Dok. A-60-1

Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring

Jan Willemson, Peeter Laud, Aivo Jürgenson, Märt Laur

Tehniline dokument

Versioon 1.1
15. juuli 2011. a.
52 lk

Uuring on koostatud Euroopa Liidu SF programmi “Infoühiskonna teadlikkuse tõstmine” raames Riigi Infosüsteemi Ameti ning Majandus- ja Kommunikatsiooniministeeriumi riigi infosüsteemide osakonna tellimusel.

SISUKORD

1 SISSEJUHATUS

See aruanne on sündinud eesmärgiga koondada teaduskirjanduses ja erinevates rahvusvahelistes uuringutes esitatud soovitused krüptograafiliste süsteemide ja -algoritmide kasutamiseks riigi infosüsteemides. Aruanne on suunatud tarkvaraarhitektidele, IT-audiitoritele jt tehnilistele spetsialistidele, kellel on sõnaõigus uute infosüsteemide loomisel ja vanade turbe tagamisel.

Aruanne on kirjutatud 2011. a, ajahorisondiga ligikaudu viis aastat. See on aeg, mille jooksul jõuavad arvutustehnika arengus ja krüptoanalüüsi meetodites toimuda piisavalt suured muudatused, et kaugemaleulatuvad ennustused usaldusväärsuse kaotaks. Seetõttu tuleb hiljemalt 2016 aruanne uuesti läbi vaadata ja ajakohastada. On oluline, et see dokument ei jääks ühekordseks ettevõtmiseks.

Vajadus

Krüptosüsteemid ei murdu reeglina üleöö, vaid järkjärgult. Samuti juhtub sageli, et mõne primitiivi kasutamine muutub ebaturvaliseks ainult mõnes kasutusstsenaariumis, sellal kui teiste stsenaariumite turvalisus pole vähemalgi määral mõjutatud. Klassikaliseks näiteks on siinkohal räsifunktsioonid, mille kollisioonide leidumine ei tähenda veel, et neid ei võiks kasutada olukorras, kus süsteemi turvalisuse tagamiseks on vaja ainult kindlust originaali leidmise suhtes. Probleem tekib aga sellest, et sageli pole tarkvaraarendaja ega isegi arhitekt piisavalt kompetentne otsustamaks, kas selline stsenaarium on teadaolevate rünnete suhtes immuunne või mitte. Sageli ei anna arendaja või arhitekt endale ka aru, et loodavas süsteemis on kasutusmalle, mis vajavad turvalisuse tagamiseks krüptograafilisi meetodeid, või eiratakse nende mallide korrektset realiseerimist arenduskiiruse huvides. Halvimal juhul otsustab arendaja korrektse lahenduse asemel kasutada omaloodud krüptoalgoritme. Sellistele probleemidele on ka raske jälile jõuda, sest arendaja vaatepunktist on kõik justkui korras ning samas pole organisatsioonid sõltumatust koodiauditist huvitatud, tuues ettekäändeid alates ärihuvidest ja lõpetades riigisaladusega.

Siinkohal väärib märkimist, et krüptograafiakogukonnas on juba enam kui sada aastat tagasi võetud omaks põhimõte, mille sõnastas 1883. aastal lingvist ja krüptograaf Auguste Kerckhoffs [115]. Nn Kerckhoffsi printsiip nõuab, et **krüptosüsteemi enda kirjeldus ei tohi olla salajane, mis tähendab, et kogu saladus peab sisalduma võtmes.** Seda põhimõtet on ajaloos erinevatel põhjustel korduvalt eiratud ning see eiramine on sageli viinud olukorrani, kus muidu nõrka krüptosüsteemi üritatakse kaitsta hämamisega (ingl. k. *security by obscurity*), kuid pärast süsteemi avalikukstulekut on see kiiresti murtud. Üks tuntumaid juhtumeid selles vallas on II maailmasõja aegne Saksa šifrimasin Enigma, kuid hoiatavaid näiteid leiab ka meie päevist, näiteks erinevate raadiovõrkude turvamiseks kasutatud algoritmid A5 ja RC4 (vt jaotised 2.1.2 ja 2.1.3).

Infoturbespetsialistide kaasamine suuremate infosüsteemide loomisesse on tänapäeval vältimatu. Ka Eesti Vabariigi andmekogudele kehtestatud kolmeastmeline etalonturbe süsteem ISKE näeb ette erinevaid krüptograafilisi lahendusi ning nende kasutuselevõtu organisatsioonilisi mehhanisme (vt [33], meede M 2.161 “Krüptokontseptsiooni väljatöötamine”, meede M 4.90

“Krüptoprotseduuride kasutamine ISO/OSI etalonmudeli eri kihtides” jt).

Krüptograafiliste primitiivide kasutamiseks kõrgema taseme protokollide ehitamisel on sageli erinevaid võimalusi ja mitte kõik neist pole ühtviisi turvalised. Keerukust lisab asjaolu, et osaliselt murdunud primitiiv võib osutuda teatud kontekstis jätkuvalt turvaliseks. Otsus selle primitiivi väljavahetamiseks on suuresti majanduslik ning eeldab turvariskidest tuleneda võivate kahjude hindamist. Selle aruande eesmärk ei ole ega saagi olla kõigi Eesti Vabariigi infosüsteemide jaoks niisuguse hinnangu andmine. Aruanne annab vaid alusmaterjali ning mõned lähtekohad vastava analüüsi jaoks; analüüs ise tuleb aga igal üksikjuhul eraldi läbi viia süsteemi arhitektuuri kavandamisel.

Välisriikide kogemus

Krüptograafiliste primitiivide hetkeseisu seirega tegeleb Euroopa Liidu kompetentsikeskus ECRYPT II, mis koondab endas regiooni juhtivate uurimisasutuste spetsialiste. Keskus üllitab igal aastal raporti, kus leiavad kajastamist põhilised krüptograafilised algoritmid ja protokollid ning kust võib leida üldiseid soovitusi nende kasutamiseks. Hetkel kehtivad soovitused on avaldatud 31. märtsil 2010 [47]. Ka käesolev aruanne tugineb olulises osas sellele dokumendile.

USAs koostab sarnaseid aruandeid National Institute of Standards and Technology (NIST) Computer Security Resource Center. Hetkel kehtivaid soovitusi USA valitsusasutuste andmete turvamiseks sisaldab 2009. aasta detsembris koostatud raport [51], mida käesoleva aruande koostamisel samuti kasutatud on. 2005. aastal avaldas USA National Security Agency (NSA) omapoolsed soovitused valitsusasutustes kasutatavate avalike krüptoalgoritmide kohta, mida tuntakse ühisnimetaja Suite B all [12]. Eksisteerib ka eriti tundlike andmete käitlemiseks kasutatav Suite A, kuid selle kirjeldus pole avalik. Suite B algoritmid on standardiseeritud ka mitmete protokollistike osana, näiteks IPsec [132], TLS [175] jt.

Jaapani valitsuse poolt kasutatavate krüptograafiliste primitiivide kohta annab soovitusi CRYPTREC projekt [5], kahjuks pärineb nende viimane inglisekeelne aruanne aastast 2002 [27].

Taust ja metoodika

Nagu eelpool mainitud, oli selle uuringu üks ajendeid Eesti riigile olulistes infosüsteemides kasutatavate krüptovahendite kaardistamine, kuna täpsem teadmus selle kohta on lünklik või puudub üldse. Olime valmis nägema isetehtud (ja seetõttu kaheldava turvalisusega) infosüsteeme ning protokolle, mille kohta ka loojad ei tea, et tegu on krüptoprotokolliga. Eeldasime (ja eeldame praegugi), et üldine turbetase on nõrk, kuni pole tõestatud vastupidist.

Kahjuks ei olnud asutused, kelle huvides peaks olema turbetaseme tõstmine, varmad koostööd tegema. Info kogumiseks saatsime valitud asutustele (maavalitsused, ministeeriumid, ametid ja inspektsioonid, Riigikantselei, Rahvusarhiiv ning IT-tugiasutused, pluss 6 eraettevõtet, kokku 51 asutust) kahel korral ringkirja teabepalvega. Valdav enamus asutustest ignoreeris kirju, aga andmete mitteavaldamise põhjenduseks toodi ka riigisaladus ehk Vabariigi Valitsuse määrus nr. 262 [16], mis on klassikaline näide *security by obscurity* rakendamisest. Ühel juhul toodi põhjenduseks ka kahtlustus, et töö teostaja ei jää erapoolikuks ning kasutab saadud teavet oma huvides.

Põhiline probleem on siinjuures mõjutusvahendi puudumine. Pöördumised tehti Riigi Infosüsteemide Osakonna poolt, kellel on põhimääruse alusel õigus saada riigiasutustelt vajalikke dokumente ja informatsiooni, kuid põhjustel, mille üle spekuleerida pole mõtet, asutused seda

teavet ei avaldanud.[1] Tulemusena jäi kahjuks täitmata ka üks selle projekti eesmärke – saada ülevaade mittestandardsete (st omaloodud) krüptoprotokollide kasutamisest ning anda soovitusi nende parandamiseks.

Seetõttu näeme selget vajadust viia läbi jätkuprojekt, mille ülesanne on saada teavet kõigilt asutustelt, kes praeguses projektivoorus ei vastanud. Vajalik on see kasvõi selleks, et aru saada, kui aladokumenteeritud on asutuste infosüsteemid tegelikult. Soovitatav on ka algatada diskussioon eesmärgiga muuta riigiasutustes kasutatavaid krüptovahendeid puudutav teave avalikuks, vähemalt ametialases ringis. Pikemas plaanis on oluline riigiasutuste nõustamine üldise turbeteadlikkuse tõstmise eesmärgil.

Praegu tuleb vaid resümeerida, et Eesti riigil pole jätkuvalt aimu, milliseid krüptovahendeid olulistes infosüsteemides kasutatakse ja kuidas neid hallatakse, ega pole ka head võimalust selle teadasaamiseks. Arvestades projekti väikest ajalist ja rahalist mahtu ei olnud võimalik teostada riigiinfosüsteemidele lähtekoodi- ega konfiguratsiooniauditit, kuid tungivalt soovitatav on leida võimalusi selle läbiviimiseks tulevikus.

Lõpetuseks – tegelikult on vaja tekitada IT-juhtides arusaama, et süsteemi realisatsioonidetailide varjamine ei asenda tegelikku turvet; kindlustunnet, et turvanõrkuste avastamine ei tähenda häbiposti, vaid asjakohast abi ja oskusteavet; ning mõistagi ka ressursse infosüsteemide tehniliseks dokumenteerimiseks ja selle ajakohasena hoidmiseks.

2 KRÜPTOGRAAFILISED ALGORITMID

2.1 SÜMMEETRILISED KRÜPTOALGORITMID

Sümmeetrilised krüptoalgoritmid jagunevad kahte suurde klassi: plokk- ja jadašifrid. Jadašifrite peamised kasutusvaldkonnad on tänu nende efektiivsusele erinevad madala taseme multimeediaprotokollid, valdavas enamikus teistes rakendustes kasutatakse plokkšifreid. Teisest küljest on plokkšifrid jadašifritega võrreldes krüptoanalüüsile paremini vastu pidanud. Sellest johtuvalt keskendume ka käesolevas aruandes peamiselt plokkšifritele ning käsitleme jadašifreid eeskätt esituse täielikkuse huvides.

2.1.1 DES, TDEA/3DES

Ajalooliselt oli DES esimene laialt kasutust leidnud standardiseeritud plokkšiffer, mille standardis NIST esimest korda 1976. aastal kui FIPS-46. Standardit on hiljem kolmel korral uuendatud; viimane versioon FIPS-46-3 pärineb aastast 1999 [6]. Oma 56-bitise efektiivse võtmepikkuse tõttu peetakse DESi tänapäeval ebaturvaliseks ning ka FIPS-46 standard on 2005. aastast ametlikult tühistatud. USA valitsusasutustel on lubatud DESi kasutada ainult kolmekordse krüptimisalgoritmi TDEA (tuntud ka tähise 3DES all) komponendina kuni aastani 2030 [53]. Kuigi TDEA/3DESi efektiivne võtmepikkus on 168 bitti, sisaldab algoritm palju teadaolevaid nõrkusi, mistõttu on tema ründamiseks vajalik reaalne töö suurusjärgus 2^{112} operatsiooni [47].

[1] Sellega seoses loodame, et (kirjutamishetkel veel idee tasemel olev) riigi infosüsteemide arhitekt saab niisuguse info hankimiseks vajalikud volitused ja mõjutusvahendid.

Sellest johtuvalt tuleks ka TDEAd võimalusel kõigis rakendustes vältida, seda enam, et AESi näol on olemas väga hea alternatiiv.

2.1.2 A5, Kasumi

Kuigi 2. põlvkonna GSM-mobiilsideplatvormil kasutatud algoritme A5/1 ja A5/2 pole kunagi ametlikult avalikustatud, pöördprojekteeriti nende disain 1990ndate aastate lõpus ning peatselt leiti mõlemast olulisi turvanõrkusi [63, 50]. 2010. aastal demonstreerisid saksa uurijad Nohl ja Munaut praktilist seadet, mille abil on võimalik GSM kõnesid ka reaalselt pealt kuulata [158]. Nad ei avalikustanud küll kogu rakenduse lähtekoodi, kuid kirjeldasid kogu ründevektorit piisava põhjalikkusega, et motiveeritud ja piisava tehnilise taustaga ründaja suudaks nende seadme sõltumatult uuesti luua.

Juba esimeste nõrkuste leidmise järel hakkasid standardiorganisatsioonid GSMA ja 3GPP algoritme A5/1 ja A5/2 aktiivsest kasutusest kõrvaldama. Samuti õpiti suletud disaini ohtudest ning järgmise generatsiooni (3G/UMTS) mobiilside krüptoalgoritmi loomise protsess oli avatum kui GSMi korral. Valituks osutus MISTY šifri modifikatsioon Kasumi [18], mida kasutab võtmejada genereerimiseks ka uus GSM standard A5/3. 2010. aastal leiti Kasumi vastu lähedaste võtmete rünne (*related-key attack*), mis võimaldab süsteemi poolt kasutatava võtme täielikult taastada, nõudes ainult nelja lähedast võtit ja ühe tavaarvuti poolt pakutavat arvutusjõudlust [92]. Hetkel pole selge, kas see rünne võib viia uue põlvkonna mobiilside krüptograafia murdumisele mõnes praktilises olukorras, kuid on kindel, et Kasumi on nõrgem kui tema loojad algselt kavandasid (paradoksaalselt isegi nõrgem kui algne MISTY šiffer). Samuti on 3. põlvkonna UMTS-protokolli vastu leitud vahendusründeid (*man-in-the-middle attack*), mis võimaldavad ära kasutada GSM-võrgust pärinevaid nõrkusi [144].

Samuti on ebaselge, kas lähiajal on oodata mobiilsidevõrkude turvastandardite uuendamist, kuid kardetavasti pole see kuigi lihtne, eeldades suure hulga nii võrgu- kui mobiilseadmete väljavahetamist. Paraku ei saa ka Eesti riik selle protsessi kiirendamiseks midagi ette võtta – ainus võimalus on eriti tundlikku informatsiooni üle mobiilvõrkude (GSM, 3G) mitte edastada.

2.1.3 RC4

Jadašifri RC4 töötas 1987. aastal RSA Security juures välja Ronald Rivest (ka RC šifri nimes tähendab mitteformaalselt “Ron’s code”). Šifri kirjeldust pole kunagi ametlikult avaldatud, kuid tema väidetav lähtekood lekkis 1994. aastal Internetti ning kuna lekkinud koodi väljund vastab ametlikule RC4 binaarrealisatsioonile, on alust uskuda, et see on autentne. Tänu oma efektiivsele tarkvaralisele realiseeritavusele on RC4 kasutusel paljudes IT-süsteemides ja standardites (SSL/TLS [86], IEEE802.11 standardipere (WiFi) [11], Microsofti MPPE protokoll [161] jpt).

Jadašifrina peaks RC4 ideaalis genereerima juhuslikust bitijadast eristamatu jada, kuid kahjuks on RC4 väljundist leitud arvukalt sõltuvusi, mis võimaldavad teha järeldusi šifri sisemise oleku ning kasutatava võtme kohta [136, 162, 137, 163, 98, 117]. Kleini analüüs [117] viis 2007. aastal utiliidi aircrack-ptw väljatöötamiseni, mis suudab WEP-protokollis kasutatud 104-bitise RC4 võtme leida vähem kui minuti jooksul [199]. Sarnase jõudlusega ründe pakkusid samal aastal välja ka Vaudenay ja Vuagnoux [201].

Kokkuvõtteks võib öelda, et RC4 on murtud nii teoorias kui ka praktikas ning tema kasutamine on praktiliselt kõigi turvaeesmärkide saavutamiseks mittesoovitatav. Samuti tuleb turvalisust nõudvates keskkondades hoiduda WiFi-võrkude turvamisest WEP-protokolliga ning kasutada selle asemel protokolli WPA2 (vt ka jaotist 3.5.3).

2.1.4 Blowfish

Šifri Blowfish töötas 1993. aastal välja Bruce Schneier [178] ning see oli omal ajal üks esimesi patendivabu plokkšifreid, mis on sellest ajast peale leidnud realiseerimist paljudes krüptoteekides ja -protokollistikes. Blowfishi disain on sobilik kasutamiseks nii tark- kui riistvaras ja vaatamata pikaajalisele analüüsile pole tema vastu olulisi ründeid leitud. Seega sobib Blowfish (sobiva võtmepikkusega) kasutamiseks praktiliselt kõikjal, kus vajatakse turvalist plokkšifrit. Tõsi, pärast AESi standardimist on Blowfishi populaarsus teataval määral vähenenud.

2.1.5 AES

Standardiorganisatsioon NIST kuulutas 1997. aastal välja konkursi uue põlvkonna plokkšifristandardi (Advanced Encryption Standard, AES) loomiseks ning valikuprotsess jõudis lõpule 2001. aastal, mil võitjaks kuulutati Belgia krüptogtaafide loodud šiffer Rijndael [157]. Peale *de jure* standardistaatuse on AES kujunenud ka *de facto* kõige laiemalt kasutatud (ja rünnatud) plokkšifriks. Kuna tugevus klassikalise lineaarse ja diferentsiaalse krüptoanalüüsi vastu oli juba AESi disainieesmärk, pole need meetodid tema vastu ka efektiivseks osutunud. 2002. aastal leiti teoreetilisi võimalusi AESi algebralise struktuuri nõrkuste ärakasutamiseks [82, 151], kuid need pole viinud praktiliste rünneteni [77, 133].

2009. aastal leidsid Biryukov ja Khovratovich lähedaste võtmete ründe AES-192 ja AES-256 vastu [62]. Ründe keerukus on üllatuslikult kõige väiksem AES-256 jaoks (suurusjärgus $2^{99,5}$ operatsiooni), kuid ründe õnnestumiseks on vaja, et ründaja suudaks veenda süsteemi krüptima tema poolt ette antud lähedaste (Hammingi kauguse mõttes) võtmetega. Kui reaalne selle eelduse täidetus on, sõltub konkreetsest süsteemist ning nõuab eraldi analüüsi. AES-128 korral pole lähedaste võtmete ründe keerukus enam täielikust võtmeruumi läbivaatusest väiksem.

Kõige suuremat ohtu kujutavad AESi analüüsi seisukohast kõrvalkanali ründed (*side channel attacks*), eriti protsessori vahemälu ajastuse mõõtmine (*cache timing attacks*), [59, 160]. Ka nende rünnete praktilisus sõltub suuresti konkreetsest rakenduskeskkonnast, nõudes paljude eelduste üheaegset täidetust.

Põhjaliku ülevaate erinevatest teadaolevatest rünnetest AESi vastu leiab projekti ECRYPT aruandest [76].

Kokkuvõtteks võib öelda, et viimase 10 aasta jooksul on AESi põhjalikult analüüsitud, kuid vaatamata rahvusvahelise krüptograafiakogukonna jõupingutustele pole seda siiani sisuliselt murda suudetud. Seega võib AESi nii võtmepikkusega 128, 192 kui ka 256 bitti pidada selle aruande ajahorisondi ulatuses kasutamiseks piisavalt turvaliseks.

2.2 ASÜMMEETRILISED KRÜPTOALGORITMID

Asümmeetrilised krüptoalgoritmid tuginevad erinevate algebraliste ja arvuteoreetiliste ülesannete lahendamise raskusele. Seetõttu eeldab sobivate algoritmide valik vastavate ülesannete lahendamise meetodite hetkeseisu analüüsi. Analüüsi käigus keskendutakse järgmistele probleemidele:

- täisarvude tegurdamine, RSA probleem;
- diskreetne logaritm ja Diffie-Hellmani probleem jäägiklassiringide multiplikatiivsetes rühmades;
- jagamise probleem elliptiliste kõverate rühmades.

2.2.1 RSA

RSA on üks vanemaid ning praktikas kõige laiemalt levinud avaliku võtme krüptograafilisi algoritme, mille pakkusid 1978. aastal välja Ron Rivest, Adi Shamir ja Leonard Adleman [170].

2.2.1.1 Mooduli tegurdamine

RSA krüptosüsteemi turvalisus tugineb suurte kordarvude tegurdamise raskusele ja seetõttu on tegurdamine ka kõige rohkem läbiuuritud meetod RSA ründamiseks.

Parim hetkel teadaolev meetod (*number field sieve*) suurte kordarvude tegurdamiseks võimaldas rahvusvahelisel teadlaste rühmal 2009. aastal tegurdada 768-bitise arvu [118]. Selleks kulus orienteerivalt 2000 aastat arvutiaega; tänu paralleliseerimisele jäi füüsiline ajakulu suurusjärku 2 aastat. Ründe autorite hinnangul on 1024-bitise RSA ründamine sama meetodiga umbes 1000 korda aeglasem, mistõttu esimese RSA-1024 mooduli murdumist on oodata mitte varem kui viie, aga mitte hiljem kui kümne aasta pärast. Arvestades ka tegurdamismeetodite võimalikku arengut, ei saa RSA-1024 soovitada kasutada kauem kui 5 aastat. RSA-2048 on suure tõenäosusega jätkuvalt turvaline üle käesoleva aruande ajahorisondi.

Stsenaarium, kus ründajal on RSA ründamiseks kasutada vaid avalik moodul, on küll praktikas kõige lihtsamini saavutatav, kuid sugugi mitte ainuvõimalik.

2.2.1.2 Realisatsiooni ründamine

Kuna oma algse matemaatilise definitsiooni alusel realiseeritud RSA krüptosüsteem oleks semantiliselt ebaturvaline, kasutatakse seda peaaegu eranditult kapseldatuna PKCS#1 vormingusse. 1998. aastal leidis Daniel Bleichenbacher ründe, mis võimaldas standardi PKCS#1 versiooniga 1 krüptitud avatekste taastada [66]. Seejärel töötati välja tõestatavate turvagarantiidega versioon 2 ning hetkel kehtib standardi versioon 2.1 [110].

Sellest näitest lähtuvalt anname kaks soovitust. Esiteks, krüptograafilistest standarditest on alati soovitav kasutada viimaseid versioone, ja teiseks, krüptograafiliste algoritmide rakendamisel tuleb alati tugined laialt tunnustatud ja testitud realisatsioonidele, näiteks OpenSSL [13] või Bouncy Castle [2].

2.2.1.3 RSA ründamine lisainfo olemasolul. Kõrvalkanali ründed

Alati ei kehti eeldus, et ründajal on krüptosüsteemi ründamiseks kasutada ainult protokolli disaineri poolt avalikuks deklareeritud väärtused (RSA puhul näiteks avalik moodul ja avalik võti). Välise vaatleja jaoks võivad infot lekitada näiteks aeg, mis kulub avaliku mooduli jaoks vajalike algarvude genereerimiseks või salajase eksponendiga astendamiseks, vool, mida arvutamine nõuab jne (vt Paul Kocheri ja tema töörühma artiklid [126, 124, 125]). Seega on realistlik eeldada, et ründaja võib omada osalist juurdepääsu salajastele väärtustele ning üritada süsteemi nende abil murda.

Esimene uurimus selles vallas pärineb juba 1985. aastast RSA loojate Ron Rivesti ja Adi Shamiri sulest [172], kes tõestasid, et jämedalt $n/3$ biti teadmine n-bitisest RSA moodulist võimaldab mooduli polünomiaalses ajas tegurdada.[2] Hilisemad arvukad uurimused on seda tulemust oluliselt parandanud, lisatud on teistsuguseid eeldusi ja ründevektoreid. Nii näiteks tõestas Michael J. Wiener 1990. aastal, et kui RSA salajane eksponent d on (näiteks dekrüptimise kiirendamiseks) valitud nii väike, et $d < N^{0.25}$ (kus N on avalik moodul), siis on RSA

[2] Paneme tähele, et ühe RSA mooduli teguri pikkus on keskmiselt $n/2$ bitti.

süsteem polünomiaalses ajas murtav [206]. Hiljem parandasid Boneh ja Durfee seda tulemust kuni piirini $N^{0.292}$ [69].

Häid ülevaateid RSA ründamise hetkeseisust ning viimastest tulemustest leiab näiteks Alexander May ja tema töörühma artiklitest [105, 138].

Kõigist rünnetest ülevaate andmine ning analüüs, millised neist iga konkreetse infosüsteemi jaoks ohtlikud on, jääb käesoleva aruande raamidest kaugele välja. Üldine soovitus võimalike ohtude vastu on sama, mis jaotises 2.2.1.2 – kasutada läbitestitud ja aktiivse kogukonna poolt toetatud teeke, mida uute rünnete ilmnemisel keskselt uuendatakse, mitte aga tugineda kohalikule realisatsioonile. Kaitse kõrvalkanalirünnete (nt voolutarbe mõõtmise või ajastusrünnete) vastu sõltub suuresti konkreetsest rakendusest ning hõlmab erinevaid varjestusmeetodeid ja sama võtme erinevate kasutuskordade arvu piiramist. Head ülevaadet kõrvalkanalirünnetest ning võimalikest kaitsemeetmetest võib lugeda Paul Kocheri jt artiklist [125].

2.2.2 DSA, ElGamal

RSA kõrval teise olulise avaliku võtme krüptosüsteemi pakkus 1985. aastal välja Taher ElGamal [96]. Võrreldes RSA-ga olid ElGamali süsteemil mõned olulised eelised. Esiteks oli ta juba definitsiooni tasemel randomiseeritud (leevendades seega oluliselt RSA puhul kerkivat semantilise ebaturvalisuse probleemi) ning teiseks polnud ta patentidega kaetud. Need asjaolud motiveerisid USA valitsust kasutama ElGamali süsteemi alusena oma digitaalsignatuuri algoritmi (*Digital Signature Algorithm*, DSA) standardimisel digitaalsignatuuri standardi DSS osana [10]. ElGamali süsteem on defineeritud üle suvalise (tsüklilise) rühma, seetõttu võib tema realiseerimisel kasutada näiteks elliptiliste kõverate rühmi (ning see on DSS standardi praeguses versioonis ka ette nähtud).

On võimalik tõestada, et ElGamali krüptosüsteemi ründamine on arvutuslikult samaväärne Diffie-Hellmani võtmevahetuse murdmisega, mis omakorda on mitte raskem kui diskreetse logaritmi ülesande lahendamine allolevas rühmas [194].

2.2.3 Diffie-Hellmani võtmevahetus

Oma algsel kujul, nagu Whitfield Diffie ja Martin Hellman selle 1976. aastal välja pakkusid, lahendab Diffie-Hellmani (DH) võtmevahetusprotokoll probleemi, kuidas kaks osapoolt võivad üle autentse, kuid mittekonfidentsiaalse kanali leppida kokku ühise saladuse, mida pealtkuulaja ei suuda tuvastada [87]. Tänaseks on DH oma erinevates realisatsioonides saanud valdava enamuse Interneti-protokollistike poolt kasutatavaks võtmevahetusmeetodiks, olles standarditud näiteks IETFi poolt kui RFC2631 [168] ning *Internet Key Exchange* (IKEv2) standardi osana kui RFC5996 [113], NSA poolt Suite B osana [12] jm.

2.2.4 Elliptiliste kõverate süsteemid

Elliptiliste ning hüperelliptiliste kõverate omaduste kasutamise krüptograafias pakkusid 1980. aastate teisel poolel sõltumatult välja Neil Koblitz [120, 121] ja Victor Miller [147]. Võrreldes RSAga on süsteemi kirjeldus keerulisem ning ühe osa avalikest parameetritest moodustab kasutatav elliptiline kõver, mille valik on mittetriviaalne. Rida USA valitsusasutustes rakendamiseks sobivaid elliptilisi kõveraid spetsifitseerib NISTi standard FIPS 186-3 [10]. Soovitused standardi rakendamiseks annab juhend [36], mis täpsustab paljusid reaalsete kõverate ja võtmete genereerimise, hoidmise ning kasutamisega seotud üksikasju. Matemaatiliste primitiivide realiseerimist kirjeldab juhend [34].

Abstraktse algebra seisukohast kasutavad elliptiliste kõverate süsteemid samuti diskreetse logaritmi leidmise raskust teatud rühmades, konkreetsemalt elliptiliste kõverate punktide rühmades. Sellest tulenevalt saab neile tuginedes ehitada DSA/DSS-põhise signatuuriskeemi [10] ning Diffie-Hellmani põhise võtmevahetusprotokolli [52]. *L*-bitiste võtmete puhul töötavad parimad üldiste rühmade jaoks välja töötatud diskreetse logaritmi arvutamise meetodid (nt Pollardi ρ-algoritm või *baby-step giant-step* algoritm, vt [143] peatükk 3.6) ajakeerukusega $2^{L/2}$. Daniel Brown on esitanud argumente, mis näitavad, et elliptiliste kõverate rühmad käituvad paljuski üldiste rühmade sarnaselt, millest tulenevalt võib loota, et oluliselt paremaid meetodeid nende kasutamisel saadavate krüptosüsteemide ründamiseks ei leidugi [71]. Seda seisukohta on kritiseerinud Stern, Pointcheval, Malone-Lee ja Smart [191] ning viimaste seisukohta omakorda Koblitz ja Menezes [122]. Aktiivne teaduslik diskussioon selles vallas alles areneb, kuid kindlalt võib öelda, et jäägiklassiringide kohta on teada märksa rohkem sisemist struktuuri, mis aitab ründeid tunduvalt tõhusamalt realiseerida.

Elliptiliste kõverate algebralist struktuuri on edukate rünnete realiseerimiseks õnnestunud ära kasutada vaid mõnel üksikul erijuhul. 1993. aastal näitasid Menezes, Okamoto ja Vanstone, et kui kasutatava elliptilise kõvera rühma elementide arv jagab suurust $q^k - 1$ mingi algarvu q ja väikese astendaja k korral, saab elliptilise kõvera diskreetse logaritmi ülesande taandada diskreetse logaritmi ülesandele korpuse $GF(q^k)$ multiplikatiivses rühmas ning selle ülesande jaoks on omakorda teada subeksponentsiaalse ajalise keerukusega algoritmid [142]. Semaev [179], Smart [188] ning Satoh ja Araki [177] näitasid sõltumatult, et kui elliptilise kõvera punktide arv langeb kokku alloleva korpuse elementide arvuga (kõver on *anomaalne*), siis saab vastava diskreetse logaritmi ülesande lahendada polünomiaalses ajas. Mõlema ründe rakendumise kriteeriumid on parameetrite genereerimisel lihtsasti kontrollitavad ja neid ründeid on seega lihtne ära hoida. Süstemaatilise ülevaate ellptiliste kõverate krüptosüsteemide ründamisest ning hetkeseisust annavad Koblitz, Menezes ja Vanstone artiklis [123].

Patendid

Kuigi elliptiliste kõverate krüptosüsteemid pakuvad RSA ja diskreetse logaritmi süsteemidega võrreldes sama taseme turvalisust palju lühemate võtmetega ja seega palju efektiivsemalt, pole elliptilised kõverad praktilistes rakendustes sugugi laialt levinud. Peamiseks põhjuseks on siin juriidilised asjaolud, sest paljud vastavate süsteemide realiseerimiseks vajalikud meetodid on kaetud erinevate patentidega. Patendid on laiali erinevate firmade käes (sh Sun Microsystems, Apple Computer, Certicom, Cylink) ning on segane, millised neist firmadest ja milliste väidetega võivad esile tulla. Näiteks Sun Microsystems on omalt poolt välja kuulutanud "patendirahu" ning annetanud oma elliptiliste kõverate meetodite lähtekoodi kasutamiseks teegis OpenSSL. Samal ajal kaebas Certicom, kes omab üle 130 vastavasisulise patendi, 2007. aastal Sony kohtusse intellektuaalomandi õiguste rikkumise pärast. Kohtuasi lükati 2009. aastal küll tagasi, kuid küsimus sellest, milliste juriidiliste järelmitega peab elliptiliste kõverate süsteemi realiseerija arvestama, on jätkuvalt lahtine. Miinimumina tuleb arvestada rahvusvahelises kohtus käimise kuludega. Certicomi õigust elliptiliste kõverate süsteemide intellektuaalomandile pretendeerida tunnustas ka NSA, kes 2003. aastal litsentseeris firma tehnoloogiat 25 miljoni dollari eest USA valitsusasutustes kasutamiseks. Väärib tähelepanu, et NSA Suite B kasutab avaliku võtme krüptograafiast ainult elliptilistel kõveratel baseeruvaid algoritme [12].

Kokkuvõttes et saa elliptilistel kõveratel baseeruvat krüptograafiat hetkel ilma sügavama juriidilise ja patendianalüüsita soovitada. Halvemal juhul tuleb arvestada märgatavate litsentsitasude maksmisega.

2.3 RÄSIFUNKTSIOONID

Võrreldes üsnagi hästi läbiuuritud võtmepõhise krüptograafia algoritmidega on turvaliste räsifunktsioonide loomise meetodite kohta teada oluliselt vähem. Peaaegu kõik praktikas levinud räsifunktsioonid kasutavad mingil määral *ad hoc*-konstruktsioone ning nende formaalseid turvatõestusi on tunduvalt raskem anda. Seetõttu on räsifunktsioonid viimastel aastatel olnud ühed enimmurtud krüptograafilised primitiivid.

2.3.1 MD5

Asendamaks nõrkustega funktsiooni MD4, töötas Ronald Rivest 1991. aastal välja räsifunktsiooni MD5 ning see standarditi IETF-is 1992. a [171]. Esimesed nõrkused leiti uuest standardist juba 1996, mil Hans Dobbertin demonstreeris MD5 raundifunktsiooni kollisiooni [88]. 2004. aastal leidis professor Wangi töörühm kollisiooni ka kogu räsifunktsioonile [204]. Hiljem on selle ründe efektiivsust oluliselt tõstetud, nii et nüüdseks on MD5 kollisioone hariliku lauaarvuti abil võimalik leida sekunditega [192, 208]. Kuigi leitavad kollisioonid pole täiesti vabalt valitavad, vaid peavad vastama üsna paljudele kitsendustele, on teadaolevatest aluskollisioonidest lähtuvalt leitud kollisioonie ka reaalsete semantikaga dokumentidele nagu PostScripti failid [84] või X.509 sertifikaadid [193].

Kõik ülalöeldu kehtib spetsiifiliselt kollisioonide leidmise kohta. Räsiväärtusest originaali arvutamise mõttes on MD5 endiselt tugev. Parim teadaolev rünne suudab originaali leida $2^{123.4}$ operatsiooniga, sellal kui MD5 väljundil on üldse 2^{128} teoreetiliselt võimalikku väärtust [176].

Samas ei saa MD5-t enam kollisioonide leidumise tõttu kasutamiseks soovitada, sest krüptoprotokolle realiseerivad programmeerijad ei mõista sageli, millistel eeldustel on see funktsioon turvaline. Tegelikult ei anna infosüsteemide ehitajad endale sageli aru isegi sellest, et nad krüptoprotokolli loovad. Seega soovitame üldise turvataseme tõstmiseks kasutada ainult niisuguseid primitiive, mis on turvalised praktiliselt kõigis kontekstides.

Kui aga mingitel (näiteks eelarvelistel) põhjustel pole MD5 täielik väljavahetamine võimalik, tuleks tööde planeerimisel juhinduda järgmistest põhimõtetest.

- Räsifunktsiooni MD5 ei tohi kasutada olukorras, kus ohtlikuks osutub stsenaarium, mille korral ründaja võib saavutada kasu ise kahte räsitavat faili ette valmistades (näiteks lepingute või sertifikaatide signeerimine, sõnumi moodustamine ajatembelduseks või räsipõhiseks logimiseks jmt).
- Räsifunktsiooni MD5 võib selle aruande ajahorisondi (5 aastat) ulatuses kasutada olukorras, kus turvaomadused tuginevad räsiväärtusest originaali või teise originaali leidmisele.
- Räsifunktsiooni MD5 võib ka edaspidi kasutada mittekrüptograafilistel eesmärkidel (näiteks ühekordsete identifikaatorite moodustamisel või failide kontrollsummade arvutamisel juhuslike vigade avastamiseks).

Samas on süsteemi projekteerides väga raske kindlustada, et seda ei hakata kunagi kasutama stsenaariumis, mille suhtes antud primitiiv enam turvaline ei ole. Seega soovitame käesolevas aruandes rusikareeglina MD5 kasutamisest ikkagi üldse loobuda.

2.3.2 SHA-1 ja SHA-2

MD5 potentsiaalne nõrkus oli selge juba 1990. aastate alul, sest tänu suhteliselt lühikesele väljundi pikkusele (128 bitti) on temas kollisiooni leidmise keerukus sünnipäevaründega suurusjärgus 2^{64} räsiarvutust.

1993. aastal kuulutas NIST välja uue standardi FIPS 180, mida hakati kutsuma SHA (*Secure Hash Algorithm*) ja hiljem SHA-0. NISTi väitel avastati sellest peatselt nõrkusi, mille parandamiseks töötati välja kergelt modifitseeritud SHA-1 ning avaldati see 1995. aastal standardina FIPS 180-1. Hiljem on seda standardit veel kahel korral uuendatud (2002 ja 2008). Hetkel kehtib FIPS 180-3 [17], mis peale 160-bitise väljundiga SHA-1 spetsifitseerib lisaks funktsioonid SHA-224, SHA-256, SHA-384 ja SHA-512 (kus number viitab väljundi pikkusele). Viimast nelja tuntakse ka koondnimetuse SHA-2 all ning nad sisaldavad võrreldes SHA-1-ga erinevaid täiendusi, mis on kõik suuremal või vähemalt määral *ad hoc*.

Kasutades MD5 kollisioonide leidmisele viinud metoodikat, on SHA-1 kollisiooni leidmine võimalik 2^{63} räsiarvutusega [79] (sünnipäevarünne nõuaks 2^{80} arvutust) ning hetkel uuritakse võimalusi selle keerukuse vähendamiseks. 2010. aastal avaldati meetod, mis võimaldab leida kollisiooni 73-raundilise SHA-1 puhul [94]. Kuna SHA-1 täisversioon kasutab 80 raundi, on tema lõplikku murdumist oodata juba lähiajal.

SHA-2 pere algoritmide vastu on leitud vaid ründeid, mis töötavad piiratud raundide arvuga (täpsemalt 24 raundi korral, kusjuures SHA-224 ja SHA-256 kasutavad 64 raundi, SHA-384 ja SHA-512 aga 80 raundi) [107]. Niisiis võib SHA-2 pere algoritme käesoleva aruande ajahorisondi ulatuses pidada turvalisteks; vt ka [47, 27].

2.3.3 SHA-3

Kuna kahest kõige laiemalt kasutatavast räsifunktsioonist (MD5 ja SHA-1) üks on juba murdunud ning teine kohe-kohe murdumas, kuid asenduseks pakutav SHA-2 perekond on võrdlemisi *ad hoc* disainiga, kuulutas NIST 2006. aastal sarnaselt edukaks osutunud AES-i konkursiga välja järgmise põlvkonna räsifunktsiooni konkursi. 2010. aasta detsembris avalikustati viis finalisti, mille seast tehakse valik 2012. aasta lõpuks [156].

Selle projekti raames pole lootust konkursi tulemust ette aimata, seega piirdume siinkohal soovitusega võtta 2013. aastast kasutusele uus standardiseeritud SHA-3 algoritm.

2.3.4 RIPEMD pere

Esimene RIPEMD pere räsifunktsioon RIPEMD töötati välja EL projekti RIPE (*RACE Integrity Primitives Evaluation*) käigus aastatel 1988–1992. Juba 1995. aastal leidis Hans Dobbertin ründe, mis andis kollisiooni RIPEMD-le, mille raundide arv oli piiratud 3-lt 2-le [89]. Kuigi täisversiooni kollisioon leiti professor Wangi töörühma poolt alles aastal 2004 [203], hakati algset RIPEMD-d kohe täiendama. 1996. aastal pakkusidki Hans Dobbertin, Antoon Bosselaers ja Bart Preneel välja uued räsifunktsioonid RIPEMD-128, RIPEMD-160, RIPEMD-256 ja RIPEMD-320 [90]. Viimased kaks ei lisa esimesele kahele muud turvalisust peale kaks korda pikemate väljundite, mis aitab vaid sünnipäevaründe-laadsete jõumeetodite vastu. Kuna sünnipäevaründega RIPEMD-128 nõuab vaid suurusjärgus 2^{64} operatsiooni ning lisaks on piiratud raundide arvuga RIPEMD-128-st leitud olulisi nõrkusi [141], ei soovitata RIPEMD-128-t enam kasutada. Kuna sarnane rünne RIPEMD-160 vastu ei tööta, siis võib RIPEMD-160 käesoleva aruande ajahorisondi ulatuses pidada turvaliseks.

2.4 SÕNUMIAUTENTIMISKOODID (MAC-KOODID)

Sõnumiautentimiskoodidest (ingl. k. *Message Authentication Codes (MAC)*) võib lihtsustatult mõelda kui digitaalsignatuuride sümmeetrilistest analoogidest, mida saab kasutada olukorras, kus sõnumi autentsuse ja tervikluse tagamine on oluline ainult kahe osapoole vahel. Sel juhul

pole reeglina vaja kasutada tehnoloogiliselt nõudlikumat asümmeetrilist krüptograafiat, vaid saab lähtuda ka ühiselt jagatud sümmeetrilisest võtmest.

MAC-koodid pole enamasti iseseisvad primitiivid, vaid tuginevad teistele primitiividele, tüüpiliselt plokkšifritele ja räsifunktsioonidele. Krüptograafilises kogukonnas on välja pakutud erinevaid võimalikke lähenemisi, milledest osade kohta on võimalik anda ka formaalseid turvatõestusi. Nii näiteks on HMAC konstruktsiooni [127] kohta tõestatud, et ta on turvaline, kui aluseks võetud räsifunktsiooni kompressioonifunktsioon käitub pseudojuhusliku funktsioonina [55]. Huvitav on märkida, et see turvatõestus ei sõltu alloleva räsifunktsiooni kollisioonivabadusest ja seega ei välista MD5-s leitud kollisioonid otseselt tema kasutamist HMAC-i konstrueerimisel. Küll aga on näidatud, kuidas kasutada räsifunktsiooni teadaolevaid nõrkusi ära HMAC-i eristamisel juhuslikust funktsioonist [116]. Sellest johtuvalt on mõistlik ehitada HMAC teadaolevalt turvalistele räsifunktsioonidele (nt SHA-2).

Teine laialt levinud meetod MAC-ide saamiseks on kasutada plokkšifrit mõnes sobivas töörežiimis, näiteks CBC-režiimis [14], saades tulemusena konstruktsiooni, mida tuntakse kui CBC-MAC [4]. On võimalik tõestada, et see konstruktsioon osutub turvaliseks, kui autenditava dokumendi pikkus on ette teada ja fikseeritud [56]; samas ei ole see eeldus praktikas enamasti täidetud. Algne CBC-MAC-i definitsioon kasutab ühte plokkšifrit kahe erineva võtmega. Süsteemi praktilisel realiseerimisel on kiusatus need kaks võtit võrdseks võtta, kuid niisugune lähenemine toob endaga kaasa olulise turvanõrkuse, võimaldades luua kasutaja poolt mõeldust erineva dokumendi, mille MAC valideerub. Niisuguste probleemide ennetamiseks on soovitav kasutada CBC-MAC-i mõnd modifikatsiooni, näiteks CMAC-i [190] või OMAC-i [108].

2.5 VÕTMEPIKKUSTE VÕRDLUSED JA SOOVITUSED PARAMEETRITE VALIKUKS

2.5.1 Võtmepikkuste võrdlused

Ükskõik kui hea disainiga krüptograafilised algoritmid ka ei oleks, on neid alati võimalik rünnata jõumeetodil, mis näiteks sümmeetrilise krüptograafia korral tähendab sisuliselt võtme äraarvamist võtmeruumi täieliku läbivaatuse teel või räsifunktsiooni kollisiooni leidmist sünnipäevaründe abil. Asümmeetrilise krüptograafia korral kasutatakse valdavalt algebralisi konstruktsioone, mille sisestruktuuri kohta on reeglina palju teada, ja nii peame asümmeetrilisel juhul lisaks arvestama ründeid aluseks võetud struktuuri vastu.

Krüptograafiliste primitiivide üks olulisi disainieesmärke on saavutada nende käitumine "ideaalilähedaste" objektidena, st selliselt, et nende vastu ei leiduks jõumeetodist oluliselt efektiivsemaid ründeid. Sümmeetrilise primitiivi turvamiseks piisab sel juhul kindlustada, et läbivaatuseks vajalik võtmeruum on piisavalt suur.

Algoritme, mille vastu jõumeetodist paremad ründed leitakse, peetakse nõrkadeks. Nõrku sümmeetrilisi algoritme ja räsifunktsioone soovitatakse mitte kasutada. Asümmeetrilisel juhul on teatavad struktuursed nõrkused enamasti paratamatud ja seetõttu tuleb krüptosüsteemi parameetrid valida nii, et parimad teadaolevad ründed oleks liiga keerukad. Praktiliselt kõik võtmega krüptoalgoritmid on disainitud võimaldama erineva pikkusega võtmeid, mis suurendab läbivaatuseks vajalikku võtmeruumi ja muudab algoritmi sisestruktuuri ärakasutamise praktikas liiga keeruliseks.

Kui keeruline on "praktikas liiga keeruline" ülesanne? 2006. aastal hinnati kogu maailmas tol hetkel olemas olnud arvutusvõimsust 2^{85} operatsioonile aastas [76]. Arvestades Moore'i seadust, mille üks võimalik tõlgendus ütleb, et olemasolev arvutusvõimsus kahekordistub poo-

Ründaja	Eelarve	Vajalik turvatase
Lihthäkker	0	53
Lihthäkker	$400	58
Pahavaralooja	0	73
Väike organisatsioon	$10k	64
Keskmine organisatsioon	$300k	68
Suur organisatsioon	$10M	78
Luureagentuur	$300M	84

Tabel 1: ECRYPT2 ründajate klassifikatsioon

leteise aastaga[3], võib kirjutamishetkeks (mai-juuni 2011) pidada realistlikuks hinnanguks 2^{90} operatsiooni aastas.

Kuigi on ebareaalne eeldada, et kogu maailma arvutusvõimsus oleks suunatud ühe algoritmi või võtme ründamisele, annab see siiski mõistliku ülemise hinnangu praktilisele turvavajadusele.

Teine parameeter, millega peab turvavajaduse hindamisel arvestama, on potentsiaalse ründaja motivatsioon ning sellest tulenev valmisolek investeerida ründeressurssidesse. ECRYPT2 aruanne [47] kasutab tabelis 1 esitatud ründajate ning nende motivatsioonist tulenevate vajalike turvatasemete klassifikatsiooni. Turvavatase on määratud ideaalse sümmeetrilise šifri võtmebittides, st ründamaks šifrit turvatasemega b peaks ründaja tegema 2^b operatsiooni (sisuliselt kogu võtmeruumi läbi vaatama). Kuna aruande [47] soovitused pärinevad kirjutamishetkel umbes pooleteise aasta tagant, võib Moore'i seadust arvestades kõigile turvavajadustele tänase seisuga lisada umbes ühe biti.

Nagu eelpool mainitud, tuleb asümmeetrilise krüptograafia võtmepikkuste soovitamisel arvesse võtta algebralisest struktuurist lähtuvaid ründeid. Tabel 2 esitab ECRYPT2 [47] aruande hinnangud asümmeetriliste ja sümmeetriliste võtmete pikkuste võrdlusele. RSA ja diskreetse logaritmi põhiste süsteemide (ElGamal, DSA, Diffie-Hellmani võtmehavetus) turvataset võib sama võtmepikkuse juures pidada praktiliselt võrdseteks. Sealjuures on elliptilistele kõveratele tuginevate süsteemide vajadused võtmepikkuse mõttes tunduvalt madalamad, jäädes turvataseme b saavutamiseks suurusjärku $2b$.

Tabel 3 esitab ECRYPT2 hinnangud RSA ja diskreetse logaritmi süsteemide enamlevinud võtmepikkuste vastavusele ideaalse sümmeetrilise šifri turvatasemele.

2.5.2 Soovitused

Aruande üks eesmärke on anda ka soovitusi kasutamiseks sobivate krüptoalgoritmide osas. Järgnevas esitame kolm nimekirja, kuhu on jaotatud selles peatükis käsitletud algoritmid. Esimene nimekiri sisaldab primitiive, mille kasutamine on ebaturvaline juba täna, teises nimekirjas on primitiivid, mille kasutamisest tuleks loobuda aruande ajahorisondi (st 5 aasta) jooksul, ning kolmanda nimekirja primitiivide kasutamine on suure tõenäosusega turvaline ka üle selle ajahorisondi. Aruande autorid saavad samas aru, et teatud primitiivide korral (näiteks mobiilsi-

[3]Kuigi tänapäeval sõnastatakse Moore'i seadust üldise arvutusvõimsuse terminites ning hinnatakse arengukiiruseks kahekordistumine 18 kuuga, pole see ajalooliselt siiski päris täpne. Gordon Moore'i algses artiklis [149] kõneldakse loogikaelementide arvust kiibi (pinnaühiku) koha ning pakutakse kahekordistumise ajaks ühte aastat. Hiljem muutis Moore ise oma ajahinnangut kahe aasta peale [150]. Praeguseks folkloorsena kokku lepitud hinnangut 18 kuud võib pidada paljude erinevate ekspertide arvamuste keskmiseks.

Turvatase	RSA / DLOG	EC
48	480	96
56	640	112
64	816	128
80	1248	160
112	2432	224
128	3248	256
160	5312	320
192	7936	384
256	15424	512

Tabel 2: ECRYPT2 võtmepikkuste ekvivalentsuse tabel

RSA/DLOG võtmepikkus	Turvatase
512	50
768	62
1024	73
1536	89
2048	103

Tabel 3: ECRYPT2 asümmeetriliste võtmepikkuste turvatasemete tabel

des kasutatavates krüptoalgoritmides) on nende kasutamist Eestis raske lõpetada Eesti-siseste organisatsiooniliste meetmete abil.

Ebaturvalised primitiivid DES, A5/1, A5/2, RC4, RSA-512, RSA-768 (ja diskreetse logaritmi põhised süsteemid mooduli pikkusega kuni 768 bitti), MD5, RIPEMD-128.

Primitiivid, mille kasutamine tuleks 5 aasta jooksul lõpetada TDEA/3DES, RSA-1024 (ja diskreetse logaritmi põhised süsteemid mooduli pikkusega 1024 bitti), SHA-1, Kasumi.

5 aasta jooksul turvalised primitiivid Blowfish, AES (kõik standardsed võtmepikkused), RSA-2048 (ja diskreetse logaritmi põhised süsteemid mooduli pikkusega 2048 bitti), SHA-2, SHA-3, RIPEMD-160.

3 KRÜPTOGRAAFILISED PROTOKOLLID

Krüptograafilistest primitiividest üksinda ei piisa turvaliste süsteemide ehitamiseks. Neid primitiive tuleb ka õigesti kasutada, muidu võib juhtuda, et süsteemi saab rünnata primitiividest mööda minnes. Oluline krüptograafia kasutamise koht on süsteemikomponentide vaheline suht-

lus ning eeskirjadeks, mis ütlevad, millise koha peal mingit krüptoalgoritmi rakendada, on krüptograafilised protokollid.

3.1 PROTOKOLLIDE TURVAGARANTIIDE OLEMUS

Krüptograafilised protokollid on märksa keerulisemad objektid kui eelmises peatükis vaadeldud primitiivid. Ka krüptograafias konstrueeritakse keerulisemaid objekte lihtsamatest. Lisaks konstruktsioonile püütakse reeglina anda ka tõestus, et kui konstruktsiooni osadena kasutatud lihtsamad objektid on turvalised, siis on ka konstrueeritav objekt turvaline. Oluline on märgata, et sellise tõestuse andmiseks pole tarvis midagi teada tegelikult kasutatavate lihtsamate objektide turvalisusest; turvatõestus (ehk *reduktsioon*) annab ainult implikatsiooni.

Sellest tulenevalt peaks ideaalis ka iga krüptograafilise protokolli juurde kuuluma tema turvalisuse reduktsioon protokollis kasutatavate primitiivide turvalisusele. Tegelikkuses on kõik see, mida protokoll lisaks primitiividele sisaldab – side osapoolte vahel, ründaja interaktsioon toimuva sidega, implementatsiooni iseärasused – aga detailirikas, raskesti formaliseeritav ja arutletav ning seetõttu on paljud protokollid esitatud ilma turvareduktsioonideta. Heal juhul on protokolli turvareduktsioon esitatud kunagi hiljem, halvemal juhul üksnes arvatakse, et protokoll on turvaline.

Küll aga järeldub eelmisest lõigust, et võrreldes primitiividega on protokollide jaoks võimalike turvaotsuste ulatus laiem. Kui primitiivi kohta saab teha ainult otsuse “ebaturvaline” või “pole teada, et ebaturvaline” (võimalik, et kvalifitseeritult selle primitiivi kasutusparameetrite võimalike väärtustega), siis krüptograafilise protokolli kohta võib otsustada ka seda, et ta on “turvaline, kui kasutatavad primitiivid on turvalised” või “ebaturvaline isegi siis, kui kasutatavad primitiivid on turvalised”.

Krüptograafiliste protokollide turvareduktsioonide kasutamise üks põhjusi on krüptoprimitiivide turvadefinitsioonide keerukus. Nn. *arvutuslikus mudelis* [100] spetsifitseerib turvadefinitsioon selle ründe, mis ründajal peab ebaõnnestuma; sellel definitsioonil põhinevad turvareduktsioonid peavad näitama, kuidas krüptoprotokolli vastu sooritatav rünne on teisendatav primitiivi turvadefinitsiooniga antud ründeks. Lihtsustamaks krüptoprotokollide turvalisuse põhjendusi on välja pakutud *krüptograafia sümboliline mudel* [91]. Mudel ei keskendu mitte sellele, mida ründaja teha ei suuda, vaid sellele, mida “mõistlikku” ta teha suudab. Mõned näited ründaja jaoks “mõistlike” operatsioonide kohta:

- genereeri uus juhuslik väärtus;
- olles omandanud (s.t. genereerinud, kinni püüdnud või arvutanud) kaks teadet, moodusta neist paar (või rakenda neile või neist ühele mõnda teist operatsiooni);
- omades krüptoteksti ja sellele vastavat võtit, leia vastav avatekst.

“Mõistlik” operatsioon ei ole näiteks ainult krüptotekstist ilma võtit teadmata teise krüptoteksti loomine, nii et vastavad avatekstid oleksid omavahel mittetriviaalsel viisil seotud. Teatud tegevuste klassifitseerimine “mõistlikeks” põhineb eeldusel, et kõik muud operatsioonid viivad ainult tähenduseta sõnumite tekkimiseni, mis on sellise sõnumi saaja poolt lihtsasti väljafiltreeritavad. Krüptograafia sümboliline mudel eeldab, et ründaja sooritab ainult “mõistlikke” arvutussamme.

Krüptograafia sümboliline mudel on osutunud mugavaks ja väga viljakaks abstraktsioonitasemeks krüptograafiliste protokollide analüüsil, mida tõendab lähenemisviiside paljusus. Analüüsimetooditena on välja pakutud paljude erinevate tehnikate kohandusi, näiteks mudelikontrolli [134, 54, 148], andmevooanalüüsi [67, 154], formaalloogikat [72, 197, 83], tüübisüsteeme [40, 101, 102], resolutsiooni [64]. Väga oluline on ka see, et suur osa neist metoodikatest on

osutunud piisavalt lihtsaks, et konstrueerida automaatseid või siis vähemalt poolautomaatseid analüüsivahendeid [139, 135, 187, 189, 45, 65], mis on piisavalt efektiivsed, et neid rakendada reaalsetel protokollidel. Selliste analüüsivahendite kasutus tagab, et protokolli turvalisuse põhjendus ei sisalda hooletusvigu (kuid seal võivad olla vead, mis on omakorda tingitud vigadest analüsaatori lähtekoodis; seetõttu tuleb loota, et analüsaatoril on piisavalt kasutajaid selleks, et vead kiiresti avastataks).

Kui protokoll on turvaline sümbolilises mudelis, siis mida tähendab see arvutuslikus mudelis? Esimene tähelepanek on, et kui protokoll ei oleks arvutuslikus mudelis turvaline, siis peaks mõne kasutusel oleva krüptoprimitiivi või nende kombinatsiooniga olema võimalik sooritada mõni "ebamõistlik" operatsioon. Kuigi primitiivide turvatõestused seda ei välista, oleks see ikkagi väga märkimisväärne avastus. Küsimust, kas ja millal protokolli turvalisusest sümbolilises mudelis järeldub tema turvalisus arvutuslikus mudelis on viimase kümne aasta jooksul uurinud mitmed tööd [43, 129, 146, 145, 81, 109, 80, 48, 49]. Tänaseks on saadud tulemused piisavalt põhjalikud, võimaldades väita, et kui me oleme näidanud mõne krüptoprotokolli, mis kasutab ainult enamlevinud primitiive (krüptimine, signeerimine, räsimine) ning neid ainult "mõistlikul" viisil, siis järeldub sümbolilisest turvalisusest arvutuslik turvalisus.

3.2 AUTENTIMIS- JA VÕTMEVAHETUSPROTOKOLLID

Autentimisprotokolli kasutatakse juhul, kui kaks või enam olemit (isikut / arvutit / ...) soovivad kindlaks teha, et teine osapool (või -pooled) on hetkel olemas. Võtmevahetusprotokolli kasutatakse, kui olemid soovivad kokku leppida värske võtme, mida ei tea keegi teine peale nende. Enamjaolt on kasutatavatel protokollidel nii autentimis- kui ka võtmevahetuseesmärk. See tähendab, et protokollis osalev olem saab nii teadmuse, et teised olemid on elus (s.t. protokolli töö ajal aktiivsed), kui ka võtme, mida teavad ainult need teised olemid.

Et väita midagi teise olemi kohta, on tarvis tema kohta midagi teada. Tüüpiliselt on selleks teadmuseks olemi avalik võti, mis leitakse tema sertifikaadist. (Sertifikaatide levitamise viisid jäävad selle aruande teemakäsitlusest välja.) Siinkohal eeldatakse, et olemi nime järgi on võimalik leida tema kehtiv sertifikaat (mis on üks ja ainuke, vähemalt käesolevas rakenduses) ning sellest tema avalik võti. Mõnes rakenduses võib olemi kohta teada olla midagi muud – näiteks tema parool või midagi, mis sellest sõltub.

3.2.1 SSL/TLS

Transport Layer Security (TLS) (värskeim versioon 1.2 [86]) ja tema eelkäija *Secure Sockets Layer (SSL)* on ilmselt levinuimad protokollistikud kahe osapoole vaheliseks autentimiseks ja võtmevahetuseks ning sellele järgnevaks turvaliseks suhtluseks (vt. jaotis 3.5). TLS spetsifitseerib võtmevahetusprotokolli (mille käigus toimub ka ühe või mõlema osapoole autentimine) ja transpordiprotokolli. Siinkohal vaatame võtmevahetusprotokolli, transpordiprotokolli käsitleb jaotis 3.5.

Võtmevahetusprotokoll (*handshake protocol*) on üsna lihtsa struktuuriga. Kõigepealt saadavad osapooled teineteisele oma sertifikaadid ning lepivad kokku kasutatavates algoritmides. Seejärel saadab klient serverile ühe teate ning võib-olla server kliendile ühe teate (kui lepiti kokku Diffie-Hellmani võtmevahetuses), millest mõlemad osapooled suudavad arvutada ühe ja sama, kõigi ülejäänud isikute jaoks salajase väärtuse (*pre-master secret*). Võtmevahetus on sellega saavutatud. Samuti on server kliendile autenditud. See, kas klient on serverile autenditud, sõltub sellest, kas kliendisertifikaat võimaldab signeerimist ja kas server seda sertifikaati tunnistab.

Lisaks eelmises lõigus kirjeldatud struktuurile on TLS-protokollil veel palju valikuliselt rakendatavaid osi ning nende kõigi arvessevõtmine turvatõestuste juures on töömahukas (olgugi, et kontseptuaalselt võrdlemisi lihtne). Paulson [165] on kasutanud TLS-võtmevahetusprotokolli turvaomaduste tõestamiseks sümbolilises mudelis [91] formaalloogilisi meetodeid [164], tõestused on läbi viidud teoreemitõestusassistendi Isabelle/HOL [155] abil. Ta uurib protokolli põhilist teadetevoogu (mis koosneb kuni kaheksast teatest) ning näitab, et selle teadetevoo juures on tegu turvalise võtmevahetusprotokolliga. He jt. [104] kasutavad protokollide komponeerimisloogikat [83], et näidata umbes samasuguse teadetevoo turvaomaduste säilimist ka protokolli käitamisel suurema süsteemi alamprotokollina. Gajek jt. [99] näitavad, et TLS-i võtmevahetus- ja transpordiprotokolli koos võib abstraheerida turvalisi sideseansse; see abstraktsioon on korrektne krüptograafia arvutuslikus mudelis. Bhargavan jt. [60] analüüsivad üht konkreetset TLS-i realisatsiooni, mis on kirjutatud keeles F# [196], ning leiavad, et tegu on turvalise protokolli ja implementatsiooniga.

TLS-võtmevahetusprotokolli ja tema realisatsioonide kohta on teada ka mitmeid nõrkusi, mis demonstreerivad toodud turvaanalüüside piiri. Bleichenbacheri rünnet [66] kirjeldati jaotises 2.2.1.2. Klima jt. [119] näitavad kuidas seda rünnet laiendada juhule, kus server küll realiseerib Bleichenbacheri ründe vastumeetmed, kuid kontrollib nähtavalt, et kliendi saadetud *pre-master secret* sisaldab õiget versiooninumbrit. Need ei ole ründed protokolli enda, vaid seal kasutatava krüptoprimitiivi vastu. Augustis 2009 leiti aga rünne päris TLS-protokolli vastu, mis seostub teadetejadaga, mida eelnevad analüüsid ei uurinud. TLS-protokollis on mõlemal osapoolel võimalik algatada uue *pre-master secret*'i kokkuleppimine. Sel hetkel on võimalik ründajal lisada enda valitud sõnum kliendi ja serveri vahelisse kanalisse ning jätta mulje, et see tuli kliendilt [167]. Selle ründe vältimiseks lisati TLS-protokollile täiendus, mis selle autentsuse kadumise probleemi parandab [169]. See täiendus on realiseeritud OpenSSL-i uuemates versioonides.

3.2.1.1 ID-kaardi kasutamine TLS-protokollis

Olemasolevat standardit PKCS #11 toetavad krüptoteegid lubavad TLS-võtmevahetusel kliendi võtmevahetusteate signeerimist kiipkaardis oleva salajase võtmega. Nii kasutatakse ka Eesti ID-kaarti. Väärib rõhutamist, et eelkirjeldatud rünnet kliendi autentimise vastu [167] ei saa ära hoida üksnes ID-kaardi kasutamisega.

3.2.1.2 Mobiil-ID

Kui võtmevahetuse ajal jääb klient serverile autentimata, siis võib kliendi autentimine olla osaks edasisest suhtlusest; üks levinud viis selleks on paroolide kasutamine. Kuna server on kliendi jaoks autenditud, siis võib klient loodud ühenduse kaudu saata parooli, mida ainult serveril on lubatud teada. Selles skeemis on nõrkused juhul, kui klient ei ole tähelepanelik ning on ühenduse loonud ründaja, mitte soovitud serveriga. Sel juhul võib ründaja ise serveriga ühendust võtta ning kliendilt saadud parooli serverile edasi saata (klassikaline vahendusrünne). ID-kaarti kasutades ei ole selline rünne võimalik, sest ründaja ei ole võimeline kliendi nimel signatuure koostama ning kliendi koostatud signatuure (mis on mõeldud võtmevahetuseks ründajaga) server ei aktsepteeri.

Mobiil-ID on samuti protokoll kliendi autentimiseks. Selles protokollis püütakse klienti autentida sellega, et serveri genereeritud väljakutse (*challenge*) signeeritakse kliendi sertifikaadis olevale avalikule võtmele vastava salajase võtmega, mis asub aga kasutajale kuuluva mobiiltelefoni SIM-kaardil. Seega peab protokoll viima serveri väljakutse mobiiltelefonini ning seejärel veenduma, et klient soovis tõepoolest end sellele serverile autentida. Serveri ja mo-

biiltelefoni vahel suhtlemiseks kasutatakse DigiDocService-veebiteenust [32], mida haldab AS Sertifitseerimiskeskus. Mobiiltelefon arvutab saadud väljakutsest (millest ühe poole genereeris server ja teise poole DigiDocService) kontrollkoodi, mis koosneb neljast kümnendnumbrist. Sama kontrollkoodi saadab server ka kliendile, mis seda kasutajale näitab. Kasutaja võrdleb kahte kontrollkoodi ning nende kokkulangemisel annab mobiiltelefonile korralduse signeerida. Tervet protokolli kirjeldab DigiDocService'i spetsifikatsioon [32].

Mobiil-ID autentimisprotokolli turvaanalüüs [130] näitas, et tema tehnilises lahenduses on mitmeid nõrkusi, mis teatud juhtudel võivad ründajal lasta end mõne teise kasutajana autentida. Leiti, et DigiDocService, kes peaks olema kõigest vahendaja serveri ja mobiiltelefoni vahel, on protokolli praeguses versioonis usaldatud osapool. Probleem tuleneb järgmistest asjaoludest.

- Väljakutse signeerimisel signeeritakse ainult väljakutse. Krüptoprotokollide ettevaatliku kavandamise põhimõtted [42] soovitavad juhul, kui signatuur on mõeldud kontrollimiseks mingile konkreetsele osapoolele, lisada selle osapoole nimi signatuuri alla.
- Osa väljakutsest genereerib DigiDocService. Kuna kontrollkoodidel on ainult 10 000 võimalikku väärtust, saab DigiDocService'i kontrollkoodide väärtusi vabalt valida ning kollisioone tekitada.
- Kontrollkoodi, mida klient kasutajale näitab, arvutatab välja DigiDocService-veebiteenus, mitte server.

Lisaks leidis turvaanalüüs [130], et Mobiil-ID protokoll on vahendusrünnetele sama haavatav kui parooliga autentimine.

Ülaltoodud nõrkuste parandamine vajab muudatusi DigiDocService'i ning Mobiil-ID-d kasutavate serverite töös. Et muuta autentimiseks kasutatavat signeerimisoperatsiooni, on tarvis välja vahetada ka SIM-kaardid. Kuni nende muudatuste tegemiseni soovitame Mobiil-ID-d kasutades aktsepteerida ainult sertifikaate, mille on väljastanud AS Sertifitseerimiskeskus.

3.2.1.3 iPizza pangalink

Pangalink on Eesti pankade poolt kasutusele võetud firmapärane standard, mis võimaldab teenusepakkujatel autentida kasutajaid läbi internetipanga. Pangalingi kasutamine on Eestis üsna laialt levinud – väga tihti kasutatakse seda teise võimalusena ID-kaardiga autentimise kõrval. Pangalingi tehnilist standardit kirjeldab näiteks Swedbanki dokument [195].

iPizza protokollis vahetatakse sõnumeid üle HTTPS-protokolli. Edastatavad sõnumid on veel omakorda signeeritud ning selleks kasutatakse SHA-1 räsialgoritmi ning RSA PKCS #1 versioon 1.5 [22] signatuuride polsterdusviisi (*padding*). RSA võtmetena on toetatud kuni 4096-bitised võtmed.

3.2.1.4 TUPAS pangalink

TUPAS on Soome finantsettevõtete liidu *Finanssialan Keskusliitto* poolt loodud protokoll [20], mille abil saavad teenusepakkujad autentida kasutajaid läbi internetipanga. Protokolli kasutab Eestis Nordea pank.

TUPAS-protokollis edastatavad sõnumid on varustatud sõnumiautentimiskoodiga (MAC), mis võimaldab teenusepakkujal ja pangal kontrollida, et sõnumit ei ole vahepeal muudetud ning sõnum on saabunud salajast võtit jagavatelt osapooltelt (teenusepakkuja ja pank). Tegu ei ole krüptograafilise HMAC-sõnumiautentimiskoodiga, kuna salajane võti lisatakse räsitava stringi lõppu ning seejärel kasutatakse tavalist räsialgoritmi. Selline MAC-funktsiooni loomise meetod on teadaolevate nõrkustega [166, jaotis 4.2] ning võib ebaturvaliste räsialgoritmide kasutamisel

võimaldada protokolli liiklust vahendaval kliendil panga ja teenusepakkuja vahelisi sõnumeid modifitseerida.

Räsimiseks saab protokollis kasutada algoritme MD5, SHA-1 ja SHA-256. Standardi kohaselt peavad pangad ja teenusepakkujad loobuma 2011. aasta jooksul MD5 ja SHA-1 algoritmide kasutamisest ning võtma kasutusele SHA-256. Sõnumiautentimiskoodi salajaseks võtmeks kasutatakse vähemalt 256-bitist võtit.

3.2.2 IPsec (IKE)

IPsec on protokollistik võrguliikluse turvamiseks võrgukihis. See jaotis käsitleb IPsec-protokollistiku võtmevahetusprotokolle, peamiselt IKE-t (*Internet Key Exchange*).

IKE [113] avaldati esmakordselt 1998. aastal. Tegu on tüüpilise autenditud Diffie-Hellmani võtmevahetusega, mille osapooled rakendavad omavahelistele teadetele (vt. jaotis 2.2.3) mingit meetodit nende tervikluse tagamiseks. Selleks meetodiks võib olla signeerimine või sõnumiautentimiskoodide kasutamine. Esimesel juhul peavad osapooltel olema sertifikaadid, teisel juhul peab neil juba varasemast olema mingi ühissaladus.

IKE-võtmevahetusprotokolli on formaalsete meetoditega uurinud näiteks Meadows [140] ning Canetti ja Krawczyk [74]. Protokoll ei ole kuigi keeruline. Olgu ka mainitud, et IKE järglaseks pakutud *Just Fast Keying (JFK)* protokoll [44] on samuti formaalsete meetoditega (protokollianalüsaatoriga ProVerif [64]) analüüsitud [41].

3.2.3 Kerberos

Kerberos [153] on autentimis- ja võtmevahetusprotokoll, mis põhineb piletitel ning järgib ühekordse sisselogimise põhimõtteid. Kui klient soovib ühenduda mingi serveriga mingist domeenist, siis esmalt võtab ta ühendust selle domeeni autentimisserveriga. Autentimisserver tuvastab kliendi ning saadab talle pileti, milles sisaldub (sümmeetriline) seansivõti soovitud serveriga suhtlemiseks. Pileti abil autendib ka server kliendi. Autentimisserver võib kliendi tuvastada erinevate meetodite abil: protokolli esimestes versioonides olid selleks meetodiks jagatud võtmed, aga protokolli laiendustes lubatakse kasutada ka avaliku võtmega sertifikaate [209].

Butler jt. [73] analüüsisid Kerberost sümbolilise krüptograafia mudelis, leides, et üldjoontes on protokoll turvaline. Siiski võib väita, et see analüüs ei ole otseselt Kerberosele kohandatav, sest Kerberos kasutab krüptoprimitiive viisil, mida ei loeta üldjoontes turvaliseks. Seevastu on Boldyreva ja Kumar [68] näidanud, et Kerberos-protokollis on selline kasutusviis turvaline.

3.2.4 SKIP

SKIP (*Simple Key-Management for Internet Protocol*) [46] oli üks võimalikke võtmevahetusprotokolle IPsec-protokollistikus enne IKE kasutusele võtmist. SKIP-protokolli kasutab X-tee [131].

Protokolli kasutades eeldatakse, et kõigi osapoolte Diffie-Hellmani võtmevahetuse avalikud võtmed on teistele osapooltele autentsel viisil teatavaks tehtud (näiteks sertifikaatide abil). Sel viisil on iga kahe osapoole jaoks implitsiitselt määratud nende ühissaladus ning seda (või sellest tuletatud väärtust) saab kasutada pikaealise sümmeetrilise võtmena nende kahe osapoole vahel. Pikaealist sümmeetrilist võtit kasutatakse lühiealiste võtmete kokkuleppimiseks.

SKIP-võtmevahetusprotokoll on seega väga lihtsa struktuuriga ning lühiealise võtme kokkuleppimiseks polegi tarvis eraldi teateid vahetada. Selle hinnaks on aga transpordiprotokolli suurem kulukus, mis oli ka põhjus, miks SKIP-i ei valitud IPSec-protokollistikku [26].

3.2.5 Soovitused tulevikuks

Mobiil-ID-ga autentimine on hea näide omaloodud protokollist, mille kavandamisel pole läbi viidud põhjalikku turvaanalüüsi. Selle tagajärjel on protokollis mitmeid nõrkusi, mida tema loojad pole osanud ette näha. Kokkuvõttes soovitame Mobiil-ID protokollide ümberkavandamist, alustades soovitud turvaomaduste formaalsest kirjeldamisest ja jätkates protokollisõnumite fikseerimise ning formaalse tõestamisega (näiteks Dolev-Yao mudelis), et protokoll neid turvaomadusi ka tõepoolest rahuldab. Allikas [130] pakub ka välja võimalikke viise kõigi nende tegevuste läbiviimiseks. Selles allikas väljapakutud protokollimuutused eeldavad aga ka muudatusi kõigis Mobiil-ID-d kasutavates serverites.

SKIP-protokolli vähese kasutatavuse tõttu pole uuringu autoreil õnnestunud leida selle protokolli turvaanalüüse. Seetõttu soovitame teha ühte alljärgnevast:

- vahetada X-tees SKIP-protokoll mõne levinuma võrgukihi turvaprotokolli vastu;
- viia läbi SKIP-protokolli (X-tees kasutatavate valikute mahus) formaalne analüüs.

3.3 ARVUTI JA ID-KAARDI VAHELINE SUHTLUS

PKCS #11 [28] on standard, mis kehtestab rakendusprogrammide liidese kiipkaartidele ja teistele krüptograafilistele riist- ja tarkvaraelementidele. Standard kehtestab loetelu krüptograafilistest operatsioonidest, mida kiipkaart võib osata sooritada. Võimalike operatsioonide liike on palju, nende alla kuuluvad

- uute võtmete loomine nii sümmeetriliste kui ka asümmeetriliste krüptosüsteemide jaoks,
- krüptograafiliste objektide (võtmete) sisestamine kaarti,
- andmete räsimine, krüptimine, dekrüptimine, signeerimine, verifitseerimine, sõnumiautentimiskoodide arvutamine,
- ühe võtme krüptimine teisega,
- krüptitud võtme lahtikrüptimine ja selle edaspidine kasutamine kaardis,
- kaardis sisalduva võtme modifitseerimine teatud viisil (XOR-operatsiooniga) ja teised operatsioonid kaardis salvestatud võtmetega.

Samuti saab iga operatsiooni läbi viia erinevate krüptoalgoritmide ning nende parameetritega. Parameetrite hulka kuuluvad näiteks plokkšifri töörežiim, avateksti polsterdusviis jms. Iga krüptograafiline element võib realiseerida ainult osa standardist. Näiteks Eesti ID-kaart realiseerib väga väikese osa – ta on võimeline saadud sisendit teatud algoritmidega räsima ning kaardis sisalduvate RSA-võtmetega dekrüptima / signeerima / verifitseerima. Seejuures võib signeerimine / dekrüptimine toimuda kas ilma polsterduseta (mehhanism RSA_X_509) või PKCS #1 standardi versioonis 1.5 [22] spetsifitseeritud polsterdustega (mehhanism RSA_PKCS) krüptimiseks (polsterdusviis RSAES-PKCS1-v1_5) ja signeerimiseks (polsterdusviis RSASSA-PKCS1-v1_5).

Standardi PKCS #11 rikkalikkus annab palju võimalusi erinevateks rünneteks krüptograafiliste elementide vastu. Näiteks, kui mõni võti K kaardis on märgitud kasutatavaks nii dekrüptimisel kui ka teiste võtmete krüptimisel, siis on võimalik teised kaardis olevad võtmed leida, nad kõigepealt võtmega K krüptides (kaart tagastab krüptoteksti) ning seejärel saadud krüptoteksti võtmega K dekrüptides (kaart tagastab avateksti). Nende rünnete kasutatavus teatud kaartidel on ka praktikas järgi kontrollitud [78, 85, 70].

Eesti ID-kaart realiseerib väikese osa PKCS #11-liidesest. Seetõttu ei ole kirjeldatud ründed talle rakendatavad. Aleksei Gornõi bakalaureusetöös [103] läbiviidud rünnetes ei rünnata mitte ID-kaarti, vaid arvutit.

PKCS #11-standard spetsifitseerib mitmeid krüptoalgoritme, mida kiipkaart või muu krüptograafiline element võib sisaldada. Uute algoritmide (näiteks SHA-3 või elliptilistel kõveratel põhinevad krüptosüsteemid) lisandudes tuleb seda standardit täiendada.

3.4 DIGIALLKIRJASTAMISPROTOKOLLID

Digiallkirjastamise[4] eesmärk on *salgamistõrje* (*non-repudiation*) – kui üks isik on mingi dokumendi allkirjastanud, siis peab allkirjastatud dokumendi saanud isikutel olema võimalus kolmandaid osapooli veenda, et esimene isik selle dokumendi tõepoolest allkirjastas. Samas on oluline ka *korrektsus* – kui esimene isik pole mingit dokumenti allkirjastanud, siis ei tohi teised osapooled uskuma jääda, et ta seda tegelikult tegi.

Digiallkirjastamisel on oluline silmas pidada, kus tehakse otsus mingi dokument allkirjastada. Sageli on otsustajaks inimene, kes ei ole tehniliselt suuteline ise digiallkirja koostama, vaid delegeerib selle edasi mingile arvutile. Inimene peab arvutit usaldama, et see digiallkirjastaks just selle dokumendi, mida inimene soovis allkirjastada. Samuti peab inimene usaldama seda seadet, mis talle dokumente näitab ja mille kuva põhjal ta teeb otsuse mingi dokument allkirjastada. Selline *usaldatud arvutusbaasi* küsimus kerkib digiallkirjastamisel alati. Järgnevad jaotised iseloomustavad, kui suur on see usaldatud baas erinevates protokollides. Eelistada tuleks protokolle, kus usaldatud baas on väiksem.

Eesti kontekstis on peamised digiallkirja andmise vahendid ID-kaart ja Mobiil-ID ning kasutatavad protokollid põhinevad DigiDoc-teegil [186]. Seetõttu eeldabki järgnev analüüs, et nii digiallkirja konstrueerimisel kui ka kontrollil kasutatakse nimetatud vahendeid. Eesti kontekstis võib olla oluline ka välisriigis moodustatud digiallkirja kontrollimine DigiDoc-vahenditega. Sel juhul on turvaotsuste tegemiseks siiski tarvis teada ka viisi, kuidas digiallkiri moodustati. Võib arvata, et kui moodus on sarnane Eestis levinud vahenditega, siis on ka saadavad turvagarantiid sarnased.

3.4.1 Allkirjastamine ID-kaardiga

Allkirjastamine arvutis ID-kaarti on võimalik digiallkirjastamiseks kasutada mitmel viisil. Kontseptuaalselt lihtsaim on DigiDoc-teegi kasutamine oma arvutis. Sellisel juhul kutsub DigiDoc-funktsionaalsust välja mingi rakendus, mille ülesannete hulka kuulub signeeritava dokumendi kasutajale näitamine, selle dokumendi esitamine õiges vormingus ning edastamine DigiDoc-teegile. Too rakendus kuulub seega usaldatud arvutusbaasi.

DigiDoc-teek arvutab digiallkirjastatava dokumendi sõnumilühendi. Sõnumilühend edastatakse ID-kaardile, mis viib läbi astendamise RSA salajase astendajaga. Saadud väärtus lisatakse digiallkirja objektile. Samuti saab teeki kasutada OCSP kinnituste saamiseks moodustatud digiallkirjale. OCSP kinnituse küsimine toimub peale digiallkirja moodustamist.

ID-kaardiga allkirjastamine on väga lihtne protokoll. Tema usaldatud arvutusbaasi kuulub DigiDoc-teek ning rakendus, mis seda teeki kasutab. Loomulikult kuuluvad usaldatud arvutusbaasi ka kasutaja arvuti ning ID-kaart. Arvuti ning ID-kaardi vahelist suhtlust reguleerib PKCS #11 [28] standard. OCSP-protokoll on samuti standardiseeritud [152].

Protokoll OCSP See protokoll on allkirjale kehtivuskinnituste küsimiseks. Moodustatud allkiri saadetakse sertifitseerimiskeskusele, mis tagastab signeeritud vastuse, kus öeldakse, kas esialgse allkirja moodustanud isiku sertifikaat on hetkel kehtiv. Kehtivuskinnitust võib küsida

[4] Eesti õigusaktides ja tavakeeles kasutatakse *digital signature* tõlkevastena "digitaalallkiri". Korrektne termin on "digitaalsignatuur".

nii allkirja moodustaja kui ka keegi teine, näiteks verifitseerija. Protokoll on lihtne, koosnedes vaid päringust ja päringuvastusest. Protokoll võib olla tundlik taasesitusrünnete suhtes – kui allkirja verifitseerija teeb OCSP-päringu, siis võib tema vastuseks esitada mõne varasema vastuse, mis on sama sertifikaadi kohta käinud päringu kohta tehtud. Ründe vastu aitab nonsi (*nonce*) kasutamine päringus ja vastuses; see on samuti standardis [152] spetsifitseeritud.

Allkirjastamine DigiDoc-portaalis DigiDoc-portaalis digitaalallkirjastamiseks laadib kasutaja allkirjastatava(d) dokumendi(d) portaali. Seejärel käivitatakse kasutaja arvutis Java-plugin, mis saab ühe parameetrina ette signeeritava dokumendikonteineri räsi. Programm eeldab, et see räsi on arvutatud SHA-1 algoritmiga – ta nõuab, et räsi pikkus oleks 20 baiti ning enne järgmise sammu tegemist lisab ta sellele räsile SHA-1 algoritmiidentifikaatori (vt. jaotis 4.1.4). Ainus samm signeerimisel, milleks seda pluginat kasutatakse, on ID-kaardil oleva salajase võtmega RSA-astendamine. Astendamise tulemus saadetakse tagasi DigiDoc-portaalile, mis viib läbi ülejäänud sammud signeeritud dokumendi loomisel, k.a sertifikaadi kehtivuskinnituse küsimise.

Kuna portaal teostab enamuse signeerimiseks vajalikest sammudest, k.a signeeritava räsi arvutamise, siis lisandub ta usaldatud arvutusbaasile. Kui ründaja saavutab portaali üle kontrolli, siis on ta võimeline mistahes dokumente signeerima, andes Java-pluginale ette endavalitud räsiväärtuse.

3.4.2 Allkirjastamine Mobiil-IDga

Mobiil-IDga allkirjastamiseks tuleb kasutada DigiDocService-veebiteenust. Veebiteenusele saadetakse allkirjastatavad failid või nende räsid. Teenusepakkuja saadab allkirjastatava konteineri räsi edasi mobiiltelefonile. Mobiiltelefon näitab kasutajale kontrollkoodi ning kasutaja peab seda võrdlema koodiga, mida näitab talle veebiteenuse poole pöördunud rakendus. Nagu näitab protokollianalüüs [130], kuuluvad sellisel juhul usaldatud arvutusbaasi kasutaja arvuti, DigiDocService-veebiteenuse pakkuja, kasutaja mobiiltelefon, telefonis olev SIM-kaart ning mobiilioperaator. Kui ründaja neist mõnda kontrollib, siis saab ta ükskõik millist dokumenti allkirjastada.

3.4.3 Digiallkirja kontrollimine

Digiallkirja on võimalik kontrollida kas kasutaja enda arvutis või DigiDoc-portaalis. Kontrollimine võib sisaldada allkirjastamisel kasutatud sertifikaadi suhtes OCSP-päringute tegemist. Portaalis allkirja kontrollimise protokoll on lihtne – kasutaja laadib signeeritud dokumendi portaali ning portaal vastab, kas allkiri kehtib või mitte. Loomulikult tuleb sealjuures portaali usaldada.

3.4.4 Soovitused tulevikuks

Meie soovituseks signeerijale on võimalusel kasutada signeerimisviisi, mille korral usaldatud arvutusbaas on väiksem. Muuhulgas peaksime positiivseks ka arengut, kus mingis veebikeskonnas (näiteks internetipangas) loodud dokumenti signeerides oleks kasutajal olemas valik, millise rakendusega (veebikeskkonnas või kasutaja arvutis jooksvaga) dokument allkirjastatakse. Eriti teravalt ilmneb see probleem just internetipangas maksekorraldusi signeerides, sest kasutajal puudub igasugune kontroll selle üle, milline dokument tegelikult signeeritakse.

Sertifitseerimiskeskus AS soovitab, et digiallkirjastamist pakkuvad veebikeskkonnad võimaldaksid kasutajal tutvuda allkirjastatava dokumendiga enne ning loodud digiallkirjaga peale

allkirjastamist [37]. Kui veebikeskkond seda soovitust järgib (mitte kõik ei tee seda) ning kasutaja kontrollib, et õige dokument digiallkirjastati, siis märkab ta, kui keskkond allkirjastatava dokumendi ära vahetas. Samas ei eemalda ka see soovitus veebikeskkonda usaldatud arvutusbaasist, sest valele dokumendile antud digiallkiri on olemas ka juhul, kui kasutaja pärast märkab, et ta soovis allkirjastada hoopis teist dokumenti. Valele dokumendile antud digiallkirja kehtetuks tunnistamine on ilmselt seotud juriidiliste raskustega.

Mis aga puutub standardite ja tarkvara tuge digiallkirjastamisele Eestis, siis soovitame neid mõlemaid muuta modulaarsemaks. Toetatud räsi- ja signeerimisalgoritmide ning võtmepikkuste nimekirja muutmine tuleb teha lihtsamaks nii DigiDoc-standardis kui ka DigiDoc-teegis ja ilmselt siis ka DigiDocService-veebiteenuses.

Räsialgoritmide valiku osas soovitame loobuda SHA-1 algoritmi toetamisest ning enne SHA-3 konkursi lõppu võtta kasutusele SHA-2 perekonna algoritmid. Hetkel ei ole alust arvata, et ühegi selle perekonna räsifunktsiooni vastu leitaks lähemate aastate jooksul tõsiseid ründeid (vt. jaotis 2.3.2), seetõttu ei ole meil alust anda soovitust sellest perekonnast mingi konkreetse algoritmi eelistamiseks.

Rõhutame siinkohal ka seda, et turvamehhanism on nii nõrk kui nõrk on tema kõige nõrgem lüli. DigiDoc-formaadis signeeritud dokumendi koostamisel kasutatakse räsialgoritme kõigepealt signeeritavate dokumentide räsimiseks ning seejärel DigiDoc-konteineri räsimiseks. Kui nendes kohtades kasutada erinevaid algoritme, siis signatuuride võltsimiskindlus on piiratud neist kahest algoritmist nõrgema kollisioonikindlusega.

Lisaks soovitame, et kui allkirja verifitseerimisel kasutatakse sertifikaadi kehtivuse kindlakstegemiseks OCSP-protokolli, siis tuleks taasesitusrünnete (*replay attacks*) vältimiseks lisada päringule nonss, nii nagu vastav standard seda ette näeb ja nagu DigiDoc-teek seda ka teeb.

3.5 TRANSPORDIPROTOKOLLID

Pärast turvalise võtmevahetusprotokolli tööd on selle protokolli osapooltel olemas ühine saladus, millest tuletada võti või võtmed, mida kasutada edaspidise osapooltevahelise liikluse kaitsmiseks. *Transpordiprotokoll* kirjeldab, milliseid krüptograafilisi operatsioone tuleb rakendada saadetavatele andmetele, et tagada nende konfidentsiaalsus ja terviklus.

Kuigi teoreetiliselt on teada, kuidas konstrueerida turvalisi kanaleid krüptoalgoritmidest ja sõnumiautentimiskoodidest [57, 58, 174], esineb protokollistike transpordiprotokollides palju erinevaid konstruktsioone. Enamus neist on turvalised, kuid mingitel täiendavatel eeldustel krüptimis- ja sõnumiautentimisalgoritmidele, mida teoreetilised konstruktsioonid ei vaja.

3.5.1 SSL/TLS

SSL/TLS-i transpordiprotokollis (*record protocol*) [86] rakendatakse saadetavale paketile ja tema järjekorranumbrile kõigepealt sõnumiautentimiskoodi. Pakett ja kood konkateneeritakse ning neile lisatakse polsterdus, et teate kogupikkus oleks kasutatava plokkšifri plokipikkuse kordne. Polsterdusbaitide väärtus ja nende arv on omavahel seotud. Seejärel teade krüptitakse ja saadetakse välja. See konstruktsioon ei vasta täpselt teooriale, mis soovitab kõigepealt krüptida ning seejärel autentida. Teadaolevad konstruktsioonid [57], kus see konstruktsioon nõrgaks osutub, on küll puhtteoreetilised. Samuti on näidatud, et kui krüptoalgoritmiks on plokkšiffer CBC-režiimis (nagu SSL/TLS-i transpordiprotokollis) või jadašiffer, siis on ka kõigepealt autentiv ja seejärel krüptiv konstruktsioon turvaline [128]. Küll aga leidub realistlik rünne [200], juhul kui transpordiprotokolli realisatsioon vigaselt formaaditud paketi saamisel avalikustab,

millise koha peal tema dekodeerimine ebaõnnestus – kas oli polsterdus vale pikkusega või ebaõnnestus sõnumiautentimiskoodi kontroll. OpenSSL (alates versioonidest 0.9.6i ja 0.9.7a) realiseerib kaitse selle ründe vastu.

3.5.2 IPsec-protokollistiku transpordiprotokoll

Encapsulating Security payload (ESP) [114] on võrgupakettide kodeerimisviis IPsec-protokollistikus. Muuhulgas kasutab seda kodeerimisviisi X-tee [131]. Sel viisil IP-paketti kodeerides tema sisu (või terve pakett, sõltuvalt kasutusmoodist) kõigepealt krüptitakse ning seejärel rakendatakse krüptitud paketile sõnumiautentimiskoodi. See vastab teoreetilistele soovitustele [57]. SKIP-võtmevahetusprotokolli kasutades peab kodeeritud pakett täiendavalt sisaldama selle paketi krüptimiseks kasutatud võtit (mis on krüptitud pikaealise võtmega).

3.5.3 Wi-Fi transpordiprotokollid

IEEE 802.11 (Wi-Fi) traadita võrgu standard spetsifitseerib võrgutaseme transpordiprotokollid võrguliikluse kaitsmiseks. Neist esimeses – *Wired Equivalent Privacy (WEP)* [31, jaotis 8.2.1] – on mitmeid nõrkusi ning tänaseks on ta murtud. Leidub aktiivne rünne, mille abil võib kasutatava transpordivõtme leida vähem kui minutiga [199].

WEP-protokolli nõrkusteks olid muuhulgas jadašifri RC4 (jaotis 2.1.3) kasutamine, tema võtmejada ebakorrektne initsialiseerimine ja kasutu mehhanism andmepakettide tervikluse tagamiseks. Protokoll TKIP (*Temporal Key Integrity Protocol*) [31, jaotis 8.3.2] parandab WEP-protokolli mitmel viisil. Iga paketi jaoks kasutatakse eraldi krüptimisvõtit, mis genereeritakse pikaealisest võtmest. Samuti on terviklusmehhanismi parandatud, olgugi et see on ikka veel nõrk. Krüptimisalgoritmina on endiselt kasutusel RC4. TKIP on standardiseeritud kui *Wi-Fi Protected Access (WPA)*.

WPA vastu on teada rünne [198, 159], mille abil on võimalik leida terviklusmehhanismis kasutatav võti, misjärel on ründajal võimalik teeselda suhtluse osapooleks olemist ning teisele osapoolele endavalitud sisuga pakette saata. Rünne ei võimalda krüptimisvõtmeid leida või osapooltevahelist liiklust dekrüptida.

Protokolli CCMP (*Counter Mode with Cipher Block Chaining Message Authentication Code Protocol*) [31, jaotis 8.3.3] nimi sisaldab juba suure osa sellenimelise plokkšifrist andmepakettide kodeerimisviisi konstruktsiooni detailidest. Tema põhiosaks on plokkšifri kasutamine CCM-režiimis [205]; lisaks kehtestab standard, kuidas teatud autentimisandmeid arvutada. CCMP, kus plokkšifriks on 128-bitise võtmega AES, on standardiseeritud kui *Wi-Fi Protected Access II (WPA2)*.

CCM-režiim on tõestatavalt turvaline [111] ning WPA2-kodeerimisviisi vastu teadaolevaid ründeid ei leidu. Küll aga on CCM-režiimi kritiseeritud tema vähese efektiivsuse, kontseptuaalse keerulisuse ja detailide peensuse tõttu [173].

Selle jaotise lõpetuseks tasub veel kord rõhutada, et et kui soovitakse Wi-Fi liiklust turvata, siis tuleb kasutada WPA2-mehhanismi. Varasemad mehhanismid on ebaturvalised.

3.6 SOOVITUSED TULEVIKUKS

Selle aruande põhiline soovitus on kasutada tõestatavalt turvalisi protokolle, sest protokollide turvatõestusi osatakse esitada ning neist lähtuvate garantiide ulatusest saadakse aru. Kui infosüsteemis kasutatavatel protokollidel on turvaomadused, mida need infosüsteemid vajavad, siis ei ole tarvis neid protokolle tulevikus välja vahetada. Seega puudub vajadus infosüsteeme nii ulatuslikult modulariseerida, et näiteks terve võtmevahetusprotokoll koos oma loogi-

kaga oleks lihtsasti vahetatav. Küll aga on tarvis, et protokollis kasutatavad krüptograafilised primitiivid ning nende võtmepikkused oleksid vahetatavad, mistõttu protokollide realisatsioonid peavad olema modulaarsed ning kasutatavad primitiivid ei tohi olla lähtekoodis fikseeritud (*hard-coded*), vaid peavad jääma koodist sõltumatult konfigureeritavaks.

4 TARKVARASÜSTEEMIDE TUGI

4.1 PROTOKOLLIDE REALISATSIOONID

See jaotis kirjeldab krüptoprimitiivide kasutust mõnes Eesti jaoks olulises protokollistikus ning analüüsib krüptograafiliste algoritmide vahetuse mõju vastavatele süsteemidele. Arvestades, et aruande koostamise käigus ei õnnestunud saada ei juurdepääsu vajalikele Eesti riigi infosüsteemidele ega nende süsteemide tehnilist dokumentatsiooni, piirdub analüüs üldkasutatavate avalike protokollistike ning ID-kaardi ja sellega seotud tehnoloogiatega.

4.1.1 Ülemaailmse levikuga rakendused

4.1.1.1 Krüptoteegid: OpenSSL, Bouncy Castle

OpenSSL [13] on C-keeles kirjutatud raamistik, mis realiseerib protokollid SSLv2, SSLv3 ja TLSv1 ning neis protokollides vajalikud primitiivid. Oluline osa OpenSSL-ist on sertifitseeritud NIST-i poolt vastavana FIPS 140-2 [24] nõuetele. Sertifitseeritud primitiivide hulka kuuluvad AES, 3DES, DSA, RSA, SHA-1, SHA-2 perekonna liikmed ja neile räsifunktsioonidele vastavad HMAC-konstruktsioonid. OpenSSL-i arendab suur rahvusvaheliselt tunnustatud kogukond ning enam kui kümneaastase arendusperioodi tulemusena on saavutatud väga stabiilne, optimeeritud ja hästi modulariseeritud lähtekood. Tänu paindlikele Apache'i-laadsetele litsentsitingimustele on OpenSSL lihtsasti kasutatav nii äri- kui ka vabavaratoodetes, ning C-põhistes süsteemides tuleb teda kindlasti eelistada krüptograafiliste primitiivide ise realiseerimisele.

Java- ja C#põhiste süsteemide puhul tuleb iseleiutamisele samuti eelistada suurema kogukonna poolt arendatavat teeki. Üheks selliseks on Bouncy Castle, mis pole küll FIPS 140-2 sertifitseeritud, kuid mille litsentsitingimused (MIT-laadne litsents), stabiilsus ja suur krüptoprimitiivide hulk rahuldab enamiku praktikas esinevatest vajadustest. Toetatud primitiivide hulka kuuluvad AES, Blowfish, DES, IDEA, RSA, ElGamal, SHA-1, SHA-2 perekonna räsifunktsioonid jpt primitiivid; täieliku nimekirja leiab spetsifikatsioonist [3].

4.1.1.2 Apache SSL soovitatav konfiguratsioon

Veebiserveriga Apache kasutatakse enamasti mod_ssl moodulit [97], mis realiseerib TLS-protokolli toe, võimaldab serveerida HTTPS URL-idega veebisaite ning samuti autentida X.509 sertifikaatidega kasutajaid. mod_ssl mooduli konfiguratsioonis on võimalik reguleerida, milliseid algoritme SSL-protokolli kokkuleppimisel server aktsepteerib ning seeläbi juhtida kliendiga kokkulepitava SSL-seansi turvalisust.

Vaikimisi konfiguratsioonis on `SSLCipherSuite` direktiivi väärtus `"ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP"`, mille puhul võivad osad

brauserid kokku leppida ka näiteks DES-krüpteerimisalgoritmi, mille kasutamist see aruanne ei soovita. Debian 6.0 distributsioonis kasutatakse vaikimisi mõnevõrra turvalisemat stringi "HIGH:MEDIUM:!ADH", kuid ka sellisel juhul lubatakse näiteks räsialgoritmi MD5, mille kasutamist see aruanne samuti ei soovita.

Soovitame kasutada direktiivi väärtusena näiteks stringi "!ADH:!DES:!3DES:HIGH", mis lubab DH ja RSA võtmevahetusalgoritmid, RSA ning DSS signeerimisalgoritmid, AES krüptoalgoritmi 128- ja 256-bitise transpordivõtmega ning SHA-1 räsialgoritmi. Viimast kasutatakse HMAC-sõnumiautentimiskoodi moodustamiseks, millist rakendust võib käesoleva aruande ajahorisondi jooksul pidada turvaliseks. Ka teiste algoritmide valik läheb kokku käesoleva aruande muudes peatükkides toodud soovitustega.

Juhul kui aja jooksul võetakse kasutusele 2048-bitistel võtmetel põhinevad X.509 sertifikaadid ning OpenSSL-i, mod_ssl'i ning Apache'i uuendatakse ning võetakse kasutusele SHA-2 ja SHA-3 räsialgoritmid, siis peaks soovitatud direktiivis olev HIGH makro lubama automaatselt ka turvalisemaid algoritme. Siiski soovitame perioodiliselt direktiiviga lubatud algoritmide nimekirja üle vaadata, ebaturvalised algoritmid jõuga keelata ning järjestada lubatud algoritmid, arvestades serverite jõudlust ja turvaeesmärke.

4.1.2 ID-kaardi baastarkvara

ID-kaardi baastarkvara [38] koosneb järgmistest komponentidest [112]:

- libdigidoc – DigiDoc-tüüpi failide haldamise teek. Sisaldab funktsionaalsust [29] failide allkirjastamiseks, allkirjastatud failidele OCSP-kinnituse võtmiseks, allkirjade kontrollimiseks, andmete krüpteerimiseks ja dekrüpteerimiseks vastavalt XML-ENC standardile [106].
- libdigidocpp – BDoc-tüüpi failide haldamise teek. Sisaldab funktsionaalsust failide allkirjastamiseks ja allkirjade kontrollimiseks. Krüptimisfunktsionaalsust ei ole. Kirjutatud keeles C++.
- JDigiDoc – DigiDoc- ja BDoc-tüüpi failide haldamise teek, mis on kirjutatud Javas. Võimaldab faile nii signeerida kui ka krüptida.
- ID-kaardi utiliit ja smartcardpp.
- Veebilehitsejate pluginad.

Loetletud komponendid kasutavad järgmisi üldlevinud teeke:

- OpenSC [39] on standardi PKCS #11 [28] realisatsioon. See standard kehtestab rakendusliidese kiipkaartidega suhtlemiseks. OpenSC ei kasuta niivõrd ise krüptot, kuivõrd vahendab krüptograafilisi päringuid rakenduse ja kaardi vahel.
- PC/SC spetsifikatsiooni [35] realisatsioon. See komponent korraldab arvuti ja kaardi(lugeja) vahelist suhtlust. Komponendil ei ole krüptograafilist funktsionaalsust.
- OpenSSL.
- Teegid XML-i parsimiseks.
- Teegid käivitatava koodi dünaamiliseks laadimiseks.

Järgnevas kirjeldame krüptoprimitiivide kasutust neis komponentides ja nende konfigureeritavust. Samuti püüame anda soovitusi, mida neid komponente kasutades või edasi arendades võiks silmas pidada. Ülevaate aluseks on komponentide lähtekood [38]. Rõhutame, et koodi on loetud, kuid ei ole püütud käivitada.

4.1.2.1 libdigidoc

See teek kasutab krüptograafilisi primitiive järgmistel signeerimisega seotud juhtudel:

OCSP-kontrollipäringute koostamine Päringu koostamiseks kasutatakse OpenSSL-teeki, kuid algoritmide identifikaatorid antakse ette libdigidoc-teegis. Konkreetselt antakse ette räsialgoritmi identifikaator (SHA-1), mida kasutatakse päringukonteineri räsimiseks enne selle signeerimist. Signeerimiseks ning võtme lugemiseks PKCS #12 [23]-standardile vastavast konteinerist kasutatakse samuti OpenSSL-i, nii et võimalike signeerimisalgoritmide tugi sõltub ainult OpenSSL-ist.

OCSP-päringuvastuste kontroll Signatuuri kontrolliks kasutatakse OpenSSL-i. Kontroll, kas saadud päringuvastus üldse kehtib asjassepuutuva sertifikaadi kohta, sooritatakse libdigidoc-is. Selleks räsitakse sertifikaadi väljaandja nimi ja tema avalik võti fikseeritud räsialgoritmiga (SHA-1), seejuures kontrollimata, kas päringuvastuses on kasutatud seda algoritmi või mõnda teist.

Signeerimine Signeerimisel moodustatakse kõigepealt konteiner, kuhu lisatakse signeeritavate failide sõnumilühendid ning vajadusel teised signatuuriga kaasaskäivad atribuudid. Sellest konteinerist arvutatakse enne signeerimist omakorda sõnumilühend. Teek võimaldab esimesena mainitud sõnumilühendit arvutada räsialgoritmiga SHA-1 või SHA-256. Viimast sõnumilühendit arvutab ta algoritmiga SHA-1. Signatuuri arvutamisega tegeleb füüsiliselt ID-kaart. Enne 1. jaanuari 2011 välja antud kaardid kasutasid 1024-bitist RSA-d, millest soovitame järgmise 5 aasta jooksul loobuda. Alates 1. jaanuarist 2011 väljastatavad kaardid kasutavadki juba 2048-bitist RSA-d, mida võib kaartide kehtivusaja piires pidada piisavalt turvaliseks.

Signatuuride verifitseerimine teek eeldab, et tegu on RSA-ga loodud signatuuridega, kus kasutatav räsialgoritm on SHA-1 või SHA-256. RSA-signatuuri verifitseerides kasutatakse OpenSSL-i madala taseme funktsioone, mis signatuuri väärtusest lähtudes taastavad dokumendi räsi. Seda räsi võrreldakse dokumendi räsiga. Toetatud on ka elliptilistel kõveratel baseeruvad signatuurid, mida kontrollitakse OpenSSL-i funktsioonidega.

Fikseeritud algoritmide identifikaatorid on programmikoodi sisse kirjutatud, nii et nende muutmine ja/või konfigureeritavaks tegemine nõuab koodi läbivaatust. Samuti on mitmes kohas programmikoodi sisse kirjutatud krüptograafilise algoritmi (peamiselt räsifunktsiooni) väljundi eeldatav pikkus. Seetõttu ei pruugi teegi uuendamisel piisata kõigi nende kohtade läbivaatamisest, kus esineb sõne `sha1` või `SHA1` või `SHA256`.

Teek libdigidoc kasutab krüptograafilisi primitiive ka järgmistel krüptimisega seotud juhtudel:

Transpordivõtme(te) genereerimine libdigidoc kasutab standardset hübriidse krüptimise meetodit, kus dokument krüptitakse sümmeetrilise võtmega ja see sümmeetriline transpordivõti dokumendi vastuvõtja avaliku võtmega. Transpordivõti tuleb iga kord uuesti genereerida. libdigidoc-teegis on fikseeritud, et sümmeetrilise krüptoalgoritmina kasutatakse 128-bitise võtmega AES-i ahelarežiimis (*Cipher Block Chaining (CBC) mode*) [14, 207]. Võtme genereerimiseks küsitakse operatsioonisüsteemilt juhuslike baite, mis segatakse OpenSSL-i pseudojuhuarvude generaatori olekusse. Seejärel küsitakse pseudojuhuarvude generaatorilt juhuslikke baite arvul, millest jätkuks nii võtmele kui ka ahelarežiimi initsialiseerimisvektorile. Neid baite ei kasutata aga otse võtme ja initsialiseerimisvektorina, vaid hoopis sisendina OpenSSL-teegi funktsioonile `EVP_BytesToKey`, mille ülesanne on konstrueerida võti argumentidena etteantud paroolist ja soolast (*salt*). Konstruktsioon kasutab ka räsifunktsiooni, milleks libdigidoc'is on fikseeritud MD5.

Krüptimine Krüptimisel kasutatakse kõigepealt genereeritud transpordivõtit, et krüptida dokument, ning seejärel saaja sertifikaadist saadud avalikku võtit, et krüptida transpordivõti. Dokumendi krüptimiseks kasutatakse 128-bitise võtmega AES-i ahelarežiimis. Transpordivõtme krüptimiseks kasutatakse RSA-d. libdigidoc ei sea piiranguid RSA võtme pikkusele. Transpordivõtme krüptimisel kasutatakse standardi PKCS #1 versioonis 1.5 [22] spetsifitseeritud polsterdamist.

Dekrüptimine Dekrüptimisel leitakse kõigepealt transpordivõti ja seejärel dokument, tehes krüptimisel läbiviidavate operatsioonide pöördoperatsioone.

Transpordivõtmete genereerimise lahendusele tuleb teha mitmeid etteheiteid. *Esiteks*, pseudojuhuarvude generaatori väljund on juba otse kasutatav võtmena (ja initsialiseerimisvektorina), nii et `EVP_BytesToKey` kasutamine on ebavajalik ja kasutu (kuid tähelepanu – see pole sama, mis "kahjulik"). *Teiseks*, `EVP_BytesToKey` instrueeritakse kasutama MD5-räsifunktsiooni, mida peaks igasugustes krüptograafilistes rakendustes vältima. *Kolmandaks*, `EVP_BytesToKey` realiseerib PKCS #5 standardis [30] esitatud võtme- ja initsialiseerimisvektori genereerimise meetodi PBKDF1, mis eeldab, et räsifunktsiooni väljundi pikkus on vähemalt sama suur kui võtme ja initsialiseerimisvektori pikkused kokku. Siintoodud juhul see nii ei ole (MD5 väljundi pikkus on 128 bitti, AES-i võtme pikkus on kirjeldatud juhul samuti juba 128 bitti ning initsialiseerimisvektori pikkus samuti 128 bitti). Sellisel juhul on `EVP_BytesToKey` käitumine ebastandardne.

PKCS #1 v1.5 polsterdusviis on teatud olukordades ebaturvaline [66]. Olgugi et dokumenditranspordi kontekstis need riskid ilmselt ei realiseeru, peab dekrüptimisrakenduses olema ettevaatlik, et mitte luua pahatahtlikul saatjal võimalust teada saada, kas vastuvõtja leidis krüptitud dokumendist õigesti polsterdatud transpordivõtme. Kui vastuvõtja kasutab dekrüptimiseks oma ID-kaarti[5], siis on selle võimaluse välistamine ID-kaardi ülesanne. On küllalt tõenäoline, et ID-kaart seda ei tee – kui transpordivõtme polsterdus ei ole korrektne, siis on tulemuseks veateade, ning libdigidoc logib selle veateate.

4.1.2.2 libdigidocpp

See teek kasutab krüptograafilisi primitiive samadel signeerimisega seotud juhtudel, mis libdigidoc. Võrreldes libdigidoc-teegiga on siin räsifunktsioonide valik suurem – lisandunud on SHA-2 perekonna räsifunktsioonid. Nende valik on enamuses koondatud klassi `crypto/Digest`, mis muudab uute räsialgoritmide lisamise lihtsamaks. Samas on ka selles teegis SHA-1 mõne koha peal signatuuride loomise klassides koodis fikseeritud.

4.1.2.3 JDigiDoc

JDigiDoc on Java-keeles realiseeritud DigiDoc-tüüpi failide haldamise teek, mis kasutab krüptograafilisi operatsioone läbi `javax.crypto` liidese. Ajatemplite ning OCSP-päringute ja vastuste töötlemiseks kasutab JDigiDoc teeki Bouncy Castle [2]. Sarnaselt libdigidoc-teegiga on ka JDigiDoc kasutatav failide allkirjastamiseks, signatuuride kontrolliks, failide krüptimiseks ja dekrüptimiseks.

Ka JDigiDoc-teeki on SHA-1 kasutamine sisse kirjutatud ning sellest lahtisaamine vajab koodi täielikku läbivaatust. Muude algoritmide kasutus on aga paremini parametriseeritud kui libdigidoc-teegis. Muud algoritmid näivad olevat fikseeritavad konfiguratsioonifailis. Samas ei paista teegis olevat mehhanisme, mille abil mingeid algoritme ebaturvaliseks tunnistada.

[5] ID-kaardil oleva autentimisvõtme kasutamine dekrüptimiseks läheb iseenesest vastuollu põhimõttega, et erinevate kasutusstsenaariumite jaoks on olemas erinevad võtmed

Dokumentide krüptimisel kasutatakse transpordivõtme krüptimiseks samuti PKCS #1 v1.5 polsterdusviisi, mis võib teatud tingimustel nõrkuseks osutuda. Teisi polsterdusviise ei lubata, aga koodis on olemas kohad, kus seda konfigureerida. Transpordivõti genereeritakse standardsete vahenditega krüptoteegist. Koodis on fikseeritud, et transpordivõti on 128-bitine. Kasutatav krüptimisalgoritm on samas konfigureeritav.

4.1.2.4 ID-kaardi utiliit

Teek sisaldab funktsionaalsust mitmesuguste operatsioonide sooritamiseks kiipkaardiga; krüptograafiline funktsionaalsus teegis peaaegu puudub. Nende operatsioonide seas on ka kaardil olevate RSA-võtmetega RSA-astendamiste läbiviimine. Need operatsioonid saavad argumendina ette dokumendi räsi ning tagastavad signeeritud räsi. Teegi hetkeversioon sisaldab funktsioone, mis eeldavad, et räsi on moodustatud SHA-1, MD5 või SHA-1+MD5 algoritmiga. Enamate räsifunktsioonide lisandumisel võib see koht antud teegi lähtetekstis vajada ülevaatamist.

4.1.2.5 Teised komponendid

Läbiviidud analüüs ei tuvastanud ID-kaardi baastarkvara teistes komponentides krüptograafilist funktsionaalsust.

4.1.3 DigiDocService

DigiDocService on protokollil SOAP põhinev veebiteenus, mida pakub AS Sertifitseerimiskeskus [32]. Teenus on oluline eelkõige seetõttu, et läbi tema toimub isikutuvastus ja digitaalallkirjastamine Mobiil-ID-ga. Sama teenuse abil on võimalik ka ID-kaardiga digiallkirju anda ning allkirju kontrollida.

Teenuse realisatsiooni lähtekood on AS-i Sertifitseerimiskeskus kontrolli all ning nende poolt uuendatav. Aruande autorid ei ole seda lähteteksti näinud. Teenuse spetsifikatsioonist [32] selgub aga, mida selle teenuse kasutajad peaksid silmas pidama, kui kasutatavad krüptoalgoritmid muutuvad.

Mobiil-ID-ga autentimisel signeerib kasutaja telefon 160-bitise juhusliku sõnumi, millest poole on genereerinud autentimist nõudev server ning teise poole DigiDocService veebiteenus. Signatuuri pikkus võib muutuda, kui pikemad võtmed kasutusele võetakse. Juhusliku sõnumi pikkus (160 bitti) on selles rakenduses piisav.

Allkirjastamisel tuleb veebiteenusele saata päringuga kas allkirjastatavad failid või nende räsid. Failide räsimisel on ainus toetatud räsialgoritm SHA-1. DigiDoc-konteineri räsimise algoritm ei ole DigiDocService'i spetsifikatsioonis täpsustatud. Samas tekitab see veebiteenus DigiDoc-formaadile vastavalt dokumente, mis lubab sellel räsialgoritmil olla ainult SHA-1.

DigiDocService-veebiteenuse kohandamise enamate krüptoalgoritmidega peab ära tegema selle teenuse haldaja. Selle aruande kirjutajail ei ole ligipääsu DigiDocService'i lähtekoodile, seega ei ole võimalik käesolevaga hinnata, kui keeruline see töö on. Kuna tegu on veebiteenusega, piisab muutuste tegemisest ainult ühes kohas.

Kui kasutada DigiDocService-veebiteenust dokumentide allkirjastamiseks ID-kaardiga, siis laaditakse alla Java-plugin, mis saab DigiDoc-konteineri räsi portaalist. Plugin eeldab, et räsi pikkuseks on 20 baiti. Kiipkaardil laseb plugin kasutada PKCS #1 polsterdust.

Mis puutub uute (räsi)algoritmide kasutuselevõtu keerukust DigiDocService'is, tuleb arvestada järgmist. DigiDocService'i spetsifikatsioon ei täpsusta, millist räsialgoritmi kasutab sisemiselt tema realisatsioon, kui enne ID-kaardi või Mobiil-ID-ga RSA astendamist on tarvis

leida DigiDoc-konteineri räsi. Seega võiks DigiDocService'i uuendamisel uute räsialgoritmide kasutuselevõtt üpris valutult käia.

4.1.4 Dokumentide digiallkirjastamine riigi- ja DigiDoc-portaalis ning pankades

DigiDoc-portaali ja riigiportaali funktsionaalsuse hulka kuulub suvaliste failide üleslaadimine ning digiallkirjastamine. Internetipangad ei paku suvaliste failide allkirjastamise võimalust, kuid üks osa nende töövoost on maksekorralduste allkirjastamine. Aruande koostajad on neid veebikeskkondi külastanud brauseriga FireFox operatsioonisüsteemidel Linux ja MacOS X.

Nimetatud veebikeskkondades dokumente allkirjastades laaditakse alla Java-plugin, mis teeb kasutaja arvutis üheainsa sammu DigiDoc-tüüpi allkirjastatud dokumendi loomiseks – võtab DigiDoc-konteineri sõnumilühendi, saadab selle kiipkaardile ning saadab tulemuse veebikeskkonnale tagasi.

Aruande autorid uurisid pluginaid, mille saadavad DigiDoc-portaal [7], riigiportaal eesti.ee [15], Swedbanki internetipank ning SEB-panga internetipank. Java-arhiivide kujul pluginad salvestati ning neid uuriti Java dekompilaatoriga [93]. Tuvastati, et kõik neli pluginat pärinevad ühest ja samast koodibaasist ning DigiDoc-portaali ja Swedbanki internetipanga omad on praegugi praktiliselt ühesugused. Nad kõik saavad parameetrina kaasa signeeritava sõnumilühendi (DigiDoc-konteineri räsi) ning esitavad selle kiipkaardile signeerimiseks. Signeerimismehhanismiks on RSA koos PKCS #1 polsterdusega. Eeldatakse, et räsi on moodustatud SHA-1 räsialgoritmiga – PKCS #1 polsterdus nõuab, et selle algoritmi identifikaator lisataks sõnumilühendi ette ning seda kõik neli pluginat ka teevad. DigiDoc-portaali ning Swedbanki internetipanga pluginad kontrollivad lisaks, et saadud sõnumilühendi pikkus oleks 20 baiti. Kiipkaardi tagastatud RSA-astendamise tulemus saadetakse tagasi veebikeskkonnale.

On näha, et need pluginad ei ole loodud arvestusega, et kunagi võiks olla tarvis vahetada räsialgoritmi. Samas ei ole algoritmi vahetamine ka suur töö – ära tuleb muuta algoritmiidentifikaator ning sõnumilühendi eeldatav pikkus. Ilmselt vajab plugin ka täiendavat argumenti, mis ütleb, millist algoritmi kasutada. Enamik DigiDoc'i loomise ja signeerimise loogikast on aga veebikeskkonna osa, mille lähtekoodile pole aruande koostajatel juurdepääsu ning mille kohandamise keerukust seetõttu hinnata ei saa.

Soovitame siinkohal kõigil digiallkirjastamist võimaldavate veebikeskkondade haldajatel teha koostööd, sest ei ole mõtet ühe ja sama asja tegemiseks mitut koodibaasi hallata ning neis krüptoalgoritme uuendada. Veelgi enam – soovitame, et veebilehele digiallkirjastamisfunktsionaalsuse lisamiseks vajalik programmikood (paaril enamlevinud platvormil) oleks edaspidi osa ID-kaardi baastarkvarast.

Märgime lõpuks veel paari probleemi, millest vähemalt esimene tuleb ära parandada.[6]

- Seisuga 20.06.2011 on kodanikuportaali plugin signeeritud sertifikaadiga, mis aegus 23. märtsil 2008.

- Kõigi nelja plugina signeerimiseks kasutatud sertifikaadid on välja antud kas Verisigni või Thawte'i sertifikaadiga, mille nimes sisaldub sõne *Code Signing*. Nende sertifikaatide võtmekasutuslaiendid on aga sellised, mis koodi signeerimist ei toeta.

- Kodanikuportaali ja Swedbanki internetipanga pluginad on signeerinud AS Sertifitseerimiskeskus.

[6] Autoreid on teavitatud, et muudatusi DigiDocService'it ja Mobiil-ID-d kasutavate serverite töös on juba tehtud, kuid kirjutamishetkel pole Siseministeerium vastavaid detaile veel avaldanud.

4.1.5 Digi-ID

Digitaalne isikutunnistus ehk Digi-ID on riiklik digitaalne dokument, millega saab elektroonilises keskkonnas oma isikut tuvastada ja anda digitaalallkirja [9]. Erinevalt ID-kaardist ei kanta Digi-ID-kaardile isiku nime, fotot ega muud vajalikku, et teda saaks kasutada visuaalse isikut tõendava dokumendina. Krüptograafia seisukohast on tegu samasuguse kiipkaardiga nagu ID-kaart ning Digi-ID sertifikaatide väljastamisel kasutab Sertifitseerimiskeskus samu ülpõhimõtteid [184], ESTEID-kaardi sertifitseerimispoliitikat [182] ning ESTEID sertifikaatide profiili [183]. Seetõttu käsitleb see aruanne Digi-ID ja ID-kaarti samaväärsena.

4.1.6 Digitaalne tempel

Digitaalne tempel [8] on AS-i Sertifitseerimiskeskus (SK) teenus, mille abil saavad ettevõtted anda digiallkirju dokumentidele, millega lisatakse dokumendile kinnitus, et dokument pärineb allkirjastanud asutusest ning et dokumenti ei ole vahepeal muudetud. Tüüpiliselt kasutatakse seda mass-produtseeritud dokumentide tarbeks, näiteks arved, maksekorraldused, kinnitused, tunnistused jne. Teenuse tellimisel väljastab SK ettevõttele kas kiipkaardil või USB-krüptopulgal X.509-sertifikaadi. Digitaalse templi kasutamisel luuakse andmete allkirjastamisel DigiDOC-vormingus konteiner täpselt samamoodi nagu ID-kaardi või Digi-ID kasutamisel andmete allkirjastamiseks.

Digitaalse templi sertifikaatide väljastamisel kasutab SK oma üldiseid sertifitseerimispõhimõtteid [184], asutuse sertifikaatide sertifitseerimispoliitikat [181] ja sertifikaatide profiili [185], mis määravad kokkuvõttes järgmised tingimused:

- sertifikaatide kehtivusaeg on 3 aastat ja 30 päeva,
- sertifikaatide signeerimisel kasutatakse algoritmi RSA-SHA-1,
- RSA võtmete miinimumpikkus on 1024 bitti.

See aruanne soovitab 5 aasta jooksul loobuda SHA-1 ning RSA-1024 algoritmidest. Kuna väljastatud sertifikaadid kehtivad veel 3 aastat, siis peaks Sertifitseerimiskeskus juba praegu alustama teenuse edasiarendamist ning plaanima üleminekut pikematele RSA võtmetele ning turvalisematele räsifunktsioonidele.

4.1.7 Soovitused tulevikuks

Meie peamine soovitus on siinuuritud tarkvarateekide ümber kirjutamine viisil, mis lubaks kasutatavaid krüptoalgoritme ning nende võtmepikkusi kiiresti vahetada, kasutades selleks täitmis- või kompileerimisaegseid konfiguratsioonifaile. Sel viisil peaks spetsifitseerima nii neid algoritme, mida teek uute krüptograafiliste objektide loomisel kasutab, kui ka neid algoritme, mida ta kontrollimisel aktsepteerib. Sel viisil on võimalik operatiivselt lisada uute algoritmide tuge ja samas eemaldada vanu, ebaturvaliseks tunnistatud algoritme. Eriti oluline on selline refaktoreerimine räsifunktsioonide toe juures, sest toetada tuleb kõigepealt SHA-2 perekonna funktsioone ning mõne aja pärast SHA-3 räsifunktsiooni. Lisaks algoritmidele ja võtmepikkustele soovitame, et ka RSA polsterdusi oleks hõlbus valida.

Koos ülalkirjeldatud täienduste tegemisega on mõistlik ja soovitatav ka lähtekoodi refaktoreerimine – signeerimise ja verifitseerimise erinevad juhud (ainult signatuuri, või ka sertifikaatide, või ka kehtivuskinnituste loomine/kontroll) sisaldavad sarnaseid koodilõike.

Krüpteerimisel tuleb plaanida üleminek plaanida RSA PKCS #1 v1.5 polsterduselt RSA OAEP polsterdusele [25].

Täiendavalt tuleb läbi mõelda, kust on võimalik saada entroopiat Windowsis transpordivõtmeid genereerides. OpenSSL-i meetod `RAND_screen` ei pruugi anda piisavalt head tulemust. Eelistatum viis on kasutada Windowsi CryptoAPI-t.

Transpordivõtmete genereerimisel pole mõtet kutsuda välja funktsiooni `EVP_BytesToKey`. OpenSSL-i juhuarvugeneraatorist saadud väärtused on otse võtmena kasutatavad, samuti ei pea sel moel tuginema ebaturvalisele räsifunktsioonile MD5.

4.2 ANDMEVORMINGUD

4.2.1 Signeeritud andmete vormingud

Eestis on põhiliselt kasutusel andmevormingud DigiDoc [180] ja BDOC [1].

4.2.1.1 Andmevorming DigiDOC

DigiDOC [180] on andmevorming, mida kasutavad DigiDOC-süsteemi kuuluvad rakendused ning mis põhineb rahvusvahelistel standarditel XML-DSIG [95] ja ETSI TS 101903 [21] ning on alamhulk nimetatud standarditest. See andmevorming on olnud kasutusel juba aastast 2002 ning on praeguseks jõudnud versioonini 1.3. DigiDOC-andmevorminguga on võimalik esitada signeeritud andmeid ning nende andmetega seotud allkirju. Koos allkirjadega esitatakse ka nende aluseks olnud X.509 sertifikaatide kehtivust kinnitavad OCSP-kehtivuskinnitused. Praegune andmevorming määrab järgmised tingimused:

- Andmefailide räsimiseks tuleb kasutada SHA-1 algoritmi.
- Räside signeerimisel tuleb kasutada RSA PKCS #1 versioonile 1.5 [22] vastavat RSASSA-PKCS1-v1_5 polsterdusviisi.
- Signeerija sertifikaadi kehtivuskinnituse saamiseks saadetakse OCSP-päringus `nonce` väärtusena RSA allkirjast võetud räsiväärtus. Räsimiseks kasutatakse SHA-1 algoritmi.
- `CertDigest` elemendis olev räsi viitab andmefaili signeerimisel või näiteks OCSP-kehtivuskinnituse signeerimisel kasutatud sertifikaadile ning seal kasutatakse samuti SHA-1 algoritmi.

Andmevormingus pole ette nähtud valikuvõimalusi ning kasutusel olevat RSA-SHA-1 algoritmi ei saa rakendused praegu muuta. Standard XMLDSIG näeb küll ette ka DSA-SHA-1 [10] signeerimisalgoritmi kasutamisvõimalust, kuid DigiDOC-andmeformaati ei ole seda võimalust üle toodud.

4.2.1.2 DigiDOC-i muutmise keerukus

Selleks et muuta DigiDOC andmevormingus praegu spetsifitseeritud krüptoalgoritme, tuleb teha järgmist:

- Uued krüptoprimitiivid, näiteks SHA-3 ja seda kasutavad RSA PKCS #1 või DSA skeemid peavad olema realiseeritud mõnes üldlevinud krüptoteegis, näiteks OpenSSL või Bouncy Castle teekides.
- OCSP kehtivuskinnitusteenus peab olema uuendatud, et päringutes saaks kasutada uusi primitiive.
- Tuleb luua ning publitseerida uus DigiDOC-andmevormingu kirjeldus, kasutades uute primitiivide kokkulepitud URI identifikaatoreid (sarnaselt RFC4051 standardis spetsifitseeritud SHA-512 jms algoritmide identifitkaatoritele).

- Tuleb luua ning testida rakendustarkvara, mis kasutaks uuet andmevormingu versiooni. Uuendada tuleb vähemalt DigiDOC-klient, jdigidoc-teek ning DigiDOC-veebiteenus.

4.2.1.3 Andmevorming BDOC

BDOC on andmevorming, mis on loodud 2008. aastal `wpki.eu` foorumi töö käigus selleks, et asendada Eesti-spetsiifiline DigiDOC ning Läti-spetsiifiline eDoc andmevorming ühise vorminguga, mis võimaldaks esitada signeeritud andmeid ning nende andmetega seotud allkirju. BDOC-andmevormingu kirjeldus on standardiseeritud 2009. aastal Eesti ametliku standardina [1].

BDOC-vormingus tuleb kasutada samu krüptoalgoritme nagu DigiDOC-vormingus, SHA-1 räsialgoritmi ning RSASSA-PKCS1-v1_5 polsterdusviisi. Samas sisaldab standard märkust, et on tungivalt soovitatav kasutada uuemaid räsialgoritme. Kuna standard jääb kahjuks üldsõnaliseks, millist konkreetset algoritmi tuleks tegelikult kasutada, milliseid algoritme peavad kindlasti toetama signeerimis- ja valideerimisrakendused, siis tuleb nentida, et BDOC andmevorming ei paku krüptograafilises mõttes paremaid garantiisid kui DigiDOC-vorming ning praktikas tuleb uute krüptoprimitiivide kasutuselevõtmisel arvestada samalaadsete toimingutega (vaata 4.2.1.2), nagu ka DigiDOC-vormingu korral.

4.2.2 Krüptitud andmete vormingud

CDOC-andmevorming ei ole spetsifitseeritud omaette standardina, vaid kasutatakse tavapärast XMLENC [106] standardit. XMLENC on üsna lai ning lubab kasutada mitmeid krüptoalgoritme. Kohustuslikud algoritmid on toodud paksus kirjas:

- **3DES-CBC**, **AES-128-CBC**, AES-192-CBC, **AES-256-CBC**,
- **RSAES-PKCS1-v1_5**, **RSA-OAEP**,
- **SHA-1**, **SHA-256**, SHA-512, RIPEMD-160.

Praktikas kasutab DigiDOCi klienditarkvara [186] vaid alamosa nendest algoritmidest ning ei realiseeri kõiki kohustuslikke algoritme. Lähtekoodi on sisse kirjutatud AES-128-CBC, RSAES-PKCS1-v1_5 ning SHA-1 algoritmide kasutus ning kasutajal ei ole võimalik neid muuta.

4.2.3 Suurte andmehulkade krüptimine. TrueCrypt

Suuri andmehulki pole reeglina mõistlik krüptida faili või andmebaasikirje kaupa. Tüüplahenduseks on sel juhul terve failisüsteemi krüptimine, kusjuures failisüsteem võib asuda eraldi virtuaalsel kettal (mis moodustab välise süsteemi poolt vaadates ühe suure faili) või hõlmata kogu füüsilise ketta. Ketta krüptimiseks on olemas palju erinevaid äri- ja vabavaralahendusi, millest ammendava ülevaate andmine jääb selle aruande raamidest kaugele välja. Seetõttu keskendume näiterakendusena ühele konkreetsele tarkvarale TrueCrypt [19], mis on hea kompromiss avatuse, kasutusmugavuse ja võimalusterohkuse vahel ning on seetõttu Eestis võrdlemisi laia rakendust leidnud.

Failisüsteemi krüptimiseks kasutab TrueCrypt algoritme AES-256, Serpent [61] ja TwoFish [75] ning nende erinevaid kombinatsioone XTS-režiimis. Paroolist krüptovõtme tuletamiseks saab kasutada räsifunktsioone RIPEMD-160, SHA-512 ja Whirlpool [202]. Kõiki neid algoritme võib käesoleva aruande ajahortisondi ulatuses turvaliseks pidada.

TrueCryptis moodustataval konteinerfailil pole eristatavat päist ning ainus võimalus kahtlustada, et tegu on TrueCrypti konteineriga, on tuvastada, et tegu on failiga, mille pikkus on

512 biti kordne ning mis läbib statistilised juhuslikkustestid. Võimalikud on ka ründed, kus kasutatav parool/võti tuvastatakse mälu skaneerides või klaviatuuri pealt kuulates. Samas on sellised ründed ohtlikud praktiliselt kõigi kettakrüptolahenduste puhul, seega ei saa neid võtta argumentidena TrueCrypti kasutamise vastu.

TrueCrypti lähetkood on vabalt saadaval ning seega põhimõtteliselt täielikult inspekteeritav. Samas pole ükski sõltumatu osapool selle aruande koostajatele teadaolevalt seda inspektsiooni läbi viinud. Kuna TrueCrypti arendajad eelistavad jääda anonüümseiks ning nende kasutatav litsents pole OSI poolt heaks kiidetud, ei saa TrueCrypti enne täielikku koodiauditit soovitada kõrget konfidentsiaalsustaset nõudvate dokumentide krüptimiseks. Samas võib TrueCrypti pidada piisavalt turvaliseks asutusesiseseks kasutamiseks määratud dokumentide kaitsmisel.

Kui tekib vajadus muuta krüptimiseks kasutatavat plokkšifrit, tuleb luua uus konteiner ja andmed ümber tõsta. Parooli või võtme tuletamiseks kasutatava räsifunktsiooni vahetamine on võimalik ka ilma uut konteinerit loomata; seda funktsionaalsust pakub ka TrueCrypti kasutajaliides.

4.3 JUHTUMIANALÜÜS: X-TEE RÄSIFUNKTSIOONIDE VAHETAMINE

Professor Wangi töörühm avaldas 2005. aastal oma artikli räsifunktsioonide ründamisest, kus MD5 kollisioon oli eksplitsiitselt välja arvutatud ning samuti oli selge, et sarnase konstruktsiooni tõttu ei paku ka SHA-1 enam maksimaalset algses tehnilises lahenduses kavandatud turvataset [204]. Kuna mõlemad räsifunktsioonid on erinevates infosüsteemides väga laialdaselt kasutusel, sattus nende süsteemide turvalisus automaatselt küsimärgi alla. Soovides maandada räsifunktsioonide nõrkustest tulenevaid riske Eesti ühes olulisemas e-riigi infrastruktuuri komponendis X-tee, planeeriti X-tee 5. versiooni arendustööde hulka nõrkade räsifunktsioonide väljavahetamine.

Planeeritud tööde sisu oli järgmine.

1. Identifitseerida kõik X-tee komponendid, milles kasutatakse räsifunktsioone. Analüüsida, millistes koodimeetodites tuleb MD5 ja SHA-1 esinemised välja vahetada ning millistes võivad need alles jääda (nt kohad, kus neid kasutatakse kontrollsummade vm vähese turvatundlikkusega eesmärkide jaoks).
2. Valida uus räsifunktsioon.
3. Viia X-tee serveritarkvarasse sisse parandused ning asendada kõik ettenähtud nõrga räsifunktsiooni kasutamise kohad tugevamaga.
4. Koostada uut räsifunktsiooni kasutava tarkvara sujuva evituse plaan, võttes arvesse probleeme, mis tekivad kõikide sertifikaatide asendamisel.
5. Paigaldada uus tarkvara ja võtta see kasutusse; anda välja uued sertifikaadid kõigile kasutajatele.
6. Dokumenteerida kogu protsess, võttes arvesse, et seda tuleb korrata, kuna ühegi teadaoleva räsifunktsiooni eluiga ei ole praktikas võimalik ennustada pikemaks kui 10 aastat.

Kuna SHA-3 konkurss polnud arendustööde tegemise ajaks (2010. aastal) veel lõppenud, oli ainus praktiline alternatiiv mõni SHA-2 perekonna liige. Kuna kettaruumi probleemi ei olnud, otsustati kasutusele võtta räsifunktsioon SHA-512.

Algselt planeeriti tööde kogumahuks 200 inimtundi, lõpuks kulus kokku 491 inimtundi ehk umbes 3 inimkuud. Sellesse hinnangusse mahuvad ainult reaalsed programmeerimistööd, aga mitte erinevad koosolekud, nõupidamised jms. 95% töödest teostati ühe inimese poolt ning ka tööde füüsiline kogukestus oli orienteerivalt 3 kuud. Töömahu jaotus kujunes lõpuks järgmiseks.

- Räsifunktsioonide üleminekuplaani koostamine: 76 inimtundi
- Spetsifikatsiooni koostamine: 31 inimtundi
- Serverite/koodi muutmine ja parandamine: 42 inimtundi
- Verifitseerimisprogrammi muutmine: 41 inimtundi
- Vanade logide ümberräsimise programm: 188 inimtundi (Ajahinnang sisaldab ka uue inimese sisseelamisaega projekti 90h, seega reaalset programmeerimist kogunes 98h. Samas ei saa seda 90h ajakulu sarnastes projektides ka kokku hoida, sest suure tõenäosusega hakkab krüptoalgoritmide vahetust tegema inimene, kes konkreetse infosüsteemi nüanssidega eelnevalt kursis ei ole.)
- Vanade logide ümberräsimise programmi kasutusjuhend: 13 inimtundi
- Testiplaani kirjutamine: 26 inimtundi
- Keskserverisse koondräsi logimise funktsionaalsus: 24 inimtundi
- SKIP/ESP ülekolimine SHA-512-le: 18 inimtundi
- Vanade päringulogide üleräsimise dokumenteerimine: 8 inimtundi
- Tugi vana räsifunktsiooni kasutamiseks tagasiühilduvuse huvides: 8 inimtundi
- Muudatused automaattestides: 16 inimtundi.

Lisaks tuleb mainida, et X-tee räsifunktsioonide vahetus kujunes sisuliselt koodiauditiks, sest palju lähtekoodi tuli rida-realt läbi lugeda, et aru saada, kuidas on seal mingit räsifunktsiooni kasutatud, ning läbilugemise käigus tuli välja palju muid vigu, mis vajasid parandamist ja lisasid seega kaudselt töötunde. Sarnase töö ettevõtmisel tuleb niisuguse täiendava ajakuluga kindlasti arvestada.

Tööde käigus muudeti umbes 50 faili. Puhtalt räsifunktsioonide vahetamiseks muudeti 4460 koodirida (peamiselt C ja C++), koodiauditi käigus leitud muude vigade parandamiseks muudeti 2150 koodirida. Täiesti uue lähtekoodi hulk jäi marginaalseks. Kokku vahetati SHA-1 või MD5 väljakutse SHA-512 väljakutseks 50 kohas, 40 kohas näitas analüüs, et SHA-1 väljavahetamine pole vajalik. OpenSSL-i käsureautiliidis tehti SHA-1→SHA-512 muudatusi 3 ja muid SHA-512 funktsioonide väljakutseid on koodis 30. Umbes 10 funktsioonile tuli lisada tugi räsifunktsiooni parameetriga etteandmiseks ning 40 kohas tuli kasutajale väljastavates teadetes täpsustada räsifunktsiooni tüüp. MD5 väljakutsed jäid alles SSL-seansi failinmede genereerimise protseduuri ning import-eksportfailide ja varukoopiate kontrollsummade arvutamisse. Nõrga räsifunktsiooni kasutamine kontrollsummade arvutamisel võimaldab võidelda ainult juhuslike (näiteks salvestusmeedia füüsiliste) vigade vastu, kuid paljudes stsenaariumites sellest piisab ning piiratud eelarvet arvestades otsustati räsifunktsioonide vahetamine selles kontekstis projekti skoobist välja jätta.

KIRJANDUS

[1] BDOC. Digitaalallkirja vorming. EVS 821:2009, http://www.evs.ee/product/tabid/59/p-165934-evs-8212009.aspx.

[2] Bouncy Castle cryptographic toolkit. http://www.bouncycastle.org/.

[3] Bouncy Castle Specifications. http://bouncycastle.org/specifications.html.

[4] Computer Data Authentication. FIPS PUB 113.

[5] Cyptography Research and Evaluation Committees. http://www.cryptrec.go.jp/english/index.html.

[6] Data Encryption Standard (DES). FIPS PUB 46-3. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[7] DigiDoc-portaal. https://digidoc.sk.ee.

[8] Digitaalne tempel. http://www.sk.ee/teenused/digitempli-teenus/.

[9] Digitaalse isikutunnistuse vormi, tehnilise kirjelduse ja digitaalsele isikutunnistusele kantavate andmete loetelu kehtestamine. https://www.riigiteataja.ee/akt/13353744.

[10] Digital Signature Standard (DSS). FIPS PUB 186-3. http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf.

[11] IEEE 802.11: Wireless Local Area Networks (LANs). http://standards.ieee.org/about/get/802/802.11.html.

[12] NSA Suite B Cryptography. http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.

[13] OpenSSL SSL/TLS toolkit. http://www.openssl.org.

[14] Recommendation for Block Cipher Modes of Operation. NIST Special Publication 800-38A, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

[15] Riigiportaal eesti.ee. https://www.eesti.ee.

[16] Riigisaladuse ja salastatud välisteabe kaitse kord. Vabariigi Valitsuse määrus nr 262, RT I 2007, 73, 449.

[17] Secure Hash Standard (SHS) . FIPS PUB 180-3. http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

[18] Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification. http://www.3gpp.org/ftp/specs/html-info/35202.htm.

[19] TrueCrypt website. http://www.truecrypt.org/.

[20] TUPAS Identification Service for Service Providers, versioon 2.3c. http://www.fkl.fi/en/themes/e-services/tupas/Pages/default.aspx.

[21] XML Advanced Electronic Signatures (XAdES). ETSI TS 101903, http://uri.etsi.org/01903/v1.4.1/.

[22] PKCS #1: RSA Encryption Standard, versioon 1.5, 1. november 1993. http://www.rsa.com/rsalabs/node.asp?id=2125.

[23] PKCS #12 - Personal Information Exchange Syntax Standard, versioon 1.0, 24. juuni 1999. http://www.rsa.com/rsalabs/node.asp?id=2138.

[24] Security Requirements for Cryptorgaphic Modules, 2001. http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

[25] PKCS #1: RSA Encryption Standard, versioon 2.1, 14. juuni 2002. http://www.rsa.com/rsalabs/node.asp?id=2125.

[26] CCSDS Next Generation Space Internet (NGSI) — End-to-End Security for Space Mission Communications. NASA Consultative Committee for Space Data Systems, aprill 2003. http://public.ccsds.org/publications/archive/733x5o1.pdf.

[27] CRYPTREC Report 2002. Technical report, Information-technology Promotion Agency, Japan, 2003. http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/c02e_report2.pdf.

[28] PKCS #11 - Cryptographic Token Interface Standard, versioon 2.20, 28. juuni 2004. http://www.rsa.com/rsalabs/node.asp?id=2133.

[29] DigiDoc C-teegi kirjeldus koos kasutusjuhistega, versioon 2.2.5, 27. märts 2006.

[30] PKCS #5: Password-Based Cryptography Standard, versioon 2.1, 5. oktoober 2006. http://www.rsa.com/rsalabs/node.asp?id=2127.

[31] IEEE Standard for Information Technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2007. http://standards.ieee.org/getieee802/download/802.11-2007.pdf.

[32] DigiDocService spetsifikatsioon, versioon 2.123, 1. märts 2009. http://www.id.ee/public/DigiDocService_spec_est.pdf.

[33] Infosüsteemide kolmeastmelise etalonturbe süsteem ISKE. Kataloogid. Versioon 5.01, 2010. http://www.ria.ee/public/ISKE/iske_kataloogid_5_01.pdf.

[34] Mathematical routines for the NIST prime elliptic curves, 2010. http://www.nsa.gov/ia/_files/nist-routines.pdf.

[35] PC/SC Workgroup Specifications, versioon 2.01.9, Aprill 2010. http://www.pcscworkgroup.com/specifications/overview.php.

[36] Suite B Implementer's Guide to FIPS 186-3 (ECDSA), 2010. http://www.nsa.gov/ia/_files/ecdsa.pdf.

[37] Digiallkirjastamine veebis. http://www.id.ee/10824, 20. aprill 2011.

[38] ID-kaardi baastarkvara. http://id.eesti.ee/, mai 2011.

[39] OpenSC - tools and libraries for smart cards, 2011. http://www.opensc-project.org/.

[40] Martín Abadi. Secrecy by Typing in Security Protocols. In Martín Abadi and Takayasu Ito, editors, *TACS*, volume 1281 of *Lecture Notes in Computer Science*, pages 611–638. Springer, 1997.

[41] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. *ACM Transactions on Information and System Security*, 10(3), 2007.

[42] Martín Abadi and Roger M. Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[43] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[44] William Aiello, John Ioannidis, and Patrick Drew McDaniel. Origin authentication in interdomain routing. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 165–178. ACM, 2003.

[45] Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, 2005.

[46] Ashar Aziz, Martin Patterson, and Geoff Baehr. Simple Key-Management for Internet Protocol (SKIP). In *Internet Society's 1995 International Networking Conference (INET '95)*, 1995. http://wayback.archive.org/web/*/http://skip.incog.com/inet-95.ps.

[47] Steve Babbage, Dario Catalano, Carlos Cid, Benne de Weger, Orr Dunkelman, Christian Gehrmann, Louis Granboulan, Tim Güneysu, Tanja Lange, Arjen Lenstra, Chris Mitchell, Mats Näslund, Phong Nguyen, Christof Paar, Kenny Paterson, Jan Pelzl, Thomas Pornin, Bart Preneel, Christian Rechberger, Vincent Rijmen, Matt Robshaw, Andy Rupp, Martin Schläffer, Serge Vaudenay, Fré Vercauteren, and Michael Ward and. ECRYPT II Yearly Report on Algorithms and Keysizes (2009-2010). Technical report, European Network of Excellence in Cryptology II, March 2010. http://www.ecrypt.eu.org/documents/D.SPA.13.pdf.

[48] Michael Backes, Dennis Hofheinz, and Dominique Unruh. CoSP: a general framework for computational soundness proofs. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 66–78. ACM, 2009.

[49] Michael Backes and Dominique Unruh. Computational soundness of symbolic zero-knowledge proofs. *Journal of Computer Security*, 18(6):1077–1155, 2010.

[50] Elad Barkan, Eli Biham, and Nathan Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 600–616. Springer Berlin / Heidelberg, 2003.

[51] Elaine Barker, William Burr, Alicia Jones, Timothy Polk, Scott Rose, Miles Smid, and Quynh Dang. Recommendation for Key Management. Part 3: Application-Specific Key Management Guidance. Technical report, NIST, December 2009. http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_PART3_key-management_Dec2009.pdf.

[52] Elaine Barker, Don Johnson, and Miles Smid. Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. Technical report, NIST, 2007. NIST Special Publication 800-56A, http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf.

[53] William C. Barker. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. Technical report, NIST, 2008. http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf.

[54] David A. Basin, Sebastian Mödersheim, and Luca Viganò. An On-the-Fly Model-Checker for Security Protocol Analysis. In Einar Snekkenes and Dieter Gollmann, editors, *ESORICS*, volume 2808 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2003.

[55] Mihir Bellare. New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619. Springer Berlin / Heidelberg, 2006.

[56] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[57] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer Berlin / Heidelberg, 2000.

[58] Mihir Bellare and Phillip Rogaway. Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer Berlin / Heidelberg, 2000.

[59] Daniel J. Bernstein. Cache-timing attacks on AES, 2005. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[60] Karthikeyan Bhargavan, Cédric Fournet, Andrew D. Gordon, and Nikhil Swamy. Verified implementations of the information card federated identity-management protocol. In Masayuki Abe and Virgil D. Gligor, editors, *ASIACCS*, pages 123–135. ACM, 2008.

[61] Eli Biham, Ross Anderson, and Lars Knudsen. Serpent: A New Block Cipher Proposal. In Serge Vaudenay, editor, *Fast Software Encryption*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer Berlin / Heidelberg, 1998.

[62] Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg, 2009.

[63] Alex Biryukov, Adi Shamir, and David Wagner. Real Time Cryptanalysis of A5/1 on a PC. In Bruce Schneier, editor, *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 37–44. Springer Berlin / Heidelberg, 2001.

[64] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW*, pages 82–96. IEEE Computer Society, 2001.

[65] Bruno Blanchet. Automatic verification of cryptographic protocols: a logic programming approach. In *PPDP*, pages 1–3. ACM, 2003.

[66] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 1998.

[67] Chiara Bodei, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Flow logic for Dolev-Yao secrecy in cryptographic processes. *Future Generation Computer Systems*, 18(6):747–756, 2002.

[68] Alexandra Boldyreva and Virendra Kumar. Extended Abstract: Provable-Security Analysis of Authenticated Encryption in Kerberos. In *IEEE Symposium on Security and Privacy*, pages 92–100. IEEE Computer Society, 2007.

[69] Dan Boneh and Glenn Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE Transactions on Information Theory*, 46(4):1339–1349, 2000.

[70] Matteo Bortolozzo, Matteo Centenaro, Riccardo Focardi, and Graham Steel. Attacking and fixing PKCS#11 security tokens. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 260–269. ACM, 2010.

[71] Daniel R. L. Brown. Generic Groups, Collision Resistance, and ECDSA. *Designs, Codes and Cryptography*, 35:119–152, 2005.

[72] Michael Burrows, Martín Abadi, and Roger M. Needham. A Logic of Authentication. In *SOSP*, pages 1–13. ACM, 1989.

[73] Frederick Butler, Iliano Cervesato, Aaron D. Jaggard, Andre Scedrov, and Christopher Walstad. Formal analysis of Kerberos 5. *Theoretical Computer Science*, 367(1-2):57–87, 2006.

[74] Ran Canetti and Hugo Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 27–52. Springer Berlin / Heidelberg, 2002.

[75] Christophe Cannière. Twofish. In Henk van Tilborg, editor, *Encyclopedia of Cryptography and Security*, pages 638–639. Springer US, 2005.

[76] Carlos Cid, Henri Gilbert, Daniel Augot, Alex Biryukov, Anne Canteaut, Nicolas Courtois, Christophe De Cannière, Cédric Lauradoux, Matthew Parker, Bart Preneel, Matt Robshaw, and Yannick Seurin. AES Security Report. Technical Report D.STVL.2, European Network of Excellence in Cryptology, 2004. http://www.ecrypt.eu.org/ecrypt1/documents/D.STVL.2-1.0.pdf.

[77] Carlos Cid and Gaëtan Leurent. An Analysis of the XSL Algorithm. In Bimal Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer Berlin / Heidelberg, 2005.

[78] Jolyon Clulow. On the Security of PKCS #11. In Colin Walter, Çetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 411–425. Springer Berlin / Heidelberg, 2003.

[79] Martin Cochran. Notes on the Wang et al. 2^{63} SHA-1 Differential Path. Cryptology ePrint Archive, Report 2007/474, 2007. http://eprint.iacr.org/.

[80] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM Conference on Computer and Communications Security*, pages 109–118. ACM, 2008.

[81] Véronique Cortier and Bogdan Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In Mooly Sagiv, editor, *Programming Languages and Systems*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171. Springer Berlin / Heidelberg, 2005.

[82] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer Berlin / Heidelberg, 2002.

[83] Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. Protocol Composition Logic (PCL). *Electronic Notes in Theoretica Computer Science*, 172:311–358, 2007.

[84] Magnus Daum and Stefan Lucks. The Story of Alice and her Boss: Hash Functions and the Blind Passenger Attack, May 2005. Rump session presentation at Eurocrypt 2005.

[85] Stéphanie Delaune, Steve Kremer, and Graham Steel. Formal Analysis of PKCS#11. In *21st IEEE Computer Security Foundations Symposium*, pages 331–344. IEEE Computer Society, 2008.

[86] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol, Version 1.2. IETF RFC5246, http://tools.ietf.org/html/rfc5246.

[87] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[88] Hans Dobbertin. Cryptanalysis of MD5 Compress, May 1996. Rump session presentation at Eurocrypt 1996.

[89] Hans Dobbertin. RIPEMD with Two-Round Compress Function is Not Collision-Free. *Journal of Cryptology*, 10(1):51–70, 1997.

[90] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. In Dieter Gollmann, editor, *FSE*, volume 1039 of *LNCS*, pages 71–82. Springer, 1996.

[91] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[92] Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 393–410. Springer Berlin / Heidelberg, 2010.

[93] Emmanuel Dupuy. Java Decompiler — Yet another fast Java decompiler. `http://java.decompiler.free.fr`.

[94] E.A.Grechnikov. Collisions for 72-step and 73-step SHA-1: Improvements in the Method of Characteristics. Cryptology ePrint Archive, Report 2010/413, 2010. `http://eprint.iacr.org/`.

[95] D. Eastlake, J. Reagle, and D. Solo. XML-Signature Syntax and Processing. IETF RFC3275, `http://tools.ietf.org/html/rfc3275`.

[96] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In George Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin / Heidelberg, 1985.

[97] Ralf Engelschall. *User Manual, mod_ssl versioon 2.8*, 2001. `http://www.modssl.org/docs/2.8/`.

[98] Scott Fluhrer and David McGrew. Statistical Analysis of the Alleged RC4 Keystream Generator. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 66–71. Springer Berlin / Heidelberg, 2001.

[99] Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. Universally Composable Security Analysis of TLS. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 313–327. Springer Berlin / Heidelberg, 2008.

[100] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[101] Andrew D. Gordon and Alan Jeffrey. Authenticity by Typing for Security Protocols. *Journal of Computer Security*, 11(4):451–520, 2003.

[102] Andrew D. Gordon and Alan Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3-4):435–483, 2004.

[103] Aleksei Gorny. Analysis of Chip-card Based Authentication. Tartu Ülikool, bakalaureusetöö, 2009.

[104] Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C. Mitchell. A modular correctness proof of IEEE 802.11i and TLS. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 2–15. ACM, 2005.

[105] Wilko Henecka, Alexander May, and Alexander Meurer. Correcting Errors in RSA Private Keys. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 351–369. Springer Berlin / Heidelberg, 2010.

[106] Takeshi Imamura, Blair Dillaway, and Ed Simon. XML Encryption Syntax and Processing, 10. detsember 2002.

[107] Sebastiaan Indesteege, Florian Mendel, Bart Preneel, and Christian Rechberger. Collisions and Other Non-random Properties for Step-Reduced SHA-256. In Roberto Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 276–293. Springer Berlin / Heidelberg, 2009.

[108] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In Thomas Johansson, editor, *Fast Software Encryption*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer Berlin / Heidelberg, 2003.

[109] Romain Janvier, Yassine Lakhnech, and Laurent Mazaré. Completing the Picture: Soundness of Formal Encryption in the Presence of Active Adversaries. In Mooly Sagiv, editor, *Programming Languages and Systems*, volume 3444 of *Lecture Notes in Computer Science*, pages 172–185. Springer Berlin / Heidelberg, 2005.

[110] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC3447, Feb 2003. `http://tools.ietf.org/html/rfc3447`.

[111] Jakob Jonsson. On the Security of CTR + CBC-MAC. In Kaisa Nyberg and Howard Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 76–93. Springer Berlin / Heidelberg, 2003.

[112] Raul Kaidro. ID-kaardi baastarkvara kompileerimis- ja pakendusjuhend. `http://id.eesti.ee`, 19. aprill 2011.

[113] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). IETF RFC 5996, `http://tools.ietf.org/html/rfc5996`.

[114] S. Kent. IP Encapsulating Security Payload (ESP). IETF RFC4303, `http://tools.ietf.org/html/rfc4303`.

[115] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:161–191, 1883.

[116] Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1 (Extended Abstract). In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin / Heidelberg, 2006.

[117] Andreas Klein. Attacks on the RC4 stream cipher. In *Designs, Codes and Cryptography*, volume 48, pages 269–286. Springer Netherlands, 2008.

[118] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thomé, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-Bit RSA

Modulus. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 333–350. Springer Berlin / Heidelberg, 2010.

[119] Vlastimil Klíma, Ondrej Pokorný, and Tomáš Rosa. Attacking RSA-Based Sessions in SSL/TLS. In Colin Walter, Çetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 426–440. Springer Berlin / Heidelberg, 2003.

[120] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[121] Neal Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1:139–150, 1989.

[122] Neal Koblitz and Alfred Menezes. Another Look at Generic Groups. *Advances in Mathematics of Communications*, 1(1):13–28, 2007.

[123] Neal Koblitz, Alfred Menezes, and Scott Vanstone. The State of Elliptic Curve Cryptography. *Designs, Codes and Cryptography*, 19:173–193, 2000.

[124] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer Berlin / Heidelberg, 1999.

[125] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1:5–27, 2011.

[126] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[127] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. IETF RFC2104, `http://tools.ietf.org/html/rfc2104`.

[128] Hugo Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer Berlin / Heidelberg, 2001.

[129] Peeter Laud. Encryption Cycles and Two Views of Cryptography. In *NORDSEC 2002 - Proceedings of the 7th Nordic Workshop on Secure IT Systems*, pages 85–100. Karlstad University, 2002.

[130] Peeter Laud and Meelis Roos. Formal Analysis of the Estonian Mobile-ID Protocol. In Audun Jøsang, Torleiv Maseng, and Svein J. Knapskog, editors, *NordSec*, volume 5838 of *Lecture Notes in Computer Science*, pages 271–286. Springer, 2009.

[131] Märt Laur. *X-tee 5.0 turvaserveri kasutusjuhend. versioon 5.04*, 14. märts 2011. `http://ee.x-rd.net/docs/est/turvaserveri_kasutusjuhend.pdf`.

[132] L. Law and J. Solinas. Suite B Cryptographic Suites for IPsec. IETF RFC 4869.

[133] Chu-Wee Lim and Khoongming Khoo. An Analysis of XSL Applied to BES. In Alex Biryukov, editor, *Fast Software Encryption*, volume 4593 of *Lecture Notes in Computer Science*, pages 242–253. Springer Berlin / Heidelberg, 2007.

[134] Gavin Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In Tiziana Margaria and Bernhard Steffen, editors, *TACAS*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.

[135] Gavin Lowe. Casper: A Compiler for the Analysis of Security Protocols. In *CSFW*, pages 18–30. IEEE Computer Society, 1997.

[136] Subhamoy Maitra and Goutam Paul. New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4. In Kaisa Nyberg, editor, *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 253–269. Springer Berlin / Heidelberg, 2008.

[137] Itsik Mantin and Adi Shamir. A Practical Attack on Broadcast RC4. In Mitsuru Matsui, editor, *Fast Software Encryption*, volume 2355 of *Lecture Notes in Computer Science*, pages 87–104. Springer Berlin / Heidelberg, 2002.

[138] Alexander May and Maike Ritzenhofen. Implicit Factoring: On Polynomial Time Factoring Given Only an Implicit Hint. In *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2009.

[139] Catherine Meadows. The NRL Protocol Analysis Tool: A Position Paper. In *CSFW*, page 227, 1991.

[140] Catherine Meadows. Analysis of the Internet Key Exchange Protocol using the NRL Protocol Analyzer. In *IEEE Symposium on Security and Privacy*, pages 216–231, 1999.

[141] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. On the Collision Resistance of RIPEMD-160. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC*, volume 4176 of *LNCS*, pages 101–116. Springer, 2006.

[142] A.J. Menezes, T. Okamoto, and S.A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[143] Alfred Menezes, Paul van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[144] Ulrike Meyer and Susanne Wetzel. A man-in-the-middle attack on UMTS. In *Proceedings of the 3rd ACM workshop on Wireless security*, WiSe '04, pages 90–97. ACM, 2004.

[145] Daniele Micciancio and Saurabh Panjwani. Adaptive Security of Symbolic Encryption. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2005.

[146] Daniele Micciancio and Bogdan Warinschi. Soundness of Formal Encryption in the Presence of Active Adversaries. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.

[147] Victor Miller. Use of Elliptic Curves in Cryptography. In Hugh Williams, editor, *Advances in Cryptology – CRYPTO '85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin / Heidelberg, 1986.

[148] John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated analysis of cryptographic protocols using Mur-phi. In *IEEE Symposium on Security and Privacy*, pages 141–151. IEEE Computer Society, 1997.

[149] Gordon E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 8:114–117, 1965.

[150] Gordon E. Moore. Progress in digital integrated electronics. In *Electron Devices Meeting, 1975 International*, pages 11–13, 1975.

[151] Sean Murphy and Matthew Robshaw. Essential Algebraic Structure within the AES. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin / Heidelberg, 2002.

[152] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure; Online Certificate Status Protocol — OCSP, juuni 1999.

[153] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). IETF RFC4120, `http://tools.ietf.org/html/rfc4120`.

[154] Hanne Riis Nielson and Flemming Nielson. A flow-sensitive analysis of privacy properties. In *CSF*, pages 249–264. IEEE Computer Society, 2007.

[155] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[156] NIST. Cryptographic Hash Project. `http://csrc.nist.gov/groups/ST/hash/`.

[157] NIST. Announcing the Advanced Encryption Standard (AES). Technical report, CSRC, 2001. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[158] Karsten Nohl and Sylvain Munaut. Wideband GSM Sniffing. In *27th Chaos Communication Congress*, 2010.

[159] Toshihiro Ohigashi and Masakatu Morii. A Practical Message Falsification Attack on WPA. In *2009 Joint Workshop on Information Security*, Kaohsiung, Taiwan, August 2009.

[160] Dag Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Countermeasures: The Case of AES. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin / Heidelberg, 2006.

[161] G. Pall and G. Zorn. Microsoft Point-To-Point Encryption (MPPE) Protocol. IETF RFC3078, `http://tools.ietf.org/html/rfc3078`.

[162] Goutam Paul, Siddheshwar Rathi, and Subhamoy Maitra. On non-negligible bias of the first output byte of RC4 towards the first three bytes of the secret key. In *Designs, Codes and Cryptography*, volume 49, pages 123–134. Springer Netherlands, 2008.

[163] Souradyuti Paul and Bart Preneel. Analysis of Non-fortuitous Predictive States of the RC4 Keystream Generator. In Thomas Johansson and Subhamoy Maitra, editors, *Progress in Cryptology – INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 67–70. Springer Berlin / Heidelberg, 2003.

[164] Lawrence C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

[165] Lawrence C. Paulson. Inductive Analysis of the Internet Protocol TLS. *ACM Transactions on Information and System Security*, 2(3):332–351, 1999.

[166] Bart Preneel and Paul van Oorschot. MDx-MAC and Building Fast MACs from Hash Functions. In Don Coppersmith, editor, *Advances in Cryptology - CRYPT0' 95*, volume 963 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin / Heidelberg, 1995.

[167] Marsh Ray and Steve Dispensa. Renegotiating TLS, 4. november 2009. `http://extendedsubset.com/?m=200911`.

[168] E. Rescorla. Diffie-Hellman Key Agreement Method. IETF RFC 2631. `http://tools.ietf.org/html/rfc2631`.

[169] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov. Transport Layer Security (TLS) Renegotiation Indication Extension. IETF RFC5746, `http://tools.ietf.org/html/rfc5746`.

[170] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, February 1978.

[171] Ronald Rivest. The MD5 Message-Digest Algorithm, April 1992. RFC1321, `http://tools.ietf.org/html/rfc1321`.

[172] Ronald Rivest and Adi Shamir. Efficient Factoring Based on Partial Information. In Franz Pichler, editor, *Advances in Cryptology – EUROCRYPT' 85*, volume 219 of *Lecture Notes in Computer Science*, pages 31–34. Springer Berlin / Heidelberg, 1986.

[173] P. Rogaway and D. Wagner. A Critique of CCM. Cryptology ePrint Archive, Report 2003/070, 2003. `http://eprint.iacr.org/`.

[174] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions in Information and System Security*, 6(3):365–403, 2003.

[175] M. Salter, E. Rescorla, and R. Housley. Suite B Profile for Transport Layer Security (TLS). IETF RFC 5430.

[176] Yu Sasaki and Kazumaro Aoki. Finding Preimages in Full MD5 Faster Than Exhaustive Search. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 134–152. Springer Berlin / Heidelberg, 2009.

[177] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Commentarii Mathematici Universitatis Sancti Pauli*, 47:81–92, 1998.

[178] Bruce Schneier. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In *FSE*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204. Springer, 1993.

[179] I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Mathematics of Computation*, 67:353–356, January 1998.

[180] Sertifitseerimiskeskus AS. *DigiDoc formaadi kirjeldus, versioon 1.4*, 2006.

[181] Sertifitseerimiskeskus AS. Asutuse sertifikaatide sertifitseerimispoliitika, versioon 2.2. http://www.sk.ee/upload/files/Asutuse_CPv2_2.pdf, 10. mai 2010.

[182] Sertifitseerimiskeskus AS. ESTEID-kaardi sertifitseerimispoliitika, versioon 3.2. http://www.sk.ee/upload/files/SK-CP-ESTEID-v3_2.pdf, 1. jaanuar 2010.

[183] Sertifitseerimiskeskus AS. Sertifikaadid Eesti Vabariigi isikutunnistusel, versioon 3.3. http://www.sk.ee/upload/files/ESTEID_profiil_et-3_3.pdf, 1. jaanuar 2010.

[184] Sertifitseerimiskeskus AS. Sertifitseerimispõhimõtted – CPS, versioon 2.5. http://www.sk.ee/upload/files/SK_CPS_v2_5.pdf, 1. jaanuar 2010.

[185] Sertifitseerimiskeskus AS. Asutuse sertifikaatide ja tühistusnimekirja profiil, versioon 1.3. http://www.sk.ee/upload/files/Asutuse_sertifikaadi_profiil_v1_3.pdf, 14. veebruar 2011.

[186] Sertifitseerimiskeskus AS. DigiDoc teegid. http://id.ee/?id=28729, 31. jaanuar 2011.

[187] Vitaly Shmatikov and Ulrich Stern. Efficient Finite-State Analysis for Large Security Protocols. In *CSFW*, pages 106–115, 1998.

[188] N. P. Smart. The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, 12:193–196, 1999.

[189] Dawn Xiaodong Song. Athena: A New Efficient Automatic Checker for Security Protocol Analysis. In *CSFW*, pages 192–202, 1999.

[190] JH. Song, R. Poovendran, J. Lee, and T. Iwata. The AES-CMAC Algorithm. IETF RFC4493, http://tools.ietf.org/html/rfc4493.

[191] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel Smart. Flaws in Applying Proof Methodologies to Signature Schemes. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 215–224. Springer Berlin / Heidelberg, 2002.

[192] Marc Stevens. On Collisions for MD5. Master's thesis, Eindhoven University of Technology, 2007.

[193] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Osvik, and Benne de Weger. Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 55–69. Springer Berlin / Heidelberg, 2009.

[194] Douglas R. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall / CRC, 2002.

[195] Swedbank. Pangalingi tehniline spetsifikatsioon. https://www.swedbank.ee/static/pdf/business/d2d/paymentcollection/info_banklink_techspec_2011_01_01_est.pdf.

[196] Don Syme. The F# 2.0 Language Specification, Aprill 2010. http://research.microsoft.com/en-us/um/cambridge/projects/fsharp/manual/spec.pdf.

[197] Paul Syverson and Paul C. van Oorschot. On unifying some cryptographic protocol logics. In *1994 IEEE Symposium on Research in Security and Privacy*, pages 14–28. IEEE Computer Society, 1994.

[198] Erik Tews and Martin Beck. Practical attacks against WEP and WPA. In David A. Basin, Srdjan Capkun, and Wenke Lee, editors, *WISEC*, pages 79–86. ACM, 2009.

[199] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 Bit WEP in less than 60 seconds. In *Proceedings of the 8th international conference on Information security applications*, WISA'07, pages 188–202, Berlin, Heidelberg, 2007. Springer-Verlag.

[200] Serge Vaudenay. Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS... In Lars Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–545. Springer Berlin / Heidelberg, 2002.

[201] Serge Vaudenay and Martin Vuagnoux. Passive-Only Key Recovery Attacks on RC4. In Carlisle Adams, Ali Miri, and Michael Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 344–359. Springer, 2007.

[202] V.Rijmen and P.S.L.M.Barreto. The Whirlpool hash function. `http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html`.

[203] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.

[204] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 561–561. Springer Berlin / Heidelberg, 2005.

[205] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). IETF RFC3610, `http://tools.ietf.org/html/rfc3610`.

[206] Michael J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.

[207] Jan Willemson. Sissejuhatus Krüptoloogiasse. Loengumärkmed, Tartu Ülikool, 2004. `http://home.cyber.ee/jan/intro/`.

[208] Tao Xie and Dengguo Feng. How To Find Weak Input Differences For MD5 Collision Attacks. Technical report, IACR, 2009. `http://eprint.iacr.org/2009/223.pdf`.

[209] L. Zhu and B. Tung. Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). IETF RFC4556, `http://tools.ietf.org/html/rfc4556`.