, et võtme saatjaks on sama isik.

Siinkohal tuleb siis seansivõtit hoida samamoodi saladuses nagu kontrollkoodi. Kui mõlemad korraga kätte saadakse, saab ründaja end valijana tuvastada ning saada teada, kuidas hääletati. Samas, kui on võimalik kontrollkoodi turvaliselt transportida, siis peaks see olema ka seansivõtme puhul.

Seansivõtme eeliseks oleks see, et auditeerimise keskkonnas pole vaja ID-kaardi tarkvara, mis võib mõnes kohas olla keeruline paigaldada.

3.5.4 Kontrolli aken

Kontrollitavuse lisamisel tuleb kindlasti kaaluda seda, kui suur on kontrolli aken ehk kui kaua peale hääle andmist on võimalik seda häält auditeerida. Ilmselgelt peab aken olema piisavalt pikk, et kasutaja jõuaks rahulikult ning kiirustamata transportida kontrollkoodi teise keskkonda ning seda seal kasutada. Kui inimesel pole kodus teist seadet, millega auditeerida, siis peab ta selle ajaga jõudma liikuda teise kohta.

Kindlasti aga oleks vaja seda aega piirata, vältimaks erinevaid probleeme. Lihtsaimaks neist oleks tavaline inimlaiskus: auditeerimist lükatakse nii kaua edasi kuni ta üldse ära unustatakse või on interneti teel hääletamise aeg juba lõppenud ning hääl jääbki kontrollimata.

Samuti saab piiratud kontrolli akent ühendada seansivõtmetega, et teha häälte ostmine keerulisemaks. Kui müüja annab ostjale seansivõtme ning kontrollkoodi, siis saab viimane küll näha, et esimene tõesti hääletas nii. Ent kui kontrolli aken on möödunud, saab müüja uuesti hääletada kartmata, et müüja veel peale seda tema häält auditeerima läheks.

Lisatingimusena tuleks keelata auditeerimine peale interneti teel hääletamise perioodi lõppu. Siis, kui auditeerimise tulemusena avastatakse, et hääl on vale, pole veel liiga hilja et midagi muuta.

Samuti aitab see ära hoida mainerünnakuid: vastasel korral võivad inimesed peale hääletamise lõppu tulla ning valetada, et nende hääl pole tegelikult üldse selline, nagu serverisse talletati. Peale hääletamise lõppu on aga sellistele väidetel reageerimine oluliselt keerulisem.

3.6 Rakendatavus Eesti mudelile

Kuna individuaalse kontrollitavuse lihtsustatud mudelisse aitas tõesti häält muutvat rünnet tuvastada, siis on mõeldav, et see toimiks ka Eesti interneti teel hääletamise mudelis. Küll on lisaks vaja kaaluda mudeli aspekte, mis käesolevas töös kõrvale jäid–näiteks ründed ühenduskanalite või serverite vastu. Nende vastaseks kaitseks on aga Eesti mudelis teised vahendid, mis meie mudelis puudusid.

Kindlasti tuleks kaaluda kõiki välja toodud nüansse ning alternatiivseid lahendusi, ent kokkuvõttes ei kaasnenud nende kahe auditeerimise rakenduse lisamisega ühtegi suurt probleemi ega uut ohtu. Kõik, mis leidusid, olid kas vastuvõetavad või lihtsasti välditavad.

Läbivaks tundlikuks kohaks on aga kontrollkoodi salastatus: sellest tulenevalt peab enne käsitletud kontrollitavuse lisamist kindlasti tegema otsuseid, kuidas kõige paremini seda nõuet rahuldada ning kindlustada, et kasu-

tajad selle ohutult keskkondade vahel transportitud saavad.

Ilmselgelt ei pea auditeerimise meetodid olema täpselt sellised nagu prototüübis, aga kõik, mis lähtuvad samast mudelist, peaksid olema sama edukalt rakendatavad.

Peatükk 4

Kokkuvõte

Mõlemad loodud auditeerimise rakendused olid võimelised tuvastama rünnet lihtsustatud mudelis. Samuti ei toonud kumbki neist sisse suuri uusi ohte, kuna ilma kontrollvõtmeta ei olnud nende rakendustega lisatud funktsionaalsusest ründajale mingit kasu.

Rakendused ise olid ka raskesti rünnatavad, kuna edukas kahjurvara nõuaks infot hääletamise hetkel tehtud otsuste kohta. Kuna hääletamine ja auditeerimine toimuvad kahes erinevas keskkonnas on seda aga raske saavutada, sest nende vahelist suhtlust võimaldada on piisavalt keeruline, et seda riski vastu võtta.

Lõppkokkuvõttes, kui silmas pidada kõiki välja toodud nüansse ning teha kaalutletud otsuseid vastuvõetavate riskide kohta (on võimatu luua täiesti kindlat süsteemi), siis on individuaalne kontrollitavus Eesti mudelisse edukalt rakendatav. Auditeerimiseks kasutatavad rakendused võiva erineda töös loodutest, ent kui mudelis ei toimu suuri muudatusi, siis kõik toimiks.

Kuna töö skoobist jäid välja ründed ühenduskanalite ning serverite vastu, siis see võiks olla ka koht, kust siit edasi minna. Serverite poolele lisandus uus, auditeerimist toetav funktsionaalsus, mis ei paista ründajale uusi võimalusi andvat, ent seda tuleks sügavamalt uurida, et serveri turvalisuses täiesti kindel olla.

Samuti võiks sügavamalt uurida kontrollkoodi turvalist transportimist ning kuidas teha kasutajatele selle salajasena hoidmist võimalikult lihtsaks.

Viimaks tasuks leida mingi lahendus, millega saaks kerge vaevaga ümber lükata pahatahtlike valijate väiteid, et nende hääli on muudetud, kui seda tegelikult juhtunud ei ole.

Peatükk 5

Summary

Individual Verifiability For Internet Voting

Bachelor thesis

Tiit Pikma

Estonia has used internet voting in addition to traditional paper voting in elections since 2005. A problem with the system used today is that there is no way to verify if the vote cast by a voter is recorded in the voting system unmodified. This check has to be performed by the voter as he or she is the only person who knows how he or she voted.

The aim of this thesis was to see if it is possible to add individual verifiability into a simplified version of the Estonian voting model. In this model we assume that the server and communication channels used are secure and we focus on the environment in which the vote is given.

First a prototype was created, which implemented the simplified voting model, consisting of a voting application, a vote storing server and a voting counting application. These three allow users to record votes into a server, where later the results are tallied. Then an attack was added, which modified the votes given through the voting application with the user unable to detect that a change happened.

This, in theory, is a simplified version of the current state of Estonian internet voting. Of course a lot more security is involved in the full model, but if a vote is modified before being sent to the server, a voter has no way of knowing. To counter-act this, two audit applications were added into the simplified model, which can be used by a voter to see what vote is stored in the server.

Since all votes are encrypted with RSA-OAEP before being sent to the

server, recrypting the same string again yields a different result. For the audit applications to work, a control code is given to the user after voting: this is the random number used in the OAEP padding operation. When this code is entered into an audit application, encrypting a string produces the same result as it did during voting. If we now encrypt the same vote as during voting, we get the same cryptogram. The first application then sends this result to the server for comparison, while the second application asks the server for the stored cryptogram so it can perform the comparisons locally. Therefore, we have means to compare the stored vote with the users vote without needing to decrypt the stored vote.

Afterwards the model was analyzed to see the effect of the audit applications. These can be successfully used to discover a changed vote, therefore uncovering an attack on some user's vote and providing means for individual verifiability.

It was also found that if the voters follow some requirements it is sufficiently difficult to create an attack on the audit applications, which would either hide the fact of a changed vote or leak the voter's candidate. These requirements are that they keep the control code secret and that voting and auditing are done in different environments, e.g. on different computer.

The effect on vote buying is also non-existant, since every possibilty for buying a vote that exists in the auditing model, also exists in the current model: therefore no new means for vote buying were introduced.

A number of decision points for implementing individual verifiability were also discussed. These were the environments to use for voting and auditing, the best means for storing and transporting the control code, auditer authentication and time-frame for auditing. These are all points which need to be considered and weighed by anybody wishing to implement verifiability in a voting model since there are no definitive best solutions.

In conclusion it was decided that the kind of individual verifiability used in the thesis could be successfully implemented in the Estonian voting model. The question of control code security would need to be futher investigated before doing so, but no major counter-arguments or drawbacks were found and it would greatly increase the security of elections.

Kirjandus

[1] Vabariigi Valimiskomisjon. Elektroonilise hääletamise statistika 2005 - 2011. http://www.vvk.ee/valijale/e-haaletamine/e-statistika/ (viimati vaadatud 12.05.2012)

[2] Stefan Popoveniuc, John Kelsey, Andrew Regenscheid, Poorvi Vora. Performance Requirements for End-to-End Verifiable Elections. EVT/WOTE 2010.

[3] C. Andrew Neff. Election Confidence. 2003. http://www.verifiedvoting.org/downloads/20031217.neff.electionconfidence.pdf (viimati vaadatud 12.05.2012)

[4] Vabariigi Valimiskomisjon. E-hääletamise süsteemi üldkirjeldus, http://www.vvk.ee/public/dok/Uldkirjeldus_e_haaletamine_092010.pdf (viimati vaadatud 12.05.2012)

[5] Arne Ansper, Ahto Buldas, Aivo Jürgenson, Mart Oruaas, Jaan Priisalu, Kaido Raiend, Anto Veldre, Jan Willemson, Kaur Virunurm. E-hääletamise kontseptsiooni turve: analüüs ja meetmed. 2011. http://www.vvk.ee/public/dok/EH-02-02_2011-01-13.pdf (viimati vaadatud 12.05.2012)

[6] Gerhard Skagestein, Are Vegard Haug, Einar Nødtvedt, Judith E. Y. Rossebø. How to Create Trust in Electronic Voting over an Untrusted Platform. 2006.

[7] David Jefferson, Aviel Rubin D., Barbara Simons, David Wagner. A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE). 2004. http://www.servesecurityreport.org/paper.pdf (viimati vaadatud 12.05.2012)

[8] David Jefferson, Aviel Rubin D., Barbara Simons. A comment on the May 2007 DoD report on Voting Technologies for UOCAVA Citizens. 2007. http://www.servesecurityreport.org/SERVE_Jr_v5.3.pdf (viimati vaadatud 12.05.2012)

[9] SocketServer - A framework for network servers. http://docs.python.org/library/socketserver.html (viimati vaadatud 12.05.2012)

[10] SQLite Home Page. http://www.sqlite.org/ (viimati vaadatud 12.05.2012)

[11] sqlite3 - DB-API 2.0 interface for SQLite databases. http://docs.python.org/library/sqlite3.html (viimati vaadatud 12.05.2012)

[12] xml.dom.minidom - Lightweight DOM implementation. http://docs.python.org/library/xml.dom.minidom.html (viimati vaadatud 12.05.2012)

[13] The Legion of the Bouncy Castle Java cryptography APIs. http://www.bouncycastle.org/java.html (viimati vaadatud 12.05.2012)

[14] M2Crypto - A Python crypto and SSL toolkit. http://sandbox.rulemaker.net/ngps/m2/ (viimati vaadatud 12.05.2012)

[15] RSA-OAEP Encryption Scheme. ftp://ftp.rsasecurity.com/pub/rsalabs/rsa_algorithm/rsa-oaep_spec.pdf (viimati vaadatud 12.05.2012)

[16] About 2D code. http://www.qrcode.com/aboutqr-e.html (viimati vaadatud 12.05.2012)

Lisad

1. Loodud prototüübi kood ning ründeks kasutatav Bouncy Castle paik (kättesaadav internetis, `http://kodu.ut.ee/~pikma/ivoting.zip`).