

Krüptoalgoritmid ning nende tugi teekides ja infosüsteemides

Tehniline dokument

Versioon 2.0

15. juuni 2023. a.

106 lk

Dok T-184-7

Uuringu koostas Cybernetica AS Riigi Infosüsteemi Ameti tellimusel.

Projektijuhid: Martina Võrklaev (Riigi Infosüsteemi Amet)
Tõnis Reimo (Riigi Infosüsteemi Amet)
Sandhra-Mirella Valdma (Cybernetica)

Autorid: Hendrik Eerikson (Cybernetica)
Aivo Kalu, PhD (Cybernetica)
Kristjan Krips, PhD (Cybernetica)
Peeter Laud, PhD (Cybernetica)
Mart Oruaas (Cybernetica)
Jaak Pruulmann-Vengerfeldt (Cybernetica)
Raul-Martin Rebane (Cybernetica)
Jaak Ristioja (Cybernetica)
Nikita Snetkov (Cybernetica)
Jelizaveta Vakarjuk (Cybernetica)
Jan Willemson, PhD (Cybernetica)

Riigi Infosüsteemi Amet, Pärnu maantee 139a, 15169 Tallinn, Eesti
Email: ria@ria.ee, Web: <https://www.ria.ee>, Telefon: +372 663 0200

Cybernetica AS, Mäealuse 2/1, 12618 Tallinn, Eesti
E-mail: info@cyber.ee, Web: <https://www.cyber.ee>, Telefon: +372 639 7991

© Riigi Infosüsteemi Amet, 2020-2023

Sisukord

1	Sissejuhatus	6
2	Krüptoprimitiivid	7
2.1	Võtmepikkuste soovitused	7
2.2	Krüptoprimitiivide rakendamisest	8
2.3	Krüptograafilised räsifunktsioonid	8
2.3.1	SHA-2	9
2.3.2	SHA-3	9
2.4	Parooliräsamise algoritmid	9
2.4.1	bcrypt	10
2.4.2	PBKDF2	10
2.4.3	scrypt	11
2.4.4	Argon2	11
2.4.5	Balloon	11
2.5	Sümmeetrilise võtmega primitiivid	11
2.5.1	Plokkšifrid	12
2.5.2	Jadašifrid	12
2.5.3	Räsipõhised sümmeetrilised konstruktsioonid	12
2.6	Asümmeetrilised primitiivid	13
2.6.1	RSA	13
2.6.2	ElGamal krüptoskeem	13
2.6.3	Elliptikõveratel põhinevad süsteemid	13
2.7	Postkvant-krüptograafia võimalikud matemaatilised alused	14
2.7.1	Võrepõhine krüptograafia	14
2.7.2	Räsipõhine krüptograafia	18
2.8	Postkvant-krüptograafia standardimustandi hetkeseis	19
2.8.1	Võtmekapseldusmehhanismid	21
2.8.2	Signatuurskeemid	22
2.8.3	Postkvant-turvaliste krüptoalgoritmide kasutuselevõtt	26
2.8.4	Proovikivid	27
3	Protokollid ja protokollistikud	30
3.1	TLS	30
3.1.1	TLS 1.3	30
3.1.2	TLS-i analüüsid ja ründed TLS-i vastu	32
3.1.3	TLS protokollipinu osana	33
3.1.4	TLS ja osapoolte tuvastamine	37
3.1.5	Kliendi autentimine TLS-seansis	41
3.1.6	Soovitused	44
3.1.7	Muud TLS rakendamist kirjeldavad dokumendid	46
3.2	SSH	48
3.2.1	Ülevaade	48
3.2.2	Soovitused	49
3.3	IPsec	49

3.3.1	Ülevaade	49
3.3.2	Soovitused	51
3.4	WireGuard	53
4	Krüptokonteinerite vormingud	55
4.1	Cryptographic Message Syntax	55
4.2	ASiC	55
4.3	ASiC-S	55
4.4	ASiC-E	56
4.4.1	ASiC-E koos XAdES signatuuriga	56
4.4.2	ASiC-E koos CAdES signatuuriga	56
4.5	PAdES	56
4.6	ASiC ja ajatemplid	57
4.7	JAdES	57
4.8	CDOC	57
4.9	JSON Object Signing and Encryption	59
4.9.1	JSON Web Signature	59
4.9.2	JSON Web Encryption	59
4.9.3	JSON Web Token	59
4.10	PASETO	60
4.11	PGP	60
5	Autentimis-, volitus- ja delegeerimisprotokollid	61
5.1	OAuth 2.0	61
5.1.1	Protokoll	62
5.1.2	Osapooled	62
5.1.3	Skoop	63
5.1.4	Volituskinnituse hankimise viisid	64
5.1.5	Päasutõendid	65
5.2	OpenID Connect	66
5.2.1	Terminoloogia	66
5.2.2	Identiteeditõend (<i>ID Token</i>)	67
5.2.3	Skoobid	67
5.2.4	Kasutajainfo teenus	68
5.2.5	Autenditud päringud	68
5.2.6	Autentimisvood	68
5.2.7	SSO-teenuse pakkumine OpenID Connect standarditega	69
5.3	User-Managed Access	73
5.3.1	Protokoll	73
5.3.2	Födereeritud volitamine	74
5.4	FIDO ja Web Authentication	74
5.4.1	FIDO U2F	75
5.4.2	FIDO UAF	76
5.4.3	FIDO2 – WebAuthn ja CTAP	76
5.4.4	Soovitused	77
5.5	SAML	77
6	Teegid ja rakendused	79
6.1	Ülevaade levinumatest krüptoteekidest	80

6.1.1	OpenSSL	81
6.1.2	LibreSSL	81
6.1.3	BoringSSL	82
6.1.4	GnuTLS	82
6.1.5	wolfSSL	82
6.1.6	Mbed TLS	82
6.1.7	NSS	83
6.1.8	Botan	83
6.1.9	s2n	84
6.1.10	Bouncy Castle	84
6.1.11	Tink	84
6.1.12	Libgcrypt	85
6.1.13	Crypto++	85
6.1.14	Nettle	85
6.1.15	libsodium	85
6.2	RSA-PSS tugi krüptoteekides	86
6.3	Arenduskeskkonnad	86
6.3.1	C ja C++	86
6.3.2	Java	87
6.3.3	Go	87
6.3.4	Node.js	87
6.3.5	Rust	88
6.3.6	PHP	88
6.3.7	.NET	88
6.3.8	Python	89
7	Eesti ID-kaardi NFC-tugi	90
	Summary in English	91
	Kirjandus	92

1 Sissejuhatus

Käesolev uuringuaruanne on uuendus 2021. aastal valminud aruandele [110], mis kuulub samasse seeriasse kuue varasema aruandega, mis valmisid aastatel 2011-2018 [111, 112, 113, 46, 114, 145]. Uuringu eesmärk on anda ülevaade krüptograafiliste vahendite kasutamise hetkeseisust ja heast tavast. Alati on uuringu osaks olnud soovitusel kasutatavate krüptoalgoritmide ja nende parameetrite (nt võtmepikkuste jm) kohta. Aruande 2021. aasta versioon [110] pööras rohkem tähelepanu erinevatele levinud krüptograafilistele protokollidele ja nende kasutamise olulisematele asjaoludele. Samuti kirjeldas ta levinumaid krüptograafiateeke ja krüptograafiliste vahendite tuge levinud arendusplatvormidel. Käesolevas aruanderversioonis oleme neid kirjeldusi ja soovitusi uuendanud. Samuti oleme käsitletavate teekide ja vahendite nimekirja pikendanud.

Aruanne koosneb seitsmest jaotisest. Jaotises 2 vaatleme krüptoprimitiive, nende tugevust ja soovitatavaid võtmepikkusi. Jaotises 3 vaatleme transpordiprotokollistikke. Jaotises 4 vaatleme levinumaid vorminguid ning kirjeldame, millist mõju vormingutele omavad muutused krüptoprimitiivides. Jaotises 5 kirjeldame levinumaid autentimis- ja delegerimisprotokolle. Jaotises 6 kirjeldame vabavaraliste krüptograafia-teekide hetkeolukorda, sh viimasel ajal tekkinud populaarsemate arendus- ja käidukeskkondade vaatest. Käesolevas aruanderversioonis pöörame erilist tähelepanu RSA PSS signatuurivormingu toele krüptograafia-teekides. Võrreldes 2021. aastaga on aruanne täienenud jaotisega 7, kus kirjeldame NFC-tehnoloogia kasutamist infoturbe vaatepunktist.

Lähtematerjalidena on kasutatud eespool mainitud varasemaid Riigi Infosüsteemi Ameti tellitud uuringuid ning tuntud riiklike (NIST, BSI) ja rahvusvaheliste (SOG-IS) organisatsioonide samalaadseid uuringuid ja soovitusi. Lisaks toetub töö ETSI, NISTi ja IETFi standarditele ning eelretsenseeritud teadusartiklitele.

Aruande tarkvaratehnilistes osades on rakendatud Cybernetica AS pikaajalisi kogemusi krüptovormingute ja -teekide valikul ning nende kvaliteedi hindamisel.

Aruande koosseis on määratud riigihanke nr 213942 „Krüptoalgoritmid ning nende tugi teekides ja infosüsteemides“ (2020) ja raamlepingu nr. 4.2-3/219-1 alt tellitud (2022) tööde kirjeldustega.

Aruanne on suunatud lugemiseks kõigile IT-taristu ning tarkvaraarenduse valdkondade juhtidele ja spetsialistidele.

2 Krüptoprimitiivid

2.1 Võtmepikkuste soovitused

Soovituslikud võtmepikkused ning krüptoprimitiivide kasutatavuse ajahorisontide hinnangud tuginevad NISTi [10], SOG-IS [3] ja BSI [47] soovitudele.

NIST on uuendamas oma aruandeid ja standardeid, millest meie soovitusel lähtuvad. Mais 2020 anti aruandest 800-57 välja 5. versioon [10]. Võrreldes 2017. aasta krüptoalgoritmide elutsükli uuringuga [114] ei ole hinnangud muutunud; samuti ei ole me teadlikud, et peale 2021. aasta krüptoalgoritmide elutsükli uuringu [110] avaldamist oleks tehtud krüptoanalüütilisi edusamme, mis sunniks neid hinnanguid ümber hindama. NIST on asunud ka uuendama digisignatuuri standardit FIPS 186 ja avaldanud sellest uue versiooni FIPS 186-5 [128]. NISTi soovitusel krüptograafiliste algoritmide perede turvasemetele võtab kokku tabel 1.

Algoritmide turvatase määratakse bittides – mingi algoritmi b -bitine turvatase tähendab, et efektiivseim teadaolev rünne kasutab arvutusressurssi, mis kulub ligikaudu 2^b ploki krüpteerimiseks plokkšifriga. Turvaseme formaalsem definitsioon on antud NISTi aruandes 800-57 [10, jaotis 5.6.1.1].

Tabeli 1 veerg „DSA, DH“ viitab DSA signatuurialgoritmile ja Diffie-Hellmani võtmekehtestus-protokollile üle jäägiklassiringi. Nende konstruktsioonide puhul tagavad vajaliku turvaseme avaliku võtme pikkus L (s.t. jäägiklassiringi elementide pikkus) ja salajase võtme pikkus N (s.t. salajase eksponendi pikkus). Veerg „RSA“ annab soovitusel RSA algoritmi mooduli pikkusele. Nende moodulipikkuste jaoks, mida tabelis 1 ei ole, võib turvaset püüda hinnata valemiga FIPS 140-2 realisatsioonisoovitustest [93, jaotis 7.5], seda muidugi eeldusel, et moodul ise on genereeritud vastavalt standardi FIPS 186 soovitudele [128, lisa A].

Veerg „ECC“ tabelis 1 annab soovitusel elliptikõveratele tuginevate krüptoalgoritmide võtmepikkustele. Veerud „Plokkšifrid“ ja „SHA-2, SHA-3“ soovivad vastavalt sümmeetriliste plokkšifri-algoritmide võtmepikkust ning räsifunktsioonide väljundi pikkust. Peamiseks soovitatavaks sümmeetriliseks krüptoalgoritmiks on jätkuvalt AES.

Tabel 1. Krüptograafiliste algoritmide soovitatavad võtmepikkused

Turvatase	DSA, DH	RSA	ECC	Plokkšifrid	SHA-2, SHA-3
128	$L = 3072, N = 256$	3072	256...383	128	256
192	$L = 7680, N = 384$	7680	384...511	192	384
256	$L = 15360, N = 512$	15360	512+	256	512

2.2 Krüptoprimitiivide rakendamisest

Standarditud krüptoprimitiive on võimalik rakendada laias laastus kahel viisil.

- Osana mingist standarditud protokollist, näiteks TLS.
- Osana rakendusespetsiifilisest liidestusest.

Kui esimese viisiga ei ole tavaliselt probleeme ning probleemide ilmnemisel käsitlevad neid nii standardiorganisatsioonid kui infoturbega professionaalselt tegelevad avalikud ja eraõiguslikud osapooled, siis teine viis on tihtipeale problemaatilisem.

Küsimus on selles, et krüptoprimitiivi turvatase ega korralikult realiseeritud algoritm koos võtmehaldusega ei ole piisav selleks, et saavutada turvaomadusi, mida süsteemi rakendaja soovib saavutada. Halvemal juhul võib ebakorrektselt rakendatud krüptograafiliste primitiivide kombinatsioon lausa nõrgendada mõne teise krüptograafiliste meetmete kaudu seotud süsteemi turvalisust (näiteks sama privaativõtit kasutades [158]).

Krüptograafilistel protokollidel on väga palju muid nõrkusi peale krüptoprimitiivide otsese murtavuse [31, peatükk 1.4] ning tihtipeale ei ole need nõrkused väga ilmsed, kuivõrd ilmnevad sõnumivahetuse vaid väga spetsiifilistes kombinatsioonides.¹

Krüptoprimitiivide rakendamisel väljaspool standardprotokolle või standardseid vorminguid soovitage plaanitava tööle alati küsida pädeva krüptograafi hinnangut – krüptograafide rakendatavad uurimismeetodid võimaldavad selliseid nõrkusi tuvastada, vt näiteks ProVerifi rakendamist Mobiil-ID protokollu uurimiseks [116].

2.3 Krüptograafilised räsifunktsioonid

Räsifunktsioonid (*hash function*, *hash algorithm*, *digest function*) on funktsioonid, mis võtavad oma sisendiks vaid sõnumi ning väljastavad (tavaliselt fikseeritud pikkusega) sõnumilühendi ehk räsi.

On oluline tähele panna, et kuigi tehniliselt räsifunktsiooniks kvalifitseeruva funktsiooni koostamine ei ole ülearu keeruline, tohib krüptograafilistes rakendustes kasutada vaid aktsepteeritud ja läbiuuritud **krüptograafilisi räsifunktsioone** nagu näiteks SHA-2 või SHA-3 perekondade funktsioonid.

Krüptograafilise räsifunktsiooni kohustuslikeks omadusteks on kollisioonikindlus (*collision resistance*), mittepööratavus (*pre-image resistance*) ning lisaoriginaalikindlus (*second pre-image resistance*) – Rogaway ja Shrimpton seletavad need mõisted täpsemalt lahti krüptograafiliste räsifunktsioonide aluseid tutvustavas artiklis [157].

Tavaliselt on arvutuskirius krüptograafilise räsifunktsiooni puhul omaette väärtus. Eraldi kategooria moodustavad spetsiaalselt lõppkasutaja-paroolide räsimiseks projekteeritud funktsioonid. Need funktsioonid esitavad suuri nõudmisi algoritmi käitamiseks kasutatavale riistvarale, et raskendada jõuründeid. Sellised funktsioonid on näiteks bcrypt, scrypt ja Argon2.

¹<https://buttondown.email/cryptography-dispatches/archive/cryptography-dispatches-the-most-backdoor-looking/>

2.3.1 SHA-2

2001. aastal avaldas USA *National Security Agency* (NSA) SHA-2 esimese spetsifikatsiooni. Praegu spetsifitseerib funktsiooni SHA-2 standard FIPS 180-4 [163].

SHA-2 puhul on standarditud funktsioonid SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 ning SHA-512/256, kõigis neis nimedes näitab (viimane) kolmekohaline arv väljundi pikkust bittides. Vaid SHA-256 ja SHA-512 on iseseisvad algoritmid, ülejäänud variantide puhul on tegu pikema väljundiga funktsioonist lühema väljundiga funktsiooni saamisega algväärtusvektori muutmise ning väljundi bittide äralõikamise teel. Praktilistes rakendustes ei oma see küll tähendust, kuniks kasutatakse sobiva väljundipikkuse ja turvasemega räsifunktsiooni.

SHA-2 variantide turvasemete hinnangud on toodud tabelis 1, kuid töö SHA-2 krüptoanalüüsi kallal kindlasti jätkub ning neid hinnanguid vaadatakse regulaarselt üle.

Oluline on märkida, et SHA-2 vastu ei ole täna teada klassikalistest otsingualgoritmidest efektiivsemaid kvantalgoritme (va üldotstarbeline Groveri algoritm, mida on varasemates uuringutes käsitletud [114]).

2.3.2 SHA-3

Räsialgoritm SHA-3 on NISTi poolt 2015. aastal standarditud kui täiendus algoritmidele SHA-1 ja SHA-2. Seejuures nägi NIST algoritmi SHA-3 standardides ette tänaseks edukalt realiseerunud ründeid SHA-1 vastu [165].

NISTi standard pakub nelja standardset fikseeritud väljundipikkusega räsifunktsiooni – SHA3-224, SHA3-256, SHA3-384 ja SHA3-512. Viimane arv funktsiooni nimes näitab väljundi pikkust bittides ning pikkused on valitud nii, et nad langeksid kokku SHA-2 perekonna funktsioonide väljundipikkustega.

SHA-3 perekonda kuuluvad ka muutuva (lõpmatu) väljundi pikkusega räsifunktsioonid SHAKE128 ja SHAKE256 – neid saab kasutada näiteks juhuarvugeneraatorina.

SHA-3 juurutamisel tasub kontrollida, kas algoritmi teostus on saadaval kõigi seotud tarkvarasüsteemide ja -platvormide jaoks. Eriti oluline on see vanemate pärandisüsteemide puhul – näiteks Java rakenduste maailma valitsevad endiselt Java 7 ja 8 koos, kuid Java 8 krüptoalgoritmide standardnimede loetelus perekonda SHA-3 ei sisaldu.

Pikemalt võib SHA-3 ja SHAKE'i kohta lugeda 2016. aasta krüptoalgoritmide elutsükli uuringust [46].

2.4 Parooliräsamise algoritmid

Turvaintsidentide statistikast¹ on selgelt näha, et viimasel viiel aastal on toimunud erinevatest teenustest ja infosüsteemidest vähemalt paarsada infoleket, millel on järgmine muster.

- Infosüsteemis kaitstakse lõppkasutajate paroole liiga nõrgalt (paroole hoitakse kas avatekstina või räsituna mõne lihtsa meetodiga – tavapärane on standardse räsifunktsiooni ühekordne rakendamine).

¹<https://haveibeenpwned.com/PwnedWebsites>

- Sissetungija varastab kasutajate andmebaasi, mis sisaldab kasutajatunnuseid (tüüpiliselt e-posti aadress) ning paroole või parooliräsisid ning nõrkade parooliräside korral arvutab neist jõuründega paroolide väärtused.¹
- Murdunud paroolide abil saab teenuses kasutajakontosid üle võtta ning kuivõrd inimesed kasutavad sama parooli väga tihti paljudes teenustes, saab neid paroole pruukida kasutaja kontode ülevõtmiseks teisteski teenustes.

Kuigi infosüsteemide turbe kõik meetmed on olulised, siis konkreetselt sellise ründemustri vastu aitab salvestatud parooli kaitsmine räsi-algoritmiga, mille puhul on sisendiks oleva parooli jõuründega leidmine väga ressursimahukas. Järgnevalt vaatleme mõningaid räsi-algoritme. Osa neist algoritmidest on nimetatud võtmetuletusfunktsioonideks, kuid sisuliselt on tegu sama probleemi lahendamisega – mingist jagatud saladusest raskesti ennustatava baadidajada tuletamisega.

Paroolide korralik räsimine võimaldab rakendada ja/või efektiivsemaks muuta ka muid nõudeid ja soovitusi, mis NIST juba 2016. aastal paroolide kohta andis [79, jaotis 5.1.1.2].

- Paroolide regulaarset vahetamist ei tohiks nõuda.
- Kui paroolidele on kehtestatud maksimumpikkus, siis peaks see olema vähemalt 64 tähemärki.
- Paroolidele ei tohiks kehtestada muid nõudeid peale miinimumpikkuse (näiteks nõue kasutada suur- ja väiketähti segamini ja muud sellised piirangud).
- Kasutaja valitud paroole peaks võrdlema sõnastike või reeglaliste vastu, et hoida ära levinud või kompromiteeritud paroolide kasutamist.

2.4.1 bcrypt

Algoritmi bcrypt avaldasid 1999. aastal Niels Provos ja David Mazières [148] ning see on sellest ajast saati olnud kõige populaarsem parooliräsimise algoritm. Algoritm baseerub plokkšifril Blowfish, mille võtmeloomeriutiini keerukust kasutatakse parooliräsimise arvutusmahukuse tõstmiseks.

Algoritmi bcrypt positiivseks omaduseks on kahtlemata teostuste suur valik erinevate programmeerimiskeelte ja -raamistike jaoks.

Negatiivse poole pealt võib ära märkida piirangut parooli pikkusele (72 baiti) ning suhtelist nõrkust FPGA riistvaraga sooritavate jõurünnete suhtes [178]. Samuti ei kuulu bcrypt ühegi riikliku agentuuri poolt soovitatud algoritmide nimekirja.

FPGA-põhiste rünnete tõrjumiseks tuleb uutes süsteemides kasutada spetsiifilisi vastumeetmeid rakendavaid algoritme, nagu näiteks scrypt.

2.4.2 PBKDF2

PBKDF2 on IETF poolt standarditud [131] võtmetuletusfunktsioon, mis on ühtlasi ka üks vähesi NISTi poolt soovitatud võtmetuletus- ja parooliräsi algoritme [175] [79].

Kuigi PBKDF2 on defineeritud nii SHA-1 kui SHA-2 räsi-funktsioonide baasil, ei soovi me SHA-1 varianti kasutada, kuivõrd SHA-1 ise on ebaturvaliseks tunnistatud algoritm [114].

¹<https://hashcat.net/hashcat/>

Teadaolevalt ei mõjuta see otseselt PBKDF2-HMAC-SHA1 turvalisust, kuid segaduste vältimiseks on parem SHA-1 kasutamisest täielikult hoiduda.

NIST soovib PBKDF2 kasutada vähemalt 10000 iteratsiooniga [79, jaotis 5.1.1.2].

2.4.3 scrypt

Algoritmi scrypt avaldas Colin Percival 2009. aastal [142] ning tänaseks on see standarditud kui IETF RFC 7914 [143].

Algoritmi scrypt põhiline erinevus eellastest on sõltuvus mälumahust, mitte kiiremast riistvarast. See teeb märksa kallimaks rööbitised jõuründed – ründamiseks pruugitava süsteemi integraalskeemide maht peab olema suurem, mistõttu ründe efektiivsust on keerulisem parandada.

Interaktiivsete teenuste puhul soovitame kasutada scrypt-i parameetreid $r = 8$, $p = 1$ ja $N = 2^{15} = 32768$. Sellised väärtused määravad üksiku parooliräsi arvutamiseks vajaliku mälumahu suuruseks 32MB ning 3GHz Intel Core i7 protsessoril võtab sellise räsi arvutamine aega 55 millisekundit. Valikumetoodika on pärit originaalartiklist [142, osa 8].

2.4.4 Argon2

Argon2 võitis 2015. aastal *Password Hashing Competition*¹ võistluse. Aastal 2021 sai temast IETF-i standard [24].

Argon2 on tegelikult algoritmide perekond, kus iga üksiku algoritmi omadused on pisut erinevad.

Argon2d – optimeeritud GPU-põhiste rünnete vastu, kuid võib sisaldada kõrvalkanaliründeid lihtsustavaid nõrkusi.

Argon2i – optimeeritud kõrvalkanalirünnete vastu.

Argon2id – eelmise kahe hübriid. Standard [24] soovib kasutada Argon2id algoritmi kõigil juhtudel kui ülejäänud variantide eelistamiseks ei ole häid põhjuseid, samuti soovib seda kasutada BSI [47].

Soovitatud parameetrid Argon2 perekonna algoritmidele on toodud standardis [24, jaotis 4].

Argon2 on realiseeritud näiteks teegis libsodium (vt jaotis 6.1.15).

2.4.5 Balloon

NISTi digitaalidentiteedi halduse soovitused [79] loetlevad ainsa soovitusliku algoritmina PBKDF2 kõrval ka Balloon-i [28], kuid tolle puuduseks on kättesaadavate teostuste napp kogus.

2.5 Sümmeetrilise võtmega primitiivid

Sümmeetrilise võtmega krüptoprimitiivid on sellised primitiivid, mille puhul kõigi operatsioonide teostamiseks kasutatakse sama võtmematerjali. Sümmeetrilised krüptoprimitiivid jagunevad kolme suurde kategooriasse: plokkšifrid, jadašifrid ning räsipõhised konstruktsioonid.

¹<https://www.password-hashing.net/>

2.5.1 Plokkšifrid

Plokkšifffer on oma olemuselt üks-ühene teisendus n -bitiste plokkide vahel. Konkreetse teisenduse määrab k -bitine võti. Plokipikkus n ja võtmepikkus k ei pea olema võrdsed.

Selle aruande kirjutamise ajal on ainus piisavalt turvaline standarditud plokkšifrialgoritm AES (*Advanced Encryption Standard*), mille turvasemed vastavalt võtmepikkusele on toodud tabelis 1.

Plokkšifreid ei kasutata kunagi nende puhtal kujul, vaid mingis konkreetsetes režiimis (AES-GCM, AES-CBC) või mingi teise konstruktsiooni (näiteks jadašifri või juhuarvude generaatori) ehituselemendina.

2.5.2 Jadašifrid

Jadašifffer võimaldab suvalise L -bitise avateksti krüpteerimist, genereerides k -bitisest võtmest L -bitise bitijada, mis kombineeritakse avatekstiga bitthaaval XOR operatsiooni abil (ehk *modulo 2* liitmistehtega).

Jadašifrite saamiseks on mitmeid viise: jadašifrina kasutamiseks projekteeritud algoritmide, muutuva pikkusega räsifunktsioonid (SHAKE) või sobivas režiimis plokkšifrid (AES-CTR).

Jadašifrid pakuvad selle aruande kontekstis huvi ennekõike oma kasutuse tõttu TLSi šifrikomplektides, ChaCha20 algoritmi näol. Üldisemalt on neil eelis plokkšifrite ees sellistes rakendustes, mis nõuavad ettemääramata suurusega andmesegmentide krüpteerimist ja edastamist piiratud oludes – näiteks raadiosides, kus plokkšifri kasutamine šifri ploki suurusest väiksemate segmentide edastamisel oleks vajaliku täidise tõttu raiskavam.

2.5.2.1 ChaCha20

ChaCha on Daniel J. Bernsteini 2008. aastal publitseeritud [19] jadašifrite perekond, number variandi ChaCha20 nimes tähistab iteratsioonide arvu (vähema kui 20 iteratsiooniga ChaCha variandid ei ole piisava turvasemega).

ChaCha20 on kasutusel TLS 1.2 ja TLS 1.3 jaoks soovitatud šifrikomplektides. ChaCha20 kasutatakse nendes komplektides koos Poly1305 autentimiskoodiga [139].

Hinnatakse, et ChaCha20 on ilma spetsiifilise AESi toeta riistvaral kolm korda kiirem kui AES [139], seetõttu on tema kasutusvaldkond TLSis ennekõike väheste ressurssidega arvutisüsteemides (IoT, sardsüsteemid).

ChaCha20 võtmepikkus on 256 bitti ning selle turvatase on sama, kui Salsa20/20 jadašifril, millel ChaCha põhineb – 256 bitti [21].

2.5.3 Räsipõhised sümmeetrilised konstruktsioonid

Räsipõhiste konstruktsioonide alla kuuluvad ennekõike sõnumiautentimiskoodid (*hash-based message authentication code*, HMAC) [109]. Standardiseeritud HMAC koodide turvatase sõltub ennekõike kasutatavast võtmepikkusest ning SOG-IS [3] soovib kasutada HMAC võtmepikkust vähemalt 125 bitti. Tüüpiliselt on rakendustes HMAC võtme pikkus seotud kasutatava räsifunktsiooni väljundipikkusega.

2.6 Asümmeetrilised primitiivid

2.6.1 RSA

RSAst on pikemalt juttu 2016. aasta krüptoalgoritmide elutsükli uuringus [46, jaotis 3]. Turvatasemete hinnangud RSA algoritmile on sellest ajast muutunud. Võtmepikkustele vastavad turvatasemed on ära toodud tabelis 1.

Eestis vahepeal kasutusele võetud Smart-ID autentimis- ja allkirjastamisteenus kasutab oma krüptograafilise alusena ebatraditsioonilist nelja algarvulise teguriga RSA signatuuriskeemi, mille turvatase sama võtmepikkuse juures ei vasta tava-RSAle. Selle lahenduse kirjeldus ja turvatase hinnangud koos metoodikaga on toodud temaatilises teadusartiklis [38].

Artikli [38] järgi on $2N$ -bitise avaliku mooduliga Smart-ID võtmepaari turvatase standardse N -bitise RSA võtmepaari omast madalam enamalt 13 bitti. Vastavalt tabelile 1 on selle aruande kirjutamise ajal Smart-ID poolt kasutatavate 6144-bitiste võtmete turvatase vähemalt 115 bitti. Kuivõrd NIST [10, jaotis 5.6.3] annab 112-bitise turvatasemega võtmete kasutushorisoniks aasta 2030, siis kasutabki Smart-ID praegu 6144-bitised võtmeid ja seda loetakse neile võtmetele väljastatavate sertifikaatide kehtivusaja (3 aastat) piires turvaliseks.

2.6.2 ElGamal krüptoskeem

ElGamal on asümmeetrilise võtmega krüpteerimisskeem, mis põhineb Diffie-Hellmani võtmekehtestusel [74].

Eestis on ElGamali krüptosüsteemi kasutatud interneti-hääletamise süsteemis IVXV [94].

2.6.3 Elliptikõveratel põhinevad süsteemid

Elliptikõverate krüptogaafiast on pikemalt juttu varasemates uuringutes aastatest 2015 [113] ja 2016 [46], seetõttu vaatame siinkohal ainult neid aspekte, mis on nendest aruannetest saadik muutunud või mida tasub üle rõhutada.

Ühe tähelepanuväärsema uuendusena võib ära märkida Daniel J. Bernsteini väljapakutud elliptikõvera Curve25519 [20] standardimist IETF poolt RFC 7748 [115] raames. Sellest ehk veel olulisem on, et Curve25519 leidub nüüd ka NIST-i soovitusel *Special Publication* 800-186 [129] ja digitaalallkirja standardis FIPS 186-5[128].

Curve25519 on mõjukas, kuivõrd tema tööks vajalikud arvutused on kiired ja kasutus pole piiratud patentidega. Ka on sel kõveral elliptikõverate krüptograafia seisukohast olulisi kõrvalomadusi (näiteks keeruturvalisus – *twist security*), mis muudavad algoritmide teostuse lihtsamaks.¹

Rääkides Dan Bernsteini elliptikõverast Curve25519 on vaja silmas pidada järgmist terminoloogiat²:

Curve25519 – Elliptikõver ise.

X25519 – Kõveral 25519 baseeruv Diffie-Hellmani võtmekehtestusprotokoll, mis kasutab kõvera esitust Montgomery koordinaatides.

¹<https://safecurves.cr.yp.to/>

²https://mailarchive.ietf.org/arch/msg/cfrg/-9LEdnzVrE5R0Rux30o_oDDRksU/

Ed25519 – Kõveral 25519 baseeruv signatuuriskeem, mis kasutab kõvera esitust Edwardsi koordinaatides.

Teise uuendusena võib märkida EdDSA signatuuriskeemi, eriti selle konkreetset teostust, mis on tuntud kui Ed25519. Ed25519 on EdDSA digisignatuuriskeemi konkreetne isend – on oluline tähele panna, et kui Weierstraßi kõveratel baseeruvad digisignatuurid vastavad ECDSA skeemile, siis Ed25519 aluseks olev EdDSA on Schnorri signatuuriskeemi edasiarendus Edwardsi elliptikõverate jaoks. X25519 ja Ed25519 realisatsioonid on võrreldes eelmise aruandega [114] laiemalt levinud,¹ muuhulgas on NIST standardiseerinud signatuuriskeemi Ed25519 [128].

Tabelis 2 on loetletud elliptikõverad, mida SOG-IS, NIST ja IETF soovivad.

Tabel 2. Standarditud elliptikõverad

Kõvera nimi	Standard	Kuju	Märkused
Brainpool P256r1 Brainpool P348r1 Brainpool P512r1	RFC 5639 [126]	Weierstraß	
NIST P-256 NIST P-384 NIST P-521	NIST FIPS186-4 [55]	Weierstraß	P-256 ja P-384 on kasutusel Eesti Mobiil-ID ja ID-kaardi võtmepaaride jaoks.
Curve25519 Edwards25519	RFC 7748 [115]	Montgomery Edwards	Omavahel algebraliselt seotud

2.7 Postkvant-krüptograafia võimalikud matemaatilised alused

2.7.1 Võrepõhine krüptograafia

Käesolev aruanne on meie teada esimene eestikeelne kirjutis, mis annab põhjalikuma ülevaate postkvant-turvaliste krüptoskeemide konstruktsioonidest, sealhulgas matemaatilistest struktuuridest, mida need konstruktsioonid kasutavad. Leiame, et põhjendatud on alustada nende matemaatiliste struktuuride kirjeldamist baasmõistetest, sest aruanne peaks olema loetav ka lugejale, kelle vastavad taustateadmised pole sügavad.

2.7.1.1 Ringid

Olgu X mingi hulk. (*Algebraliseks*) *tehteks* sellel hulgal nimetame suvalist funktsiooni f , mis võtab argumentideks mingi fikseeritud arvu elemente hulgast X ning tagastab taas hulga X elemendi. Hulka ühes mingi komplekti tehetega nimetame *algebraliseks struktuuriks*. *Kommutatiivne ring* on algebraline struktuur, millel on antud kaks kaheargumendilist tehet — nimetame neid liitmiseks ja korrutamiseks —, mis rahuldavad teatud omadusi:

- Liitmine on *assotsiatiivne* ja *kommutatiivne*: kõigi elementide x, y, z jaoks $(x + y) + z = x + (y + z)$ ja $x + y = y + x$.
- Korrutamine on samuti assotsiatiivne ja kommutatiivne.

¹<https://ianix.com/pub/curve25519-deployment.html>

- Leiduvad *nullelement* 0 ja *ühikelement* 1, mis iga elemendi x jaoks rahuldavad tingimusi $0 + x = x$ ja $1 \cdot x = x$.
- Igal elemendil x leidub *vastandelement* y , mis rahuldab tingimust $x + y = 0$. Seda elementi y tähistame tavaliselt $-x$.
- Korrutamine *distributeerub* üle liitmise: kõigi elementide x, y, z jaoks $x \cdot (y + z) = x \cdot z + y \cdot z$.

Nii nende üldiste definitsioonide kui ka järgnevate konkreetsete näidete juures viitame taustamaterjalina Tartu Ülikooli kursuste „Algebra I“¹ ja „Arvuteooria“² materjalidele, eriti loengukonspektidele.

Täisarvude ringid Täisarvude hulka $\{\dots, -2, -1, 0, 1, 2, \dots\}$ tähistame sümboliga $\mathbb{Z}$. Täisarvude hulgal on defineeritud liitmine ja korrutamine, ühes nende tehetega on $\mathbb{Z}$ ring.

Samuti on defineeritud täisarvu *jäägiga jagamine* naturaalarvuga: kui $a \in \mathbb{Z}$ ja $q \in \mathbb{N}$ (siin $\mathbb{N}$ tähistab mittenegatiivsete täisarvude ehk naturaalarvude hulka $\{0, 1, 2, \dots\}$), kus $q \neq 0$, siis leiduvad täpselt üks $b \in \mathbb{Z}$ ja üks $r \in \{0, 1, \dots, q-1\}$, nii et $a = bq + r$. Arvu b nimetame siin jagatiseks ja arvu r jäägiks.

Olgu $q \in \mathbb{N}$ fikseeritud. Kui arvud $a, b \in \mathbb{Z}$ annavad arvuga q jagamisel ühe ja sama jäägi, siis kirjutame $a \equiv b$. Seos $\equiv$ on ekvivalentsiseos. Selle seose järgi jagunevad kõik täisarvud ekvivalentsiklassidesse. Neid klasse on q tükki, me tähistame neid arvudega $0, 1, \dots, q-1$ ja nimetame *jäägiklassideks*. Jäägiklassi r kuuluvad kõik need arvud, mis annavad q -ga jagamisel jäägi r . Olgu meil jäägiklasside hulk $\mathbb{Z}_q = \{0, \dots, q-1\}$. Hulgal $\mathbb{Z}_q$ saab defineerida liitmise (lahutamise) ja korrutamise, need tehted näevad välja nagu vastavad tehted täisarvudega, millele aga järgneb q -ga jagamine ja jäägi võtmine. Koos nende tehetega moodustab $\mathbb{Z}_q$ ringi, mida nimetame *jäägiklassiringiks*.

Olgu N mingi naturaalarv, siis $\mathbb{Z}^N$ tähistagu kõigi N -pikkuste, täisarvuliste komponentidega vektorite hulka. Defineerime neil vektoritel liitmise (lahutamise) ning täisarvuga korrutamise, defineerime need komponentide kaupa:

$$\begin{aligned}(a_1, a_2, \dots, a_N) \pm (b_1, b_2, \dots, b_N) &:= (a_1 \pm b_1, a_2 \pm b_2, \dots, a_N \pm b_N) \\ c \cdot (a_1, a_2, \dots, a_N) &:= (c \cdot a_1, c \cdot a_2, \dots, c \cdot a_N) .\end{aligned}$$

Sarnaselt tähistagu $\mathbb{Z}_q^N$ kõigi jäägiklassivektorite hulka. Neile vektoritele saame defineerida liitmise (lahutamise) ning jäägiklassiga $c \in \mathbb{Z}_q$ korrutamise.

Polünoomide ringid Allpool kirjeldatud krüptosüsteemides tehakse sageli tehteid mitte üksnes täisarvude ja jäägiklassidega, vaid ka *polünoomidega* üle nende. Formaalselt, *polünoom* üle ringi $\mathbb{Z}$ või $\mathbb{Z}_q$ on selle ringi elementide *lõpmatu* jada $(a_0, a_1, \dots)$, kus aga ainult *lõplik* arv elemente erineb nullist. Teisisõnu, peab leiduma selline indeks N , et iga $n > N$ jaoks $a_n = 0$. Vähimat sellist N -i nimetame polünoomi *astmeks*. Polünoomidel $(a_0, a_1, \dots)$ ja $(b_0, b_1, \dots)$ sarnaselt vektoritele, saame defineerida liitmise ning alloleva ringi elemendiga c korrutamise:

$$\begin{aligned}(a_0, a_1, \dots) \pm (b_0, b_1, \dots) &:= (a_0 \pm b_0, a_1 \pm b_1, \dots) \\ c \cdot (a_0, a_1, \dots) &:= (c \cdot a_0, c \cdot a_1, \dots) .\end{aligned}$$

¹<https://courses.ms.ut.ee/2022/algebra1/fall>

²<https://courses.ms.ut.ee/2023/Arvuteooria/spring>

On ilmne et mõlema operatsiooni tulemuseks on polünoom: nii liitmise kui ka ringi elemendiga korrutamise tulemuseks olevas jadas on ainult lõplik arv nullist erinevaid elemente.

Lisaks eeltoodud tehetele saame defineerida ka polünoomide korrutamise: $(a_0, a_1, \dots) \cdot (b_0, b_1, \dots) := (c_0, c_1, \dots)$, kus iga mittenegatiivse täisarvu N jaoks

$$c_N = \sum_{i=0}^N a_i b_{N-i}.$$

Ka polünoomide korrutamise tulemus on polünoom, sest et kui N on piisavalt suur (suurem, kui $(a_0, a_1, \dots)$ ja $(b_0, b_1, \dots)$ astmete summa), siis on suurust c_N defineerivas summas kõik liidetavad võrdsed nulliga. Eeltoodud viis polünoome defineerida võib näida pisut harjumatu, oleme ju harjunud neist mõtlema kui summadest $a_0 + a_1X + a_2X^2 + \dots + a_NX^N$, kus $a_0, \dots, a_N$ on ringi $\mathbb{Z}$ või $\mathbb{Z}_q$ elemendid ning $a_N \neq 0$. Seda, mis on X , me ei pruugi aga osata kuigi täpselt sõnastada. Küll aga näeme, et eeltoodud liitmise ja korrutamise definitsioonid jadadena defineeritud polünoomidel langevad kokku liitmise ja korrutamisega, mida oleme harjunud läbi viima summadel.

Käesolevas dokumendis me ei annagi suurusele X täpsemat sisu kui see, et ta on *formaalne muutuja*. Summa $p = a_0 + a_1X + \dots + a_NX^N$ on *formaalne summa*, mis hõlbustab meil polünoomidest mõtlemist, sest et tehete intuiitiivsed valemid langevad kokku formaalsete definitsioonidega. Väärtusi $a_0, \dots, a_N$ nimetame polünoomi p *kordajateks* ning formaalset üksliiget a_NX^N polünoomi p *pealiikmeks*.

Kõigi polünoomide (üle ringi $\mathbb{Z}$ või $\mathbb{Z}_q$) hulk ühes liitmise ja korrutamisega on samuti ring. Tähistame seda ringi $\mathbb{Z}[X]$ või $\mathbb{Z}_q[X]$. Selles tähistuses on sees formaalne muutuja X , mida ringi elementide üleskirjutamisel kasutame.

Olgu a mingi ringi mingi element. Ütleme, et a on *pööratav*, kui leidub selle ringi mingi element b , nii et $a \cdot b = 1$, kus 1 on selle ringi ühikelement, s.t. element, millega suvalist elementi korrutades viimane ei muutu. Elementi b nimetame elemendi a *pöördelemendiks*. Ringis $\mathbb{Z}$ on täpselt kaks pööratavat elementi: 1 ja -1 ; nad on kumbki iseenese pöördelemendid. Ringis $\mathbb{Z}_q$ on üldiselt rohkem pööratavaid elemente; tuleb välja et $a \in \mathbb{Z}_q$ on pööratav täpselt siis, kui a ja q on ühistegurita, ning pöördelemendi leidmine on arvutuslikult lihtne. Kui q on algarv, siis on pööratavad ringi $\mathbb{Z}_q$ kõik elemendid, v.a. 0 . Olgu meil antud polünoomid p ja f ringist $\mathbb{Z}[X]$ või $\mathbb{Z}_q[X]$. Ütleme, et polünoomi p saab polünoomiga f *jäägiga jagada*, kui leiduvad polünoomid r ja s niimoodi, et $p = s \cdot f + r$ ning polünoomi r aste on väiksem kui polünoomi f aste. Polünoomi s nimetame *jagatiseks* ja polünoomi r *jäägiks*. Jäägiga jagamise piisav (mitte aga alati tarvilik) tingimus on polünoomi f pealiikme kordaja pööratavus. Näeme, et see tingimus on alati täidetud ringis $\mathbb{Z}_q[X]$, kui q on algarv. Seetõttu vaatamegi allpool ainult selliseid ringe $\mathbb{Z}_q[X]$, kus q on algarv.

Eelnevalt viisime läbi ringi $\mathbb{Z}_q$ konstrueerimise, alustades ringist $\mathbb{Z}$ ja selle elemendist q . Sarnase konstruktsiooni saame läbi viia, alustades ringist $\mathbb{Z}[X]$ või $\mathbb{Z}_q[X]$ ja mingist sellesse ringi kuuluvast polünoomist f . Olgu selle polünoomi aste N ning olgu tema pealiikme kordaja pööratav. Kui polünoomid p ja t sellest samast ringist annavad f -iga jagamisel sama jäägi, siis kirjutame jällegi $p \equiv t$. Seos $\equiv$ on ekvivalentsiseos ning selle järgi jaotuvad ringi $\mathbb{Z}[X]$ või $\mathbb{Z}_q[X]$ kõik elemendid ekvivalentsiklassidesse. Tuleb välja, et igasse klassi (mida jälle nimetame jäägiklassideks) kuulub täpselt üks polünoom, mille aste on ülimalt $N - 1$. Mugav

on samastada iga jäägiklass selle polünoomiga temas. Kõigi jäägiklasside hulka tähistame $\mathbb{Z}[X]/f$ või $\mathbb{Z}_q[X]/f$.

Jäägiklasside hulk $\mathbb{Z}[X]/f$ või $\mathbb{Z}_q[X]/f$ on jällegi ring: defineerime liitmise ja korrutamise jäägiklassidel samamoodi kui liitmine ja korrutamine on defineeritud polünoomidel. Sel viisil tehned defineerides ei vii liitmistehe meid hulgast $\mathbb{Z}[X]/f$ või $\mathbb{Z}_q[X]/f$ välja. Korrutamise tulemus ei pruugi aga enam sellesse hulka kuuluda. Seega täiendame korrutamistehte konstruktsiooni: polünoomide korrutamisele järgneb jäägi võtmine polünoomiga f jagamisel.

Maatriksite ringid Kui vektorid üle mingi hulga on „ühemõõtmeline“ struktuur, millesse on asetatud selle hulga elemendid, siis *maatriksid* on „kahemõõtmeline“ struktuur. Kui R on mingi ring, näiteks ükskõik milline eelpool kirjeldatutest, siis m -realisest ja n -veerulisest maatriksist üle ringi R mõtleme nagu elementide komplektist

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix},$$

kus $a_{11}, \dots, a_{mn}$ on ringi R elemendid. Kõigi m -realiste ja n -veeruliste maatriksite (elementidega ringist R) hulka tähistame $R^{m \times n}$.

Maatriksitel võime defineerida liitmise ja ringi elemendiga korrutamise sarnaselt vektoritele — teeme tehned komponentide kaupa. Liitmisel peab liidetavatel maatriksitel muidugi olema sama arv ridu ja sama arv veerge. Maatrikseid saab ka omavahel korrutada, kui nende ridade ja veergude arvud rahuldavad teatud omadust: selleks, et korrutis $C = A \cdot B$ oleks defineeritud, peab maatriksil A olema sama palju veerge kui maatriksil B on ridu. Korrutamise tulemuseks oleval maatriksil C on sama palju ridu kui maatriksil A ning sama palju veerge kui maatriksil B . Kui c_{ij} on maatriksi C element i -ndas reas ja j -ndas veerus, siis leitakse tema väärtus järgmiselt: $c_{ij} = \sum_{k=1}^N a_{ik}b_{kj}$, kus N on maatriksi A veergude ja maatriksi B ridade arv.

Allpool kohtame ka kirjutust $A \cdot s$, kus $A \in R^{m \times n}$ ja $s \in R^n$ on vastavalt maatriks ja vektor üle ringi R . Sel juhul mõtleme vektorist s kui n -realisest ja üheveerulisest maatriksist ning korrutamisest kui maatriksite korrutamisest. Korrutamise tulemus on vektor pikkusega m .

2.7.1.2 Võred

Olgu $L \subseteq \mathbb{Z}^N$ või $L \subseteq \mathbb{Z}_q^N$. Kui L ei ole tühi ja ta on suletud liitmise ja lahutamise suhtes, siis ütleme, et L on *võre*. See tähendab, et iga $x, y \in L$ jaoks peab kehtima $x + y \in L$ ja $x - y \in L$. Näeme, et vektor $(0, \dots, 0) = x - x$ peab kuuluma igasse võresse.

Võre L *baasiks* nimetame järjendit $(b_1, \dots, b_k)$ selle võre elementidest, nii et iga $x \in L$ jaoks leidub *täpselt* üks täisarvu- või jäägiklassikomplekt $c_1, \dots, c_k$, nii et $c_1 \cdot b_1 + \dots + c_k \cdot b_k = x$. Loomulik viis selle võre esitamiseks on võre baas. On oluline, et ühel ja samal võrel leidub üldjuhul palju erinevaid baase.

Võre on määratud mõnega oma baasidest. Võre iga punkt on määratud baasvektorite lineaarkombinatsiooniga. Võrepõhises krüptograafias leidub keerukaid probleeme mitu:

- **Lühima vektori probleem (SVP – Shortest Vector Problem)** – kui on antud mingi võre mingi baas, siis leida nullist suurem, kuid lühima pikkusega vektor selles võres.

- **Lähima vektori probleem (CVP – Closest Vector Problem)** – kui on antud mingi võre mingi baas ning sellesse võresse mittekuuluv sihtvektor, siis leida võre vektorite seast talle lähim vektor.

Kirjeldatud probleemidele vahetult tuginedes siiski veel krüptograafilisi skeeme üles ehitada ei saa. Krüptosüsteemide konstrueerimiseks sobivad järgmised keerukad võreprobleemid:

- **Lühike täisarvlahendus (SIS - Short Integer Solution)** – kui on antud ühtlase jaotusega juhusuuruste maatriks $A \in \mathbb{Z}_q^{n \times m}$ ja positiivne täisarv v , siis leida selline nullist erinev vektor $y \in \mathbb{Z}^m$, et $Ay \equiv 0 \pmod{q}$ ning $0 < \|y\| \leq v$. Siin $\|y\|$ on vektori y Eukleidiline norm defineeritud kui $\|(y_1, \dots, y_m)\| := \sqrt{y_1^2 + \dots + y_m^2}$.
- Mitut tüüpi **vigadega õppimise (LWE - Learning With Errors)** probleemid. Olgu $A \in \mathbb{Z}_q^{n \times m}$ mingi ühtlase jaotusega juhusuuruste $n \times m$ maatriks ning olgu $e \in \mathbb{Z}_q^n$ mingi teatud jaotuse χ järgi valitud veavektor, mis asetab tõenäosuse enamjaolt lühikestele vektoritele.
 - **Arvutuslik LWE** – kui mingi $s \in \mathbb{Z}_q^m$ jaoks on antud paar $(A, b := As + e)$, siis leida s .
 - **Otsustuslik LWE** – kui on antud maatriks A , siis eristada vektoreid (hulgast $\mathbb{Z}_q^n$) kujul $As + e$ (kus s on vektorite hulga $\mathbb{Z}_q^m$ mingi element) vektorite hulga $\mathbb{Z}_q^n$ juhuslikult valitud elemendist.
- **Arvutuslik NTRU¹** – Olgu R polünoomide ring kujul $R = \mathbb{Z}_q[X]/f$ (kus f on taandatud polünoom, s.t. tema pealiikme kordaja on 1). Kui on valitud element $h \in R$ mingist (statistilisest) jaotusest D nii, et eksisteerib nullist erinev $(f, g) \in R^2$, mis rahuldab tingimust $h \cdot f \equiv g \pmod{q}$ ja mille Eukleidese norm on väike, siis leida see paar (f, g) .

SIS ja LWE probleemidel eksisteerivad ka algebraliselt struktureeritud variandid, vastavalt Module-SIS ja Module-LWE, kus mingi mooduli q ning polünoomi $f = X^{2^d} + 1$ jaoks ring R on polünoomide ring kujul $R = R_q = \mathbb{Z}_q[X]/f$.

2.7.2 Räsipõhine krüptograafia

Räsipõhised signatuurskeemid on konstrueeritud räsifunktsioonidest, vt jaotis 2.3. Selles aruandes kirjeldatud olekulised ja olekuvabad skeemid põhinevad Winternitzi ühekordsetel signatuuriskeemidel (**WOTS - Winternitz One Time Signatures**) [37]. Nagu nimest nähtub, ei tohi signatuuri ühtki hetkvõtit eales kasutada enamaks kui üheainsa sõnumi signeerimiseks. Kui vastane suudab hankida kaks sama hetkvõtmega loodud signatuuri, siis suudaks ta võltsida signatuuri suvalisele sõnumile.

Võtme genereerimine Winternitzi ühekordses signatuuriskeemis toimub järgmiselt:

1. Kasuta pseudojuhuslikkuse funktsiooni (**PRF – pseudo-random function**) mille abil loo seemnest *seed* ühekordsed privaativõtmed $x_1, \dots, x_n$.
2. Arvuta n räsiahelat pikkusega ℓ , kasutades sisendina räsifunktsiooni H ja loodud privaativõtmeid.
3. Lõpliku avaliku võtme arvutamiseks kasutatakse iga ahela viimast elementi $y_i = H^\ell(x_i)$.

¹NTRU - N-th degree Truncated polynomial Ring Units

4. Lõplik avalik võti on räsi kõigist elementidest $y_1, \dots, y_n$: $pk = H(y_1, \dots, y_n)$.
5. Lõplik privaativõti on $sk = (seed)$.

Sõnumi signeerimine toimub järgmiselt:

1. Sõnumilühendi m loomiseks sõnum M räsitakse, ja sõnumilühend kodeeritakse kui arv $m = \overline{m_1 \dots m_n}$ alusel ℓ , kusjuures $m_i \in \{0, \dots, \ell - 1\}$.
2. Seemnest $seed$ rekonstrueeritakse $x_1, \dots, x_n$. Signatuur on $\sigma = (H^{m_1}(x_1), \dots, H^{m_n}(x_n))$.

Ühtlasi arvutatakse signeerimisprotsessi käigus sõnumilühendi kontrollsumma ning signeeritakse see koos sõnumilühendiga.

Verifitseerimine

1. Sõnumi M sõnumilühend m kodeeritakse sama moodi nagu signeerimisprotsessis – arv $m = \overline{m_1 \dots m_n}$ alusel ℓ .
2. Olgu signatuur $\sigma = (z_1, \dots, z_n)$, arvuta $y'_i = H^{\ell - m_i}(z_i)$ iga $i \in \{1, \dots, n\}$ jaoks.
3. Arvuta $pk' = H(y'_1, \dots, y'_n)$.
4. Kontrolli, et $pk' = pk$.

2.8 Postkvant-krüptograafia standardimustandi hetkeseis

On ennustamatu, millal täpselt ehitatakse piisavalt võimas kvantarvuti, mis suudab murda RSA või ECDSA taolisi tegurdamisel ja diskreetlogaritmil põhinevaid probleeme. EvolutionQ üritas seda ajaraami hinnata, küsides kvantarvutuse ekspertidelt küsimuse "*Milliseks hindate sellise kvantarvuti loomise võimalikkust,¹ mis suudaks 2048-bitise arvu tegurdada vähem kui 24 tunniga?*" [133]. Vastuste kokkuvõte on esitatud joonisel 1.

Kvantarvutite arengut käsitlev BSI uurimus [170] esitab teabe loogiliste kvantbittide ja Toffoli loogikaelementide koguse kohta, mida vajatakse arvude tegurdamiseks ja elliptikõveratel diskreetlogaritmi arvutamiseks. Need hinnangud esitame tabelis 3. Loogilised kvantbitid on abstraktses lülitusmudelil kasutatud kvantbitid. Et realiseerida ühe loogilise kvantbiti esitamiseks vajalikud veaparandusmehhanismid, võib vaja minna palju füüsilisi kvantbitte. Vajaminevate füüsiliste kvantbittide arv oleneb füüsilisest ühendatusest ja füüsiliste loogikaelementide veatasemest. Näiteks hinnatakse, et 2048-bitise RSA mooduli tegurdamiseks võidakse vajada suurusjärgus 20 miljonit füüsilist kvantbitti (lisadetailid on leitavad artiklist [75]).

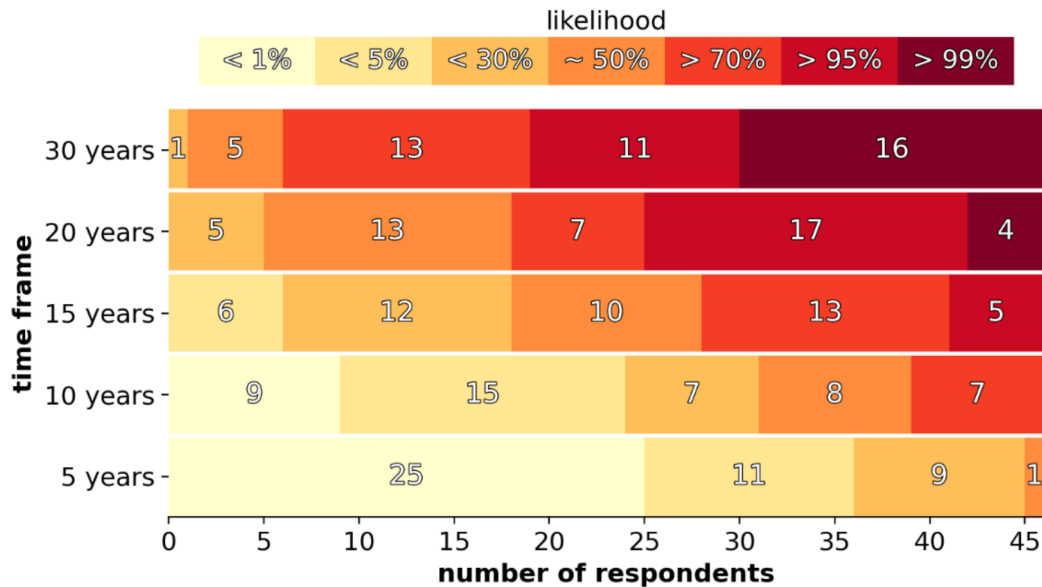
Aastal 2016 alustas USA Riiklik Tehnika ja Standardiinstituut (NIST) projektiga, mille eesmärgiks oli välja valida kvantarvutikindel võtmekapseldusmehhanism (KEM) ja kvantarvutikindel signatuuriskeem [136]. 2022.a. juulis, pärast kolme kommenteerimis- ja tagasisidevooru, kuulutas NIST standardimiseks valitud skeemid välja [130]. Need skeemid on esitatud tabelis 4.

NISTi konkursikutse nägi ette mitut turvataset, iga skeem oli varustatud neile turvatasemetele vastavate parameetritega. Turvatasemed on kokkuvõtlikult esitatud tabelis 5.

Valitud skeemide tarbeks standarditakse järgmised parameetrikomplektid:

¹statistikas eristatakse mõisteid **võimalikkus** ja **tõenäosus**

Experts' estimates of likelihood of a quantum computer able to break RSA-2048 in 24 hours



Joonis 1. 2048-bitise arvu tegurdamine vähem kui 24 tunniga [133]

Tabel 3. RSA ja elliptikõveratel põhinevate krüptosüsteemide murdmiseks vajalik kvantbittide arv

Elliptikõvera diskreetlogaritm üle algarvukorpuse $GF(p)$, kusjuures $n = \lceil \log_2(p) \rceil$		n -bitise arvu tegurdamine	
n	Loogilisi kvantbitte	n	Loogilisi kvantbitte
160	1466	1024	2050
224	2042	2048	4098
256	2330	3072	6146
384	3484	7680	15362
521	4719	15360	30722

- Kyber: Kyber-768 (turvatase 3) ning Kyber-1024 (turvatase 5) ja väikese võimsusega süsteemidele potentsiaalselt ka Kyber-512 (turvatase 1);
- Dilithium: parameetrid turvasemetele 2, 3 ja 5;
- Falcon: parameetrid turvasemetele 1 ja 5;
- Sphincs+: parameetrid turvasemetele 1, 3 ja 5, nii väike- kui kiirversioonid (lubatud räsifunktsioonid on SHA-2 ja SHAKE).

Väljavalitud skeemide standardimustandid valmivad 2023.a. jooksul ning esimesed neist peaksid saama publitseeritud aastal 2024. Koodipõhistele võtmekapseldusmehhanismidele (KEM) – klassikaline McEliece, BIKE ja HQC – peetakse täiendavalt veel ka neljas voor [169]. Lisaks sellele algatas NIST uue üleskutse, kus sihiks on üldotstarbeline signatuuriskeem, mis ei põhineks struktureeritud võrel [137].

Tabel 4. NISTi poolt standardimiseks valitud algoritmid

Nimetus	Tüüp	Perekond
Crystals-Kyber	võtmekapseldusmehhanism	võrepõhine
Crystals-Dilithium	signatuuriskeem	võrepõhine
Falcon	signatuuriskeem	võrepõhine
SPHINCS+	signatuuriskeem	räsipõhine

Tabel 5. NIST standardimiskandidaatide turvasasemed

Tase	Kirjeldus	Näide
1	128-bitise plokkšifri võtmeotsinguga võrreldav rünne	AES-128
2	256-bitise räsi(funktsiooni) kollisiooniot-singuga võrreldav rünne	SHA-256
3	192-bitise plokkšifri võtmeotsinguga võrreldav rünne	AES-192
4	384-bitise räsi(funktsiooni) kollisiooniot-singuga võrreldav rünne	SHA-384
5	256-bitise plokkšifri võtmeotsinguga võrreldav rünne	AES-256

2.8.1 Võtmekapseldusmehhanismid

Kyber Esitame lihtsustatud versiooni võtmekapseldusmehhanismist Kyber. Detailsem versioon on saadaval Kyberi dokumentatsioonis [162]. Kyber tehisenä on avaliku võtme taristule mõeldud krüpteerimisskeem, kuid teda saab Fujisaki-Okamoto (FO) tüüpi teisendusmehhanismi abil muuta võtmekapseldusmehhanismiks [73].

Kyber-i konstruktsioonis tehakse arvutusi polünoomide ringis $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$. Konstruktsioon defineerib ka mooduli astaku $k = 2, 3, 4$ (vastavalt turvasasetele 1, 3, 5). Mooduli astak määrab protokollis genereeritavate vektorite ja maatriksite suuruse. Täiendavateks parameetriteks on moodul $q = 3329$ ja ringi R_q elementide tõenäosusjaotus χ , mis on kontsentreeritud lühikestele elementidele (tuletame meelde, et ringi R_q elemente — polünoome — võime vaadata kui nende kordajatest moodustatud vektoreid).

Võtme genereerimise protsess algab polünoomidest koosneva avaliku maatriksi $\mathbf{A} \in R_q^{k \times k}$ genereerimisest. Edasi, jaotuse χ järgi valitakse juhuslikult kahe salajase polünoomide vektori $\mathbf{s}, \mathbf{e} \in R_q^k$ elemendid. Neist esimesest, $\mathbf{s}$, saab privaativõti ja teisest, $\mathbf{e}$, saab vealiige.¹ Avalik võti on $pk = (\mathbf{A}, \mathbf{t})$, kus $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ ja privaativõti on $sk = \mathbf{s}$.

Krüpteerimaks 256-bitist sõnet m (võime teda vaadelda kui polünoomi — ringi R_q elementi —, kusjuures tema kordajad kuuluvad hulka $\{0, 1\}$), on vaja juhuslikult valida kaks polünoomide vektori $\mathbf{r}, \mathbf{e}_1 \in R_q^k$ ja polünoom $e_2 \in R_q$; kõik polünoomid valitakse jaotuse χ järgi. Krüptogramm moodustub paarina $c = (c_1, c_2) \in R_q^k \times R_q$, kus $c_1 = \mathbf{A}^T \mathbf{r} + \mathbf{e}_1$ and $c_2 = \mathbf{t}^T \mathbf{r} + e_2 + \lfloor \frac{q}{2} \rfloor \cdot m$. Lõpuks tihendatakse krüptogrammi mõlemad komponendid mõne madalaima biti minemaviskamise kaudu.

¹ingl.k. *error term*

Krüptogrammi c dekrüpteerimiseks privaativõtme s abil krüptogramm hõrendatakse ning arvutatakse temast vaheväärtus $m' = c_2 - s^T c_1$. Lõpuks, sõnum m ekstrahitakse vaheväärtuse m' tihendamise teel.

Enne NISTi teadaannet standarditavate skeemide kohta olid Kyberi intellektuaalomandiõigused ebaselged, kuivõrd Kyber on kaitstud USA patendiga [56]. 2022.a. detsembris teatas NIST, et nad koos asjassepuutuvate pooltega sõlmisid litsentsilepingu [168]. Selle litsentsi kohaselt on NISTi standardit järgivad teostused maksuvabad.

Muud skeemid Klassikalist McEliece'i skeemi peetakse tugevaks kandidaadiks NISTi võtmekapseldusmehhanismile; NIST on nentunud, et nad on selle skeemi turvalisuses veendunud [130]. Siiski vajab alles otsustamist, kas spetsiifiline kasutusjuht, kuhu McEliece võtmekapseldus sobiks, üldse eksisteerib.

NIST on postkvant-krüptograafia standardi võtmekapseldusmehhanismi kandidaatidena käsitletud muidki koodipõhiseid mehhanisme – BIKE ja HQC. BIKE'i jõudlus kuulub mittevõrepõhiste kandidaatide paremikku. Paraku vaieldakse BIKE'i puhul tema kindluse osas valitud krüptotekstiga rünnete vastu, mistõttu see skeem alles vajab täiendavat analüüsi ja uurimist [130]. NIST oli rahul HQC turvaväidetega (matemaatilise tõestusega) ning turvataseme saavutamiseks vajaliku võtmepikkuse ning krüptogrammi mahuga. Kahjuks pole HQC jõudlus teiste kandidaatidega võrrelduna sugugi optimaalne [130].

NIST uuris oma konkursi neljanda vooru tarbeks elliptikõverate isogeeniaal põhinevat SIKE võtmekapseldusmeetodit. Paraku kõrvaldas avalikustatud rünne [41] SIKE'i konkursist. Kirjeldatud rünne ühetuumalisel protsessoril murrab SIKEp434 (turvatase 1) umbes tunniga [41]. Teatavaid jõupingutusi selle ründe leevendamiseks tehti, kuid pakutav lahendus toonuks kaasa võtme ja krüptogrammi pikkuse kasvu ning jõudluse vähenemise [132, 71].

NISTi standardimisprotsessi kolmanda vooru üheks alternatiivkandidaadiks oli struktureerimata võrelel põhinev FrodoKEM. See skeem annab teiste struktureeritud LWE kandidaatidega võrreldes tugevama turvagarantii, kuid tal on oluliselt väiksem jõudlus ja pikemad võtmed. Neil põhjusil jäeti FrodoKEM neljandast standardimisvoorst välja [130].

2.8.2 Signatuurskeemid

Dilithium Dilithiumi salajase võtme genereerimiseks kasutatakse polünoome ringist $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$ algarvulise mooduliga $q = 2^{23} - 2^{13} + 1$. Selle ringi juhuslikest elementidest moodustatakse üks avalik maatriks, mille mõõdud on antud parameetritega l, k , ning kaks eriotstarbelist vektorit – nn salajane komponent (vektor) pikkusega l ning nn veavektor (veaterm) pikkusega k . Nii maatriksi kui vektorite igas positsioonis paikneb seega üks polünoom ringist R_q . Võendused (*samples*) allutatakse järgmistele piirangutele: kõik võendused sooritatakse ühtlase juhusliku jaotuse põhjal; võendus sooritatakse ringis R_q ; salajase vektori ning veavektori iga polünoomi iga kordaja väärtus peab olema piisavalt väike, mistõttu need võendatakse ringi R alamhulgast $S = \{-\eta, \dots, \eta\}$, kus η on mingi väike (< 3) positiivne täisarv.

Hiljem, signeerimisel läheb vaja veel ka varjamisvektoreid, nendegi kordajad võendatakse ühtlase juhusliku jaotuse põhjal, kuid ringi R alamhulgast $S' = \{-\gamma, \dots, \gamma\}$.

Seega, Dilithiumi võtme genereerimiseks võendatakse ülaltoodud piiranguid arvestades avalik maatriks $A \in R_q^{k \times l}$, seejärel salajane vektor $s_1 \in R_q^l$ ja veavektor $s_2 \in R_q^k$. Seejärel

arvutatakse avalik vektor $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in R_q^k$. Avalik võti on $pk = (\mathbf{A}, \mathbf{t})$ ja privaatvõti on $sk = (\mathbf{s}_1, \mathbf{s}_2)$.

Signeerimiseks on kõigepealt vaja võendada varjamisvektor $\mathbf{y} \in R_q^l$, seda tehakse eelpool kirjeldatud piirangute kohaselt (võendamine ühtlane juhuslik jaotuse järgi, kus vektori elementideks olevate polünoomide kordajad on piiratud vahemikuga $S' = \{-\gamma, \dots, \gamma\}$). Nüüd saab arvutada vahevektori $\mathbf{w} = \mathbf{A}\mathbf{y}$. Vahevektor $\mathbf{w}$ on esitatav kaheosalisena $w_i = \alpha \cdot w_{iH} + w_{iL}$, kus w_{iL} märgib vektori $\mathbf{w}$ i -nda elemendi w_i (mis on polünoom) madalamajärgulisi bittide ja w_{iH} kõrgemajärgulisi. Naturaalarv α on madalamajärguliste bittide poolt moodustatud arvude ülemtõke. Tüüpiliselt valitakse ta selline, et kõrgemajärgulisi bittide oleks ca 15.

Signeeritav sõnum m esitatakse bitistringina ja räsitakse vektorit $\mathbf{w}_H$ kaasates spetsiifilise räsifunktsiooni H_0 abil. Räsifunktsioon H_0 tekitab väljundis vektori, milles tohivad esineda üksnes väärtused $\{-1, 0, 1\}$, kusjuures nullist erinevaid elemente peab olema täpselt τ . Parameetri τ väärtus valitakse tüüpiliselt $\tau \approx 60$ ning see kujutab endast sujuvat optimumi töökiiruse ja turvalisuse vahel. Räsimatehte tulemus saadakse $c = H_0(m \parallel \mathbf{w}_H)$.

Iga potentsiaalne signatuur peab läbima turvakontrolli tagamaks, et lõplik signatuur ei lekitata teavet salajase võtme kohta. Potentsiaalne signatuur arvutatakse valemiga $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$. Potentsiaalse signatuuri kontroll kätkeb endast $\mathbf{z}$ ja $\mathbf{w}$ kooskontrolle. Kontrollitakse, et

- vektori $\mathbf{z}$ elementideks olevate polünoomide kordajad ei ole liiga suured;
- vektori $\mathbf{w} - c\mathbf{s}_2$ elementideks olevate polünoomide kordajate madalamajärgulistest bittidest ei moodustu liiga suured arvud.

Kui kontrole ei läbita, siis visatakse varjamisvektor $\mathbf{y}$ ära ning uue potentsiaalse signatuuri valmendus algab $\mathbf{y}$ uuest võendusest. Kui kontrollid läbiti edukalt, siis lõplik signatuur moodustub paarist $\sigma = (\mathbf{z}, c)$.

Sõnumi m signatuuri σ kontrolliks avaliku võtmega $(\mathbf{A}, \mathbf{t})$ tuleb arvutada vahevektor $\mathbf{w}' = \mathbf{A}\mathbf{z} - c\mathbf{t}$. Sellest kõrgemaid bittide ekstraktides saadakse abivektor $\mathbf{w}'_H$. Signatuur loetakse verifitseerituks tingimusel et vektorisse $\mathbf{z}$ kuuluvate polünoomide kordajad on piisavalt väikesed ning on täidetud tingimus $c = H_0(m \parallel \mathbf{w}'_H)$.

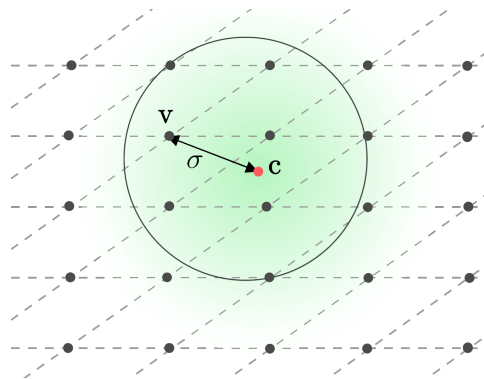
Ülalkirjeldatud on lihtsustatud variant standardimisel olevast signatuurskeemist. Täielikus variandis sisalduvad avaliku võtme tihendusalgorithmid, mis võimaldavad vektori $\mathbf{t}$ talletamisel piirduda vaid kõrgemate bittidega. Kogu skeemi detailsema kirjelduse annab [122].

Falcon Veel üks standardimiseks valitud võrepõhine signatuurskeem on Falcon. Selle struktuur erineb eelkirjeldatud Dilithiumist oluliselt.

Falconi arvutused toimuvad polünoomide ringis $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. Parameetrite konkreetsed väärtused on siin $n = 512$ või $n = 1024$ (vastab turvasemetele 1,5) ning $q = 12289$.

Võtme genereerimiseks tuleb genereerida kaks sellist polünoomide maatriksit $\mathbf{A}$ and $\mathbf{B}$, et $\mathbf{A}$ oleks pseudojuhuslik, $\mathbf{B} \cdot \mathbf{A} = 0$ ning maatriksi $\mathbf{B}$ elementideks olevatel polünoomidel oleksid väikesed kordajad. Avalik võti on $pk = \mathbf{A}$ ja privaatvõti on $sk = \mathbf{B}$.

Sõnumi m signeerimiseks tuleb arvutada selline c , et $c \cdot \mathbf{A} = H(m)$. Järgneb kõige keerukam samm, mis seisneb punktide c lähedase vektori $\mathbf{v}$ leidmises võrest, mille baasvektorid



Joonis 2. Vektori leidmine võrest

moodustavad maatriksi B . Seda sammu illustreerib joonis 2. Mõte on selles, et võre baasist B lähtudes on säärase vektori leidmine lihtne, samas kui avaliku baasi A abil selle leidmine on keeruline. See samm eeldab ujukoma-aritmeetikat, mis aga muudab turvalise teostuse keeruliseks. Signatuur $s = (c - v)$ arvutatakse punktide c ja v vahena.

Sõnumi m signatuuri s kontrollimiseks avaliku võtme A abil peab verifitseerija veenduma, et vektor s on piisavalt lühike ning et $s \cdot A = H(m)$.

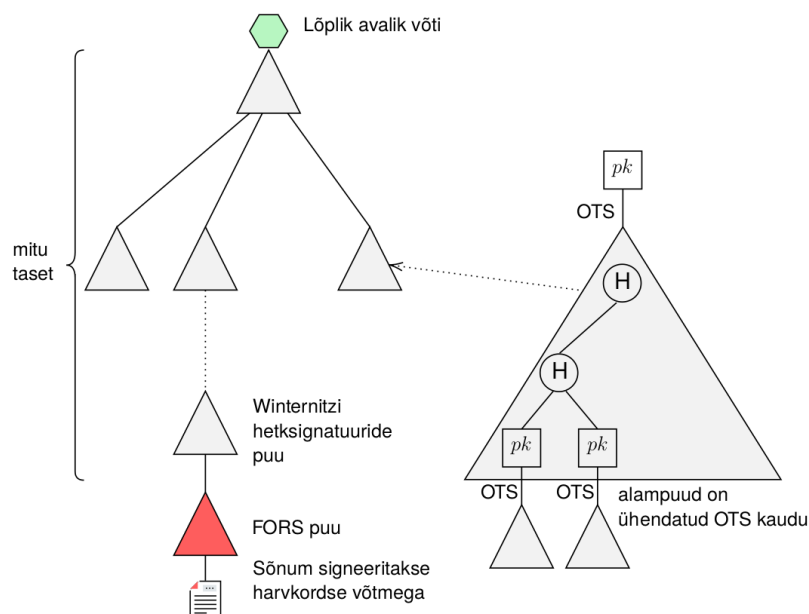
Seda signatuuriskeemi kirjeldab detailsemalt [146].

Sphincs+ Sphincs+ kasutab ühekordseid signatuure (nn hetksignatuure) (**OTS** – *one-time signature*), harvkordseid signatuure (**FTS** – *few-time signature*) ja räsipuid ning konstrueerib neist signatuuriskeemi, mis võib alamprogrammina kasutada mistahes turvalist räsifunktsiooni.

Sphincs+ signatuuride turvalisus sõltub üksnes kasutuselolevast räsifunktsioonist. Skeemil Sphincs+ on mitmeid konkreetset turvataset määratlevaid parameetreid. Igal konkreetset turvatasemel eristuvad parameetrid, mille abil saab luua kas lühemaid või, vastupidi, kiiremat signeerimist võimaldavaid signatuure.

Sphincs+'i privaativõti määratleb sertifitseerimispuu (mis omakorda koosneb alampuudest) ning avalikuks võtmeks on selle puu juur. Puu lehtedeks on kogus harvkordseid võtmeid. Kõigi harvkordsete võtmepaaride avalikud võtmed on autenditud sertifitseerimispuu abil. Sõnumi signeerimiseks valitakse välja juhuslik harvkordne võtmepaar ning sõnum signeeritakse vastava privaativõtmega. Ühtlasi lisatakse signatuurile puu juureni kulgev autentimistee. Sphincs+ kasutab Winternitzi ühekordseid signatuure (WOTS+) alampuid kokkusiduvaks signatuuriks ning juhuslike alamhulkade metsa (**FORS** – *Forest of Random Subsets*) sõnumite signeerimiseks. Joonis 3 selgitab Sphincs+ hüperpuu struktuuri. Signatuuriskeemi Sphincs+ detailse kirjelduse esitab [89].

Kuivõrd Sphincs+ signeerimiste käigus ei nõua ühegi väärtuste uuendamist, mida signeerimistevahelisel ajal meele tuleks pidada, siis liigitub ta olekuvabaks räsipõhiseks signatuuriskeemiks. On ka teistsuguseid – olekulisi – signatuuriskeeme, mis on kiiremad ja mille signatuur on lühem, kuid mis nõuavad arvepidamist sõnumite signeerimiseks juba ära kasutatud ühekordsete võtmete üle. Skeemid Leighton-Micali Signature (LMS) ja Xtended Merkle Signature Scheme (XMSS) on NIST ära standardinud ning need on kasutusvalmid [43].



Joonis 3. Signatuurskeemi Sphincs+ struktuur

Signatuurskeemide võrdlus Tabelis 6 on esitatud signatuurskeemide omaduste võrdlus turvatasemel 5. Mõõtlus pärineb jõudlustulemuste visualiseeringust [147], algoritmid on kompileeritud kõiki toetatud protsessorilaiendusi arvestades (Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz), kood on porditav.

Tabel 6. Turvatasemele 5 vastavate signatuurskeemide võrdlus

Skeem	Võtme ja signatuuri pikkus (baitides)	Tõhusus (operatsiooni sekundis)
Dilithium	avalik võti: 2 592 privaatvõti: 4 864 signatuur: 4 595	võtme genereerimine: 9991.00 signeerimine: 5224.33 signatuuri kontroll: 10612.00
Falcon	avalik võti: 1 793 privaatvõti: 13 953 signatuur: 1 280	võtme genereerimine: 39.11 signeerimine: 1403.33 signatuuri kontroll: 8671.67
Sphincs+ (kiire)	avalik võti: 64 privaatvõti: 128 signatuur: 49 856	võtme genereerimine: 226.36 signeerimine: 10.66 signatuuri kontroll: 225.77
Sphincs+ (lühike)	avalik võti: 64 privaatvõti: 128 signatuur: 29 792	võtme genereerimine: 13.82 signeerimine: 1.13 signatuuri kontroll: 438.85

Kuivõrd NIST plaanib Dilithiumi kasutada esmase signatuurialgoritmina, siis standarditakse ta esimesena [130]. Dilithiumil on kiireim võtmegenereerimine ja signeerimine. Ka on tema teostus teiste signatuurskeemidega võrreldes lihtsam.

Avaliku võtme ja signatuuri pikkuselt ning signatuuri kiire kontrollivõimaluse tõttu on Falcon sobiv valik mõningates kitsendustega stsenaariumites nagu TLS 1.3 sertifikaadid ning DNSSEC. Siiski tuleb arvestada, et Falconi võtmegenereerimise ja signeerimise protseduurid on teostuselt keerulised, sest vajavad ujukomaaritmeetikat.

Sphincs+ on standardimisele valitustest kõige konservatiivsem signatuuriskeem, kuivõrd tema turvalisus oleneb vaid tuginetava sümmeetrilise primitiivi turvalisusest. Viis, kuidas Sphincs+ on konstrueeritud, seab piirid signatuuride kogusele, mida konkreetse avaliku võtmega tohib toota. Skeemi Sphincs+ avalikud ja privaatvõtmed on lühikesed, seevastu signatuurid võrreldes teiste skeemidega väga pikad.

Muud signatuuriskeemid Signatuurskeem Rainbow [57], mis kuulub avaliku võtmega mittemuutujaliste krüptosüsteemide hulka, jõudis NIST postkvant-krüptograafia standardimiskonkursi finalistide hulka. Ta signeerimine ja signatuurikontroll on kiired, väike on ka signatuuride maht, kuid avalikud võtmed on oluliselt pikemad kui Dilithiumil and Falconil. Aastal 2022 demonstreeris Beullens praktilist rünnet [22], mis sooritab Rainbow turvatasemel 1 võtmetuletuse umbes 53 tunniga. Seetõttu eemaldati Rainbow postkvant-krüptograafiakonkursi finalistide seast.

2.8.3 Postkvant-turvaliste krüptoalgoritmide kasutuselevõtt

Peale krüptoalgoritmide standardimist on tarvis nad olemasolevates ja tulevastes infosüsteemides juurutada. Tegemist on ülimalt mittetriviaalse ülesandega, mis vajab põhjalikke ja kaugeleaatavaid plaane nii organisatsioonide, riikide kui ka rahvusvahelisel tasemel. Selle ülesande erinevaid tahke üritab loetleda NIST-i seletusartikkel [9], nimetades ligi paarkümmend tehnilist ja organisatsioonilist küsimust, millele plaani koostamisel või seda täitma hakkamisel vastata tuleks; loetelu ei ole ammendav.

2022.a. septembris avaldas NSA riikliku äriturbealgoritmi (**CNSA** - *Commercial National Security Algorithm*) komplekti (Suite) v 2.0 [138] ning tegi teatavaks ajakava versioonile 2.0 üleminekul. Komplektis on järgmised algoritmid:

- Võtmekehtestus: Kyber (parameetrid turvatasemel 5),
- Signatuurid: Dilithium (parameetrid turvatasemel 5),
- Tarkvara ja püsivara signeerimiseks: LMS ja XMSS.

Ajakava näeb ette, et tarkvara ja püsivara signeerimisel alustatakse üleminekuga kohe, teistes rakendustes 2025. ja 2027. aasta vahel (niširakendustes mitte hiljem kui 2030. aastal). Üleminek peaks olema lõpetatud 2030. ja 2033. aasta vahel, sõltuvalt rakendusvaldkonnast.

2022.a. mai kuus avaldas Saksamaa Infoturbe Liidumet (BSI) oma soovitusi sisaldava, kvantkindla krüptograafia alase juhendi [149]. BSI soovib hübriidlahendusi (vt detaile järgnevast hübriidrežiimi puudutavast arutlusest), mis kõrge turvatasemega süsteemide tarbeks kombineerivad postkvant-krüptograafiat traditsioonilisega. Võtmekapseldusmehhanismide osas otsustas BSI mitte oodata NISTi otsust ning tegi tehnilises suunises TR-02102-1 [47] ettepaneku võtta kasutusele (struktureerimata võredel põhinev) FrodoKEM ning kestussaladuste hübriidrežiimis kaitsmiseks klassikaline McEliece. Lisaks soovib BSI oma tehnilises suunises TR-02102-1 kestustoega turvasignatuuride genereerimiseks (näiteks, püsivara allkirjastamisel) räsipõhiseid olekulisi signatuurskeeme Leighton-Micali Signature (LMS) ja eXtended Merkle Signature Scheme (XMSS). BSI hinnangul sobivad üldotstarbelisteks signatuurideks Dilithium ja Falcon.

Euroopa Liidu tasemel meie teada tehnilisi juhiseid postkvant-turvalise krüptograafia juurutamiseks avaldatud ei ole. 2022.a. oktoobris avaldas Euroopa võrgu- ja infoturbeagentuur

ENISA postkvantkrüptograafia integratsiooniuringu [50], mis käsitleb proovikive postkvantkrüptograafia integreerimisel olemasolevatesse süsteemidesse ja protokollidesse. Sisu poolest võib seda pidada sarnaseks NIST-i seletusartikliga [9], kus tehnilised detailid on täpselt kirjeldatud. Uuring ei tee ettepanekut konkreetsete konstruktsioonide kasutuselevõtuks või kasutuselevõtu ajakava kehtestamiseks. Sarnaseid seletusartikleid on avaldanud ka Prantsusmaa Riiklik Infosüsteemide Turvalisuse agentuur (ANSSI) [8].

Eestiski ei ole senini avalikustatud ajakava või juhiseid postkvant-turvalistele krüptoalgoritmidele üleminekuks ning olemasolevad Eesti päritolu olukorrajeldused ja soovitusel (näiteks [167]) ei lähe kaugemale ENISA uuringus [50] avaldatust. Leiame, et tegemist on eba-meeldiva ja pisut ohtlikugi olukorraga — nii mõneski Eesti elutähtsas teenuses on keskses rollis avaliku võtmega signeerimise funktsionaalsus, mille jaoks me hetkel kasutame algoritme, mis kvantarvutite arenedes ebaturvaliseks muutuvad. Prominentseim signeerimis-funktsionaalsusest sõltuv elutähtis teenus on elektrooniline isikutuvastamine ja digitaalne allkirjastamine, mis kasutab signeerimisalgoritme realiseerituna nii tark- kui ka riistvaras. Nende väljavahetamine, eriti riistvaras, võib vajada palju aastaid, sest vajalike tegevuste seas on nii rakendusuuringud, tootearendus kui ka juurutamine.

Elektrooniline isikutuvastamine ja digitaalne allkirjastamine on kõigest üks paljudest teenustest ja infosüsteemidest, milles kasutatavad krüptoalgoritmid — signatuuriskeemid ja asümmeetrilise võtmega krüpteerimisskeemid — väljavahetamist vajavad. Tõenäoline on, et kui eri infosüsteemid omavahel suhtlevad, siis peab selline üleminek neis süsteemides toimuma samaaegselt, mõlema süsteemi kasutatavaid suhtlusprotokolle korraga uuendades.

Meenutame, et meil on olemas infosüsteemides krüptoalgoritmide väljavahetamise kogemus: näiteks aastal 2010 vahetati X-tees räsifunktsioonid MD5 ja SHA-1 räsifunktsiooni SHA-512 vastu [111, jaotis 4.3]. Ei ole aga selge, kas ja kuivõrd on toonane kogemus ülekantav praegusesse olukorda. Leiame, et Eesti kui riik, mille sõltuvus elektroonilisest identiteedist on üle keskmise, ei tohiks ootama ja lootma jääda Euroopa Liidu tasemel koostatavatele juhtnööridele, sest need võivad katta liiga vähe ja tulla liiga hilja, et olla ellu viidud tähtaegadeks, mida soovib NSA.

2.8.4 Proovikivid

Üks postkvantkrüptograafia skeemide suurimatest jõuproovidest on asjaolu, et puudub lahendus, mis oleks ühtaegu optimaalne nii sõnumipikkuse kui ka arvutuste tõhususe poolest. See tähendab, et postkvant-algoritmi tuleb valida konkreetsest kasutusstsenaariumist lähtudes. Mõnes kasutuskontekstis pole avaliku võtme pikkus kuigi oluline, teistes jällegi on, sest teda saadetakse korduvalt üle arvutivõrgu. Algoritmi sobivus tuleb tuvastada kõiki nüansse hoolikalt kaaludes.

Hübriidrežiim Kuigi NISTi standardimisprotsessi käigus on läbi viidud ulatuslik analüüs, ei pruugi postkvant-algoritmid olla eraldiseisvaks kasutuseks lõplikult küpsed. Süsteeme postkvant-lahendustele migreerides tuleb turvalisus ja võimeestik ikkagi saavutada moel, mis oleks tagasiühilduv süsteemidega, kus postkvanttugi puudub [23]. Seetõttu tuleks süsteemides kasutada postkvantalgoritme tavalistega kombineerivat hübriidrežiimi. Üldjuhul peaks võtmekapseldusmehhanism sooritama kaks võtmekehtestust – ühe postkvant-

algoritmile ja teise traditsioonilisele, ning lõpliku jaossaladuse tuletamiseks¹ need kaks saladust liitma. BSI soovitus hetkel on, et kui rakendus nõuab postkvant-turvalisust, siis kasutada võtmekapseldusmehhanismi hübriidrežiimis [149].

Signatuuride puhul pole olukord nii selge kui võtmekapseldusmehhanismide puhul. Signatuuride kombineerimiseks on mitu võimalust [23]:

- Sidurdus – kaks sõltumatut signatuuri eri skeemidest paigutatakse üksteise kõrvale;
- Nõrk pesastus – sõnum signeeritakse esimese skeemiga ning tulemus signeeritakse teise skeemiga;
- Tugev pesastus – esimese skeemiga signeeritakse sõnum, teise skeemiga signeeritakse nii sõnum kui ka signatuur esimesest skeemist;
- Pesastatud kaksiksõnum – esimese skeemiga signeeritakse esimene sõnum, teise skeemiga signeeritakse esimene sõnum, esimese skeemi signatuur ja teine sõnum. See režiim sobib X.509 sertifikaadirakendustega, kus üks sertifikaat signeeritakse esimese skeemiga, tulem manustatakse laiendina teise sertifikaati, mis omakorda on signeeritud teise skeemiga.

Siiski on veel palju lahtisi küsimusi ning on alles vaja otsustada, missuguseid meetodeid konkreetsetes kasutusmallides kasutada.

Kiipkaardid ja Smart-ID Kiipkaardid seadmetena on piiratud mälu ning arvutusvõimsusega. Need piirangud ongi peamine takistus valitud skeemide turvalisel teostamisel kiipkaartides. Mälukitsendustega seadmetele on olemas Dilithiumi ja Falconi teostused. Dilithiumiga signeerimiseks vajatakse operatiivmälu 7 KiB [30] ja Falconi puhul 40 KiB [144]. Paraku nõuab Falcon skeem ujukomaaritmeetikat, mille vaiketugi tänapäevastel kiipkaartidel puudub. Lisaks on postkvant-krüptograafia evitamisel kiipkaartidesse vaja lisada kaitse kõrvalkanalirünnete vastu. Eri tüüpi rünnete leevendamiseks on mitmeid teid, kuid need kas aeglustavad algoritmi või vajavad rohkem operatiivmälu.

Kuivõrd Smart-ID turvalisus tuleneb RSA krüptosüsteemist, siis on vaja olemasolevad primitiivid asendada turvaliste postkvant-signatuuridega. Enamgi, et jaosvõtmevõimekust säilitada, peab asendus olema lävisignatuurskeem. Üldlahendus Dilithiumi ja Falconi lävisignatuurskeemiks muutmiseks on küll olemas, kuid pole praktiliseks kasutamiseks tõhus [44]. On mitmeid lähenemisi, kuidas Dilithiumi variantidele tuginedes lävisignatuure konstrueerida [51, 176, 117], kuid need konstruktsioonid ei ühildu Dilithiumi endaga. Selle uuringu kirjutamise hetkel pole teada ühtki viisi Dilithiumi, Falconi või Sphincs+ tõhusaks lävestamiseks.

Teegid Projekt nimega Open Quantum Safe² (OQS) toetab postkvant-krüptograafia arendust ja prototüüpimist. Oma avaliku C teegiga toetab OQS postkvant-krüptograafia integreerimist protokollidesse ja rakendustesse, sh OpenSSL, BoringSSL, ja OpenSSH. Erinevates teadusprojektides leiab OQS kasutust laia valiku protokollide ja rakendusstsenaariumite testimiseks. Teek liboqs sisaldab hulka postkvant-skeeme, sh kõiki standardimiseks valitud skeeme, nagu ka alternatiivkandidaate NISTi projekti kolmandast voorust. Mõned teostused

¹<https://datatracker.ietf.org/doc/pdf/draft-stebila-tls-hybrid-design-03>

²<https://openquantumsafe.org/>

pärinevad projektist PQClean,¹ ülejäänud aga tiimide poolt NISTi projekti osana esitatud etalonteostustest ja optimeeritud teostustest.

OQS-OpenSSL-1.1.1 on OpenSSL-1.1.1 haru, mis integreerib postkvantalgoritmid protokolidesse TLS 1.3, X.509 ja S/MIME. Teeki hooldatakse aktiivselt. Lisaks puhastele postkvantalgoritmidele sisaldab teek ka hübriidrežiime võtmekapseldusmehhanismidele ja signatuuri-dele. Lisaks toetavad BoringSSL harud postkvant- ja hübriidseid võtmekehtestusmehhanis-me nagu ka avaliku võtmega autentimist postkvantvõtmete või hübriidvõtmete abil.

Tabel 7 näitab, millistes teekides on hetkel postkvant-krüptograafia tugi olemas.

Tabel 7. Kokkuvõte postkvant-krüptograafia toest olulisemates teekides.

Name	Postkvant-tugi	Märkused
OpenSSL	ei	Enne standardimisprotsessi lõppu ei lisandu OpenSSL'i ühtki postkvant-kandidaatalgoritmi.
LibreSSL	ei	
BoringSSL	ei	
GnuTLS	ei	
BouncyCastle	jah	
Tink	jah	Postkvant-krüptograafiaga eksperimenteerimise eesmärgil annab toe NISTi valikutega enamjaolt sobivale postkvant-šifrikomplektile. Tink'i praegune teostus võib erineda NISTi lõplikust spetsifikatsioonist.
libsodium	ei	
WolfCrypt	ei	
WolfSSL	jah	Dilithium, Falcon and SPHINCS+ signatuurid, Kyber KEM (hübriidrežiimis koos NIST elliptikõveratega, võimaldab FIPS-vastavust).
MbedTLS	ei	
NSS	ei	
Botan	jah	Toetab vaid üksikuid primitiive – XMSS signatuuride jaoks; MCELiece ja NewHope võtmekehtestuseks.
s2n	jah	Toetab hübriidrežiimis mitut võtmekapseldusmehhanismi – Kyber, BIKE, ja SIKE (ebaturvaline!)
Libgcrypt	ei	
Crypto++	ei	
Nettle	ei	
liboqs	jah	Hooldab OpenSSL, BoringSSL, ja OpenSSH postkvant- ja hübriidrežiimi toega harusid.

¹<https://github.com/PQClean>

3 Protokollid ja protokollistikud

3.1 TLS

TLS (*Transport Layer Security*) on valdava osa Interneti-liikluse turvamiseks kasutatav protokollistik. Eelmiste uuringutega võrreldes on kõige suurem sisuline muutus TLS versiooni 1.3 standardimine augustis 2018 [152].

3.1.1 TLS 1.3

Eelmise aruande [114] ilmutumise ajal ei olnud TLS 1.3 veel lõplikult avaldatud. Nüüd on TLS 1.3 avaldatud, selle spetsifitseerib RFC 8446 [152]. Järgnev on ülevaade TLS 1.3st.

TLS 1.3 on protokoll TLS uus, seni viimane versioon. Hoolimata minimaalsest versiooninumbrimuutusest on tegu senisest üsnagi erineva protokolliga. Protokollikirjelduse autor on TLS 1.3 projekteerimise eesmärgidena nimetanud¹:

- Koristamine: ebaturvaliste ja kasutusel mitteolevate omaduste, algoritmide jms eemaldamine;
- Parem konfidentsiaalsus: kätluse (*handshake*) osiste senisest ulatuslikum krüpteerimine;
- Parem latents: lühemate sõnumivahetustega kätlus, 2 sõnumit (1-RTT) tavalisel juhul ja 1 sõnum (0-RTT) korduval ühendumisel;
- Järjepidevus: seniste oluliste kasutusviiside jätkuv võimaldamine;
- Turvalisus: tehtavad valikud peavad olema põhjendatud turvaanalüüsiga.

Kasutajavaatest on olulisemad muudatused:

- Teadaolevalt ebaturvalised algid on keelatud. RSA võtmekehtestus, SHA-1 ja MD5 räsifunktsioonid, RC4, CBC-režiimis plokkšifrid jne on ilmutatult keelatud. Sümmeetrilist krüpteerimisalgoritmi on lubatud kasutada vaid AEAD-režiimis. RSAGA signeerimiseks kasutatakse turvalisemat signatuuriskeemi RSASSA-PSS (alates PKCS#1 v2.1 defineeritud, turvatõestustega signatuuriskeem [101, jaotis 8.1]). DSA on keelatud. Kasutaja valitud rühmad DHE võtmekehtestusel on keelatud.
- Tavakasutuses on lubatud vaid efemeersete võtmepaaridega võtmekehtestusprotokollid, mis pakuvad tulevikuturvalisust (*Perfect Forward Secrecy*). See tähendab, et sümmeetriliseks krüpteerimiseks kasutatavad seansivõtmed on iga kord uued ning poolte pikaajalise võtmematerjali (sertifikaadiga seotud salajase võtme) lekkimine ei võimalda dekrüpteerida enne leket loodud seansse.

¹<https://cs155.stanford.edu/lectures/19-rescorla.pdf>

- Lisati vastussõnumita (0-RTT) kätlusrežiim, mis võimaldab kliendil hakata kaitstud andmeid saatma enne kui server oma esimese paketi saadab või see pakett kliendini jõuab. Selliselt saadetud nn „varajased andmed“ (*early data*) on kaitstud nõrgemini kui tavalise kätluse järel vahetavad andmed ja seetõttu tuleb 0-RTT režiimi lubamisel olla ettevaatlik.¹
- Kõik kätlussõnumid, mis järgnevad serveripoolse võtmekehtestusosaku saatmisele, on krüpteeritud, sh serveri vastus laienduste kohta ja sertifikaadid. Kui varem (kuni TLS versioonini 1.2 kaasa arvatud) oli pealtkuulajal võimalik näha mõlema poole esitatud sertifikaate ja autentimissignatuure, siis versioonis 1.3 enam mitte. Protokollis versioonis 1.3 on nähtavad kliendi saadetud laiendused (sõnum *Client Hello* koos laiendustega), nt serveri nimi (SNI, *Server Name Indication*). Käib töö ka kliendi esimese sõnumi krüpteerimise võimaldamiseks, kuid 2023. aasta alguses pole see veel standardina valmis.²
- Elliptikõveratel põhinevate algoritmide kasutamine autentimiseks ja võtmekehtestuseks on nüüd põhistandardi osa. Elliptikõveratega seotud andmevahetust lihtsustati, iga toetatud kõvera jaoks on määratud oma konkreetne punktiesituse vorming³ varasema vormingute igakordse kokkuleppimise asemel.
- Kasutatava TLSi versiooni kokkuleppimine muutus, nüüd kasutatakse selleks laiendusi kätlussõnumites. TLS 1.3 kirjeldamise käigus selgus, et märkimisväärselt suur hulk Interneti-liiklust vahendavaid seadmeid (ruuterid, tulemüürid, liikluseanalüsaatorid; *middleboxes*) teevad neid läbivate pakettide kohta lubamatuid eeldusi. Seetõttu on TLS-pakettide päises oleva versiooninumbri vältimiseks kasutamine versiooni 1.3 jaoks võimatu. Lahendus: TLS 1.3 liiklus maskeeritakse TLS 1.2 seansi jätkamiseks. Selliseks maskeerimiseks saadetakse täiendavaid sõnumeid, millel ei ole TLS 1.3 jaoks tähendust.
- Erinevad seansi jätkamise mehhanismid (seansi identifikaator, seansipilet) ja eeljaotatud võtmega šifrikomplektid asendati ühe eeljaotatud võtmeid kasutava võtmekehtestusmehhanismiga. Seansi jätkamise võimaldamiseks saab server seansi käigus väljastada kui tahes palju üksteisest sõltumatuid pileteid, millest tuletatakse eeljaotatud võtme identifikaatorid. Toetatud on nii skeemid, kus olekut hoiab server kui skeemid, kus olek säilib piletis.
- Eeljaotatud saladuste mehhanism võimaldab TLS 1.3 kasutada ka ilma avaliku võtme krüptograafiata. Sellisel juhul ei kasutata efemeerseid võtmeid ja seansid ei ole tulevikuturvalised. Samas on selline TLSi variant siiski turvaanalüüsitud [58] ja nt IoT-seadmete puhul eelistatud, võrreldes omaloodud protokollidega. Väljatöötamisel on (või oli) ka vastav TLS 1.3 profiil.⁴
- Täienesid viisid, kuidas klient ja server saavad vastastikku kokku leppida, milliseid sertifikaate autentimiseks tunnistatakse. Näiteks saab kumbki pool deklareerida, milliseid signeerimisalgoritme, sertifitseerimisorganeid (*Certification Authority*, CA) ja sertifikaadilaiendusi ta teise poole sertifikaatides toetab või eeldab.

¹<https://blog.trailofbits.com/2019/03/25/what-application-developers-need-to-know-about-tls-early-data-0rtt/>

²<https://tls-wg.org/draft-ietf-tls-esni/draft-ietf-tls-esni.html>

³Elliptikõveratel põhinevates süsteemides on võtmed ja krüptogrammid punktid (täisarvudena esitatud koordinaadid) kasutataval kõveral. Nende esitamiseks baidjadadena on kasutusel olnud mitmeid erinevaid vorminguid, kusjuures erinevatele kõveratele sobivad erinevad esitused.

⁴<https://tools.ietf.org/html/draft-ietf-uta-tls13-iot-profile-05>

- Eemaldatai TLS-tasemel pakkimise tugi. Selline pakkimine võimaldas teatud tingimustel lekitada kaitstud kanalis liikunud saladusi, nt HTTP seansivõtmeid jms. Oluline on tähele panna, et samasugust lekkimist võimaldab rakendusprotokolli tasemel pakkimine ning seda ka TLS 1.3 kasutamisel – võimalusel tuleb selline pakkimine keelata või rakendada täiendavaid leevendavaid meetmeid [77].
- Kätluse kordamine seansi käigus (*renegotiation*) eemaldatai. Küll on võimalik kliendi autentimine pärast põhikätluse lõppu (*post-handshake authentication*), mis on see, milleks kätluse kordamist sageli varasemalt kasutati. Mis sellise hilisema autentimise tähendus on, sõltub rakendusprotokollist ning nt HTTP/2 jaoks on see hoopiski keelatud [18]. Samadel põhjustel on kätluse kordamine keelatud ka juhtudel, kui HTTP/2 kasutatakse koos mõne vanema TLSi versiooniga [15, jaotis 9.2.1].

3.1.2 TLS-i analüüsid ja ründed TLS-i vastu

Erinevalt varasematest versioonidest arendati TLS 1.3 tihedas koostöös arvutiteadlastega, kes olid kogenud krüptoprotokollide analüüsis. Selle koostöö käigus tekkisid detailsed protokollimudelid ja -analüüsid [36] ning edasi arenes ka krüptograafiliste protokollide analüüs kui teadusvaldkond. Seega leiti protokolliloojaka vead üles enne, kui protokoll spetsifikatsioon avaldati. Siiski on ka viimase paari aasta jooksul ilmunud teadusartikleid, mis kirjeldavad TLS-i, sealhulgas selle uusimate versioonide nõrkusi. Esitame järgnevalt ülevaate neist nõrkustest; me arvame, et see ülevaade on esinduslik. Lugejal võib olla huvitav käesolevat ülevaadet võrrelda 2016. aasta omaga [46, Ptk. 2] ning tähele panna, kuidas nõrkuste olemus on muutunud.

TLS-i protokolliloojaka ei pruugi olla alati lihtsasti mõistetav. Näiteks ei autendi TLS-i võtmekehtestusseanss loodud TCP-ühenduse otspunkte (IP-aadress ja port). Seni kuni mitmel samas domeenis pakutaval teenusel on ühine (koond)sertifikaat, on vahendusründajal võimalik suunata liiklus ühelt alamdomeenilt teisele [35]. Klient võib niimoodi saada valesti aru, millise teenusega ta suhtleb, tänu millele on võimalikud teatud (rakendus)protokollidevahelised ründed. Vastumeetmetena on võimalik kasutada SNI [61] ja ALPN [72] protokollilaiendusi. Kahjuks võivad serverid neid laiendusi ignoreerida.

Käesoleva aruande valmimise ajal on meile ainus teadaolev nõrkus TLS 1.3 protokolliloojikas nn. selfi rünne [59], mis puudutab eeljaotatud saladusi. Kui Alice'il ja Bobil on eeljaotatud saladus k ning Alice loob protokolliseansi kellegagi, kes teab saladust k , siis ta ilmselt soovib ühendust võtta Bobiga. Samuti, kui keegi loob protokolliseansi Alice'iga, kasutades eeljaotatud saladust k , siis Alice ilmselt eeldab, et ühendust soovib võtta Bob. TLS 1.3 protokolliloojikas ei ole aga takistusi sellele, et ründaja suunaks Alice'i protokolliseansialgatuse Alice'ile tagasi. Nüüd arvab Alice, et tal on kaks paralleelset seanssi Bobiga, millest ühes on ta algataja ja teises vastaja. Sarnaseid arusaamatusi võib tekkida siis, kui rohkem kui kaks osapoolt teavad ühte ja sama eeljaotatud saladust [4]. Eeldusi, mida eeljaotatud saladused peavad TLS-i turvaomaduste saavutamiseks rahuldama, kirjeldab RFC 9257 [86]. Eeljaotatud saladusi kasutades tuleb arvesse võtta selle RFC soovitusi; tehnilised standardid selle jaoks pakub välja RFC 9258 [17].

Isegi kui protokolliloojaka on vigadeta, siis nõrkused võivad ilmnedagi järgmistel sammudel — teostuses ja evitamisel. Olgugi et TLS 1.3 on lihtsam kui eelmised versioonid (vähemalt kõikvõimalike variantide ja laienduste arvu poolest), pole ta sugugi lihtne protokoll. Samuti ei saa protokolliteegid loobuda eelmiste versioonide toetamisest. TLS-i teostavate teekide, muuhulgas käesoleva aruande 6. peatükis kirjeldatute, tegelikult teostatud protokolliloojaka

süsteemilise analüüsi kohta on hiljuti taas ilmunud mõned teadusartiklid [123, 150]. Neist üks [123] on teek kombinatoorselt testinud, püüdes katta võimalikult suurt arvu kõikvõimalikest protokolliparameetritest. Teine artikkel [150] kasutab masinõpet, et tuvastada olekumasin, mille teek tegelikult realiseerib. Mõlemad artiklid raporteerivad kõrvalekalduksi TLS 1.3 spetsifikatsioonist. Muuhulgas tuvastati, et wolfSSL teegi versioonides, mis on väiksemad kui 5.2.0, olid sertifikaatide valideerimisel tehtavad kontrollid puudulikud.

Nõrkused võivad ilmned ka evitamisel. TLS-i võtmekehtestusel kasutatakse serveri autentimiseks enamasti RSA-d — server annab protokolliseansi jooksul vahetatud teadetele oma signatuuri. Allkirja arvutamine kasutab tüüpiliselt optimeerimist, kus RSA mooduli $N = pq$ järgi astendamine tehakse eraldi algarvuliste moodulite p ja q järgi (tänu millele toimuvad korrumatamised väiksemate arvudega) ning tulemused kombineeritakse hiina jäägiteoreemi abiga. Kui täpselt ühe algarvulise mooduli järgi toimuvad arvutused on vigased ning ründaja saab teada nii signeeritava teate kui ka loodud vigase signatuuri, siis suudab ründaja mooduli N tegurdada.

Kust võivad tulla sellised arvutusvead, mida ründaja kasutada suudaks? Tänapäevased dunaamilise muutmälu kiibid salvestavad bittide nii tihedalt, et kui ühte konkreetset bitti väga sageli lugeda või kirjutada, siis võib see mõjutada ka kõrvalasuvate bittide väärtust [107]. Sellel nõrkusel põhineb *Rowhammer*-rünn (või rünnetepere), mille kasutamiseks peab ründajaprotsess jooksma samal serveril rünnatavaga. Rünn toimib väga erinevates kontekstides ja on resistentne eri vastumeetmete suhtes, muuhulgas on bitivigased võimalik tekitada isegi veebilehitsejas jooksva JavaScript-koodi abil [156]. Hiljutigi on uuritud eri TLS-teekide haavatavust nii selle ründe kui ka mõnevõrra raskemini kasutatava, samuti *Rowhammer*-il põhineva ECDSA-vastase ründe suhtes [134] ning leitud, et kasutatud vastumeetmed ECDSA-vastase ründe suhtes on sageli puudulikud. Üks võimalik vastumeede, mis on RSA-signatuuride jaoks sageli juba realiseeritud — signatuuri kontrollimine enne selle tagastamist — on nüüd ECDSA-signatuuride jaoks realiseeritud wolfSSL teegi versioonis 5.5.0. Teine võimalik vastumeede lähtub TLS 1.3 disainist — erinevalt TLS 1.2-st on osa serveri poolt signeeritavast võrguliiklusest krüpteeritud ja seega ründaja jaoks raskemini leitav.

Kui mingit operatsiooni teha väga palju kordi, siis võib mõnikord sooritus olla vigane. Populaarsed veebisaidid loovad signatuure väga paljudes TLS-võtmekehtestustes. Kui mingi veebisaidi paljud kasutajad asuvad ühes ja samas alamvõrgus, siis liigub selles alamvõrgus palju signatuure, mida see veebisait on TLS-võtmekehtestustel loonud. Artikli [171] autorid salvestasid oma ülikoolide võrkudest mitme kuu jooksul ca. 2,7 miljardit serveri signatuuri (valdavalt RSA) võtmekehtestustest, mis kasutasid ülimalt TLS versiooni 1.2, ning leidsid sealt ligi 2000 vigast signatuuri. Neist signatuuridest 11 olid sellised, mis lubasid RSA mooduli tegurdada. Kõik signatuurid tulid samalt veebisaidilt ning artikli autorid arvavad, et nende loomise põhjuseks oli üks vigane riistvarakiirendi, mille töö tulemust tarkvarateek ei kontrollinud.

3.1.3 TLS protokollipinu osana

Lisaks TLSi protokollile ja kasutatavatele krüptoalgoritmidele vajavad kaitsmist ka muud ühenduse turvalisust mõjutavad aspektid.

Paljude rakendusprotokollide juures on TLS kasutusel valikuna – alustatakse mittekaitstud ühendusega ja kooskõlastuse (*negotiation*) käigus võidakse kokku leppida TLSi kasutami-

ne.¹ Kuna teadmise TLS-ühenduse võimalikkusest saab klient üle turvamata ühenduse, on võimalik madaldusrünne (*downgrade attack*), kus vahemees eemaldab kliendile saadetavatest pakettidest teadmise TLSi kasutatavuse kohta ja sunnib sedasi ka edasises ühenduses TLSi mitte kasutama (sunnib ühenduse turvaseme madalamaks, kui see oleks ilma rünnakuta). Sarnane teguviis oleks HTTP-kaudu serveeritud lehelt ümbersuunamise eemaldamine HTTPSi kasutavale lehele. Keerulisemal juhul suhtleb vahemees päris serveriga üle turvatud kanali, kuid kliendiga suhtlemiseks kasutab jätkuvalt turvamata ühendust. Ainus soovitud turvagarantiisid andev lahendus on, et klient teab algusest peale, kui peab kasutama turvatud ühendust ning omab toimivat viisi serveri autentimiseks. Sellise info edastamiseks on välja pakutud DNSi-põhiseid lahendusi (näiteks DANE ja MTA-STS²), aga ka veidi nõrgemate turvagarantiidega TOFU-stiilis (*Trust On First Use*) lahendusi (mööndustega näiteks HSTS [84]).

Järgnevalt kirjeldame lähemalt valikut mehhanisme, millega TLS ühendusi täiendavalt turvatakse.

3.1.3.1 HSTS – *HTTP Strict Transport Security*

HSTS [84] on mehhanism, mille kaudu veebiserveri omanik saab deklareerida, et tema serverit peaks kasutama vaid turvalist ühendust kasutades. See on mõeldud kaitsena madaldusrünnete vastu, aga aitab ka serveripidaja näpuvigadest põhjustatud mitteturvalise kanali kasutamise vastu.

Turvalise ühenduse kasutamise poliitika võib kehtida kas ainult täpselt sellele serverile/domeeninimele, mille poole kasutaja pöördus või ka kõigile alamdomeenidele. Koos HSTS-poliitika deklareerimisega määrab server selle kehtimise lõpu aja. Kui kasutaja brauser on HSTS-deklaratsioonist teadlik, siis asendab see kõik asjassepuutuva(te) domeeni(de) suunas tehtud mitteturvalised (`http`) päringud automaatselt turvalistega (`https`). Kui turvalise ühenduse tegemine ei õnnestu, siis annab brauser kasutajale vea ning spetsifikatsiooni kohaselt ei tohi brauser võimaldada HSTS poliitika eiramist.

Levinumad veebibrauserid sisaldavad kõik eelkonfigureeritud poliitikaid.

HSTS poliitika kasutuselevõtmisel tuleb arvestada asjaoluga, et see rakendub ka samas domeenis või selle alamdomeenis olevatele test- ja arendusserveritele. HSTS poliitika rakendamisel on mõistlik rakendada see kogu domeenile (nt mitte ainult `www.example.com`), et välistada kasutaja suunamine aadressidele, mis on küll samas domeenis (nt `vvv.example.com`) kuid HSTS-poliitikaga katmata ning kuhu seetõttu võidakse saata seansivõtmeid vms tundlikku infot sisaldavaid brauserikooke (kui nende skoobiks on määratud terve domeen). Alamdomeenidele rakenduva poliitika korral on oluline jälgida, et poliitika deklareeriks juba domeeni tipmine nimi (nt `example.com`), vajadusel peab sellisele nimele vastava veebiserveri tekitama ning muudest serveritest (nt `www.example.com`) laetud sisust tipmisele nimele viitama, et vastav poliitika kindlasti laetaks.

HSTS ei asenda mehhanisme nagu CSP³ ning brauserikookide turvalist transporti nõudvat atribuuti *secure*.

¹https://en.wikipedia.org/wiki/Opportunistic_TLS

²<https://tools.ietf.org/html/rfc8461>

³*Content Security Policy*, <https://www.w3.org/TR/CSP/>, sellega saab keelata osa sisu mitteturvalise laadimise

Olukorras, kus HTTPS on juba kasutusel ja kogu domeeni ja kõigi alamdomeenide veebiliikluse sundimine HTTPSile on võimalik, on HSTS-i kasutuselevõtmine ja HTTP kasutamisest loobumine soovitatav. HTTP kasutamisest loobumine tähendab siinkohal ka seda, et veebiserveri ressursid ei ole üldse üle HTTP kättesaadavad ning päringutele vastatakse ümbersuunamisega HTTPS peale. HSTS rakendamisel on kasulik jälgida OWASPi soovitusi.¹ Kui HSTS on korrektselt konfigureeritud ja toimib, siis tuleks turvalisuse tagamiseks esitada oma domeen lisamiseks brauseri eelkonfigureeritud nimekirjadesse.

3.1.3.2 MTA-STS – *SMTP MTA Strict Transport Security*

MTA-STS [124] on DNS-i ja HTTPS-i põhinev mehhanism, mis võimaldab DNS-domeeni omanikul deklareerida poliitikaid selle kohta, kas ja millised selle domeeni meilivahetusserveritest (MX, *mail exchanger*) on võimalised kasutama TLS-i ning mida tuleb teha juhul, kui TLS-ühenduse tekitamine mingil põhjusel ei õnnestu. DNS-is hoitakse väidet selle kohta, et domeeni kohta on olemas MTA-STS poliitika ning kehtiva poliitika versiooninumbrit. Tegelik poliitika tuleb alla laadida HTTPS-i kaudu aadressilt, mille saab MTA-STS-i rakendava domeeni nimest tuletada. Saadud poliitikat võib puhverdada, kuni DNS-is olev versiooninumber pole muutunud.

MTA-STS on meede madaldusrünnete vastu, kuid selle kasutamisel peab arvestama, et MTA-STS rakendamine ei ole saatvatele serveritele kohustuslik ning suurem osa neist seda ei teegi. Ka vastuvõttev server ei tea, millised saatjad MTA-STS-i toetavad ning nii on muu kanali kaudu edastatud infota keeruline keelduda mitteturvalistest ühendustest. Sellisena pakub MTA-STS märkimisväärselt madalamaid turvagarantiisid kui HSTS ja üldjuhul peab arvestama, et avalik e-postiliiklus ilma täiendavate meetmete rakendamiseta on kolmandatele osapooltele kergesti pealtkuulata, muudetav ja takistatav.

Sarnast probleemi lahendab DANE SMTP profiil [60]. Praktikute hinnangul on MTA-STS loodud suurte meiliteenuste poolt ning sobibki sellisena neile, samas kui DANE SMTP profiil on populaarsem väiksemate ISP-de ja meiliteenuse pakujate juures.² DANE SMTP profiili kasutamisest on juttu ka allpool, DANE kohta käiva jaotise lõpus.

Nii DANE SMTP profiili kui MTA-STS-i toetavad saatjad saavad TLS-ühenduse tekitamisel tekkinud vigadest teatada, kasutades standardiettepanekus *SMTP TLS Reporting* [125] kirjeldatud meetodeid.

3.1.3.3 DANE – *DNS-based Authentication of Named Entities*

DANE [85] on DNSSEC-i põhinev mehhanism, mille abil saab kirjeldada nõudeid TLS-serveri avalikule võtmele või avaliku võtme autentsust tõestavale sertifikaadiahelale. DANE eesmärk on vähendada TLS-ühenduste turvalisuse sõltuvust usaldatavatest kolmandatest osalistest. DANE rakendamisel kasutatakse kas täiendavat kinnistamist või väliste CA-de asendamist DNSSEC poolt pakutava usaldusmudeliga.

DANE defineerib DNS kirjetüübi TLSA. TLSA kirje seob konkreetse DNS-nime, protokoll (TCP, UDP või SCTP) ja pordiga (number) ühe või mitu objekti koos iga objekti kohta käiva

¹https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

²<https://github.com/internetstandards/toolbox-wiki/blob/master/DANE-for-SMTP-how-to.md>

metainfoga. Metainfo kirjeldab objekti tüübi (sertifikaat või avalik võti), esituse (sidumisviisi) ja kasutusviisi (st kuidas seda serveri autentimisel kasutada).

Seotav objekt on standardne sertifikaat või avalik võti.¹ Sidumise viis on kas täielik esitus või räsi. Toetatud räsialgoritmid on SHA-256 ja SHA-512. Ühe aadressi, protokoll ja pordi kombinatsiooni kohta võib leiduda mitu TLSA kirjet. Piisab, kui neist ühe määratud tingimused on täidetud. Kui DANE spetsifikatsiooni toetav klient leiab DNSSEC-mõttes turvalised TLSA-kirjed, siis on talle näidatud aadressile ja pordile TLS-ühendust algatades kohustuslik kontrollida, et serveri avalik võti ja/või lisatud sertifikaadiahel vastavad mõnes TLSA-kirjes toodud tingimusele.

DANE kirjeldab neli võimalikku kasutusviisi (nimed täiendavast RFCst 7218 [80]):

- PKIX-TA: „*CA constraint*“. Serveri sertifikaati kontrollitakse tavalisel, PKIX defineeritud [27] viisil, kasutades kliendile varem teadaolevaid ankurvõtmeid (*trust anchor*). Kirjes seotud objekt peab vastama² mõnele sertifikaadile leitud sertifitseerimisahelas. Sisuliselt on tegemist CA sertifikaadi või avaliku võtme kinnistamisega.
- PKIX-EE: „*Service certificate constraint*“. Serveri sertifikaati kontrollitakse tavalisel, PKIX defineeritud viisil, kasutades kliendile varem teadaolevaid ankurvõtmeid. Kirjes seotud objekt peab vastama ahela viimasele, serveri enda avalikku võtit sertifitseerivale sertifikaadile. Sisuliselt on tegemist serveri sertifikaadi või avaliku võtme kinnistamisega.
- DANE-TA: „*Trust anchor assertion*“. Serveri sertifikaati kontrollitakse tavalisel, PKIX defineeritud viisil, kuid ankurvõti peab vastama kirjes seotud objektile. Sel viisil saab serveri (domeeni) omanik deklareerida uue CA. Kui TLSA-kirjes on sertifikaadi või avaliku võtme räsi, peab server sertifikaadiahelas esitama ka juursertifikaadi, kuigi see tavaliselt nõutud pole.
- DANE-EE: „*Domain issued certificate*“. Serveri sertifikaat peab otse vastama kirjes seotud objektile. Erinevalt kasutusviisist PKIX-EE, ei kontrollita selle kasutusviisi korral sertifikaadiahelat. See kasutusviis võimaldab DNS tsooni (domeeni) halduril sertifitseerida avaliku võtme TLS-kasutuseks. TLS 1.3 võimaldab poolte autentimisel kasutada ka otse SPKI-struktuuri, ilma sertifikaadita – kasutusviis DANE-EE on üks võimalik viis sellise avaliku võtme sertifitseerimiseks. Kui server esitab avaliku võtme sertifikaadis, siis võib klient mõnel juhul ignoreerida sertifikaadis olevaid nimepiiranguid ja kehtivusaegu (DANE-teadliku SMTP kasutamisel isegi peab neid ignoreerima [60]).

Oportunistlikku TLSi toetavad protokollid (SMTP, IMAP, FTP jne) kasutavad TLSi samal pordil kui turvamata ühendust. See tähendab, et TLS kasutamise alustamine lepitakse kokku turvamata kanalis ning seega on võimalik madaldusrünne – ründaja saab jätta mulje, et server ei toeta TLSi. DANE algse RFC järgi ei too TLSA kirjete olemasolu üldjuhul kliendile kohustust kasutada otse turvatud ühendust või loobuda ühendusest kui ei õnnestunud TLSile üleminekus kokku leppida. Sellise nõude SMTP ja SRV-kirjete kaudu kirjeldatud teenuste jaoks defineerivad RFC 7672 [60] ja 7673 [69] – nii saab DANEi kasutada HSTS analoogina ka teiste teenuste jaoks.

¹DER-kodeeritud X.509 sertifikaat või SPKI (*Subject Public Key Information*) struktuur

²Spetsifikatsioonis kirjeldatud tähenduses

Kasutusviisid DANE-TA ja DANE-EE võimaldavad saada hakkama ilma välise CAdeta, CA rollis on DNSi tsooni (domeeni) haldaja, kes saab tekitada TLSA kirjeid. Võrreldes praeguste brauseri-CAdega on sellise mikro-CA võimalus võltsida sertifikaate (kui see osapool peaks pahatahtlik olema või vastav võti lekkima või murduma) piiratud vaadeldava tsooniga. Brauserite toetatud juur-CA-d võivad praegu väljastada sertifikaate suvalistele nimedele või ka tekitada uusi CA-sertifikaate.

DANE toimimise eelduseks on toimiv DNSSEC. DNSSECI kasutatakse järjest rohkem, kuid olukorrast, kus kõik või vähemalt enamus kasutajaid omaks võimekust DNSSEC-kirjeid valideerida, ollakse veel kaugel.¹ Sellel on mitmesuguseid põhjuseid, alustades probleemidest võrguseadmetega. Sarnaselt TLS 1.3 juurutamisel leituga – paljud olemasolevad võrguseadmed teevad kasutuselolevate protokollide osas eeldusi ning ei luba läbi või moonutavad pakette, mis neile eeldustele ei vasta. DNSSECI (ja DANE) uued, seni mitte kasutatud kirjetüübid ning signatuuride arvel senisest palju suuremad sõnumid on küll standardikohased, kuid ei tööta sellegi poolest märkimisväärselt suure osa internetikasutajate jaoks korrektselt.

Teine suur DANE kasutuselevõttu aeglustav probleem on TLSi suurima kasutuse, HTTPS-ühenduste põhikliendiks olevate brauserite loojate (majanduslikud) huvid. Brauseritootjad on ärisuhetes CAdega, mille juursertifikaate nad toetavad ning on aastate jooksul loonud mitmesuguseid lahendusi, leevendamaks CAde-põhise PKI probleeme. Nii aitab DNSikirjete võltsimise (probleem, mida DNSSEC lahendab) vastu osaliselt *DNS over TLS* ning sertifikaatide võltsimise vastu aitavad *Certificate Transparency* ja CAA-kirjed DNSis. See tähendab, et brauseritootjatel pigem pole huvi toetada lahendusi, mis soodustavad DANE levikut.

DANE-põhised lahendused teevad keerukamaks või välistavad ka legitiimse TLS-liikluse inspekteerimise antiviiuruste või tule müüride poolt.

DANE on võrdlemisi hästi toetatud SMTP jaoks.² DANE võimaldab saavutada meiliserverite vaheliste TLS-ühenduste vastastikkuse autentimise³ ning seetõttu on DANE kasutamine meililiikluse turvamiseks soovitatav.

DANE kasutamine täiendava või peamise autentimisinfo kirjeldamiseks on mõistlik, kuid sõltub kaitstavast teenusest ja selle klientidest (nende võimekusest DANE TLSA-kirjeid kätte saada ja valideerida).

3.1.4 TLS ja osapoolte tuvastamine

Oluline osa TLS kätlusest on poolte autentimine. Tavaliselt autenditakse TLS tasemel vähemalt server, teenustevaheliste ühenduste korral sageli mõlemad pooled. TLS protokollina pakub autentimiseks kas eeljaotatud saladuse kasutamist või viisi tõestamiseks, et tuvastatav pool saab kasutada deklareeritud avalikule võtmele vastavat salajast võtit. Eeljaotatud saladuse kasutamine on mõistlik vaid väga spetsiifilistes olukordades – nt seadmetes, mille jõudlus ei võimalda avaliku võtme krüptograafiat kasutada – ning on sellisena erand.

¹<https://blog.apnic.net/2020/07/10/why-has-dnssec-increased-in-some-economies-and-not-others/>

²<https://github.com/baknu/DANE-for-SMTP/wiki/4.-Adoption-statistics>

³https://www.ria.ee/sites/default/files/content-editors/kuberturve/turvaline_meilivahetus_avalikus_sektoris_2019.pdf

Avaliku võtme krüptograafia kasutamisel on autentimiseks vaja TLSi-välist viisi avaliku võtme sidumiseks vastava võtmepaari kasutaja identiteediga. Tavaliselt kasutatakse avaliku võtme kuuluvuse näitamiseks X.509 sertifikaatidel põhinevat avaliku võtme taristut (PKIX, *Public Key Infrastructure (X.509)* [27]). PKIX kõige levinuma kasutusjuhu – HTTPS serverite sertifikaatide – puhul esitab HTTPS-server sertifikaadi, mis sisaldab kokkulepitud kujul serveri DNS-nime (mille järgi klient tavaliselt pöördumist alustab) ning mille kohta server suudab näidata kliendi poolt usaldatud CAga lõppeva sertifitseerimisahela.

PKIX asemel või selle kõrval võivad pooled kasutada sertifikaatide või avalike võtmete kinnistamist (*pinning*). Sellisel juhul on poolel mingi informatsioon teise poole sertifikaadi või selles sisalduva avaliku võtme kohta. Kinnistamist kasutatakse enamasti rakenduste vahelises sides, kus osapooli on vähe ning need on eelnevalt teada. Avalikes protokollides (nt HTTPS) on probleemiks kinnistamisinfo usaldusväärne levitamine.

Võimalik probleem PKIX kasutamisel on ka võltssertifikaadid. Sertifikaatide võltsimist võimaldavad väga mitmesugused nõrkused alustades krüptograafilistest probleemidest mõne sertifikaadi või kasutatava võtmepaariga ahelas ja lõpetades organisatsiooniliste probleemidega CAdes. TLSi kontekstis on võltssertifikaadi tagajärjel võimalik vahemeherünne (MITM, *Man in the Middle*) mõne osapoole vastu. Ajalooliselt on sertifikaatide võltsimiseni viinud mõne CA võtme murdmine, pahatahtlik või lohakas (sertifikaaditaotlust mitte piisavalt kontrollinud) CA. Võltssertifikaatide probleemi leevendamiseks kasutatakse kinnistamist ja sertifikaatide läbipaistvuslogi (*Certificate Transparency*).

TLSiga seotud turvameetmena saab vaadelda ka CAde ja brauseritootjate foorumi¹ alusnõuetes kehtestatud (serveri)sertifikaatide maksimaalset lubatud kehtivusaega. Piiratud kehtivusaeg vähendab valesti väljastatud sertifikaadi ajalist mõju (kui selline sertifikaat muul viisil avastamata jääb) ning võimaldab kiiremini kehtestada uusi nõudeid sertifikaatidele. Praegu (alates 1. septembrist 2020) kehtib nõue, et avalikud CAD ei tohi väljastada sertifikaate kehtivusega rohkem kui 398 päeva. Mozilla soovib oluliselt lühemaid (kuni 90 päeva) kehtivusaegu, väites, et see soodustab sertifikaatide uuendamise automatiseerimist ja vähendab praegust, sageli veaohklikku käsitsi uuendamist. Sertifikaadi harva käsitsi uuendamine ununeb kergesti ning selle käigus tehakse sageli vigu usaldusahela konfigureerimisel jm, see kõik põhjustab käideldavusprobleeme.

3.1.4.1 CAA – *Certificate Authority Authorization*

CAA [82] on DNSi-põhine mehhanism, mis võimaldab DNS-domeeni omanikul deklareerida poliitikaid selle kohta, millistel CAdel on lubatud anda välja domeeni kohta käivaid sertifikaate või koondsertifikaate (*wildcard certificate*, *.example.com). Sellisena on see osa kaitsest võltssertifikaatide vastu.

CAA-poliitikate tarbijad on on CAD ja see kaitseviis pole mõeldud (publitseeritud) sertifikaatide lõppkasutajale. Sel põhjusel ei põhjusta CAA kasutamine ka lisatööd iga konkreetse TLS-ühenduse autentimisel. Kõik suuremate brauserite toetatud CAD peavad kohustuslikus korras toetama CAA kirjeid. Toetatud CAde deklareerimine CAA kirjetes on hea tava.

¹<https://cabforum.org/>

3.1.4.2 Sertifikaatide ja avalike võtmete kinnistamine

Kinnistamisest üldiselt ja selle mõnedest esinemisvormidest oli juttu jaotises 3.1.3.3 – DANE TLSA kirjed on sisuliselt üks viis sertifikaatide või avalike võtmete kinnistamiseks ning selle info levitamiseks. Kinnistamine vähendab CAde rolli (ainsa) usaldusallikana ning seeläbi maandab võtssertifikaatidest tulenevaid turvariske.

Sisuliselt on ka (ainult) oma CA kasutamine või autentimine eelnevalt vahetatud sertifikaatide või avalike võtmete alusel kinnistamise vorm, nagu on seda ka süsteemi või brauseri sisseehitatud CAde sertifikaadid. Tavaliselt mõeldakse kinnistamise all siiski lõppolemi sertifikaadi või mõne CA sertifikaadi või võtme kontrollimist lisaks tavalisele, süsteemseid usaldusankruid kasutavale PKIX kontrollile.

Kinnistatakse tavaliselt kas terve X.509 sertifikaat või SPKI (*subject public key info*, subjekti avaliku võtme info [27, jaotis 4.1.2.7]) struktuur selle sees. SPKI struktuur sisaldab kasutatava algoritmi identifikaatorit ning võtmematerjali – avalikku võtit ja asjasse puutuvaid parameetreid vastavalt viidatud algoritmile.

Kinnistust kasutatakse sageli rakendustevahelises suhtluses. Kui suhtlevad osapooled on eelnevalt teada ning neid ei ole palju, võib kasutatavate avalike võtmete eelnev vahetamine olla lihtsam, võrreldes kolmanda osapoole (CA) kasutamisega.

Järgnevad lõigud kirjeldavad lähemalt aspekte, mis on vaja kinnistamise juures läbi mõelda.

Sertifikaat või avalik võti? Üldjuhul on mõistlik kinnistada avalik võti, kui vastav salajane võti on kinnistaja kontrolli all, ning kinnistada sertifikaat, kui sertifikaat sisaldab kellegi teise kontrolli all oleva võtmepaari avalikku võtit.

Sertifitseerimisahelas olevat, CA kontrolli all olevat võtit võib CA meelega või kogemata kasutada teistsuguste omadustega vahesertifikaadi tekitamiseks ning vaid võtme kinnistamine aitab vältida usaldatavate osapoolte hulga ootamatut laienemist (vt ka jaotis 3.1.5.1).

Sertifikaadi kujul esitatud kinnistus on sageli kasutatav ka otse usaldusankruna, erinevalt SPKI-struktuurist. Sertifikaadi sisse saab lisada ka reegleid avaliku võtme või CA sertifikaadi abil välja antud sertifikaatide kasutusviiside jms kohta, mida PKIX valideerimisalgoritme kasutavad rakendused kasutada oskavad.

Teisalt, avaliku võtme kinnistamine võimaldab kinnistada sellise võtme, mille jaoks sertifikaati veel välja antud polegi. Näiteks võib paljude hajutatud klientidega veebiteenusel olla otstarbekas lisaks aktiivses kasutuses olevale võtmele või sertifikaadile kinnistada „varuvõti“, millele hangitakse sertifikaat siis, kui aktiivne võti mingil põhjusel kasutuskõlbmatuks muutub – salajane võti lekib või murtakse, sertifikaat aegub vms. Erakorralises olukorras võib uue kinnistuse levitamine osutuda võimalikuks, seetõttu on hea, kui seda saab levitada eelnevalt. Samamoodi – kui aegunud või mingi muu probleemiga sertifikaat asendada, siis on sageli võimalik kasutada sama võtmepaari ning kui kinnistus käib võtmepaari kohta, siis kehtib see edasi.

Terve struktuur või räsi? Kinnistuse esitamise viis sõltub peamiselt kinnistust kasutava süsteemi võimekusest (milliseid esitusi on see võimeline kinnistusena kasutama) ning sellest, kuidas kinnistust komuniqueerima peab.

Süsteemid, mis kasutavad kinnistavat sertifikaati (või ka avalikku võtit) usaldusankruna,

vajavad üldjuhul tervet struktuuri. Olukordades, kus on vajalik kinnistusinfo sage edastamine (näiteks DANE TLSA kirjetes) või algse võtmematerjali varjamine („varuvõtmed“ rakendustes, mida levitatakse olulise viivitusega, näiteks mobiilirakendused), on eelistatum räsi.

CA või lõpp-olem? Üldjuhul kehtib väide, et mida lähemal paikneb kinnistus lõppolemile, seda vähem on võtmeid ja neid kontrollivaid osapooli, mida usaldama peab. Siiski võib olla otstarbekas kinnistada CA, kui lõppolemi avalik võti muutub sageli või lõppolemeid on palju. CA sertifikaadi (või võtme) kinnistamisel peab kindlasti muul viisil (PKIX kontrollid, nimekontroll) kontrollima, kas lõppolemi sertifikaat on sobiv ja kehtib.

Üks või mitu kinnistust? Peamised kaalutlused on kinnistusi kasutava süsteemi võimekus (mõnel juhul saabki defineerida vaid ühe sertifikaadi või võtme) ning kinnistuse edastamise meetod. Kui kinnistust saab värskendada vaid sama kanalit pidi, mille turvalisust kinnistus parandab, peab kasutama mitut kinnistust. Ainult ühe kinnistuse kasutamisel on väga suur oht sellise kanali rikkumiseks ja seisundi lukustamiseks.

Kinnistuse uuendamise meetod Juhuks kui kasutusel olevate ja kinnistatud võtmete või sertifikaatidega midagi juhtub, peab leiduma meetod, kuidas kinnistust uuendada. Lihtsamal juhul võib uuendamine käia käsitsi: administraator lisab sihtsüsteemi konfiguratsiooni info uue kinnistuse kohta. Automaatse uuendamise korral peab arvestama uuenduse levimise ajaga ning hoolitsema uuenduse autentsuse eest. Kui uuendamiseks kasutatakse sama kanalit, mida kinnistus kaitseb, tuleb kogu protseduur väga hoolikalt läbi mõelda, et vältida ebaõnnestunud uuendamisest tulenevaid käideldavusprobleeme.

HPKP – HTTP Public Key Pinning HPKP [68] on standard, mis kirjeldab HTTP päise, mille abil HTTP server saab kinnistada mõne sertifikaadi oma sertifikaadi sertifitseerimisahelast. Kinnistamiseks kasutatakse avaliku võtme räsi, kinnistamine on täienduseks tavalisele PKIX valideerimisele.

Praktilises kasutuses selgus, et selline kinnistamine muudab serverid haavatavaks teenusetökestusrünnete suhtes [172] ning suuremad brauserid on HPKP toe praeguseks eemaldanud. Selle asemel soovitatakse kasutada *Certificate Transparency*’t ja sellega seotud HTTP-päist Expect-CT. HPKP kasutamine ei ole soovitatav, vajadusel tuleb sertifikaadi kinnistamiseks kasutada muid meetodeid.

3.1.4.3 *Certificate Transparency*

Certificate Transparency [118] ehk CT on mehhanism CAde väljastatud ja kasutusel olevate sertifikaatide kohta avaliku ja muutmiskindla logi pidamiseks. CT tegeleb TLS-serverites kasutatavate sertifikaatidega, mille on väljastanud avalikud, brauserite poolt vaikimisi usaldatavad CAde. RFC 6962 [118] kirjeldab kasutamise TLS-ühenduse ja serveri autentimise kontekstis, muudes kohtades – isikusertifikaadid, kliendiautentimine jm CTd ei kasutata.

CT eesmärk on vähendada väga suurt usaldusvajadust brauseritesse vaikimisi lisatud CAde vastu. CT vahenditega peetava logi abil saab tuvastada valesti väljastatud, nt sertifitseerimispoliitikale mittevastavad või domeeniomaniiku teadmata väljastatud sertifikaadid ja seega ka halvasti käituvad CAde.

Logitakse väljastatud sertifikaadid ja/või veel väljastamata eelsertifikaadid¹ koos terve sertifitseerimisahelaga lõppsertifikaadist juursertifikaadini. Sertifikaate logisse lisamiseks võib esitada igaüks, tänapäeval on eelsertifikaatide ühte või mitmesse logisse esitamine paljude CAde sertifitseerimispoliitika osa.

Vastuseks logisse lisamise päringule tagastab logipidaja sertifikaadi ajatempli (SCT, *Signed Certificate Timestamp*). CT nõuab, et TLS server esitaks kätluse käigus oma lõppolemi sertifikaadi kohta SCT ühelt või mitmelt logilt ja TLS klient kontrolliks, et vähemalt üks esitatud SCTdest on kehtiv.

Server võib SCT esitada kas laiendusena X.509 sertifikaadis, selleks loodud uues TLS laienduses (*signed_certificate_timestamp*) või TLS-sertifikaadi olekupäringu laienduses („OCSP stapling“). Kliendilt oodatakse vähemalt ühega neist kolmest meetodist edastatud SCT aktsepteerimist.

CT defineerib jälgija (*monitor*) ja audiitori (*auditor*) rollid. Jälgija ülesanne on ühest küljest jälgida, et logi lõppu kord juba kirjutatud andmeid ei eemaldataks ning teisalt võrrelda lisanud sertifikaate teadaolevate sertifitseerimispoliitikate ja sertifikaaditaotlejate kehtestatud reeglitega. Audiitor teeb pistelist logi kooskõlalisuse kontrolli ning kontrollib, et serverite esitatud SCTdele vastavad sertifikaadid tegelikult logisse jõuavad.

CT pakutav väärtus põhineb eeldustel, et a) kõik väljaantud ja kasutusel olevad sertifikaadid on avalikus logis kirjas ning b) jälgijad leiavad kahtlased sertifikaadid üles ning teavitavad neist CAde, domeeniomanikku jt. Algse lubaduse järgi võib igaüks olla jälgija ja seega otsustada ise, kas tema haldusala kohta väljastatud sertifikaadid on õiguspärased. Paraku on teada, et uute CT kirjete maht on märkimisväärselt suur – ööpäevas rohkem kui 40GB uusi andmeid [119]. Sama allikas väidab, et ükski viiest sõltumatust jälgijast, kes võimaldavad domeeniomanikel enda domeeniga seotud logitud sertifikaate otsida, ei anna täielikke vastuseid.

2021. aasta alguse seisuga on oma poliitika CT-kontrolli osas avaldanud Google Chrome² ja Apple.³ Google nõuab, et CT-ühilduvad oleks kõik EV (*Extended Validation*) sertifikaadid, mis on väljastatud pärast 1. jaanuari 2015 ja kõik teised sertifikaadid, mis on väljastatud pärast 1. maid 2018. Apple nõuab CT-ühildumist sertifikaatidelt, mis on väljastatud pärast 15. oktoobrit 2018.

Tähele peab panema ka, et CT ei tegele mitteavalike CAdega. See tähendab, et kasutaja arvutis usaldusväärseks konfigureeritud „kohalik“ CA võib välja anda sertifikaate suvaliste veebiaadresside kohta. Organisatsioonide arvutivõrkudes võib see olla soovitud käitumine (vahemeheründele ülesehitatud sisukontrolli tarkvara jaoks), aga teisalt avab sellise CA privaatsuse leke võimalusi rünneteks organisatsiooni sisevõrgu kasutajate vastu.

3.1.5 Kliendi autentimine TLS-seansis

TLS *Client Certificate Authentication* (TLS-CCA) on standardne viis kliendi autentimiseks TLS-seansis. Juhul, kui autentitakse nii server kui klient, nagu tavaliselt levinud praktika on, nimetatakse tekkivat komplekti ka kui *Mutual TLS Authentication* (mTLS).

¹<https://www.thesslstore.com/blog/ssl-precertificates/>

²https://chromium.github.io/ct-policy/ct_policy.html

³<https://support.apple.com/en-gb/HT205280>

Server võib paluda kliendilt autentimist, näidates millist kliendisertifikaati ta ootab (kriteeriumiteks võivad olla toetatud CAde eraldusnimed, toetatud algoritmid, TLS 1.3 puhul sertifikaadi laiendused ja tingimused nende väärtustele).

Klient esitab sertifikaadi ning signeerib sama, sertifikaadis näidatud võtmepaariga senised kätlussõnumid (ja sertifikaadipäringu kontekstiinfo, TLS 1.3 korral). Klient võib sertifikaadi esitamisest ka loobuda, nt kui sobivatele tingimustele vastav sertifikaat puudub. Sellisel juhul on serveri otsustada, kas seansiga saab jätkata või mitte.

TLS versioonides kuni TLS 1.2 liigub kliendi sertifikaat esmase kätluse ajal kaitsmata kujul ning on sellisena pealkuulajale nähtav. TLS 1.2 korduskätlusel ja TLS 1.3 kätluses on sertifikaat kaitstud. TLS 1.3 korral toimub sertifikaatide edastamine krüpteeritult ja kliendisertifikaat edastatakse vaid eelnevalt edukalt autenditud serverile.

Eestis levinud TLS-CCA kasutus on ID-kaardiga autentimine, aga TLS-CCAd kasutatakse ka süsteemidevahelises kommunikatsioonis, nt X-tee turvaserverite vahel.

TLS-CCA on ainuke standardne viis kliendi võtmepaari krüptograafiliseks sidumiseks TLS-seansiga.¹

Tänu kahepoolsele ja kätlusega seotud autentimisele on TLS-CCA ainus praktiliselt kasutatav viis, mille abil vältida infovahetuse krüptograafilise kaitse märkamatu nõrgenemist või kadu keskkondades, kus kasutatakse TLS liikluse inspekteerimise vahendeid (tulemüürid, antiivirused) või rakendatakse liikluse vahendamist rakenduskihis (*phishing* ründed, PSD2 *embedded* ja *decoupled* liidestused²). Mööndustega aitab sama probleemi vastu ka kinnistamine (vt jaotis 3.1.4.2), kuid ainult esimesena mainitud vahendite puhul.

3.1.5.1 TLS-CCA ja Eesti ID-kaardiga autentimine

Kuivõrd ID-kaardi kasutuse algusaegadel oli ID-kaart ja temaga tehniliselt samaväärsed digiallkirjavahendid ainsad eksisteerivad digiallkirjavahendid, läks käibelevi lihtsustatud oskusteave ID-kaardi jaoks TLS-CCA konfigureerimise kohta, mis tegeleb ainult ühe aspektiga kogu vajalikust konfiguratsioonist: usaldatud sertifitseerimisahelate määramisega.

Tänapäevaks on toimunud kaks olulist muutust:

- ID-kaartide sertifikaate väljastatakse mitmest sõltumatust sertifitseerimishierarhiast.
- ID-kaartidega samast sertifitseerimishierarhiast väljastatakse sertifikaate ka teistsugustele QSCD (*Qualified Signature Creation Device*) vahenditele, näiteks Mobiil-ID.

See tähendab, et laialt kasutusel olev konfiguratsioon,³ mille sisuks on õigesti paigaldatud sertifikaadihierarhiad, ei anna tegelikult soovitud turvaomadust – kindlust, et krüptograafilisel tasemel oleks TLS-CCA autentimine võimalik vaid ID-kaardiga.

Selleks, et saavutada korrektset konfiguratsiooni, on vaja pidada silmas järgmist.

- Sertifikaadi kuulumine kindlasse sertifikaadihierarhiasse näitab seda, et sertifikaadi sees olev informatsioon on usaldusväärne, seejuures tuleb aga silmas pidada seda,

¹Seda võimaldaks ka *Token Binding* aga selle standardimine on peatunud ja pole tõenäoline, et brauserid seda nähtavas tulevikus toetama hakkavad. Standardimist vedasid seni just brauseritootjad.

²[https://www.europeanpaymentscouncil.eu/sites/default/files/kb/file/2018-05/APIEG30-18Authenticationguidance\(SCA\).pdf](https://www.europeanpaymentscouncil.eu/sites/default/files/kb/file/2018-05/APIEG30-18Authenticationguidance(SCA).pdf)

³https://wiki.itcollege.ee/index.php/ID_kaardiga_autentimine_Apache2_veebiserveriga

et vaadata tuleb hierarhiat tervikuna, mitte ainult juursertifikaati. Sama juursertifikaadi all võib olla hierarhia kõrvalharusid, mida ei tohi etteantud otstarbeks usaldada.

- Informatsioon sertifikaadi esitaja ning lubatud kasutusotstarbe kohta on saadaval üksnes sertifikaadis endas.

3.1.5.2 Eesti ID-kaardiga autentimine Web eID abil

Web eID on üldnimetus 2021. aastal kasutusele võetud tarkvarakomponentidele, mis võimaldavad veebibrauseris kiipkaardi vahendusel autentida ja signeerida. Web eID koosneb kolmest komponendist: JavaScripti teegist, veebilehitseja laiendusest ja omarakendusest (*native application*). Lisaks Eesti riigi poolt väljastatud kiipkaartidele võimaldab Web eID kasutada ka Läti, Leedu, Soome, Horvaatia ja Belgia ID-kaarte.¹

Web eID kasutamiseks on vaja uuendada nii serveri kui kliendi tarkvara. Kui e-teenus kasutab ID-kaardi liidestamiseks varasema põlvkonna tarkvara, siis tuleb teha mõningaid muudatusi Web eID kasutuselevõtuks. Web eID integreerimiseks e-teenustesse on loodud teegid koos juhiste ja näidistega Java, .NET ning PHP platvormidele.²

Kliendi poolel on Web eID kasutamiseks vaja uuendada ID-kaardi tarkvara, mille käigus paigaldatakse Web eID brauserilaiendus ja Web eID kasutamiseks vajalik omarakendus. Praeguse raporti kirjutamise ajal omab Web eID tagasiühilduvust eelmise põlvkonna ID-kaardi tarkvaraga, mistõttu on võimalik autentida ka e-teenustesse, mis nõuavad autentimist TLS-CCA abil. Web eID arendamise ajakava näeb ette, et 2023. aastal tagasiühilduvus kaob.³

Võrreldes varasemalt kasutusel olnud tarkvaraga muudab Web eID peamiselt autentimisega seonduvat. Web eID autentimisprotokoll omab ühiseid jooni nii TLS-CCA kui WebAuthn-ga, kuid erinevalt TLS-CCA-st toimib Web eID rakenduskihis ja autentimistõend edastatakse REST API kaudu. Seetõttu ei nõua Web eID veebiserverilt TLS eriseadistust.

Sarnaselt varasemalt kasutusel olnud autentimisprotokolliga loob Web eID kasutav e-teenus juhuslikult genereeritud pretensiooni (*challenge*), mille klient peab ID-kaardil oleva autentimiseks kasutatava privaativõtmega signeerima. Erinevalt eelmise põlvkonna tarkvarast ei puhverda Web eID enam ID-kaardi PIN1, mistõttu igal autentimisel küsitakse seda uuesti ning seetõttu ei ole pärast e-teenusest väljalogimist vaja brauserit sulgeda. Teise uuendusena küsib Web eID autentimisprotsessi alguses kliendilt luba sertifikaadi edastamiseks e-teenusele. Täpsem protokollikirjeldus on toodud Web eID dokumentatsioonis [177, Joonis 5].

Web eID kasutuselevõtt muudab mõnevõrra turvagarantiisid. Varasemalt kasutusel olnud TLS-CCA erines paljudest levinud autentimisprotokollidest nagu näiteks OpenID Connect, Mobiil-ID, Smart-ID selle poolest, et TLS-CCA takistas seansi kaaperdamist ja vahendusründeid. Kuna Web eID ei kasuta TLS-CCA, siis pole Web eID TLS seanss krüptograafiliselt seotud kliendi võtmepaariga ning seetõttu tuleb kasutada teisi turvameetmeid vahendusrünnete ja seansi kaaperdamise takistamiseks. Näiteks vahendusrünnete tuvastamiseks laseb Web eID kliendil allkirjastada lisaks e-teenuse poolt

¹Riikide nimekirja pärineb raporti kirjutamise ajal <https://www.id.ee/artikkel/web-eid/> veebilehel kuvatud andmetel. Tehnilise poole pealt on olemas võimekus võtta kasutusele ka teiste Euroopa riikide ID-kaarte.

²<https://www.id.ee/artikkel/web-eid/#mida-pean-tegema>

³<https://www.id.ee/artikkel/web-eid/#ajakava>

genereeritud pretensioonile (*challenge*) ka veebisaidi allika (*origin*). Täpsem ülevaade seansi kaaperdamise ja vahendusrünnete riskist on toodud Web eID turvaanalüüsis [5], analüüsi võtab kokku ülevaade ID.ee lehel.¹

3.1.6 Soovitused

Siin jaotises anname meie soovitused TLS-i konfigureerimiseks. Need soovitused põhinevad meie teadmistel TLS-i eri võtmevahetusviiside ja andmete kodeerimise turvalisuse kohta, samuti meie soovitustel kasutatavate krüptoprimitiivide ja võtmepikkuste kohta peatükis 2.

3.1.6.1 TLS versioonid

Kasutada võib TLS versiooni 1.3 ja mööndustega ka versiooni 1.2. TLS 1.2 võib kasutada vaid juhul kui kasutatavad šifrikomplektid ja laiendused on teadaolevalt turvalised.

3.1.6.2 Kasutatavad krüptograafilised primitiivid

TLS 1.2 šifrikomplektid määravad algoritmid võtmekehtestuse, (autentimisel) signeerimise ja sümmeetrilise krüpteerimise jaoks ning fikseerivad kasutatava räsifunktsiooni.

RSA võtmekehtestuse kasutamine ei ole enam lubatud, sest seni pole õnnestunud seda turvaliseks muuta. Tõepoolest, alates sellest ajast, kui Bleichenbacher kirjeldas rünnet [26] RSA PKCS#1 v1.5 standardis [102] välja pakutud avatekstide polsterdusmehhanismi (enne nende RSA-ga krüpteerimist) vastu, pole suudetud välja pakkuda viise, mis võtaksid ründajalt praktilise võimaluse seda rünnet läbi viia. Pigem on leitud uusi võimalusi selle ründe läbiviimiseks [158], kus arvutiriistvara ja operatsioonisüsteemide poolt rakendustele pakutavate abstraktsioonide mittetäielikkus loob kanaleid, mille abil ründaja talle vajaliku informatsiooni kokku kogub. Oluline on ka, et kuna rünnakud RSA võtmekehtestuse vastu võimaldavad võltsida RSA signatuuri, siis saab ebaturvalist võtmekehtestust pakuva serveri abil rünnata ka servereid, mis kasutavad sedasama RSA võtmepaari ainult autentimiseks (nt TLS 1.3 servereid). Seepärast on võtmekehtestuseks lubatud kasutada vaid efemeersete² Diffie-Hellmani võtmepaaridega võtmekehtestust (ECDHE ja DHE).

Kui koostalitlusvõime tagamiseks on RSA võtmekehtestuse kasutamine kasvõi ajutiselt vältimatu, peab hoolitsema, et kasutataks unikaalseid, vaid selle serveriga seotud võtmeid. Võimalusel tuleb sellise serveri kasutamist piirata ka kliendi IP-aadressi järgi või vähendada võimalikku ründepinda mingi muu meetodiga. Üldisemalt ei ole soovitatav sama võtmepaari kasutamine mitmes iseseisvas TLS-serveris. Eriti ettevaatlik peab olema võtmetega, millele on olemas *wildcard*-sertifikaadid (nimedega kujul *.example.com) või mitut nime sisaldavaid sertifikaadid ning mida sellisena on mugav kasutada mitmes, sageli erineva TLS-protokolli teostuse või konfiguratsiooniga serveris.

RSA kasutamine suhtluspoolte autentimiseks (st võtmetüübina serveri või kliendi sertifikaadis) on lubatud.

Efemeersete võtmetega Diffie-Hellmani võtmekehtestuseks astendamisega jäägiklassirühmas (DHE) jaoks peab kasutama piisavalt suure järguga rühma. DHE-võtmekehtestuse korral sõltub võtmekehtestusprotokolli abil kokkulepitud peasaladuse turvalisus otseselt ja ainult

¹<https://www.id.ee/artikkel/sessiooni-kaaperdamise-runaku-ja-vahendusrunde-risk-web-eid-s/>

²Ühekordsete, lühikese elueaga. Täpsemat käsitlust vt [31, jaotis 1.5.7].

kasutatud rühma turvaomadustest ja mitte näiteks serveri autentimissertifikaadis kajastatud võtmepaari omadustest.¹ TLS 1.2 ja vanemad lubavad kasutada isegengereeritud rühmi, mille turvaomadusi on raske hinnata. TLS 1.3 lubab kasutada vaid valitud hulka, teadaolevalt turvalisi rühmi ning RFC 7919 [76] lisab nimega rühmade toe ka varasematele TLSi versioonidele. Kasutada tohib vaid rühmi, mis on piisava turvasemega vastavalt tabelile 1. Kui DHE kasutamine pole vajalik tagasiühilduvuse jaoks, soovitame seda mitte kasutada, sest elliptiköveratepõhine ECDHE on parema jõudlusega ja seda toetab suurem osa tänapäevaseid TLS-teostusi (sh värskemad brauserid).

Sümmeetrilise krüpteerimise jaoks on lubatud kasutada vaid autenditud töörežiime. Plokkšifritest võib kasutada vaid AESi, jadašifritest toob TLS 1.3 sisse ChaCha20. Lubatud šifrid/töörežiimid on AES-GCM, AES-CCM ja CHACHA20-POLY1305. AES-CCM kasutamine on lubatud, aga vaid siis, kui teised kaks pole kasutatavad.

Šifrikomplektis määratud räsifunktsiooni kasutatakse sõnumiautentimiskoodi (HMAC) koosseisus. Sümmeetriliste šifrite autenditud töörežiimides (AEAD – *Authenticated Encryption with Additional Data*) kasutatakse sõnumiautentimiskoodi vaid pseudojuhusliku jada generaatori (PRF – *Pseudorandom Function*) sisendi komponendina. Kui šifrikomplekt räsifunktsiooni eraldi ei maini, kasutatakse TLS 1.2 ja uuemate juures vaikimisi SHA-256. See on ka soovitus – kasutada SHA-256 või tugevamat.

3.1.6.3 Lubatud šifrikomplektid

TLSis kasutatavaid šifrikomplekte ja nende kahebaidiseid identifikaatoreid registreerib IANA.² Samas registris on toodud ka väli iga konkreetse šifrikomplekti soovitatavuse kohta.

Vastavalt eelmisele jaotisele on TLS 1.2 jaoks praegu lubatud vaid šifrikomplektid:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_DHE_RSA_WITH_AES_128_CCM (see ja kõik järgmised üksnes turvalise DHE rühmaga! Vaata ka ülalolevat lõiku DHE võtmekehtestuse kohta – see on lubatud kuid võimalusel peaks seda vältima)
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_CCM
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256

TLS 1.2 kasutamisel tuleb keelata kätluse kordamine juba loodud seansis (*renegotiation*).

¹<https://weakdh.org/>

²<https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4>

TLS 1.3 šifrikomplektid määravad vaid sümmeetrilise šifri ja räsifunktsiooni ning seetõttu neid TLSi eelmiste versioonidega kasutada ei saa.

Lubatud TLS 1.3 šifrikomplektid on:

- TLS_AES_128_CCM_SHA256 (vaid juhul, kui AES-GCM või ChaCha20-Poly1305 pole kasutatavad)
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

3.1.6.4 Andmete pakkimine

TLS 1.2 kasutamisel on oluline keelata ka andmete pakkimine (*compression*) TLSi kihis, sest see muudab TLSi rünnatavaks. Võimalusel tuleb keelata ka andmete pakkimine TLS-seansi sees kasutatavas rakendusprotokollis (nt HTTP), sest selline pakkimine võimaldab andmete lekkimist sarnaselt pakkimisele TLS-kihis [77]. Kui pakkimise keelamine pole võimalik, peab rakendama muid leevendavaid meetmeid nagu päringuvõltsimise (*Cross-Site Request Forgery*) vastane kaitse ja väljundi pikkuse juhuslik muutmine, et peita pakkimise tekitatud pikkuseerinevusi, mida seda tüüpi ründed kasutavad.

3.1.6.5 Autentimine Eesti ID-kaardiga

Selleks, et tuvastada, kas autentimisel kasutatud X.509 sertifikaat on ID-kaardi autentimis-sertifikaat ning usaldusväärne, tuleb lisaks sertifikaadi väljastanud CA-le kontrollida ka sertifikaadi laiendust `X509v3CertificatePolicies`, mis sisaldab sertifikaadi väljastamise aluseks olevate poliitikate identifikaatoreid (OID-sid).¹ Selle aruande kirjutamise ajal kehtivad OID-d on loetletud tabelis 8 – ainult sellised sertifikaadid, mis on väljastatud tabelis loetletud CAde poolt ning milles sisaldub vastav poliitika OID, on ID-kaardi (ja sarnaste kiipkaartide) autentimissertifikaadid.

Sertifikaadihierarhiate konfigureerimise osas soovitame pöörduda vastavate tehniliste juhendmaterjalide poole^{2,3}.

3.1.7 Muud TLS rakendamist kirjeldavad dokumendid

TLS turvalise konfigureerimise soovitusi annavad ka muud organisatsioonid. Seetõttu on kasulik neid soovitusi regulaarselt kontrollida. Viidatud veebilehti on lihtsam uuendada kui käesolevat teksti ning teiste dokumentide uuendamistsükkel võib olla käesoleva teksti omast erinev. Järgnevalt toome mõned viited materjalidele, mida jälgida.

IETF haldab heade tavade kirjeldust, sellest on väga hiljuti avaldatud uus versioon: RFC 9325 [166].

¹vt jaotis 2.2.3 dokumendis https://www.skidsolutions.eu/upload/files/SK-CPR-ESTEID2018-EN-v1_3_20220217.pdf

²<https://github.com/SK-EID/smart-id-documentation/wiki/Secure-Implementation-Guide>

³<https://www.id.ee/artikkel/id-kaardiga-isikutuvastus-veebilehel-serveriseadistused-probleemilahendused-jms-2/>

Tabel 8. ID-kaardi autentimisvõtme sertifitseerimispoliitikad

Kirjeldus	Väljastav CA	OID
Eesti kodanikele väljastatud ID-kaartide poliitika	ESTEID2018	1.3.6.1.4.1.51361.1.1.1
EU kodanikele väljastatud ID-kaartide poliitika	ESTEID2018	1.3.6.1.4.1.51361.1.1.2
Digi-ID-kaartide poliitika	ESTEID2018	1.3.6.1.4.1.51361.1.1.3
E-residentidele väljastatud ID-kaartide poliitika	ESTEID2018	1.3.6.1.4.1.51361.1.1.4
Pikajalise elamisloa ID-kaartide poliitika	ESTEID2018	1.3.6.1.4.1.51361.1.1.5
Ajutise elamisloa ID-kaartide poliitika	ESTEID2018	1.3.6.1.4.1.51361.1.1.6
EU kodanike pereliikmetele väljastatud elamisloa ID-kaardi poliitika	ESTEID2018	1.3.6.1.4.1.51361.1.1.7
Diplomaatidele väljastatud ID-kaardi poliitika	ESTEID2018	1.3.6.1.4.1.51455.1.1.1
Varasemalt väljastatud ID-kaartide poliitika. Kaarte väljastati teisest usaldusahelast ning need peaksid enamuses olema praeguseks aegunud	ESTEID-SK 2015	1.3.6.1.4.1.10015.1.1
Varasemalt väljastatud Digi-ID-kaartide ja e-residentide Digi-ID-kaartide poliitika. Kaarte väljastati teisest usaldusahelast ning need peaksid enamuses olema praeguseks aegunud	ESTEID-SK 2015	1.3.6.1.4.1.10015.1.2

OWASPi on spikker (*cheat sheet*) TLSi turvaliseks konfigureerimiseks,¹ muuhulgas leiab sealt konfiguratsiooni testimiseks mõeldud tööriistade nimekirja.

Brauseritootja Mozilla haldab soovitusi TLS serverite turvaliseks konfigureerimiseks.² Mozilla soovitusel on jagatud kolme gruppi vajaliku tagasiühilduvuse taseme järgi: moodsad brauserid/kliendid (TLS 1.3 toega, tagasiühilduvust pole vaja), keskmine/möödukas tase (soovituslik konfiguratsioon üldkasutatavale serverile) ja vana (erandlik tagasiühilduvus; kasutamiseks vaid erandkorras).

Ka NSA on avaldanud soovitusel [62] vanematest TLSi versioonidest ja nõrkadest algoritmidest loobumise kohta. Üheks oluliseks erinevuseks selle aruande ja NSA soovitusel vahel on, et meie soovitame RSA võtmekehtestusest loobuda.

Veebibrauserites kasutatava TLS ja PKI osas lepidavad brauseritootjad ja CA-d kokku

¹https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

²https://wiki.mozilla.org/Security/Server_Side_TLS

omavahelises foorumis.¹ Selle organisatsiooni kaudu kehtestatakse alusnõuded (*baseline requirements*),² mis on kohustuslikud CAdele, kes soovivad oma juursertifikaatide levitamist brauserites.

3.2 SSH

3.2.1 Ülevaade

Secure Shell (SSH) on krüptoprotokoll, mis on projekteeritud võrguteenuste turvamiseks. SSH esimene rakendus oli kaugterminal, turvamata protokollide nagu *telnet* ja *rexec* asendusena. Praktikas võib SSH-d kasutada igasuguste TCP-põhiste võrguprotokollide turvamiseks.

Tarkvarapaketi SSH esmaversiooni, mis toetas vaid protokoll SSH-1, avaldas Tatu Ylönen 1995. aastal. Protokoll SSH-1 sisaldab mitmeid vigu³ mille tulemusena ei soovitata seda enam kasutada.

IETF standardis protokoll SSH-2 2006. aastal [121]. Protokoll on krüptograafiliselt parema lahendusega kui SSH-1 ning sisaldab tuge suurema hulga kasutusjuhtude jaoks (näiteks rakenduse-ühenduste multipleksimine üle ühe TCP/SSH ühenduse⁴).

Protokoll SSH-2 krüptograafilisi omadusi määratakse kolme komponendiga: šiffer (*Encryption Algorithm*), autentsustõendi algoritm (*MAC Algorithm*) ja võtmekehtestuse algoritm (*Key Exchange Method*). Normeeritud väärtused nendele parameetritele on kirjas IANA vastavas registris,⁵ kuid realisatsioonid ei pea piirduma loetletud algoritmidega, vaid võivad lisada omi.

Seetõttu on raske anda üksikasjalikku juhist SSH šifrite konfigureerimiseks (erinevalt näiteks TLSist). On võimalik anda mõned järgimiseks sobivad suunised, kuid pärandüsteemide ühilduvusnõuded võivad seada omad nõuded.

Andmevahetuse kaitsmisest sõltumatult võidakse krüptograafilisi meetodeid kasutada ka kasutaja autentimiseks. Siinkohal on oluline tähele panna, et kord genereeritud võtmepaare kasutavad kasutajad autentimisvahendina aastaid ja seega peab võtmete genereerimisel valima piisavalt pika eeldatava elueaga võtmetüübi ja piisavalt pikaks ajaks turvalisust pakkuva võtmepikkuse. Samuti peab regulaarselt kontrollima süsteemide kaugpääsu võimaldavate võtmete turvataset ja kui see pole enam piisav, siis nõrgaks osutunud võtmed asendama või eemaldama. Lisaks võtme tüübile on oluline ka kasutatav autentimisalgoritm. Näiteks on kõik esimeses selleteemalises RFCs [121] kirjeldatud algoritmid praeguseks kasutuskõlbmatud, sest kasutavad SHA-1 räsi.

Milliseid algoritme konkreetne SSH realisatsioon toetab, tuleb selgitada vastava tarkvara juhendmaterjalidest.

¹<https://cabforum.org>

²<https://cabforum.org/baseline-requirements-documents/>

³<https://www.cvedetails.com/cve/CVE-2001-1473/>

⁴<https://en.wikibooks.org/wiki/OpenSSH/Cookbook/Multiplexing>

⁵<https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml>

3.2.2 Soovitused

SSH turvalise kasutamise kohta on oma soovitused andnud BSI [49]. Meie soovitused tuginevad neile. Samuti tuginevad need meie soovitustele krüptoprimitiivide kohta, mille andsime eelmises peatükis. Vastavad seadistused tuleks sisse viia nii SSH serveri kui kliendi konfiguratsiooni, et vältida madaldusrünnet.

- Šifrite hulgast tuleks kasutada ainult ChaCha20-Poly1305 ning AES-GCM variante, nende puudumisel AES-CTR.
- Autentimiskoodide hulgast tuleks kasutada ainult HMAC algoritme SHA-2 perekonna räsidega.
- Võtmekehtestuse algoritmide hulgast tuleks valida sellised, mis kasutavad DH või ECDH võtmekehtestust, SHA-2 või SHA-3 perekonna räsialgoritme ning selliseid Diffie-Hellmani rühmi või elliptikõveraids, mis vastavalt tabelile 1 pakuvad vähemalt 128-bitist turvataset (näiteks DH rühmad *group1* ja *group14* ei kvalifitseeru, sest nende järgu bitipikkus ei ole piisav).
- Serveri ja kasutaja (kui kasutatakse kasutaja autentimist võtmepaariga) võtmetena tuleb kasutada piisava turvatasetmega (v.t tabel 1) elliptikõveratel baseeruvaid võtmeid ja neile vastavaid autentimisalgoritme.

Internetis leidub hulgaliselt materjale¹ SSH turvalise konfigureerimise kohta, kuid nendega tutvudes tuleb alati veenduda, et need pole aegunud ning kehtiks lugejat huvitava SSH realiseerimise ja versiooni kohta.

3.3 IPsec

3.3.1 Ülevaade

IPsec (RFC 4301 [164] jt) on IETFi defineeritud turvaarhitektuur ja protokollistik, mis lisab krüptograafilise kaitse OSI-mudeli võrgukihis (vastandina nt TLSile, mis teeb seda transpordikihis). IPseci abil turvatakse näiteks asutustevahelist sidet (kahe paljudest arvutitest koosneva IP-võrgu ühendamine üle üldkasutatava sidevõrgu) aga ka kaugpääsu, mille korral ühenduvad üksikud arvutid turvaliselt kaitstud võrku või serverisse.

IPsec turvaparameetrite konfigureerimise kohta on soovitused andnud ka BSI [48] ja NIST [11].

IPsec kirjeldab:

- turvühendi (*security association*) mõiste,
- protokollid turvühendi kokkuleppimiseks ja haldamiseks ning
- protokollid turvaliseks andmevahetuseks kokkulepitud turvühendit kasutades.

IKEv2 (*Internet Key Exchange version 2*) [103] on protokoll poolte vastastikuseks autentimiseks ning turvühendite tekitamiseks ja haldamiseks. Praegune versioon 2 on edasiarendus varasematest IKE v1 ja ISAKMP (*Internet Security Association and Key Management Protocol*) protokollidest. IKE v2 pakub võrreldes IKE v1ga hulka mugavusteenuseid (laiendatava autentimise tugi, IP-rändluse tugi, seansi elutukse jms) kuid

¹Näiteks: <https://linux-audit.com/audit-and-harden-your-ssh-configuration/>

ka IKEv1-l ei ole teadaolevaid nõrkusi ning vajadusel võib ka seda kasutada turvaühendi kokkuleppimiseks. Eelistatud on siiski IKEv2 kasutamine.

IP-pakettide krüptograafilise kaitsega tegelevad protokollid AH (*Authentication Header*) [104] ja ESP (*Encapsulating Security Payload*) [105]. AH tagab tervikluse paketi lastile (*payload*) ja valitud päiseväljadele saatja identiteedi kontrolli. ESP tagab lisaks AH pakutavale ka paketi sisu konfidentsiaalsuse ja piiratud kaitse liiklusvoo analüüsi vastu. IPsec arhitektuur võimaldab AH ja ESP kasutamist ükshaaval või kombinatsioonis, konkreetne kasutus sõltub vajadusest.

Mõlemad protokollid toetavad transportrežiimi ja tunnelirežiimi.

Transportrežiimis lisatakse algsele pakatile, algse päise järele, lisapäis, mis seob paketi turvaühendi ja sisaldab (ESP korral valikuliselt) tervikluse kontrolliks vajalikke andmeid. Marsruutimiseks kasutatakse algse paketi päist ning selles olnud aadresse. Transportrežiimis ei saa tagada IP-paketi muutuvate väljade terviklust ning ESP korral on paketi päis pealtkuulajale nähtav (võimalik on liiklusvoo analüüs).

Tunnelirežiimis on kogu algne pakett uue, turvalüüside vahel saadetava paketi lastiks (*payload*) ning seda ei saa paketi edastamisel muuta. Terviklust kontrollitakse kogu (algse) paketi ulatuses. ESP korral krüpteeritakse kogu pakett – pealtkuulaja ei näe algse paketi lasti ega ka päist.

Soovitame kasutada ESP protokollitunnelirežiimis.

AH ja ESP poolt kasutavad krüptoalgoritmid, parameetrid ja võtmed määratakse turvaühendi poolt ning need lepitakse kokku IKEt kasutades. Võtmete käsitsi konfigureerimine ilma IKEt kasutamata on tehniliselt võimalik, kuid seda ei peaks lubama, kuivõrd seda on keerulisem hallata ning siis jäävad pooled ilma võtmevärskendamise võimalusest.

IKE protokollit esimese sammuna lepivad IKE pooled kokku IKE turvaühendis ja teise sammuna autendivad üksteist. Esimeses sammus tuletatakse DH-kehtestatud saladusest ja vahetatud nonssidest põhisaladus, millest tuletatakse edaspidi kõik kasutatavad võtmed. IKE turvaühendis kokkulepitud algoritme kasutatakse IKE liikluse (sh ka autentimissõnumite) kaitsmiseks ja vastastikuseks autentimiseks.

Autentimiseks tuletatakse esimestest vahetatud pakettidest ja kokkulepitud saladustest autentimissõnumid, mille kumbki pool peab signeerima. Jagatud saladuse (parooliga) autentimisel kasutatakse sõnumiautentimiskoodi, mille sisenditeks on tuletatud autentimissõnum ja paroolist tuletatud väärtus. IKEv2 jagatud saladusega autentimisprotokoll ei sisalda meetmeid parooli jõuga murdmise vastu. Kasutaja peab tagama parooli piisava keerukuse. Alternatiivselt kasutatav EAP autentimisprotokoll sisaldab meetmeid ka jõuründe vastu.

Pärast IKE turvaühendi kokkuleppimist loovad pooled vähemalt ühe alam-turvaühendi (*child security association*) konkreetsete liikluse-selektorite jaoks. Liikluse-selektor (*traffic selector*) on IP-aadresside vahemik, IP-protokoll ja pordivahemik, millele turvaühendiga määratud kaitset rakendatakse.

Olemasoleva IKE turvaühendi kontekstis saavad pooled luua uusi turvaühendeid ja juba kokkulepitud turvaühendite võtmeid värskendada (*rekey*). On šifrite töörežiime, mille korral on sama võtmega krüpteeritav andmemahut piiratud (nt AES-GCM korral on see umbes 64GB) ja vajalik on võtmete regulaarne uuendamine. Mõned IPsec teostused ei uuenda võtmeid pärast ettenähtud andmemahu täitumist. Sellisel juhul peab kasutama šifreid, millel

mahupiirangut pole või lühendama võtmevärskenduse perioodi nii, et võti uuendataks enne, kui kriitiline andmemahut täituda saab.

IPsec-i protokollistiku turvaanalüüse oleme me kirjeldanud varasemas aruandes [112], kus leidsime, et protokolliloogikas nõrkusi ei ole. Artiklites [174, 81] on analüüsitud IPsec-i teostuste Libreswan ja strongSwan mõningaid versioone (kasutades kas hägutestimist või olekumasina õppimist) ja leitud, et puudu on teatud kontrollid serverinimede või protokollisõnumipäiste parsimisel. Me arvame, et põhjalikum IPsec-i teostustest nõrkade kohtade otsimine on alles ees.

3.3.2 Soovitused

IKE on protokoll turvaühendite kokkuleppimiseks. Kõik kokkulepitavad algoritmid on nummerdatud, vastavaid registreid haldab IETF. Järgnevalt on lubatud algoritmid nimetatud IETF-i registrites näidatud nimede ja numbritega. Meie soovitused algoritmide valikuks või mittevalikuks lähtuvad jällegi meie soovitustest krüptoprimitiividele peatükis 2.

Kõik IKE abil kokkulepitavad võtmed tuletatakse pseudojuhusliku funktsiooni (*Pseudo Random Function, PRF*) abil esimeste sõnumitega kehtestatud Diffie-Hellmani saladusest ja vahetatud nonssidest. See tähendab, et turvalisus sõltub oluliselt DH parameetritest ja kehtestatud saladuse turvasemest.

3.3.2.1 DH rühmad

DH rühma abil kehtestatud saladust kasutatakse peasaladuse tuletamisel (IKE turvaühendis) ja võidakse kasutada lisasisendina alamturvaühendite loomisel.

Lubatud rühmad¹ on esitatud järgnevas loetelus. Kirjete juures sulgudes olev arv on rühma identifikaator IETF registris.

- 3072-bitine MODP rühm (15)
- 4096-bitine MODP rühm (16)
- 256-bitine juhuslik ECP rühm (19). See on IETF nimi ja identifikaator rühmale NIST P-256.
- 384-bitine juhuslik ECP rühm (20). See on IETF nimi ja identifikaator rühmale NIST P-384.
- 521-bitine juhuslik ECP rühm (21). See on IETF nimi ja identifikaator rühmale NIST P-521.
- brainpoolP256r1 (28)
- brainpoolP384r1 (29)
- brainpoolP512r1 (30)

Soovitame kasutada ECDHd koos Brainpooli elliptilise kõvera, viimasel peab olema sobiv turvatase.

Alamturvaühendite saladused (kasutatavad võtmed) tuletatakse peasaladusest ja turvaühendi loomisel vahetatud uutest nonssidest. Protokoll lubab ka (valikulist) uut DH võtmekehtestust, mille abil tekitatakse lisasaladus alamturvaühendi saladuse tuletamiseks. Kui

¹DH rühmade register: <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-8>

kasutatav IPsec teostus võimaldab sellist täiendavat võtmekehtestust, siis soovitame seda kasutada. Kasutama peaks sama või tugevamat DH rühma kui see, mida kasutati algse (IKE-)turvaühendi kokkuleppimisel.

Ilma täiendava saladuseta sõltub alamturvaühendi saladus ainult peasaladusest ning peasaladuse murdmisel on kõik alamsaladused võimalik tuletada pealtkuulatud sõnumitest – täiendav DH võtmekehtestus annab alamturvaühenditele tulevikuturvalisuse.

3.3.2.2 Pseudojuhuslik funktsioon

Pseudojuhuslikku funktsiooni kasutatakse Diffie-Hellmani protokolliga kehtestatud jagatud saladusest krüptovõtmete pärimiseks.

Lubatud funktsioonid¹ on:

- PRF_AES128_XCBC (4)
- PRF_AES128_CMAC (8)
- PRF_HMAC_SHA2_256 (5)
- PRF_HMAC_SHA2_384 (6)
- PRF_HMAC_SHA2_512 (7)

Pseudojuhusliku funktsiooni väljundi pikkus peab olema vähemalt sama pikk, kui on kasutatava šifri võti. See tähendab, et 256-bitise võtmega šifreid kasutades ei tohi kasutada AES128-põhiseid funktsioone IETF registrinumbritega 4 ja 8.

3.3.2.3 Tervikluse kontrolli algoritmid

Tervikluse kontrolli algoritme² kasutatakse IKE, AH ja ESP protokollides sõnumi tervikluse tagamiseks. Nende kasutamine IKE ja ESP korral on vajalik vaid siis, kui kasutatav šiffer ise terviklust ei taga.

Lubatud sõnumiautentimise algoritmid on:

- AUTH_AES_XCBC_96 (5)
- AUTH_HMAC_SHA2_256_128 (12)
- AUTH_HMAC_SHA2_384_192 (13)
- AUTH_HMAC_SHA2_512_256 (14)

3.3.2.4 Šifrid

Šifreid kasutatakse protokollides IKE ja ESP lasti (*payload*) krüpteerimiseks (konfidentsiaalsuse saavutamiseks). Mõnedel šifritel/töörežiimidel on tervikluse tagamise omadus.

Lubatud šifrid³ on:

¹Pseudojuhuslike funktsioonide register: <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-6>

²Tervikluse kontrolli algoritmide register: <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-7>

³Šifrite register: <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-5>

- ENCR_AES_CBC (12) (koos sellega peab kasutama eraldi tervikluse tagamise algoritmi!)
- ENCR_AES_CTR (13) (koos sellega peab kasutama eraldi tervikluse tagamise algoritmi!)
- ENCR_AES_GCM_16 (20)
- ENCR_AES_GCM_12 (19)
- ENCR_AES_CCM_16 (16)
- ENCR_AES_CCM_12 (15)
- ENCR_CHACHA20_POLY1305 (28)

Kõigi loetletud šifritega võib kasutada võtmeid pikkusega 128 ja 256 bitti, vastavalt šifri võimalustele.

3.3.2.5 Autentimisalgoritmid

Autentimisalgoritme kasutatakse protokollis IKE tõestamaks, et identiteediväidet esitav pool teab mingit jagatud saladust või deklareeritud avalikule võtmele vastavat salajast võtit.

Lubatud algoritmid¹ on:

- ECDSA kõveral secp256r1 (NIST P-256), SHA-256 (9)
- ECDSA kõveral secp384r1 (NIST P-384), SHA-384 (10)
- ECDSA kõveral secp521r1 (NIST P-521), SHA-512 (11)
- ECDSA kõveral brainpoolp256r1, SHA-256 (14)
- ECDSA kõveral brainpoolp384r1, SHA-384 (14)
- ECDSA kõveral brainpoolp521r1, SHA-512 (14)
- RSASSA-PSS, kasutades 4096-bitist RSA-võtit, SHA-384 (14)

3.3.2.6 Turvaühendi eluiga

IKE turvaühendi maksimaalne soovituslik eluiga on 24 tundi ja alamturvaühendi eluiga maksimaalselt 4 tundi [48]. Arvestama peab ka kasutatavast šifrist tingitud võimalikke andmemahupiiranguid.

3.4 WireGuard

WireGuard² on suhteliselt värske ja moodne VPN-protokoll. Alates märtsi lõpust 2020 sisaldub WireGuard Linuxi tuumas (tuuma versioon 5.6). Praeguseks leidub WireGuard teostusi muuhulgas ka Android, Windows, macOS, iOS, FreeBSD ja OpenBSD platvormide jaoks.

Kuigi ühest küljest kannab WireGuardi protokollidokumentatsioon³ märkust „*Draft revision*“, on protokoll siiski formaalselt verifitseeritud⁴ ning protokollidokumentatsioonis omadustes võib seega suhteliselt kindel olla.

¹<https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-12>

²<https://www.wireguard.com/>

³<https://www.wireguard.com/papers/wireguard.pdf>

⁴<https://www.wireguard.com/formal-verification/>

Kas WireGuardi kasutada või mitte, sõltub väga ümbritsevast keskkonnast, võrgutopoloogiast ja kättesaadavast oskusteabest – WireGuard on projekteeritud ennekõike kakspunktvõrgutopoloogias kasutamise jaoks ning keerulisemate võrkude ülesseadmine võib olla komplitseeritud. Ei tohi ka unustada, et konservatiivsemate organisatsioonide jaoks ei ole väärske eesliinitehnoloogia peaaegu kunagi hea valik.

WireGuardi krüptograafiaprimitiivide konfiguratsiooni osas ei saa selle aruande kirjutamise ajal soovitusi anda, kuna WireGuardi konfiguratsioon on fikseeritud, aga see sisaldab siiski krüptograafilisi primitiive, mida aruande muudes osades on soovitatud (X25519, ChaCha20, Poly1305). Ühtegi mittesoovitavat või keelatud primitiivi kasutuses ei ole.

Fikseeritud šifrikonfiguratsioon on WireGuardi üks suurematest nõrkustest, kuna mõne šifri murdumine ei võimalda sujuvat üleminekut, vaid tarkvara tuleb uuendada mõlemas otspunktis korraga. Suuremate installatsioonide puhul võib see olla problemaatiline.¹

¹<https://blog.ipfire.org/post/why-not-wireguard>

4 Krüptokonteinerite vormingud

Väga paljudes rakendustes ja protokollides on tarvis edastada krüptograafiliselt kaitstud infot mingis mitte-sidusas režiimis, tavaliselt kas failina või päringuparameetrina mingi suurema/üldisema protokollis sees. Kõige tuntumad näited Eestis on ilmselt laialdaselt kasutatavad ASiC-E ja CDOC konteinerid, vastavalt digitaalallkirja ning krüpteeritud sõnumite edastamiseks.

Neid vorminguid on aga rohkem, anname neist järgnevates jaotistes ülevaate koos soovitusetega.

4.1 Cryptographic Message Syntax

Cryptographic Message Syntax (CMS) [88] on ASN.1 ja PKCS#7 põhine vorming, mida kasutakse ka standardsete CAdES [42] ja PAdES [141] signatuuride alusena. Probleemiks on vähene tuntus ning ASN.1 kodeeringu keerukus koos sellest omakorda tulenevate turvaprobleemidega. Samuti on CMS vanem versioon [87] haavatav asendusrünnete (vt [88, *Abstract*] – sisuliselt tähendab see kõiki täna kasutusel olevaid realisatsioone, kuivõrd täiendus on väga uus).

CMS kasutamine rakenduste ehitusprimitiivina ei ole otseselt ebasoovitatav, kuid arendaja peaks hoolikalt jälgima, kas tal on kogu vajalik kogemus ja oskusteave olemas, et niivõrd keerulist vormingut korrektselt rakendada.

4.2 ASiC

Associated Signature Container (ASiC) on ETSI poolt standarditud [67] konteinerivorming seostamiseks andmefailide signatuuride või ajatemplitega. ASiC oma erinevates variatsioonides ei ole midagi muud kui etteantud struktuuriga ZIP-fail. ASiC standard ise krüptograafilisi omadusi ei sätesta. ASiC standard sisaldab lihtsutatud ASiC-S konteinerivormingut ja laiendatud ASiC-E vormingut. Mõlemad konteinerivormingud võivad sisaldada kas XML (XAdES) või CMS (CAdES) vormingus signatuure või ajatempli tokenit. Lisaks, võib ASiC-E konteiner sisaldada ka tõenduskirjeid.

4.3 ASiC-S

Associated Signature Container Simple (ASiC-S) on krüptokonteiner, mis sisaldab ühte allkirjastatavat andmefaili ja ühte signatuurifaili (mis võib sisaldada mitut signatuuri) XAdES või CAdES formaadis või ajatempli RFC 3161 (õigemini selle uuenduse RFC 5816 [161]) formaadis. Ajatempleid sisaldavaid konteinereid käsitleme põhjalikumalt jaotises 4.6. Allkirjastatav andmefail võib olla ka muuhulgas ZIP-fail, mis sisaldab mitut faili.

ASiC-S krüptokonteiner on võimekuselt alamhulk ASiC-E krüptokonteinerist. ASiC-S tugi on lihtsamini teostatav, see formaat on sobilik dokumentide digiallkirjastamiseks. Kahjuks aga ei ole ASiC-S laialt levinud. Eesti kontekstis ei suuda me ka soovitada kvaliteetset teeki ASiC-S konteinerite loomiseks ja verifitseerimiseks.

4.4 ASiC-E

Associated Signature Container Extended (ASiC-E) krüptokonteiner sisaldab ühte või mitut allkirjastatavat andmefaili ja CAdES või XAdES formaadis signatuure. ASiC-E konteiner võib sisaldada ka ajatempli tokenit RFC 5816 [161] formaadis ja RFC 4998 [78] või RFC 6283 [25] formaadis tõenduskirjet.

4.4.1 ASiC-E koos XAdES signatuuriga

Eestis on ilmselt kõige levinumaks krüptokonteineriks ASiC-E koos XAdES (*XML Advanced Electronic Signatures*) signatuuriga, mille baasprofiili annavad ETSI EN 319 162-1 [67] ja ETSI EN 319 132-1 [65] ning mis on täpsustatud Eesti kohaliku BDOC spetsifikatsiooniga [13].

Eestis kasutatakse seda vormingut kõigi igapäevaste digiallkirjade korral.

Kuivõrd tegu on tehniliselt keeruka vorminguga, on oluline ASiC-E digiallkirjadega töötamisel kasutada mõnd head teeki valmisolevate hulgast,¹ vastavalt oma platvormile.

4.4.2 ASiC-E koos CAdES signatuuriga

ASiC-E koos CAdES signatuuriga on vastava XAdES vorminguga analoogne, kuid XML asemel kasutatakse seal signatuuri tehnilise kandjana vormingut CMS. ETSI EN 319 162-1 [67] annab baasprofiili ka ASiC-E konteinerile koos CAdES (*CMS Advanced Electronic Signatures*) signatuurivorminguga. Signatuurivormingut kirjeldab ETSI EN 319 122-1 [64].

Eestis ei ole CAdES laialdaselt kasutusel, kuid nõue teistes Euroopa Liidu riikides kasutatavate digisignatuuri-vormingute aktsepteerimiseks tähendavad, et ka Eesti infosüsteemid peavad olema valmis sellised allkirju vähemalt valideerima, kasutades heakvaliteedilist valmisteeki.

4.5 PAdES

PAdES (*PDF Advanced Electronic Signatures*) on laiendus PDF vorming, mis võimaldab lisada PDF-failile täiustatud digisignatuure. PAdES on standarditud ETSI poolt standardis EN 319 142-1 [66]. PAdES vorming koosneb CAdES vormingus signatuurist PDF konteineris.

PDF dokumente saab allkirjastada PAdES vormingus näiteks LibreOffice ja Adobe Acrobat tarkvaraga. PAdES signatuuride valideerimiseks on ETSI loonud veebirakenduse.²

PAdES vormingul on olnud turvaprobleeme seoses spetsifikatsiooni mitmetimõistetavusega,³ seetõttu tuleb veenduda, et PAdES-allkirjaga dokumentide verifitseerimiseks ja lugemiseks oleks kasutusel teenused, teegid ja PDF-vaaturid, milles on kõik teadaolevad turva-

¹<https://github.com/open-eid/>

²<https://signatures-conformance-checker.etsi.org/pub/index.php>

³<https://www.pdf-insecurity.org/>

vead parandatud. Vastasel korral on võimalik meelitada kasutaja aktsepteerima sellist kuvatud dokumendi ja digiallkirja kombinatsiooni, mida pole tegelikult allkirjastatud. Näiteks on võimalik, et allkirjastatud on ainult üks osa dokumendist, aga kasutajale jääb mulje, et signatuuri all on terve dokument.

4.6 ASiC ja ajatemplid

ASiC-S CAdES, ASiC-S XAdES ja ASiC-E CAdES krüptokonteinerid võivad sisaldada RFC 3161 uuenduses RFC 5816 [161] spetsifitseeritud formaadile vastavat ajatempli tookenit. RFC 3161 ajatemplid põhinevad avaliku võtme krüptograafial ja on signeeritud usaldatud kolmanda osapoole ehk kvalifitseeritud usaldusteenuse pakkuja poolt.

RFC 3161 ajatempli taotluse loomiseks, verifitseerimiseks ja signeerimiseks on olemas funktsioonid OpenSSL teegis. Krüptoteegid Tink ja BouncyCastle sisaldavad ajatempli taotluse loomiseks vajalikke krüptoprimitiive, ent taotluse vormindamiseks funktsioon puudub.

EL määruse 910/2014 (eIDAS) [151] järgi kvalifitseeritud usaldusteenuse pakkujad Eestis¹ on GuardTime OÜ ja SK ID Solutions AS.

4.7 JAdES

JAdES on JSON formaadis AdES signatuur, mis põhineb JSON Web Signature vormingul. JAdES vorming on kirjeldatud standardis ETSI TS 119 182-1 [63].

JAdES täiendab JWS vormingut täiendavate signeeritud ja signeerimata atribuutidega. JAdES signeeritud atribuutide hulgas on signeerimise aeg, sertifikaadi lühend, signeeritavate andmete ajatempel ja signeerija atribuutid. Need täiendavad atribuudid kaitsevad mitmete rünnete nagu näiteks sertifikaadiasendusründe eest [45].

JAdES signatuuri standard defineerib neli signatuuride taset, mis määravad, millised atribuudid peavad signatuuris kaasatud olema. Tase B-B on baastase, B-T nõuab ajatempli olemasolu, tasemed B-LT ja B-LTA nõuavad atribuute allkirja pikaajaliseks valideerimiseks.

JAdES signatuure saab luua Digital Signature Service teegiga.²

CB-AdES

Töös on mustand standardiseerimaks CBOR (Concise Binary Object Representation) [29] formaadis AdES signatuurid. CBOR on JSON-i andmemudelil põhinev atribuudi ja väärtuse paaridest koosnev formaat kahendkodeeritud andmete edastamiseks.

4.8 CDOC

CDOC on *XML Encryption* [92] baasil koostatud Eesti kohalik vorming krüpteeritud andmete edastamiseks. Krüpteerimiseks kasutatakse avalikule võtme abil krüpteerimist (RSA PKCS#1 v1.5, *ECC Integrated Encryption Scheme* (IES)). Selliselt krüpteeritud sõnumite dekrüpteerimiseks vajalikke liideseid pakuvad vaid ID-kaart ja krüptopulgad (Mobiil-ID ja Smart-ID ei ole toetatud).

¹<https://esignature.ec.europa.eu/efda/tl-browser/#/screen/tl/EE>

²<https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Digital+Signature+Service+++DSS>

CDOC vorming ei paku sõnumi autori autentsust. Kui autentsuse tuvastamine on vajalik, tuleb edastatavad materjalid kas enne või pärast krüpteerimist digitaalselt allkirjastada. Samal ajal tuleb tähelepanu pöörata sellele, et selline kaheastmeline krüptograafiliste meetodite rakendamine ei pruugi anda soovitud kaitset: ründajale jääb tihtipeale võimalus tulemust manipuleerida [52, p. 4.3] [108]. Täpseid vastumeetmeid on keeruline ette kirjutada, kuivõrd need sõltuvad konkreetsest rakendusest kus CDOCi kasutatakse, kuid üldkasutatav soovitus on allkirjastada enne krüpteerimist ning hoolitseda selle eest, et allkirjastatud sisu hulgas oleks ka see info, kellelt kellele see sõnum (dokument) on mõeldud liikuma [7].

Oluline on märkida, et CDOC vorming ei paku tulevikuturvalisust, mistõttu ei tohi seda kasutada pika salastusajaga materjalide krüpteerimiseks. Hoiatav näide sellest, et tulevikuturvalisuse puudumine on probleem, oli 2017. aasta ROCA intsidendi kriis, mille käigus selgus, et ID-kaartide RSA privaatvõtmed on murtavad. Muuhulgas tähendab see ka seda, et sellistele võtmetele saadetud CDOC konteinerid on piisavalt võimekate ründajate poolt avatavad [114].

CDOC 2.0

CDOC uue versiooni 2.0 väljatöötlust alustati Riigi Infosüsteemi Ameti projekti „CDOC2.0 analüüs (info pikaajalise salastamise lahenduse arhitektuur)“ käigus ning need tööd jätkuvad [140].

Lisaks CDOC-ist pärinevale töörežiimile, kus salastamiseks kasutatakse vastuvõtja pikaajalist asümmeetrilist võtit, pakub CDOC 2.0 uudse lähenemisena võtmeedastusserveri kasutamist. Selline lähenemine eraldab transpordil ära sümmeetriliselt krüpteeritud sõnumi (mis liigub otse vastuvõtjale näiteks e-posti vahendusel) ning võtmekapsli, mis saadetakse läbi serveri. See tähendab, et osapool, kes jälgib otsekanalit, ei saa ligipääsu asümmeetrilisele krüptogrammile, millega põhisõnumi krüpteerimiseks kasutatud sümmeetrilist võtit kaitstakse. See omakorda tähendab, et isegi kasutatava asümmeetrilise krüptograafia murdumisel säilib sellise ründaja vastu sõnumi salajasus.

Loomulikult tuleb serverile ligipääsu eraldi turvata, nii näiteks saab võtmekapsli allalaadimist autentida Eestis kasutusel olevate eID vahenditega. Hetkel realiseeritav versioon eeldab ka dekrüpteerimisvõtme haldamiseks riistvaralist vahendit (ID-kaarti või krüptopulka), kuid võtmeserveri arhitektuur võimaldab kasutada ka teisi lahendusi. Näiteks on tehniliselt võimalik jagada dekrüpteerimisvõti ühissalastuse abil mitmeks osaks ning levitada võtmeosakuid erinevate serverite kaudu. Sel juhul piisab dekrüpteerimiseks vaid autentimisest, mida saab realiseerida ka SmartID või mobiil-ID baasil.

Teine vajadus, millega CDOC 2.0 pikemas perspektiivis arvestab, on üleminek postkvant-krüptoalgoritmidele. Hetkel pole veel turul riistvaralisi postkvant-võtmeseadmeid, aga kui need ilmuvad, on nende lisamine CDOC 2.0 spetsifikatsiooni lihtne. Samuti on võimalik postkvant-krüptoalgoritmide võtmeid ühissalastuse abil tükkide kaupa erinevate võtmeedastusserverite kaudu levitada.

CDOC 2.0 kasutab võrreldes oma eelkäijaga ka teistsuguseid andmevorminguid. Edastatavad failid arhiveeritakse, kasutades POSIX tar vormingut [173]. Arhiveeritud failid pakitakse meetodiga ZLIB, mis on standarditud kui IETF RFC 1950 [54]. Metainfoga varustatud konteiner jadastatakse lõpuks FlatBuffers¹ vormingu abil.

¹<https://google.github.io/flatbuffers/>

4.9 JSON Object Signing and Encryption

JSON Object Signing and Encryption (JOSE) [12] on üldnimetaja JSON-põhiste krüptograafiliste sõnumivormingute perekonnale¹ - JWS, JWE ja JWT.

Kuigi JOSE vorminguperekond on moodne ja lihtsasti kasutatav, on tema suureks puuduseks (JSONist pärinev) skeemikirjeldusvahendite puudumine, mis sunnib igal platvormil programmeerijat väga hoolikalt läbi mõtlema, kuidas JOSE struktuure turvaliselt parsida ning töödelda, samuti tekib probleeme ühilduvusega, kui on vaja vahetada JOSE struktuure erinevate platvormide vahel. On teada, et olemasolevad teegid ei taga kõiki vajalikke turvakontrolle, võimaldades näiteks asendusründeid konteinerite metaandmete vastu. Teiseks puuduseks on JOSE standardiga määratud krüptoalgoritmide suletud nimekiri, mis ei võimalda JOSE struktuure kasutada standardiga määratud registris² loetlemata algoritmidega (näiteks lävikrüptograafiat kasutades). Määratud krüptoalgoritmide hulgas on ka valikud kasutada mitte mingit krüpteerimist või valitud krüptogrammiga rünnaku suhtes ebaturvalist RSA PKCS#1 v1.5 [26].

JOSE kasutamine (väljaspool avalikke standardeid, mis seda ette kirjutavad) ei ole otseselt ebasoovitav, kuid tuleb arvestada, et tegu on keerulise ja kõrge riskiastmega vorminguga, mille korrektseks rakendamiseks on vaja kaasata arendustegevusse turvainsener, kes koostab konkreetse rakenduse jaoks sobivad profiilid ning töötlusreeglid.

Riiklikes infosüsteemides kasutamisel (eriti kehtib see *OpenID Connecti* ja JWT kohta) soovitame killustumise vähendamiseks asutustevahelist kooskõlastamist ja profiilide koostamist.

4.9.1 JSON Web Signature

JSON Web Signature (JWS) [97] on vorming andmete tervikluse krüptograafiliseks kaitsmiseks kas digitaalsignatuuri või autentimiskoodi abil.

Oluline on märkida, et JWS ei kuulu eIDASe poolt tunnustatud allkirjavormingute hulka.

4.9.2 JSON Web Encryption

JSON Web Encryption (JWE) [100] on vorming andmete krüpteerimiseks (sh autentiva krüpteerimise abil).

4.9.3 JSON Web Token

JSON Web Token (JWT) [98] annab standardse viisi väidete (*claims*) edastamiseks kahe osapoole vahel JWS või JWE kujul ning määrab kindlaks mõned standardsed väited.

Erinevalt JWS ja JWE standarditest määrab JWT ära avateksti struktuuri (JSON objekt) ja jadastusreeglid, samuti reeglid vormingu lõppkujule, et seda oleks võimalik kasutada piiratud mahu ja tähestikuga vormingutes – näiteks HTTP URL osisena.

Kõige tuntum ja levinum JWT standardrakendus on OpenID Connect protokollistik [159].

¹Hoolimata sellest, et vormingu aluseks on JSON, võib kaitstav avatekst olla ka muule süntaksile vastav baidijada.

²<https://www.iana.org/assignments/jose/jose.xhtml>

4.10 PASETO

PASETO¹ (*Platform-Agnostic SEcurity TOkens*) on vorming pääsutalongidele ja küpteeritud sõnumitele. PASETO on disainitud asendama JOSE vormingut (jaotis 4.9); disaini eesmärgiks on lihtsustada vormingu(te) korrektset kasutamist. PASETO standardi versioonid fikseerivad üheselt kasutatavad krüptograafilised primitiivid, võttes arendajalt valikuvõimaluse.

PASETO-vormingut ei standardiseeri ükski standardeid arendav organisatsioon ja ei sponsoreeri ükski suurettevõtte. Vormingut toetavad teegid on olemas Node.js, Go ja Pythoni jaoks,¹ kuid neid kasutatakse vähe ja uuendatakse harva. Vormingul ja teekidel puuduvad turvaanalüüsid.

PASETO kasutamine ei ole soovitatav.

4.11 PGP

PGP (standarditud kui OpenPGP [70]) on paremate valikute puudumisel endiselt suhteliselt populaarne, kuid nagu krüptograafid ja infoturbeekspertid on korduvalt^{2,3} tähelepanu juhtinud, on tema „parim enne“ tähtaeg ammu möödas. Ennekõike on probleemiks ülearu komplitseeritud ja pika aja jooksul *ad hoc* meetodil arendatud konteinervorming ning ajaproovile mitte vastu pidanud projekteerimisotsustel baseeruv etalonteostus (GnuPG). Seda illustreerib turvanõrkus *SigSpoof* numbriga CVE-2018-12020,⁴ mis taas kord osutab sellele, kui ohtlik on andme- ja juhtkanalite omavaheline segamine tarkvarasüsteemides (tavaline *SQL injection* tuleneb samast veast).

Vormingu omapärade tõttu ei ole võimalik neid probleeme ka parandada [52, p. 3.1].

Realisatsiooni ja vormingu puudused üheskoos tekitavad probleeme nii vahetatavate sõnumite tervikluse kui konfidentsiaalsusega [135].

Sarnaselt CDOCiga puudub ka PGPI tulevikuturvalisuse omadus.

PGP kasutamine ei ole soovitatav.

¹<https://paseto.io>

²<https://blog.cryptographyengineering.com/2014/08/13/whats-matter-with-pgp/>

³<https://latacora.micro.blog/2019/07/16/the-pgp-problem.html>

⁴<https://cert.europa.eu/static/SecurityAdvisories/2018/CERT-EU-SA2018-016.pdf>

5 Autentimis-, volitus- ja delegeerimisprotokollid

5.1 OAuth 2.0

OAuth 2.0 [83, 53] on karkass (*framework*), mis defineerib volitusprotokolli (kuigi täpsem oleks öelda „delegeerimisprotokoll“). OAuth võimaldab kolmanda osapoole klientrakendusel saada piiratud pääsuõiguse HTTP teenustele. Kõige tavalisem kasutus on, et kasutaja annab ühele teenusele piiratud päringuõiguse mingis teises teenuses hallatavale ressursile.

Näiteks: kasutaja annab fototöötlusrakendusele piiratud ajaks õiguse lugeda ja muuta fotohoidlas asuvaid ja talle kuuluvaid pilte. Kasutaja on ressursi omanik. Fototöötlusrakendus on klientrakendus. Õigus lugeda ja kirjutada on delegeeritav piiratud õigus. Fotohoidla on teenus, millele klientrakendus juurdepääsu vajab. Pildid hoidlas on ressurss, mille kasutamise jaoks õiguseid delegeeriti (volitus anti).

On oluline silmas pidada, et OAuth 2.0 ei ole protokoll universaalseks volitustehalduseks, vaid ainult konkreetse kasutaja õiguste täielikuks või osaliseks delegeerimiseks mõnele rakendusele.

OAuth karkassile on ehitatud palju mitmesuguseid laiendusi nagu OpenID Connect¹ autentimisinfo standardiseeritud edastamiseks ja User Managed Access² pääsupoliitika haldamiseks kasutajakeskselt, aga ka laiendused protokoll erinevatele aspektidele nagu parameetrite edastamine, klientrakenduse autentimine, volitusserveri parameetrite avastamine, reeglid pääsutõendite kasutamise kohta jms. OAuth volitusprotokolli nüansside ja kasutuste erinevates kontekstides kirjeldamisega tegeleb IETF *Web Authorization Protocol* töörühm.³

OAuthi turvamudelit, st ohtusid millega on protokoll projekteerimisel arvestatud ning seda kuidas nende vastu kaitstakse, kirjeldab IETF RFC 6819 [120]. Oluline on märkida, et OAuth ei ole autentimisprotokoll⁴ ja selle kasutamine autentimiseks ilma sobivate täiendusteta (nagu seda enne protokoll OpenID Connect standardimist sageli tehti) võimaldab mitmesuguseid ründeid.⁵ Näiteks ei ole klientrakendusel üldjuhul võimalik kontrollida, et pääsutõend väljastati just talle või samas seansi vältel.

OAuthi abil väljastatud pääsutõendeid saab kasutada ka muudes protokollides peale HTTP, kuid OAuthi protokolliga kirjeldatud sõnumite vahendamine üle mingite muude protokollide

¹https://openid.net/specs/openid-connect-core-1_0.html

²<https://kantarainitiative.org/confluence/display/LC/User+Managed+Access>

³<https://tools.ietf.org/wg/oauth/>

⁴<https://oauth.net/articles/authentication/>

⁵<http://www.thread-safe.com/2012/01/problem-with-oauth-for-authentication.html>

peale HTTP pole OAuthi turvamudelis kirjeldatud ning sellised kasutusjuhud vajaksid eraldi turvaanalüüsi.

OAuth 2.0 projekteerimisel on arvesse võetud varasemate versioonide (1.0 ja 1.0a) kogemusi, kuid ta erineb teostuselt oluliselt. OAuth 2.0 parandas mitmed varasemates versioonides esinenud nõrkused; varasemad versioonid ei ole enam kasutamiseks turvalised. Standardimisel on ka versioon 2.1,¹ mis lihtsustab praegust kirjeldust ning koondab senised head turvatavad otse põhistandardisse. Olulisemad kavandatud uuendused, mis peaks standarditama versioonis 2.1 on järgmised.

- PKCE [160] on edaspidi volituskoodi kasutamisel kohustuslik.
- Naasmisaadresside võrdlemist on täpsustatud.
- Kaudne kinnitus (*Implicit Grant*) on standardist eemaldatud.
- Ressursiomaniku parooli kasutamine kinnituseks on standardist eemaldatud.
- Esitajatõendite (*bearer token*) edastamine URLi argumentidena on standardist eemaldatud.
- Värskendamispiletid peavad edaspidi olema seotud konkreetse kliendiga või olema vaid ühekordseks kasutamiseks.

5.1.1 Protokoll

OAuth2 protokoll järgi väljastab volitusserver klientrakendusele pääsutõendi(d) (*access token*), mida klientrakendus saab kasutada (HTTP) ressursiserverile päringute tegemiseks.

Abstraktsel tasemel koosneb protokoll kolmest sammust:

1. Volituskinnituse (*authorization grant*) hankimine. Klientrakendus hangib kasutajalt (võimalik, et volitusserveri vahendusel) volituskinnituse.
2. Pääsutõendi hankimine. Volitusserver väljastab volituskinnituse alusel klientrakendusele pääsutõendi.
3. Pääsutõendi kasutamine. Klient kasutab pääsutõendit päringus ressursiserverile.

Erinevate omaduste ja võimekustega klientrakenduste teenindamiseks defineerib OAuth erinevad viisid volituskinnituse ja pääsutõendi hankimiseks.

Tavaliselt kasutatakse volituskoodiga kinnitamist (*Authorization Code Grant*), kus pääsutõendi väljastamine algab veebibrauseri vahendusel edastatava volitustaotlusega. See võimaldab volitusserveril korraldada kasutaja autentimist ja/või küsida kasutaja nõusolekut pääsutõendi väljastamise kohta.

5.1.2 Osapooled

Loogiliselt on igas OAuth voos vähemalt kolm osapoolt: klientrakendus, volitusserver ja ressursiserver. Lisaks neile osaleb protokollis tavaliselt ka neljas osapool (ressursi omanik), kelleks on enamasti inimkasutaja.

¹<https://oauth.net/2.1/>

Klientrakendus (Client) on pool, keda kasutaja volitab enda õigustes tegutsema. Klientrakenduste ehitamisel peab arvestama, et protokoll jagab need avalikeks (*public*) ja saladusevõimelisteks (*confidential*). Esimese olek on selle kasutajatele (ja võimalikele ründajatele) täielikult nähtav ja võimalused volitamisserveri võimalused sellist klienti turvaliselt autentida seega piiratud. OAuth põhistandard kirjeldab olukorda, kus klientrakendus on veebirakendus, hilisemad laiendused [53] ja laienduste mustandid¹ spetsifitseerivad ka kasutamise omarakendustes (*native application*, sh mobiilirakendused) ja tervenisti brauseris käivates rakendustes.

Ressursiserver (Resource Server) pakub klientrakendusele kaitstud (HTTP) ressursi. OAuth kontekstis peab ressursiserveri arendaja mõtlema läbi viisid, kuidas selgitada välja ressursiserverile esitatud pääsutõendi tegelik tähendus ja kehtimine – kelle kinnitusel ja mille jaoks see tegelikult väljastati ning kas see on piisavalt värske, et selle alusel pääsuotsuseid teha. Turvakriitilistes süsteemides võib olla vajalik kehtivusinfo sage või igakordne kontrollimine volitusserverist. Arendamisel peaks arvestama pääsutõendite kasutamise spetsifikatsioonidega nagu nt RFC 6750 [99].

Pääsutõendi tähendus on tavaliselt see, et ressursi omanik delegeerib klientrakendusele mingid oma õigused ressursi suhtes. Pääsuotsuse tegemisel peab ressursiserver muuhulgas kontrollima, kas ressursiomanikul endal ikka olid delegeeritavad õigused (vt ka jaotis 5.1.3)

Volitusserver (Authorization Server) väljastab klientrakenduse soovil ja vajalike tingimuste täidetuse korral klientrakendusele pääsutõendi (*access token*). Tavaliselt on üheks tingimuseks ressursiomaniku kinnitus (*authorization grant*).

Klientrakendus peab olema volitusserveris eelnevalt registreeritud. Olemas on ka protokollilaiendused, mille abil saab klientrakendus end dünaamiliselt registreerida [155] ja sellist registreeringut hallata [154]. Klientrakenduste ja nendega seotud naasmisaadresside (*URL*) registreerimine ja kontrollimine on vajalik vältimaks kasutaja võimalikku suunamist ründaja ettesöödetud aadressile. Olukord, kus kasutaja saadetakse pärast edukat ja korrektset aadressil toimunud autentimist ja volituskinnituse küsimist võltssaidile lisab sellisele võltssaidile teenimatut usaldusväärust. Selle probleemiga peab arvestama ka klientrakenduste dünaamilise registreerimise võimaldamisel.

Pääsutõendite (ja muude volitusserveri väljastatud piletite ja tõendite) kehtivuse ja sisu kontrolliks ja edastamiseks on laiendus *Token Introspection*, RFC 7662 [153].

Ressursiomanik (Resource Owner) on kasutaja või süsteem, kellel või millel on õigus ja võimekus kaitstud ressursi kasutamise üle otsustada. Kui ressursiomanik on inimene, siis võib tema kohta öelda ka lõppkasutaja (*End-user*).

5.1.3 Skoop

Volitusserver võib defineerida ja pääsutõenditega siduda skoopid. Klientrakendus saab volituspäringus tellida mingi hulga skoope ning volitusserver võib (näiteks kasutaja valikute põhjal) väljastada pääsutõendi mingi teistsuguse skoopide komplekti jaoks. Kasutatavate skoopide hulk võib olla iga konkreetse klientrakenduse jaoks erinevalt piiratud. Samuti

¹<https://tools.ietf.org/html/draft-ietf-oauth-browser-based-apps-07>

võib olla defineeritud vaikekomplekt, mille volitusserver seob pääsutõendiga, kui rakendus skoobitellimust üldse ei edastanud.

OAuth standard ei defineeri skoobi tähendust ega kasutusi, see on jäetud konkreetsete teostuste ülesandeks – skoop on silt, mille tähenduse defineerib volitusserver. Ressursiomanikult pääsutõendi väljastamiseks loa küsimisel kuvab volitusserver tavaliselt mingil viisil ülevaate skoopidest, mille kavatseb pääsutõendiga siduda ning võib võimaldada kasutajal skoopide hulka muuta.

Skoope saab kasutada ja on kasutatud näiteks OAuth protokollide laienduste kokkuleppimiseks – OpenID Connect'i kasutamiseks peab klient kasutama skoopi `openid`. Levinud on ka skoopide kasutamine ressursiserveris rakendatavate pääsuõiguste kodeerimiseks, näiteks näitamaks, millistele ressurssidele (meiliaadress, nimi, elukoht jne) ning milliste tegevuste jaoks (lugemine, kirjutamine, kustutamine jne) pääsutõend õiguse annab.

Hea tava on, et klientrakendus küsib ja volitusserver annab pääsutõendi minimaalse vajaliku skoopidekomplekti jaoks. Sh võib kasutada skeeme, kus kasutajalt võetakse kinnitus rohkemate skoopide kohta, kuid pääsutõend väljastatakse vaid küsitud skoopidele. Sedasi saab klientrakendus vajadusel hiljem kasutajat segamata küsida uue, teistsuguse skoopiga pääsutõendi, samas kui iga konkreetse pääsutõendi lekkimisel tekkiv kahju on minimaalne.

Eraldi väärib rõhutamist, et ressursiomanik (kasutaja) saab delegeerida vaid neid õiguseid ja juurdepääse, mis tal endal juba olemas on. Skoopide lisamise või eemaldamisega ei saa (ega tohi saada) seda õiguste hulka laiendada. Näiteks kui kasutajal on mingitele andmetele ainult lugemisõigus, aga volitusserver kirjeldab ka skoobi, mille tähendus on samade andmete kirjutamisõigus, siis on ressursiserveri ülesanne tagada, et sellise, kasutaja kinnitatud ja kirjutamis-skoopiga seotud pääsutõend siiski ei tähendaks kirjutamisõigust.

5.1.4 Volituskinnituse hankimise viisid

Oluline osa OAuth karkassist kirjeldab volituskinnituse hankimist. Sõltuvalt ressursiomaniku ja klientrakenduse võimekusest pakub algne karkass välja neli erinevat viisi [83, jaotis 4] ning olemas on ka mitmesuguseid sellekohaseid laiendusi – näiteks UMA (*User-Managed Access*, vt jaotis 5.3) jt.

OAuth2 kasutamisel volitamisprotokollina oma algsel kujul (st mitte mõne muu otstarbega laienduse osana) piisab tegelikult volituskoodiga kinnituse kasutamisest. Kaudne kinnitus on praeguseks oma algse mõtte kaotanud ja muud kirjeldatud kinnitusviisid on algusest peale olnud mittedoovitatavad üleminekutööriistad.

Volituskoodiga kinnitus (*Authorization Code Grant*) on kõige tavalisem ja enimkasutatud viis kinnituse hankimiseks. Selle algsel kujul kasutamise eeldus on saladusevõimeline (*confidential*) klient, mis suudab end volituskoodi pääsutõendiks vahetamise jaoks turvaliselt volitusserverile autentida.

Avalike klientide jaoks loodi laiendus PKCE [160], mis võimaldab siduda volituskoodipäringu ja hilisema pääsutõendi päringu. Selline sidumine on kasulik ka saladusevõimelise kliendi korral ja selle laienduse kasutamine on soovitatav igal juhul.¹

¹OAuth 2.1 teeb PKCE kasutamise kohustuslikuks

Klientrakendus ja volitusserver peavad hoolitsema, et volituskood ei lekiks – see peaks olema ühekordseks kasutamiseks ja lühikese kehtivusajaga, klientrakendus peab selle ära kasutama esimesel võimalusel. Kuna volituskood edastatakse klientrakendusele parameetrina ümbersuunamisaadressis, siis satub see sellisena kasutaja brausimisajalukku ning volitusserveri päringulogisse.

Kaudne kinnitus (*Implicit Grant*) on lihtsustatud ning piiratud turvaomadustega autentimisvoog, kasutamiseks avalikes klientides, peamiselt kasutaja brauseris käivitatavates JavaScripti rakendustes. Selline lahendus (kaudne kinnitus) oli vajalik, sest vanemate brauserite turvamudel ei võimaldanud teha pääsutõendi väljastamiseks vajalikku varjatud POST-päringut volitusserveritele, mille allikas (*origin*) erines rakenduse enda omast. Tänapäeval on see AJAXi ja CORSi abil võimalik ning kaudse kinnituse asemel peab kasutama volituskoodi koos laiendusega PKCE.¹

Kaudse kinnituse korral ei kasutata pääsutõendi väljastamiseks eraldi päringut ega autendita klientrakendust. Kaudse kinnituse korral ei ole volitusserveril lubatud tagastada värskendamispiletit.

Ressursiomaniku parool kinnitusena (*Resource Owner Password Credentials Grant*) ja **Kliendi mandaadiga kinnitus (*Client Credentials Grant*)** on ajaloolised kinnituse hankimise viisid, mille eesmärk oli soodustada üleminekut pääsutõendite-põhisele pääsuvaldusele. Nende kirjelduse leiab OAuth2 spetsifikatsioonist [83]. Nende kasutamine ei ole tänapäeval soovitatav.

OAuth võimaldab ka muid kinnituse viise, näiteks RFC 7522 [39] defineerib viisi SAML 2.0 esitajakinnituste (*SAML Bearer Assertion*) kasutamiseks volituskinnitusena ja kliendi autentimiseks. Standarditud laiendusi volituskinnitusele registreeritakse muuhulgas näiteks IETF URN OAuthiga seotud kasutuste alamnimetuses [40].

5.1.5 Pääsutõendid

OAuth protokollil eesmärk on väljastada klientrakendusele sobivate omadustega pääsutõend. Klientrakenduse vaatest on pääsutõend üldjuhul läbipaistmatu (tähenduseta) sõne, mis on mõeldud esitamiseks ressursiserveri(te)le.

Kasutatakse kahte lähenemist – pääsutõend võib olla kas juhuslikult genereeritud sõne või mingi andmestruktuuri esitus baidijadana.

Juhuslikult genereeritud pääsutõendit oskab tõlgendada vaid volitusserver ise. Seega peab server kogu volituskonteksti (kellele, millise ressursi kohta, millal, mis asjaoludel, mille jaoks ja kui kauaks tõend väljastati) säilitama kuni tõendi kehtivusaja lõpuni.

Struktuuriga pääsutõendisse saab saada salvestada volituskonteksti või selle osad ning sellise tõendi kontrollimiseks vajalikud metaandmed. Sellisel otstarbel saab kasutada nt signeeritud ja/või krüpteeritud JWT-struktuuri. Sellist tõendit ei pea volitusserver tingimata ise meeles pidama – vajadusel saab ta ise (või signeerimise korral ka nt ressursiserver) pääsutõendi verifitseerida ja vajalikud andmed tõendist välja lugeda.

Millist lähenemist kasutada, sõltub konkreetsest rakendusest. Juhuslikult genereeritud pääsutõend on tavaliselt lühem ja sellest ei saa lekkida volituskonteksti asjaolusid. Sellist

¹<https://tools.ietf.org/html/draft-ietf-oauth-browser-based-apps-07>

tõendit on ka lihtsam tühistada või muul moel muuta sellega seotud andmeid – ressursiserver peab tõendi kasutamisel selle kehtimise ja kasutamise asjaolude kohta volitusserverile päringu tegema ja sealt saadavad vastused võivad ajas erineda.

Struktuuriga tõendi eelisteks on volitusserveri poolt hoitava andmestiku väiksem maht ning ressursiserveri võimekus tõendit iseseisvalt valideerida. Samas on keerulisem nii tõendiga seotud andmete muutmine pärast tõendi väljastamist kui ka tõendi enneaegne tühistamine (st kehtetuks muutmine varem kui tõendis endas sisalduv tõendi kehtivuse lõpu ajahetk).

5.2 OpenID Connect

OpenID Connect [159] on OAuth 2.0 karkassile ehitatud identiteediinfo kiht. OpenID Connect spetsifitseerib viisid, kuidas delegeerida kasutaja autentimine autentimisteenusele ja autentimise tulemus klientrakendusele tagastada. OpenID Connect defineerib ka viisid kasutaja profiili-info standardseks edastamiseks.

OpenID Connect defineerib viisid kuidas identiteediinfot kodeerida, kuidas leppida kokku edastava identiteediinfo sisus ning kirjeldab turva- ja privaatsusaspektid protokoll kasutamisel. Sarnaselt OAuthile ei kehtesta OpenID Connect kuidas täpselt volitamisserver kasutaja autentima peab, küll aga võimaldab standardsel viisil kommunikeerida toimunud autentimise asjaolusid ja klientrakenduse soove autentimise omadustele.

OpenID Connect kirjeldust haldab OpenID Foundation.¹ Sama organisatsioon on varasemalt avaldanud standardi OpenID versioonid 1.0, 1.1 ja 2.0, mille arendamisest on praeguseks loobunud. Kuna need varasemad standardid ei võta aluseks OAuthi, vaid kirjeldavad iseseisvat autentimisprotokoll, on oluline veenduda, millisest protokollist räägitakse, kui viidatakse OpenIDle.

5.2.1 Terminoloogia

Ajaloost ja traditsioonidest tulenevalt nimetatakse autentimisprotokollides (sh varasemates OpenID protokollides) protokoll osapooli teisiti kui OAuthis tavaks:

OpenID pakkuja, OP (*OpenID Provider*) volitusserver, mis toetab OpenID Connect protokoll – st korraldab lõppkasutaja autentimise ning esitab selle tulemuse identiteeditõendina.

toenduja, RP (*Relying Party*) klientrakendus, mis kasutab OpenID Connect protokoll kasutaja identiteedi tuvastamiseks ja muude OP pakutavate teenuste tarbimiseks.²

autentimispäring (*Authentication request*) OAuth 2.0 volituspäring OPle, milles kasutatakse OpenID Connect skoopt ja muid laiendatud parameetreid, eesmärgiga autentida lõppkasutaja ning edastada identiteediinfo RPle.

lõppkasutaja (*End-user*) inimkasutaja, kelle OP autendib ja kelle identiteediinfo toendujale edastatakse.

¹<https://openid.net/foundation/>

²„Toenduma“ on A. V. Kõrvi 1936. aasta uudissõnastikus antud tähendusega „(millelegi) kellegi v. millegi päälle toetuma (näiteks kõnes).“ Vastava nimisõna sobib suurepäraselt *Relying Party* vasteks, väljendades osalist ja usaldavat toetumist kellelegi teisele.

5.2.2 Identiteeditõend (*ID Token*)

OpenID Connect defineerib viisi lõppkasutaja identiteedi, st lõppkasutaja kohta käivate identifikaatorite ja muude atribuutide edastamiseks OPlt toendujale. Selleks laiendatakse pääsutõendi väljastamise teenust, mis OpenID Connect standardi kohaselt väljastab lisaks tavalisele pääsutõendile ka JWT-kujul identiteeditõendi (*ID Token*). Identiteeditõendi terviklus ja autentsus tagatakse signeerimise või HMAC (räsipõhise autentimiskoodi) kasutamisega – toenduja peab saama veenduda, et tõendi väljastas õige OP.

Identiteeditõendis seotakse ära omavahel kasutaja identifikaatorid ja atribuudid ning konkreetne autentimisjuhtum ja autentimispäring.

Tähtis on tähele panna, et identiteeditõend edastab infot ennekõike autentimisjuhtumi, mitte lõppkasutaja kohta [159, jaotis 2]. Identiteeditõendi ülekoormamine lõppkasutaja kohta käiva infoga ei ole standardikohane lahendus. Lõppkasutaja kohta detailse informatsiooni edastamiseks on ette nähtud kasutajainfo teenus [159, jaotis 5.3].

Identiteeditõendis peavad kindlasti sisalduma

- 1) väljastaja (OP) identifikaator,
- 2) lõppkasutaja muutumatu ja mõne teise lõppkasutaja jaoks mitte (taas)kasutatav identifikaator tõendi väljastaja juures,
- 3) loetelu tõendi võimalike kasutajate identifikaatoritega (st: „kellele tõend väljastati“), toenduja identifikaator OPs peab selles hulgas sisalduma,
- 4) kehtivuse lõpuaeg sekundi täpsusega,
- 5) väljastamise aeg sekundi täpsusega.

Kokkuleppel toendujaga võib OP identiteeditõendisse lisada ka suvalisi muid toendujale kasulikke väiteid (*claims*).

Lõppkasutaja identifikaatorina võib kasutada ka mõne kolmanda osapoole (näiteks riigi) väljastatud identifikaatorit, kui sellel on nõutud omadused. OpenID Connect defineerib ka valikulise skeemi, kus lõppkasutaja identifikaator iga konkreetse toenduja jaoks on unikaalne (*pairwise identifier type*). See väldib toendujate omavahelise koopereerumise ilma kasutaja ilmutatud loata.

Soovi korral võib identiteeditõendi ka krüpteerida. See võib olla kasulik, kui identiteeditõend liigub avalikes või lekkimisohuga kanalites, näiteks kaudse kinnitamisega voos või hübriidautentimisvoos (vt 5.2.6 ja 5.1.4).

5.2.3 Skoobid

OpenID Connect defineerib skoobi *openid*. Selle kasutamine OpenID Connect autentimispäringutes on kohustuslik, andmaks OPlle teada, et kehtivad OpenID Connect sätestatud reeglid.

Täiendavalt defineerib OpenID Connect valiku skoobinimesid, mille abil toenduja saab tellida erinevate atribuudikomplektide lisamise identiteeditõendisse või nende tagastamise kasutajainfo teenuse kaudu. Standardi versioon 1.0 defineerib skoobid *profile*, *email*, *address* ja *phone*.

OpenID Connect kirjeldab ka skoobi `offline_access` kasutamise värskendamispileti tellimiseks ning sellega seotud soovituslikud reeglid.

5.2.4 Kasutajainfo teenus

Muude kasutajaga seotud atribuutide hankimiseks defineerib OpenID Connect (OP)le kasutajainfo teenuse (*UserInfo Endpoint*). See on tavaline OAuth kaitstud teenus, mille kasutamiseks peab toenduja kasutama talle väljastatud pääsutõendit ja mis tagastab hulga kasutaja kohta käivaid väiteid, kas tavalise JSON-objekti või signeeritud ja/või krüpteeritud JWT kujul.

Kasutajainfo teenust saab kasutada selliste kasutaja kohta käivate väidete esitamiseks, mida pole identiteeditõendis mõistlik esitada nende ajas muutumise, mahu või privaatsuskaalutluste tõttu.

Kasutajaatribuutide edastamiseks standardsel viisil kirjeldab OpenID Connect JWT väidete registrisse¹ hulga võimalikke väiteid koos nende esitamise kuju ja tähendusega.

5.2.5 Autenditud päringud

Mitmed OAuth ja OpenID varasemate versioonide nõrkused on tingitud asjaolust, et päringud volitamisteenusele ei ole autentitud. OpenID Connect defineerib viisi, kuidas päringu parameetreid esitada JWT-kujul, mis võimaldab nende signeerimist ja soovi korral ka krüpteerimist.

JWT-kujul päringu parameetrid esitatakse väidetena JWTs. JWT lisatakse tavalisele OAuth päringule kas otse või viida kaudu (URLiga). Parameetrid `response_type`, `client_id` ja `scope` peab edastama (ka) tavaliste OAuth parameetritena, et OP tunneks ära OAuth päringu ja OpenID Connect kasutamise. Muude parameetrite kasutamine JWT kõrval on lubatud. Tavalisel viisil ja JWTs esitatud sama nimega parameetri korral eelistatakse JWTs sisalduvat väärtust.

5.2.6 Autentimisvood

OpenID Connecti eesmärk on võimaldada toendujal hankida kasutaja kohta usaldusväärne identiteediinfo. Selle saavutamiseks tagastatakse koos pääsutõendiga ka identiteeditõend (ning pakutakse täiendava identiteediinfo jaoks kasutajainfo teenust). Kuna OpenID Connect tegeleb lõppkasutaja autentimisinfo, siis kasutatakse ainult volituskoodiga ja kaudset kinnitamist, mis mõlemad algavad kasutaja brauseri suunamisega volitusteenusele. OpenID Connect laiendab ja täpsustab OAuth defineeritud volitusteenuse parameetreid. Näiteks on parameetrite `scope` ja `redirect_uri` kasutamine autentimispäringus kohustuslik. Täiendavalt saab toenduja täpsustada OP käitumist. Nii näiteks saab olemasolevaid (OP) seansse taaskasutada või tellida veateate kui seanssi pole, nõuda spetsiifilisi autentimistasemeid, anda vihjeid toendujale teadaoleva varasema autentimise kohta ja anda vihjeid kasutaja kasutajaliidesega seotud eelistuste osas.

OpenID Connect defineerib kolm autentimisvoogu:

1. Volituskoodiga voog (*Authorization Code Flow*). Volituskoodi saamine ning pääsutõendiks ja identiteeditõendiks vahetamine sarnaselt volituskoodiga kinnitusele OAuth pro-

¹<https://www.iana.org/assignments/jwt/jwt.xhtml#claims>

tokollis. Volitusserver tagastab koos pääsutõendiga ka identiteeditõendi. See on kõige levinum voog autentimiseks veebirakendustes.¹

2. Kaudne voog (*Implicit Flow*). Volitusteenust kasutatakse nagu kaudse kinnitamise korral OAuth protokollis. Sarnaselt OAuthi kaudsele kinnitusele pole see voog enam soovitatav ja selle asemel peaks kasutama volituskoodiga voogu või hübriidvoogu koos laiendusega PKCE.
3. Hübriidvoog (*Hybrid Flow*). Volitusteenust kasutatakse nagu volituskoodiga kinnituse korral, kuid OP tagastab naasmisaadressile lisatud fragmendis volituskoodi ja ühe või mitu tõendit, sarnaselt kaudsele kinnitamisele. Seda voogu peaks kasutama vaid laiendusega PKCE [160].

5.2.7 SSO-teenuse pakkumine OpenID Connect standarditega

OpenID teenusepakkuja (OP) saab toetada kasutaja autentimistulemuse taaskasutamist nõnda, et OP ning grupp RP-sid moodustavad kasutaja jaoks tihedalt integreeritud süsteemi ning kasutaja saab ühe OP kasutusseansi (OP-seansi) käigus kasutada mitme RP teenuseid, ilma mitmekordselt autentimata. See võimaldab kasutajale pakkuda suuremat mugavust ning hoida kokku autentimiste arvu.

Sellise integratsiooni jaoks on vajalik standardiseerida järgmised teemad:

1. SSO-teenuse kasutamise tehniliste detailide kirjeldamine ning RP-dele levitamine (metainfo),
2. kasutaja autentimistulemuse edastamine RP-le (see on võimalik standardi OpenID Connect tavapärase kasutamisega),
3. RP teavitamine kasutaja OP-seansi lõpetamisest,
4. RP poolt kasutaja OP-seansi lõpetamise tellimine.

OpenID Foundation on 2022. aastal kirjeldanud kolm võimalust, kuidas edastada RP-dele kasutaja OP-seansis toimuvaid muutusi:

1. *OpenID Connect Session Management*²
2. *OpenID Connect Front-Channel Logout*³
3. *OpenID Connect Back-Channel Logout*⁴

ning ühe võimaluse, kuidas RP saab tellida OP-seansi lõpetamise:

1. *OpenID Connect RP-Initiated Logout*⁵

Eestis on realiseeritud teenus GOVSSO, mis on muutumas *de facto* standardiks⁶ ning mis on juhindunud samuti OpenID standarditest. Kirjeldame siinkohal lühidalt OpenID pakutavaid võimalusi ning võrdleme neid GovSSO praktilise realisatsiooniga.

¹<https://e-gov.github.io/TARA-Doku/TehnilineKirjeldus>

²https://openid.net/specs/openid-connect-session-1_0.html

³https://openid.net/specs/openid-connect-frontchannel-1_0.html

⁴https://openid.net/specs/openid-connect-backchannel-1_0.html

⁵https://openid.net/specs/openid-connect-rpinitiated-1_0.html

⁶<https://e-gov.github.io/GOVSSO/TechnicalSpecification>

5.2.7.1 OpenID Connect Session Management

Tavapäraselt, OIDC standardi järgi, peaks RP kontrollima kasutaja seansi kehtivust uue autentimispäringuga, lisades sinna parameetrid `id_token_hint` ja `prompt=none` ning kui OP-seanss kehtib, siis vastab OP uue autentimiskinnitusega. Selliste päringute tegemine võib aga olla ressursimahukas ning mobiilseadmete korral ebasoovitav.

OIDC Session Management standard defineerib viisi, kuidas kasutaja brauseris töötav RP veebileht saaks teada samas brauseris salvestatud OP-seansi oleku muutustest ning saaks seeläbi teada, kui kasutaja on OP-seansi lõpetanud ning ka RP veebileht peaks selle kasutaja seansi lõpetama. Tavapäraselt ei ole RP veebilehel õigusi OP veebisaidi küpsiseid või HTML5 `localStorage` atribuute lugeda ning otse ta sellist infot ei saa.

Standard kirjeldab kuidas järgmiseid mehhanisme kasutades seda siiski korraldada:

1. OP edastab RP-le OIDC autentimisvastuse parameetris `session-state` OP-seansi olekut tähistava stringi. Selle väärtuse arvutamisel kasutatakse OP seansiküpsiseid, RP `client_id` väärtust ning juhuslikku soola.
2. RP lisab oma veebilehele peidetud fragmendi (*iframe*), kuhu laaditakse spetsiaalne OP alamleht, mis pakub kasutaja brauseris töötavale RP veebilehele HTML5 `postMessages` suhtlusliidest. Kuna sellesse fragmenti laaditud OP alamleht on pärit OP veebisaidist, siis ta pääseb OP veebisaidi küpsistele ligi.
3. RP saadab perioodiliselt OP alamlehele HTML5 `postMessages` päringuid küsimusega, kas `session-state` väärtus kehtib. OP alamleht arvutab seansiküpsiste abil uuesti `session-state` väärtuse ning kui ta saab teistsuguse stringi, siis vastab ta RP veebilehele, et kasutaja OP-seanss on muutunud.
4. RP peab seejärel kontrollima näiteks OIDC autentimispäringuga `prompt=none`, kas kasutaja on tõesti välja loginud ning seejärel ka enda seansi lõpetama.

Üks näide selle standardi rakendamisest ning vastavast OP alamlehe koodist, on toodud IBM WebSphere dokumentatsioonis.¹

Selle standardi järgimine võimaldab küll kokku hoida ülevõtteid võrgupäringuid, kuid samas on selle standardi puuduseks, et osad brauserid on hakanud blokeerima või plaanivad blokeerida selliseid taustapäringuid, mis ei ole seotud parasjagu fookuses oleva veebisaidi aadressiga. Seetõttu võib juhtuda, et RP veebilehe fragmendile OP alamlehe laadimine ebaõnnestub või siis ei pääse OP alamleht OP veebisaidi küpsistele ligi. Mõlemal juhul ei saa RP OP-seansi kehtivuse kohta kinnitust ning peab järeldama, et kasutaja on välja loginud.

5.2.7.2 OpenID Connect Front-Channel Logout

See standard kirjeldab viisi, kuidas OP saab kasutaja brauserit kasutades edastada RP-le infot kasutaja OP-seansi muudatuste kohta. Standard ei kasuta OP veebisaidist laaditud veebilehe fragmente. Selle asemel defineerib spetsiaalse väljalogimise URI (metainfo parameeter `frontchannel_logout_uri`) ning registreerib selle OP juures. Kui kasutaja lõpetab oma brauseriga OP juures seansi, siis lisab OP enda veebilehele peidetud fragmendi, kuhu brauser laadib RP väljalogimise URI. Sedaviisi edastatakse RP veebisaidile signaal, et kasutaja OP-seansis on toimunud muutusi.

¹<https://www.ibm.com/docs/en/was-liberty/base?topic=liberty-invoking-session-management-endpoint-openid-connect>

OP võib saata väljalogimise URI-le järgmiseid parameetreid:

1. `iss` – OP identifikaator
2. `sid` – seansi identifikaator

Sarnaselt eelmise standardiga (5.2.7.1), on ka selle standardi puuduseks, et osad brauserid on hakanud blokeerima või plaanivad blokeerida selliseid taustapäringud, mis ei ole seotud parasjagu fookuses oleva veebisaidi aadressiga. Seetõttu võib juhtuda, et OP veebilehe fragmendile RP URI laadimine ebaõnnestub või siis ei pääse peidetud fragmendile laaditud RP veebileht kasutaja seansiandmetele ligi. Standard soovib püüda selliseid olukordi tuvastada ning probleemide korral kuvada kasutajale veateid, kuid nendib, et nende olukordade avastamine ei pruugi olla alati võimalik.

5.2.7.3 OpenID Connect Back-Channel Logout

See standard kirjeldab viisi, kuidas OP saab info kasutaja OP-seansi muudatuste kohta edastada RP tagasüsteemile. Kasutaja brauser ei ole enam nende päringute vahendajaks ning seetõttu ei ole see lahendus mõjutatud sellest, kas brauseritootjad hakkavad mingeid päringuid blokeerima või mitte. Samas nõuab see standard, et RP serverid peavad olema OP serveritele juurdepääsetavad.

Kasutaja OP-seansi muutuse korral edastab OP RP tagasüsteemi väljalogimise URIle (metaandmete parameeter `backchannel_logout_uri`) spetsiaalse JWT andmestruktuuri, mida nimetatakse väljalogimise talongiks (*logout token*).

Väljalogimise talong järgib JWT standardit ning sisaldab tavapäraseid väljasid `iss`, `sub`, `aud`, `iat`, `jti` kuid ka spetsiaalseid lisaväljasid:

1. `events` – kohustuslik väli, mille sisaldab omakorda JSON objekti, milles on väli `http://schemas.openid.net/event/backchannel-logout` ja väärtus `{}`. Sellise objekti lisamine näitab, et tegemist on väljalogimise talongiga ning annab ühilduvuse standardiga RFC8417 [90]. Soovitav on veel lisaks kasutada talongi tüübi defineerimist JWT päisevälja `typ` väärtusega `logout+jwt`.
2. `sid` – valikuline väli, millega saab edastada OP-seansi identifikaatori.

Tähelepanuväärne on, et väljalogimise talongi ei tohi lisada välja `nonce`. See on selleks, et rangelt eristada autentimiskinnitusena kasutatavat JWT struktuuri (`id_token`) väljalogimise talongi JWT struktuurist (`logout_token`) ning takistada väljalogimise talongi taas-esitamist uues autentimisseansis kui autentimiskinnitust.

Talong peab olema signeeritud, et vältida võltsitud talongide saatmisega teenusetökes-tusrünnet ning talong ise edastatakse RP-le URI `backchannel_logout_uri` parameetris `logout_token`.

5.2.7.4 OpenID Connect RP-Initiated Logout

Viimane selle perekonna standard kirjeldab viisi, kuidas RP saab edastada info OP-le, et kasutaja OP-seanss tuleks lõpetada. Selleks suunab RP kasutaja brauseri OP väljalogimise URL-ile (metaandmete parameeter `end_session_endpoint`) ning võib lisada URL-ile järgmised parameetrid:

1. `id_token_hint` – OP poolt eelnevalt väljastatud autentimiskinnitus.
2. `client_id` – RP identifikaator
3. `post_logout_redirect_uri` – RP veebisaidi URI, kuhu kasutaja brauseri peale väljalogimist tagasi suunata
4. `state` – Väljalogimispäringu ja tagasisaabunud päringu seostamise parameeter

Väljalogimispäringu saamisel peab OP kuvama kasutajale küsimuse, kas ta soovib OP-seansi lõpetada. Peale kasutaja väljalogimist võib OP kasutada mõnda eelpool kirjeldatud standardit (5.2.7.1, 5.2.7.2, 5.2.7.3) teiste RP-de teavitamiseks, et kasutaja OP-seanss on nüüd lõppenud.

5.2.7.5 GovSSO

GovSSO (kuni 20. märtsini 2023 GOVSSO) on RIA poolt loodud autentimisteenus, mis salvestab kasutaja brauseri küpsistes OP-seansi andmed ning võimaldab kasutajal senist autentimistulemust taaskasutada järgmiste RP-de juures. Kasutaja autentimiseks kasutab GovSSO riigi autentimisteenust TARA.

GovSSO kasutab OpenID Connect perekonna standardeid. Dokumenteeritud kasutusviis muutus käesoleva aruande kirjutamise ajal — 20. märtsil 2023. Järgnevalt kirjeldame, kuidas GovSSO kasutas OpenID Connect protseduure enne seda kuupäeva.

1. OP-seansi identifikaator edastatakse RP-le `id_token` struktuuri sees atribuudiga `sid`.
2. OP-seansi kontrollimiseks peab RP tegema GovSSO-le OIDC autentimispäringuid, lisades parameetrid `id_token_hint` ning `prompt=none`.
3. OP-seansi lõppemisel saadab GovSSO RP-le 5.2.7.3 kohase väljalogimise päringu koos talongiga `logout_token`.
4. OP-seansi lõpetamiseks võib RP kasutaja suunata GovSSO väljalogimise lehele (metainfo parameeter `end_session_endpoint`), vastavalt jaotises 5.2.7.4 kirjeldatud protseduurile.

GovSSO eeldab, et RP-d teevad perioodiliselt OP-seansi kontrollimise päringuid. OP-seansi aegumise taimer on sihilikult valitud väga lühike, hetkel 15 minutit. See meede peaks tagama, et kui kasutaja lõpetab RP veebisaidi aktiivse kasutamise, kuid ei logi RP veebisaidist sihilikult välja, siis aegub ka globaalne OP-seanss ning kasutaja brauseris olev OP-seansi küpsis muutub kehtetuks. Samas, kui kasutaja jätkab siiski RP veebisaidis aktiivset tööd, siis pikendatakse OP-seansi kehtivust automaatselt.

Kahjuks tuleb nentida, et sellise turvanõude jaoks ei ole tänapäeva maailmas häid lahendusi ega toetatud standardeid. OpenID Connect perekonna standardid ei käsitle sellist globaalse seansi värskendamise võimalust. Selle põhjuseks võib olla, et brauseritootjate vaatenurgast on selline tegevus väga laias ulatuses kasutaja tegevuste jälgimine ning privaatsust riivav. Isegi jaotistes 5.2.7.1, 5.2.7.2 kirjeldatud standardid hoiatavad, et ka nende poolt kirjeldatud lahendused, kuidas läbi brauseri informeerida RP-d OP-seansi muudatustest, ei pruugi tulevikus enam toimida, kuna brauserid hakkavad sellist infovoogu blokeerima. Samale järeldusele on jõudnud ka GovSSO arendajad, kes töötavad alternatiivsete mehhanismidega. Üks oluline muudatus 20. märtsist 2023 on parameetriga `prompt=none` OIDC autentimispäringute kasutamisest loobumine, asendades selle värskendustalongidel põhineva lahendusega.

Soovitame selle teema juures pöörata tähelepanu W3C poolt loomisel olevale API-le *Federated Credential Management API*,¹ mis hakkab võib-olla asendama isegi tavapäraseid OIDC HTTP autentimispäringuid, kuna brauseritootjad peavad ka selliseid päringuid liialt kasutajate privaatsust ohustavaks. Võib juhtuda, et turvaeesmärkidel RP- ja OP-seansse väga tihedalt integreeriv, kogu riigi laiune GovSSO-lahendus ei pruugi kokku sobida brauseritootjate privaatsusmudeliga.²

Nagu näeme, on see valdkond käesoleva aruande kirjutamise ajal väga kiiresti muutuv. Lugejale saame ainult soovitada jälgida ning vajadusel aktiivselt osaleda asjakohastes standarditöögruppides, et plaanida enda infosüsteemide sobivaid arendusi. Samuti tuleb tänapäeval krüptoalgoritmide ning infoturbeprotokollide rakendamisel vältimatult silmas pidada kasutajate privaatsusnõudeid.

5.3 User-Managed Access

User-Managed Access (UMA) on Kantara Initiative³ hallatav spetsifikatsioonidekomplekt. UMA laiendab OAuth karkassi uue rolliga. Päringuesitaja (*requesting party*) on füüsiline või juriidiline isik, kelle nimel klientrakendus tegutseb. Päringuesitaja rollis võib olla (aga tavaliselt ei ole) ressursiomanikuga sama isik. UMA lisab uue volituskinnituse tüübi, mis võimaldab asünkroonset (ressursiomaniku vahetu sekkumiseta) volitamist ja pääsutõendi väljastamist. Ressursi omanik (samuti füüsiline või juriidiline isik) kehtestab päringuesitaja ja klientrakenduse suhtes pääsupoliitika, mille alusel volitusserver otsustab pääsutõendi väljastamise konkreetsele klientrakendusele.

5.3.1 Protokoll

Protokolli eeldus on, et ressursiomanik on mingil (UMAs) täpsustamata viisil kehtestanud volitusserveris pääsupoliitika. Protokoll kirjeldab klientrakenduse, ressursiserveri, volitusserveri ja päringuesitaja suhtluse; ressursiomaniku suhtlus volitusserveri ja ressursiserveriga ei ole UMA käsitusallas. Samuti ei käsitle UMA viise, kuidas klientrakendus saab teadlikuks ressursiserveri pakutavatest teenustest.

1. Klientrakendus teeb päringu ressursiserverile, kuid ei pane kaasa pääsutõendit.
2. Ressursiserver vastab klientrakendusele (pääsuõiguste puudumise) veaga ja lisab veateatele viite volitusserverile ja algse pääsupileti.
3. Klientrakendus küsib volitusserverilt (OAuth pääsutõendite väljastamise teenuse kaudu) päringuesitaja pääsutõendi (*RPT, requesting party token*). Klientrakendus esitab volitusserverile ressursiserverilt saadud algse pääsupileti ja võib esitada täiendavaid väiteid (*claims*) päringuesitaja, klientrakenduse või millegi pääsuotsuse tegemiseks relevantse kohta.
4. Volitusserver hindab esitatud väiteid.
5. Volitusserver võib vastata veateatega väidete ebapiisavuse kohta ning viidata liidesele, mille kaudu päringuesitaja saab interaktiivselt esitada täiendavaid/puuduvaid väiteid.
6. Väidete vastuseks väljastab volitusserver pääsupileti.

¹<https://fedidcg.github.io/FedCM/>

²Vt näiteks <https://github.com/michaelkleber/privacy-model> ja <https://datatracker.ietf.org/doc/html/rfc7258>

³<https://kantarainitiative.org/>

7. Klientrakendus küsib volitusserverilt pääsupileti alusel uuesti pääsutõendi (RPT).
8. Volitusserver väljastab klientrakendusele pääsutõendi (RPT).
9. Klientrakendus pöördub ressursiserveri poole, esitades ka pääsutõendi.
10. Ressursiserver kontrollib pääsutõendi ja positiivse tulemuse korral jätkab töötlusega.

Tehniliselt defineerib UMA uue volituskinnituse tüübi – UMA pääsupileti ning viisid selle hankimiseks, uuendamiseks ja kasutamiseks pääsutõendi väljastamisel.

Sarnaselt OAuth põhikarkassile ei kirjelda UMA volituskinnituse spetsifikatsioon ressursiserveri ja volitusserveri vahel toimuvat kommunikatsiooni.

5.3.2 Födereeritud volitamine

UMA 2.0 valikuline lisaspetsifikatsioon kirjeldab födereeritud volitamist (*federated authorization*). Lisaspetsifikatsioon võimaldab ressursiserveril kasutada sõltumatu (UMA) volitamisserveri teenuseid. Selleks defineeritakse viis, kuidas ressursiserver saab enda hallatavad (pääsureguleerimist vajavad) ressursid ning nendega seotud skoobid volitamisserveris registreerida ning viis, kuidas ressursiserver saab volitamisserverilt küsida esialgse pääsupileti (klientrakendusele tagastamiseks).

Födereeritud volitamise abil saab ressursiomanik delegeerida endale kuuluvate ressursside haldamise mingile usaldatavale või sobival viisil poliitikakirjeldamist võimaldavale volitusserverile.

5.4 FIDO ja Web Authentication

*FIDO Alliance*¹ on tööstusühendus, mille missiooniks on autentimisstandardite abil vähendada maailma sõltuvust paroolidega autentimisest. Organisatsioon edendab autentimis- ja atesteerimisstandardite loomist, kasutamist ja nende kasutamise ühilduvuskontrolli.

FIDO Alliance töötab välja standardeid, mis võimaldavad kasutajatel ühe autentimistegurina (*authentication factor*) kasutada asümmeetrilise krüptosüsteemi võtmepaari. Kogu protokollistiku oluline erinevus näiteks Eestis harjumuspärasest ID-kaardi taristust seisneb selles, et igal kasutajal on palju võtmepaare ning iga võtmepaar on kasutusel autentimiseks vaid ühe rakendusega. See on oluline meede mitmesuguste asendus- ja vahendusrünnete vältimiseks, samuti teeb see raskemaks kasutaja tegevuste korreleerimise erinevate toendujate vahel.

Üldjuhul haldab selliseid võtmepaare mingi iseseisev riistvaraline seade (autentur, *authenticator*) – *FIDO Alliance* standardid kirjeldavad madala taseme kommunikatsiooni selliste seadmetega ning kaugautentimiseks kasutatavaid autentimis- ning atesteerimisprotokolle, mille käigus noid võtmepaare kasutatakse. Tüüpiline selline seade on USB ja NFC liidesega „pulk“.²

Täielikult spetsifikatsioonile vastavad autenturid toetavad ka atesteerimist, ehk seadme enese füüsilise ja funktsionaalse tervikluse kontrolli krüptograafiliste meetmetega. Atesteerimine võimaldab toendujal veenduda, et kasutaja privaativõti on kogu oma elukaare jooksul nõuetekohaselt kaitstud.

¹<https://fidoalliance.org/>

²<https://www.yubico.com/ee/product/yubikey-5-nfc/>

Senised spetsifikatsioonid jagunevad kolme gruppi.

- FIDO U2F (*FIDO Universal 2nd Factor*; tuntud ka kui CTAP1) – vanem pärandprotokollistik, millele tuleks võimalusel alati eelistada WebAuthn [34] ja CTAP (*Client To Authenticator Protocol*) kasutamist. Defineerib ennekõike krüptvõtmeid haldava füüsilise autentimisseadme ja selle kasutuse tugeva teise autentimistegurina.
- FIDO UAF (*FIDO Universal Authentication Framework*) – kui U2F eesmärk on pakkuda turvalist teist autentimistegurit, siis UAF kirjeldab suhtluse ka mitmikautentimist võimaldava autenturiga, mis eemaldab vajaduse parooliga autentimise järele. Lisaks U2F autenturitele võimaldab see kasutada autentimisseadmeid, mis enne võtme kasutamist kontrollivad kasutajasamasust ka muul viisil: näiteks biomeetrilise kontrolliga.
- FIDO2 (katab standardid CTAP ja WebAuthn) – täna kehtiv standard tugeva autentimise toetamiseks veebirakendustes.

CTAP spetsifikatsioon [33] kirjeldab protokollid kaht eri versioon: CTAP1/U2F ja CTAP2.

Tänapäeval müüdavad FIDO2 (WebAuthn) autentimisseadmed realiseerivad CTAP2 protokollid ning osa seadmeid realiseerib tagasiühilduvuse huvides ka CTAP1/U2F protokollid.

WebAuthn on W3C standard [34], mis defineerib autentimise veebirakenduses koos asjakohaste protokollide ja API-dega. WebAuthn abil on võimalik autentimiseks kasutada nii U2F kui UAF autentimisseadmeid.

Uutes süsteemides tuleb igal juhul järgida standardit *WebAuthn Level 1* [34]. Ettevalmistamisel on ka standard *WebAuthn Level 2* [32].

WebAuthn toe lisamine infosüsteemidele ei ole paraku triviaalne, see nõuab tuge nii veebirakenduse raamistiku autentimisfunktsioonidelt kui halvemal juhul ka rakenduse kasutajate halduse meetodite täiendamist WebAuthn jaoks vajaliku dünaamilise võtmehaldusega.

5.4.1 FIDO U2F

FIDO U2F on esimene komplekt *FIDO Alliance* poolt välja antud standardeid. FIDO U2F kirjeldab autentimisseadet (*U2F device*), mis oskab genereerida ja atesteerida võtmepaare ning neid hiljem kasutaja autentimiseks kasutada. Toimingud (võtmete genereerimine ja kasutamine) peavad olema kaitstud „kasutaja kohaloleku kontrolliga“ (*user presence*), kasutaja peab vajutama nuppu või muul moel tehingud kinnitama. Igale veebisaidile või rakendusele genereeritakse oma sihtotstarbeline võtmepaar, standard kirjeldab U2F autentimisseadet metafooriga „kasutaja füüsiline võtmerõngas“.

Autentimisseade pakub lihtsat liidest autentimisvõtme loomiseks (kasutaja registreerimiseks) ning hilisemaks autentimiseks. FIDO U2F spetsifikatsioonid kirjeldavad ka veebibrauseris kasutatavad liidesed kummagi operatsiooni jaoks.¹

Toenduja (*relying party*) on üldjuhul veebirakendus.

U2F mudelis on võtmepaar vaid üks lisa-autentimistegur. Eeldatakse, et kasutaja on eelnevalt mingil muul viisil (nt lihtsa PINiga) autenditud ning toenduja võib igal hetkel nõuda täiendava tegurina võtmepaari kasutamist.

¹Need liidesed on praeguseks mittesoovitavad. Kasutama peaks liideseid, mille spetsifitseerib WebAuthn.

U2F autentimisseadme kasutuselevõtmiseks konkreetses rakenduses kasutatakse registreerimisoperatsiooni. Selle käigus genereerib autentimisseade uue võtme koos võtmepidemega (*key handle*), seob selle toenduja rakenduseidentifikaatori (*application id, origin*) ja kasutajaidentifikaatoriga.

Autentimise õnnestumiseks on vaja, et sobiksid omavahel nii rakenduse poolt autentimiseks küsitud võtmepide, kasutajaidentifikaator kui rakenduseidentifikaator.

5.4.2 FIDO UAF

FIDO UAF spetsifikatsioonid laiendavad ja üldistavad U2F standardeid. Defineeritakse üldine abstraktne autentur (*authenticator*), mis on võimekam variant U2F autentimisseadmest.

FIDO UAF autentur on abstraktne liides seadmele, mis võimaldab kasutaja autentimist FIDO UAF standardis defineeritud viisidel.

UAF võimaldab kasutaja identiteedi kinnitamise erinevatel turvasemetel:

- kasutaja kohalolu ega samasust ei kontrollita (*Silent Authenticator*);
- kasutaja kohalolu kontroll (*User Presence Check*) – kasutaja peab kinnitama oma kohalolu, vajutades nuppu, viibates NFC-seadmega vms;
- kasutajasamasuse kontrollimine (*User Verification*) – autentur kontrollib kasutaja õigust registreeritud võtit kasutada, tuvastades kasutaja näiteks PINi või sõrmejälge abil. Sisuliselt tähendab see koos võtme kasutamisega kaksikautentimist (kontrollitakse midagi, mida kasutaja teab või milline kasutaja on; võtmepaar on miski, mis kasutajal on).

UAF autentur ja sellega suhtlemise protokollid on kirjeldatud viisil, mis võimaldavad kasutada ka U2F autentimisseadmeid spetsiifilise võimekusega UAF autenturina.

5.4.3 FIDO2 – WebAuthn ja CTAP

Kolmas komplekt FIDO spetsifikatsioone ühtlustab ja üldistab sõnumivahetuse autenturi ja UAF kliendi vahel (CTAP, *Client To Authenticator Protocol*) ning kirjeldab liidese veebibrauserist FIDO autenturite kasutamiseks (*Web Authentication, WebAuthn*).

WebAuthn [34] on projekti FIDO2 peamine tulem ja ta defineerib kasutajate autentimise veebirakendustes. WebAuthn spetsifikatsioon on valminud *FIDO Alliance*'i ja *World Wide Web Consortiumi* (W3C) koostöös ja on avaldatud W3C soovitusena (*recommendation*).

WebAuthn defineerib oma autenturi mudeli, CTAP-ühilduv autentur on selle mudeli üks võimalik teostus. WebAuthn jagab autentureid:

- Ühendatuse järgi: platvormiautentur on seotud klientseadmega ega pole sellest eraldatav. Platvormiülene autentur (*cross-platform authenticator*) võib suhelda teise seadmega. Kommunikatsiooni vahendava sobiva kihi olemasolul on võimalik (nt mobiiltelefoni sisseehitatud) platvormiautenturit kasutada mingis teises kontekstis platvormiülese autenturina.
- Salajaste võtmete hoidmise viisi järgi: kohalike mandaatidega (*resident credential*, võtit ja sellega seotud atribuute hoitakse autenturi kohalikus hoidlas) autentur ja võtmeid võtmepidemesse kodeeriv autentur. Autentur võib toetada mõlemat mandaatide

(võtmete) hoidmise viisi, mandaadi loomisel saab valida, kas mandaatide hoidmine autenturis on kohustuslik.

- Kasutajasamasuse kontrollimise (*user verification*) toetamise järgi: mitmikautentimist (*multi factor authentication*) toetavad autenturid ja seda mittetoetavad autenturid. Mitmikautentimist toetav autentur võib toetada ka rohkem kui kahte autentimistegurit.

CTAP omakorda defineerib sõnumid FIDO kliendi (ehk lõppkasutajaseadmes oleva FIDO tarkvarapinu) ja tegeliku autenturi vahel toimuvaks suhtluseks. CTAP1 on U2F ühilduv, CTAP2 on FIDO2 muude standarditega kooskõlaline uuem protokoll. Reeglina võimaldavad FIDO2 kliendid kasutada ka U2F autentureid. Madala taseme transpordiprotokoll jääb mõlema CTAP versiooni jaoks samaks, kirjeldatakse USB, NFC ja BLE (*Bluetooth Low Energy*) kasutamine.

5.4.4 Soovitused

WebAuthn/CTAP kasutatavate krüptoalgoritmide loetelu on alamosa COSE algoritmidest.¹ Reaalselt kasutatavate algoritmide loetelu piirab ühelt poolt konkreetsete autenturite poolt toetatud krüptalgoritmide valik, teiselt poolt peab aga iga toenduja enda süsteemis piirama kasutatud krüptoalgoritmid ja nende parameetrid vastavalt oma turvaprofiilile.

Meie soovime igal juhul järgida selle aruande soovitusi krüptoprimitiivide osas. Mõnikord võib see osutuda problemaatiliseks, sest paljud füüsilised tokenid toetavad ebapiisava turvasemega 2048-bitiseid RSA võtmeid, kuid osa vabavaralisi WebAuthn realiseerimisi ei võimalda RSA võtmetele piiranguid konfigurida.

WebAuthn ökosüsteemi autenturite puudus on nendega seotud käideldavusprobleem: kui autentur või selles olev võtmematerjal peaks hävima, võib see tekitada autenturi omanikule probleeme infosüsteemidesse ligipääsemisel. Kui näiteks mõne asutuse sisekasutuse kontekstis ei ole nii suur mure, sest IT-abi saab kasutajale pärast isiku tuvastamist uue autentimisvõtme registreerida, siis avalike teenuste puhul võib isiku sidumine uue autenturiga olla keeruline – erinevalt näiteks ID-kaardist, mille puhul on olemas sissetöötatud protseduurid uue kaardi väljastamiseks ja isikustamiseks. Seda tuleb WebAuthn rakendamisel kindlasti silmas pidada.

Igal juhul tuleb WebAuthn protokollistikku lugeda selle aruande kirjutamise ajal veel pigem eksperimentaalseks – seda just keerukuse, mahuka dokumentatsiooni ja väheste realiseerimiste tõttu. See on ka põhjus, miks me ei anna täpseid soovitusi krüptoalgoritmide osas – WebAuthn juurutamiseks on igal juhul vaja kaasata spetsialist, kes hoolitseb terviklahenduse eest infosüsteemi enese, selle kasutajate ning potentsiaalselt pruugitavate autenturite koosmõju arvestades.

5.5 SAML

SAML (*Security Assertion Markup Language*)² on XML-põhine turvaväidete – ennekõike autentimis- ja volitusinfo – edastamise standard, mis määrab asjakohased vormingud ja protokollid.

¹<https://www.iana.org/assignments/cose/cose.xhtml#algorithms>

²<https://wiki.oasis-open.org/security/FrontPage>

SAMList on olemas kaks generatsiooni: SAML 1 (versioonid 1.0 ja 1.1) ning SAML 2.0. SAMLi versioone 1.0 ega 1.1 ei tohiks mujal kui pärandüsteemides kasutada, sest nende turvatase on sisuliselt teadmata. Ka OWASP SAML *Security Cheat Sheet*¹ annab soovitusi vaid SAML 2.0 põhjal.

Nimetatud OWASP soovitus peab olema iga SAMLi rakendaja regulaarne lugemisvara, lisaks tuleb pidevalt jälgida kasutatava tarkvara turvauuenduste infot. Ainuüksi 2020. aastal registreeriti vähemalt 27 CVE-baasi kirjet, mis puudutasid kuidagi SAMLi, s.h SAML dokumendi signatuurivalideerimise viga laialt kasutatud raamistikus *Spring Security*.²

SAMLi nõrkuste hulka tuleb kahtlemata lugeda tema baseerumine XMLil: XML korrektne valideerimine on väga keeruline³ ning selle teostamisel on väga lihtne teha vigu [96].

Viimane ametlik täiendus SAML 2.0 tuumikstandardile pärineb aastast 2012, kuigi ka hiljem on avaldatud mõningaid mustandeid ja täpsustusi. SAML 2.1 koostamise tööd on peatunud aastal 2013. Võimalusel tuleks SAMLile eelistada aktiivsemalt arendatud protokollistikke, näiteks OpenID Connect, kuivõrd kogu asjakohane tehnoloogiline keskkond on viimastest uuendustest möödunud aja jooksul radikaalselt muutunud ja see toob iseenesest kaasa riske.

¹https://cheatsheetseries.owasp.org/cheatsheets/SAML_Security_Cheat_Sheet.html

²<https://tanzu.vmware.com/security/cve-2020-5407>

³https://cheatsheetseries.owasp.org/cheatsheets/XML_Security_Cheat_Sheet.html

6 Teegid ja rakendused

Kui rakenduses on tarvis kasutada mingeid krüptograafilisi primitiive, struktuure või protokolle, siis on alati vajalik valida rakenduse keskkonda sobiv kõrge kvaliteediga krüptoteek – keerulisematel juhtudel ka sobitada omavahel krüptoteegi ning arendus- ja käidukeskkondade valikut, sest keskkonnad ei ole krüptograafiliste teekide kättesaadavuses osas võrdsed (vt jaotis 6.3). Paljudel juhtudel ei ole valitud keskkonna jaoks piisavalt universaalseid krüptoteeke üldse olemas ning sel juhul tuleb valida mõni teek, millel on olemas sobiv mähis (tavaliselt on selleks teegiks OpenSSL).

Krüptograafiliste primitiivide või konstruktsioonide teostamine mingit äriprotsessi toetava rakenduse arenduse käigus ei ole üldiselt hea plaan, kuivõrd selline arendus kipub keskenduma vaid ärirakenduse funktsiooniga seotud vajadustele, mitte aga niivõrd teostatud krüptograafia turvalisuse nõuetele (jättes näiteks tähelepanuta algoritmide kõrvalkanalikindluse).

Kindlasti tuleb arendusprojekti jaoks krüptoteegi valikul ja kasutamisel teha järgnevat.

- Tutvuda enne kasutuselevõttu teegi versioonidega ning võimalike paralleelselt eksisteerivate peaversioonidega (*major version*). Käesolev aruanne ei soovita meelega ühtegi kindlat tarkvaraversiooni, sest olukord muutub väga kiiresti ning põhjapanevaid teadustöid sel teemal ei ole.
- Hinnata, kui suur on teegi üldine populaarsus (*Google Trends* on selleks väga hea vahend), vähepopulaarsetel teekidel on kalduvus muutuda jäätvaraks.
- Hinnata, millise kiirusega on arendusmeeskond seni teegile turvaparandusi väljastanud.
- Hinnata, kui suur meeskond teegi arendamisega tegeleb ning kas võib eeldada meeskonna jätkusuutlikkust arendatava projekti eluea piires. Paratamatult on siinkohal eelistatud teegid, millel on mõne suurkorporatsiooni, valitsuse või suurema MTÜ tugi.
- Kasutada alati teegi mõne toetatud haru viimast versiooni ning hoolitseda, et oleks olemas protseduurid kasutatud teekide korraliseks ülevaatuseks ja uuendamiseks. Tähelepanu tuleb pöörata ka sellele, et süsteemne pakihaldus (näiteks APT, RPM) ei rikuks ära arendaja plaane ega paigaldaks toodangukeskkonda dünaamiliselt lingituna vanemat ning vigast versiooni krüptoteegist.
- Pidada silmas, et teegid sisaldavad ühilduvuse huvides ka mitmesuguste selliste primitiivide ja protokollide teostusi, mida ei peaks üheski uues ega uuendatavas süsteemis enam kasutama. Mingi konstruktsiooni olemasolu teegis ei ole põhjus selle kasutamiseks, iga kasutamine tuleb otsustada pädevatele ja värskelele turvahinnangutele tuginedes.

Käesolevas peatükis oleme vaadelnud eelkõige teekide ja arenduskeskkondade viimaseid stabiilseid versioone.

6.1 Ülevaade levinumatest krüptoteekidest

Kokkuvõtlik ülevaade levinumatest krüptoteekidest on tabelis 9.

Tabel 9. Krüptoteekide ülevaade

Nimi	Keel(ed)	RSA-PSS tugi	Soovitatud
OpenSSL	C	Jah	Jah
LibreSSL	C	Jah	Ei
BoringSSL	C	Jah	Ei
GnuTLS	C	Jah	Ei
wolfSSL	C	Jah	Jah
Mbed TLS	C	Jah	Jah
NSS	C	Jah	Jah
Botan	C++	Jah	Ei
s2n	C	Jah	Ei
BouncyCastle	Java, C#	Jah	Jah
Tink	Java, C++, Objective-C, Go, Python	Jah ¹	Jah
Libgcrypt	C	Jah	Jah
Crypto++	C++	Jah	Ei
Nettle	C	Jah	Ei
libsodium	C	Ei	Ei

Võrreldes aruande eelmise versiooniga [110], on tabelile 9 lisandunud veerg “RSA-PSS tugi”. Aruande tulevastes versioonides võib sarnaseid veerge lisanduda teiste krüptokonstruktsioonide jaoks, samuti võib neid jälle ära kaduda, kui küsimus vastava konstruktsiooni toe kohta aktuaalsust kaotab.

Meenutame, et RSA-PSS (*Probabilistic Signature Scheme*) [14] on meetod, mille abil RSA-st konstrueerida tõestatavalt turvaline signatuuriskeem. PSS pakub välja tõenäosusliku polsterdusmehhanismi, nii et sel viisil polsterdatud teatele RSA-permutatsiooni (mille määrab RSA salajane eksponent) rakendamisel järeldeb signatuuride võltsimatus RSA-permutatsiooni (mille määrab RSA avalik eksponent) pööramise (ilma salajast eksponenti teadmata) arvutuslikust keerukusest. Turvalisuse tõestatavus on see aspekt, mille poolest PSS erineb laiemalt levinud PKCS#1 v1.5 polsterdusest [102]. Tänu oma lihtsamale realiseerimisele (näiteks ei ole tarvis juhuarve genereerida) on viimane neist ikka veel populaarsem, kui peaaegu 20 aastat tagasi standardiseeritud PSS [101].

Meenutame ka, et PKCS#1 v1.5 polsterdusest *signeerimise* jaoks ei ole leitud olulisi turvanõrkusi, vastandina samas dokumendis väljapakutud polsterdusest *krüpteerimise* jaoks [26]; see võib olla seletus RSA-PSS kasutuselevõtu aeglusele. Mainime veel, et hiljuti on ilmunud turvatõestused polsterdustele, mis ei ole päris samad, mis PKCS#1 v1.5 polsterdus, kuid omavad sellega olulisi sarnasusi [95].

¹Arendusseis. Viimases stabiilses versioonis 1.7 polnud RSA-PSS tuge veel Go teegis.

6.1.1 OpenSSL

Alustatud 1998. aastal selleks hetkeks juba kolm aastat vana teegi SSLeay baasil, on OpenSSL¹ kasutuselolevatest laiatarbe krüptoteekidest pikima ajalooga ning tuntuim krüptograafiat pakkuv teek maailmas. OpenSSL teostab suure hulga krüptoprimitiive (räisialgoritmid, sümmeetrilised ja asümmeetrilised krüptoalgoritmid), vorminguid (näiteks X.509 ja CMS) ja protokolle (SSL, TLS, DTLS). Kuigi OpenSSL oli vaba tarkvarana pikalt väga laialdast kasutust leidnud, oli selle arendusaktiivsus vähese välise toetuse ja vähese rahastuse tõttu 2014. aastani suhteliselt tagasihoidlikuks jäänud. Häirekellaks kasutajatele sai mitmete turvanõrkuste (nagu näiteks Heartbleed²) avastamine, pärast mida on märkimisväärselt kasvanud nii leitud turvanõrkuste arv, OpenSSL arenduse tempo kui ka projekti rahaline toetus, sealhulgas paljude suurfirmade poolt. Mitmete uute täiskohaga arendajate abiga lasti 2021. aasta septembris välja ka uus versioon 3.0, mis on Apache 2.0 litsentsi all, ent jätkuvalt toetatakse veel ka vanemat OpenSSL 1.1.1 seeriat.

Kuigi üleminek OpenSSL versioonile 3.0 ei tähenda ilmtingimata palju muudatusi seda kasutavate programmide lähtekoodis, on siiski muutunud mitmed programmeerimisliidesed, masinkoodiliidesed ja ka mitmed olulised vaikekäitumised. Seetõttu tuleks uuele versioonile üleminekul OpenSSL teeki kasutavad lähtekoodid lähtuvalt OpenSSL migreerimisjuhendis³ kirjeldatud muudatustele kriitiliselt üle vaadata ja vajadusel korrigeerida. Eriliselt tähelepanelik tuleb olla nende muudatuste osas, mis ei pruugi ehitus- ja käitusaegselt otseseid vigu anda, kuid millest tulenevalt võib programmi käitumine keeruliselt tuvastatavalt ebakorrektselt muutuda. Näiteks on oht võimaldada mälu turvalisuse rikkumist kui ignoreerida OpenSSL versiooniga 3.0 lisatud nõuet teatud EVP_PKEY_ nimeruumi funktsioonide tagastusväärtusi mitte muuta. Soovitame võimaluse korral lõpetada kõikide taunitud liideste kasutamine, mida ka OpenSSL versiooniga 3.0 rohkesti lisandus.

6.1.2 LibreSSL

LibreSSL⁴ on OpenBSD poolt loodud OpenSSL kloon, mis tehti 2014. aastal pärast Heartbleed nõrkuse avastamist, põhjuseks rahulolematuse OpenSSL lähtekoodi kvaliteediga. LibreSSL arendajad tegid mitmeid olulisi turvaparandusi, millest osa viidi ka OpenSSL koodibaasi tagasi. Esialgne optimism LibreSSL kui OpenSSL alternatiivi suhtes kadus, kui ilmnid mitmed tõsised ühilduvusprobleemid seoses LibreSSL kasutuselevõtuga ning sai selgeks, et laiemalt toetatakse siiski OpenSSL projekti.⁵ Seetõttu on paljud projektid hüljanud katsed toetada ka LibreSSL teeki ning suur osa Linuxi distributsioone LibreSSL pakke ei levita. LibreSSL on ka toetatud funktsioonide osas OpenSSL teegist oluliselt maha jäänud (näiteks TLS 1.3 täielik tugi tekkis OpenSSL teegist kaks aastat hiljem) ning selle arendusaktiivsus on madalam kui OpenSSL teegil. Olles alguse saanud OpenSSL teegi pordist OpenBSD platvormile, käib LibreSSL arendus peamiselt OpenBSD CVS koodihoidlas. Toetamiseks ka teisi platvorme peale OpenBSD kasutatakse LibreSSL Portable⁶ nimelist ehitusraamistikku, mis lisab ka puuduolevad ühilduvuskihid.

¹<https://www.openssl.org/>

²<https://heartbleed.com/>

³https://www.openssl.org/docs/man3.0/man7/migration_guide.html

⁴<https://www.libressl.org/>

⁵<https://lwn.net/Articles/841664/>

⁶<https://github.com/libressl-portable/portable>

6.1.3 BoringSSL

BoringSSL¹ on Google'i sisemiseks tarbeks loodud OpenSSL kloon, mille lähtekood avalikustati mõned kuud pärast Heartbleed nõrkuse avastamist OpenSSL teegis. BoringSSL arendajate sõnul kasutatakse teeki küll Chrome/Chromium veebilehitsejas, Androidi osades ja muudes Google rakendustes, ent selle kasutamine kolmandate osapoolte poolt ei ole soovitatav. Arvestades BoringSSL kloonis tehtud muudatusi, erinevusi OpenSSL teegiga, arenduse hetkeseisu ja pigem langustrendis arendusaktiivsust võib selle professionaalse hinnanguga igati nõustuda.

6.1.4 GnuTLS

GnuTLS² on vabavaraline TLS ja DTLS teostus, mis loodi 2003 aastal põhjusel, et OpenSSL litsents polnud GPL litsentsiga ühilduv. Kuigi GnuTLS katvus funktsionaalsuse mõttes on võrreldav OpenSSL teegiga ja selle liidesed on programmeerijatele ehk kohati mugavamad, jääb ta oma populaarsuselt siiski OpenSSL järel teiseks. Madalama populaarsuse üheks märgiks võib olla ka väiksem arv CVE-kirjete arv³ kui OpenSSL teegil. Siinjuures on märkimisväärne, pärast Heartbleed nõrkuse avastamist OpenSSL teegis ei laienenud sarnane tähelepanu GnuTLS teegile. GnuTLS projektil on Red Hat toetus, kelle ridadesse kuulus nii GnuTLS üks loojatest Nikos Mavrogiannopoulos kui ka praegune peamine koodihaldur Daiki Ueno. Samas võivad just Red Hat sisesed suunamuutused olla põhjuseks, miks GnuTLS koodihoidla statistika alusel on arendusaktiivsus kalendriaastate lõikes viimasel neljal aastal olnud poole aeglasem kui neile eelnenud kaheksal aastal. Kuigi neis asjaoludes pole veel midagi kriitilist, on OpenSSL jõudsad arengud pärast Heartbleed nõrkuse avastamist siiski tugevaks põhjuseks, miks eelistada pigem OpenSSL teeki, mille viimastel versioonidel on ka sobilikum Apache 2.0 litsents.

6.1.5 wolfSSL

wolfSSL⁴ on nii vabavaralise kui ka kommerts litsentsi all pakutav kergekaaluline TLS ja DTLS teostus, mis on orienteeritud kasutusele manusseadmetes. wolfSSL on ristkompileeritav paljudele riistvaraarhitektuuridele ja operatsioonisüsteemidele ning pakub ka OpenSSL teegiga ühilduvat liidest. wolfSSLi (algsest CyaSSL) on arendatud alates aastast 2004, tema arendusaktiivsus on vähemalt viimase kümne aasta jooksul järjepidavalt kasvanud ning tema arendajad on olnud kiired lisama tuge uutele standarditele ja algoritmidele, sealhulgas postkvant-krüptograafiale. Siin nimekirjas olevatest teekidest pakub ta hetkel ainsana DTLS 1.3 tuge. wolfSSL madalama taseme krüptograafiliste primitiivide moodulil wolfCrypt on ka FIPS 140-2 sertifikaat. wolfSSL nõrkuseks võib siiski lugeda tema suhteliselt vähest populaarsust, mille üheks põhjuseks võib olla ka see, et wolfSSL teeki pakutakse vaid GPL 2.0+ ja kommerts litsentsi all.

6.1.6 Mbed TLS

Mbed TLS⁵ on TLS ja SSL teostusi pakkuv teek. Mbed TLS juured on Christophe Devine poolt loodud XySSL nimelises teegis, mille arendus jätkus pärast soikumist eraldiseisva

¹<https://boringssl.googlesource.com/boringssl/>

²<https://www.gnutls.org/>

³<https://www.cvedetails.com/product/4433/GNU-Gnutls.html>

⁴<https://www.wolfssl.com/products/wolfssl/>

⁵<https://www.trustedfirmware.org/projects/mbed-tls/>

PolarSSL nimelise projektina. 2014. aastal ostis PolarSSL teegi ära Arm ning nimetas selle ümber oma Mbed manusseadmete platvormi järgi. Mbed TLS sisaldab ka Arm seadmete turvaarhitektuuri PSA (Platform Security Architecture) API referentsteostust. Kuigi Apache 2.0 litsentsiga Mbed TLS on populaarne eelkõige manusseadmetes, on leidnud ta kasutust ka mitmetes teistes sortisvara programmides, ning on saadaval enimkasutatud Linuxi distributsioonides. Mbed TLS kiiret arendust veavad eest eelkõige Arm enda arendajad, kuid arenduses osalevad ka mõned teised manusseadmete tarkvaraga tegelevad firmad. Aastatel 2021-2022 tehti Mbed TLS koodihoidlasse enam kui 2000 koodikehtestust rohkem kui OpenSSL koodihoidlasse.

6.1.7 NSS

NSS ehk Network Security Services¹ on TLS, SSL, PKCS#5, PKCS#7, PKCS#11, PKCS#12, S/MIME ja X.509 v3 teostusi pakkuv teek. NSS koodi juured ulatuvad üheksakümnendatesse, olles saanud alguse Netscape veebilehitsejate koodist, mis pärast Netscape Communicator 4.0 väljalaset pakendati esmalt teegiks nimega HCL ("Hard Core Library"), mille versioon 2.0 sai juba nimeks NSS ning tehti kättesaadavaks ka väljaspoole Netscape'i. Hiljem on NSS teeki arendatud eelkõige Mozilla kommuuni poolt, ning see on kasutuses muuhulgas veebilehitsejas Mozilla Firefox, meilikliendis Mozilla Thunderbird ning mitmetes dokumentide signatuure töötlevates sortisvarades. Kuigi NSS näib olevat kõrge kvaliteediga, pole ta laiemalt siiski väga populaarne ning ka NSS pigem tagasihoidlikku arendustempot veavad eest suuresti ainult Mozillaga seotud arendajad.

6.1.8 Botan

Botan on Jack Lloyd poolt loodud TLS teostus, mille esimene avalik väljalase oli aastal 2001. Kuigi tegu pole väga populaarse teegiga, on Botan siiski saadaval enimkasutatud Linuxi distributsioonides. Botan on olnud BSI huviorbiidis vähemalt alates aastast 2007, mil FlexSecure GmbH abiga lisati teegile Card Verifiable Certificates (CVC) tugi esmalt InSiTo nimelise teegi näol.² Ka BSI projekti 197³ raames, mille eesmärgiks oli kvaliteetse üldotstarbelise krüptoteegi loomine vastavalt BSI nõuetele, valiti 18 kandidaatteegi seest arenduse baasiks välja just Botan⁴, mis viidi hiljemalt 2017. aasta esimeses kvartalis vastavusse BSI nõuetele. Projekti kolmas faas nägi ette ka Botan teegi hooldamist ja ajakohasena hoidmist mitmete aastate jooksul vastavalt arengutele krüptograafias, standardites ja rakendusstsenaariumites. BSI projekti 197 alltöövõtja Rohde & Schwarz on jätkanud Botan teegi arendustöid ka uurimisprojekti KBLS⁵ raames. Hoolimata BSI ja tema partnerite abist on Botan teegi arendus jätkuvalt suures osas Jack Lloyd õlgadel. Üheski stabiilses Botan teegi versioonis pole veel TLS 1.3 tuge, kuid see on lisamisel. Positiivseks võib pidada Botan teegi üleminekuid uuematele C++ keele standarditele ning vastavate turvalise C++ programmeerimise praktikate järk-järgulist kasutuselevõttu.

¹<https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS>

²[https://en.wikipedia.org/w/index.php?title=Botan_\(programming_library\)&oldid=1055456913](https://en.wikipedia.org/w/index.php?title=Botan_(programming_library)&oldid=1055456913)

³https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kryptografie/Kryptobibliothek-Botan/kryptobibliothek-botan_node.html

⁵https://www.rohde-schwarz.com/de/loesungen/cybersicherheit/about-us/research/kbbs_254244.html

⁵Teised kandidaadid olid BeeCrypt, cryptlib, Crypto++, GnuTLS, Libgcrypt, LibreSSL, libsodium, LibTomCrypt, MatrixSSL, Mbed TLS, NaCl, Nettle, NSS, OpenSSL, POCO, s2n ja wolfSSL. Esimeses analüüsivoorus hinnati teekide funktsionaalsuse olemasolu lähtuvalt BSI juhistes toodud soovitudele,

6.1.9 s2n

s2n¹ (algsest SignalToNoise) või s2n-tls on Amazoni loodud ja 2015. aastal avalikustatud TLS ja SSL teostus, mis on eelkõige mõeldud serveritele ja mida Amazon ka enda pilveteenustes kasutab. s2n arenduse poliitika sisaldab mitmeid kasulikke mehhanisme teegi turvalisuse tagamiseks. Nende hulgas on ka meetmeid koodi hulga väiksena hoidmiseks, mistõttu ei teosta s2n mitmeid TLS laiendusi ega krüptoprimitiive. Krüptoprimitiivide jaoks kasutab s2n kas OpenSSL, LibreSSL, BoringSSL teeki või Apple Common Crypto raamistikku. Kuigi s2n ei paku otseseid liideseid krüptoprimitiivide iseseisvaks kasutamiseks, võib keskendumist TLS ühendustele pidada ka tugevuseks. s2n dokumentatsioonis leidub väide, et s2n läbib korralisi turvaanalüüse valitsuste ja äriorganisatsioonide poolt, kuid täpsem info nende analüüside kohta peitub konfidentsiaalsuslepingute taga. Väline huvi s2n vastu näib olevat pigem piiratud, ning kuna s2n teegist sõltuvaid projekte on vähe, puudub s2n ka paljudest Linuxi distributsioonidest. Sellest hoolimata on tegu paljutöotava projektiga, mille elujõulises arenduses on aktiivselt kaasa aidanud ka mitmed arendajad, kellel puudub nähtav seos Amazoniga. Ettevaatlikuks teeb s2n puhul siiski tema vähene populaarsus, mistõttu ei pruugi ta olla saanud piisavalt kriitilist tähelepanu ning välitestimist.

6.1.10 Bouncy Castle

Bouncy Castle² näol on tegu kahe vabavaralise krüptoteegiga vastavalt Java ja C# keeltele, mis teostavad paljusid krüptoprimitiive, vorminguid kui ka protokolle. Kuigi funktsionaalsuse osas on mõlema Bouncy Castle teegi katvus võrreldav OpenSSL teegiga, jääb Bouncy Castle populaarsuse poolest OpenSSL teegile siiski tugevalt alla. Bouncy Castle C# teeki ei paku siiski päris kõike Bouncy Castle Java teegi funktsionaalsusest, ning ka selle arendusaktiivsus on madalam. Mõlemas keeles või õigemini mõlemal platvormil (JVM ja .NET) on tegu *de facto* peamiste krüptoteekidega, kui vastavatesse platvormidesse sisse ehitatud funktsionaalsus välja arvata. JVM platvormil töötab Bouncy Castle eelkõige JCA ja JSSE pistikteegina (vt jaotis 6.3.2).

6.1.11 Tink

Google poolt arendatud Tink³ on kõige uuem projekt siin nimekirjas (versioon 1.0.0 avaldati septembris 2017), mis pakub programmeerimise headest tavadest lähtuvalt võimalikult turvaliseks kasutuseks abstraheeritud liideseid põhiliste krüptograafiliste operatsioonide (AEAD, MAC, PRF, digitaalsignatuurid, hübriidkrüpteerimine) ja võtmehalduse teostamiseks. Tink käsitlusalasle ei kuulu madalama taseme primitiivid ega ka kõrgema taseme protokollid ja vormingud. Õigemini on Tink näol tegu mitme erineva teegiga eri programmeerimiskeelte jaoks (Java, C++, Objective-C, Go, Python, eelvaate kvaliteediga on olemas ka JavaScript/TypeScript tugi), mis vahetarkvarana kasutavad sisemiselt vastava platvormi krüptograafilisi primitiive OpenSSL, BoringSSL, BouncyCastle, Conscrypt, Oracle JDK,

arendusaktiivsust, kvaliteeti, dokumentatsiooni, teste, porditavust, arendusaktiivsust, litsentsivalikut ja väljalasete tsükli sobivust kommertskasutuseks. Esimeses voorus suurimad punktisummad saanud Botan, LibreSSL ja NSS läbisid ka teise, detailsema analüüsivooru, kus valideeriti muuhulgas X.509 teostusi, hinnati vastupidavust kõrvalkanalirünnetele ja teostuse modulaarsust. Kuigi Botan oli kõrvalkanalirünnetele suhtes haavatavam kui LibreSSL ja NSS, peeti Botan teeki projekti eesmärkide saavutamiseks oluliselt paremaks tema disaini, modulaarsuse, koodikvaliteedi, testide katvuse ja BSI nõuetele vastavuse tõttu.

¹<https://github.com/aws/s2n-tls>

²<https://www.bouncycastle.org/>

³<https://github.com/google/tink>

OpenJDK, Go crypto ja Web Crypto API teekidest või teostustest. Sedasi võivad viimaste nõrkused mõjutada ka Tink teekide turvalisust. Tink teekide suurimaks tugevuseks võib lugeda programmeerimisliideseid, mida on keeruline ebaturvaliselt kasutada ning mille kasutust on lihtne auditeerida.

6.1.12 Libgcrypt

Libgcrypt¹ on GnuPG lähtekoodist lahku löödud krüptoprimitiivide teek, mis teostab sümmeetrilisi ja asümmeetrilisi krüptoalgoritme, räsialgoritme, sõnumiautentimiskooode, võtmetuletusalgoritme, juhuarvugeneraatoreid ning vahendeid S-avaldiste, suurte arvude ja algarvude käsitlemiseks. Kuigi Libgcrypt kood kuulus algselt GnuPG alla, on ta eraldiseisva teegina leidnud praeguseks kasutust paljudes sortisvara projektides. Aastatest 2020-2022 katavad selle tähtsamaid primitiive ka paljud FIPS 140-2 sertifikaadid. Libgcrypt teegil on mitmeid arendajaid ja arendustempo on aastatel 2021-2022 olnud kiirenev.

6.1.13 Crypto++

Crypto++² on Wei Dei poolt loodud krüptograafiliste algoritmide teek, mille esimene avalik väljalase toimus aastal 1995. Crypto++ pole leidnud väga laialdast kasutuselevõttu, hoolimata sellest, et aastatest 2003-2007 on Crypto++ versioonidel 5.0.4, 5.2.3 ja 5.3.0 ka FIPS 140-2 sertifikaadid mõningatele algoritmidele, ning et teegile on olnud saadaval ka kommertstugi. Sellest hoolimata on Crypto++ saadaval enimkasutatud Linuxi distributsioonides olnud populaarne eelkõige akadeemilistes ringkondades, sest teostab laia valiku krüptograafilisi algoritme, sealhulgas ka mitmeid vähemtuntud algoritme. Crypto++ arendus tundub olnud olevat Wei Dei ühemeheprojekt, mis jõudis aastatel 2012-2014 varjusurma, kuid võeti 2015. aastal jõudsalt üle Jeffrey Walton poolt, kelle nimel on igal aastal tehtud pea kümme korda rohkem koodimuudatusi. Sellest hoolimata on Crypto++ arendus viimasel paaril aastal olnud pigem taas hooldusrežiimil.

6.1.14 Nettle

Nettle³ on krüptoprimitiivide teek, mis teostab sümmeetrilisi ja asümmeetrilisi krüptoalgoritme, räsialgoritme, sõnumiautentimiskooode, võtmetuletusalgoritme, juhuarvugeneraatorid ja AEAD teostust. Nettle leiab kasutust mitmetes sortisvara projektides, ent on pigem vähepopulaarne. Nettle küllaltki stabiilne arendustempo ei näita hääbumise märke, kuid see on väga suures osas vaid tänu Nettle looja Niels Mölleri tööle.

6.1.15 libsodium

Libsodium⁴ või lihtsalt Sodium on edasiarendus Daniel J. Bernsteini krüptoteegist NaCl, mis ise on tänaseks jäätvaraks muutunud. Libsodium pakub põhilisi madala taseme krüptoprimitiive, millest keerulisemate konstruktsioonide loomine nõuab vastavat pädevust. Libsodium on ristkompileeritav paljudele platvormidele, sh JavaScript ja WebAssembly, ja teeki on mähitud ka paljude levinud programmeerimiskeelte jaoks.⁵ Murettekitavaks

¹<https://gnupg.org/software/libgcrypt/>

²<https://www.cryptopp.com/>

³<https://www.lysator.liu.se/~nisse/nettle/>

⁴<https://doc.libsodium.org/>

⁵https://doc.libsodium.org/bindings_for_other_languages

võib lugeda, et Libsodium koodihoidlas, mille ajalugu ulatub 2013 aasta jaanuarini, on muudatuste arv on viimastel aastatel näidanud langustrendi ning üle 92% kõikidest tehtud muudatustest on projekti algataja Frank Denis' nimel. Teiste autorite tehtud sisulisi muudatusi krüptograafilises koodis on vähe. Ka libsodiumi populaarsus pole suur ning ei Nacl ega ka libsodium kohta pole ühtegi CVE kirjet, mistõttu võib kahtlustada, et ka selle kood pole piisavalt välist tähelepanu saanud.

6.2 RSA-PSS tugi krüptoteekides

Enamus ülalkirjeldatud teekidest toetavad RSA-PSS formaadis signatuuride loomist, nagu näeme tabelist 9. OpenSSL toetas seda formaati juba enne tema kloonide (LibreSSL ja BoringSSL) loomist. Selle kasutamiseks tuleb vastavad RSA-PSS võtmed luua funktsiooni `EVP_PKEY_CTX_new_id` ja võtmetüübi `EVP_PKEY_RSA_PSS` abil. Loodavate võtmete parameetreid saab muuta `EVP_PKEY_CTX_set_rsa_pss_keygen_md`, `EVP_PKEY_CTX_set_rsa_pss_keygen_mgf1_md` ja `EVP_PKEY_CTX_set_rsa_pss_keygen_saltlen` funktsioonidega.

GnuTLS teegis on RSA-PSS signatuuriskeemiga signeerimiseks võimalik kasutada funktsiooni `gnutls_privkey_sign_hash` lipuga `GNUTLS_PRIVKEY_SIGN_FLAG_RSA_PSS`. RSA-PSS võtmete võtmeparameetreid saab muuta läbi `gnutls_x509_spki_t` struktuuri, mille peal peab kutsuma välja funktsiooni `gnutls_x509_spki_set_rsa_pss_params` ning mis seostatakse võtmetega kasutades erinevaid `gnutls_*_set_spki` nimedega funktsioone.

wolfSSL teegis saab RSA-PSS signatuuriskeemi kasutada funktsioonide `wc_RsaPSS_Sign` ja `wc_RsaPSS_Verify` abil. Bouncy Castle teegis on RSA-PSS signatuuriskeem kasutatav moodulisse `org.bouncycastle.crypto.signers` kuuluva klassi `PSSSigner` abil.

RSA-PSS signatuuriskeem on ka toetatud Mbed TLS teegis alates versioonist 0.99-pre4 (kui teegi nimeks oli veel PolarSSL), NSS teegist alates versioonist 3.22 (tugi TLS-is lisandus alates versioonist 3.24 ning tugi sertifikaatides ajates versioonist 3.34), Botan teegis alates versioonist 2.4.0 (tugi TLS-is lisandus alates versioonist 2.5.0), s2n teegis alates versioonist 0.10.0 (TLS-i jaoks), Libgcrypt teegis alates versioonist 1.5.0, Crypto++ teegis alates versioonist 2.1 (uuendatud versioonides 3.0 ja 5.1 lähtuvalt IEEE standardist 1363a-2004 [91]), Nettle alates versioonist 3.4.

Tink teegid toetavad RSA-PSS signatuuriskeemi, kui signeerida vastavalt genereeritud RSA-PSS võtmete abil. Kahjuks pole vastav tugi veel jõudnud üheski stabiilses versioonis Go keele jaoks kirjutatud Tink teeki. Libsodium teegil pole RSA-PSS signatuuriskeemi tuge. Funktsioon `crypto_sign` kasutab signeerimiseks hoopis EdDSA signatuuriskeemi realisatsiooni Ed25519.

6.3 Arenduskeskkonnad

6.3.1 C ja C++

Programmeerimiskeeltesse C ja C++ ega nende standardteekidesse ei ole sisse ehitatud krüptograafilisi vahendeid, mistõttu tuleb neis keeltes kasutada väliseid krüptograafiat pakkuvaid teke. Võrreldes teiste arenduskeskkondadega leidub selliseid samas programmeeri-

miskeeles arendatud teeke suhteliselt palju, ning mitmeid nende hulgast kirjeldab ja soovitab ka see aruanne jaotises 6.1.

6.3.2 Java

Krüptograafiliste meetodite kasutamine Java (JVM) platvormil toetub *Java Cryptography Architecture* (JCA) standardile, mis on Java platvormi normatiivne osa. JCA sätestab krüptoprimitiivide kasutamise liidesed ning sisaldab ka osa liideste teostusi.

Kuni Java versioonini 1.3 olid krüpteerimise ja võtmekehtestusega tegelevad liidesed defineeritud eraldi *Java Cryptography Extension* (JCE) nime all, kuid hilisemates versioonides seda lahusust enam ei ole. JCE nimi on endiselt kasutuses, kuid tihtipeale lihtsalt JCA sünonüümina.

JCA defineerib oluliselt suurema hulga primitiive kui ta ise teostab ning seetõttu on tihtipeale vaja kasutada kolmanda osapoole pistikteeke (*provider*), näiteks Bouncy Castle (vt jaotis 6.1.10). Iga JCA versiooni poolt defineeritud primitiivide loetelu on võimalik leida JCA vastava versiooni dokumentatsiooni hulgast (*Java Cryptography Architecture Standard Algorithm Name Documentation*, *Java Security Standard Algorithm Names*).

JCA juurde käib eraldi arhitektuurse moodulina *Java Secure Socket Extension* (JSSE), mis sätestab SSL, TLS ja DTLS kasutamise liidesed Java platvormil. JSSE projekteerimisprintsiibid on samad kui JCA-l.

JCA ning JSSE dokumentatsioon on põhiline alguspunkt arendajale, kellel on vaja Java (JVM) rakendustes krüptograafilisi primitiive või turvalisi sideprotokolle kasutada.

Tuleb hoolitseda, et nii Java käigukeskkond (*runtime environment*) kui pistikteegid oleks alati uuendatud ja turvanõrkused eemaldatud.

6.3.3 Go

Go standardteek sisaldab paketti `crypto` koos selle alampakettidega, milles on realiseeritud väike valik krüptoprimitiive, X.509 sertifikaatide objektimine ja jadastus ning osaliselt ka TLS 1.2 ja TLS 1.3 protokollid. Lisaks on täiendav komplekt primitiive ja protokolle realiseeritud eraldi moodulis, mis asub nimeruumis `golang.org/x/crypto`. Neil põhineb ka Google'i enda poolt Go jaoks arendatud krüptoteek nimega Tink (vt jaotis 6.1.11).

Go jaoks leidub ka mitmeid mähiseid ka OpenSSL teegile, ent kahjuks katavad need vaid osi OpenSSL funktsionaalsusest ning aruande kirjutamise ajal polnud neist ühegi näol tegu väga elujõulisena näiva projektina. Samas on C teekide liidestamine Go programmidega suhteliselt lihtne (see on keele Go osa), mistõttu on võimalik ka otse OpenSSL või mõnda teist C teeki kasutada. Muudel juhtudel soovitame jääda kasutama standardteegi paketti `crypto`, moodulit `golang.org/x/crypto` või teeki Tink.

6.3.4 Node.js

Node.js standardteek sisaldab mooduleid `crypto` ja `tls`, mis mähivad osa Node.js käivituskeskonnaga vaikumisi paigaldatava OpenSSL teegi krüptoprimitiividest ja TLS teostusest. Mitmete algoritmide ja protokollide jaoks, mida standardteek ei paku, leidub pakihaldussüsteemi *npm* pakettidena levitatavad mooduleid, ent nende kvaliteet võib olla

varieeruv. Kuna JavaScript pole väga riistvaralähedane jooksutuskeskkond, võib selles realiseeritud krüptograafilistel algoritmidel olla probleeme kõrvalkanalikindlusega.

Node.js pakub ka Node-API (varem N-API) nimelist laiendusliidest JavaScript moodulite ja muu funktsionaalsuse lisamiseks kasutades C++ keelt, mille abil on võimalik liidestada ka C ja C++ teek, ent selle kasutamine on pigem keeruline. Kuivõrd OpenSSL on juba vaikimisi osaliselt liidestatud, ei ole selle järele krüptograafia kontekstis ka suurt vajadust.

6.3.5 Rust

Rust keele standardteek ei sisalda krüptograafilisi algoritme ning nende teostamine on jäetud välistele teekidele, mis on üldiselt kättesaadavad ka pakihaldussüsteemi *Cargo* kaudu. Seetõttu on väliseid krüptoalgoritmide teostusi palju^{1,2} ning nende kvaliteet ei ole ühtlane [127].

Eraldi võib ära märkida siiski OpenSSL mähise³ ning Rustls⁴ nimelise TLS 1.2 ja TLS 1.3 teostuse. Jätkuvalt on arendusjärgus ka värske Tink teegi (vt jaotis 6.1.11) mitteametlik Rust versioon,⁵ millega tehti algust septembris 2020.

Programmeerimiskeel Rust, mille projekteerimiseesmärkide hulga kuuluvad rangem tüübi- ja mäluurvalisus, võimaldab küll välistada teatud vigade klassid ent pole siiski imevahend kõikide programmeerimisvigade vastu. Seetõttu ei tohi ka Rust keeles kirjutatud koodilt alati automaatselt paremat kvaliteeti oodata. Ometi tasub hoolikalt kaaluda Rust keele kasutamist komponentides, mille funktsionaalsusest olulise osa moodustab krüptograafiliste algoritmide kasutamine.

6.3.6 PHP

PHP pakub rakendustele mõningaid krüptograafilisi laiendusi,⁶ millest kõige märkimisväärsim on OpenSSL mähis, mis võimaldab kasutada mõningaid OpenSSL funktsioone. Juhul, kui PHP rakenduses on vaja krüptograafilisi funktsioone kasutada, soovitame eelkõige pöörduda vastavate mähise poole.

Leidub ka mitmeid mittestandardseid PHP krüptoteeke, ent nende kvaliteet on varieeruv. Neist üks populaarseim on *phpseclib*,⁷ mis pakub mõningaid sümmeetrilisi ja asümmeetrilisi krüptoalgoritme, X.509 ja SSH tuge. Kuna PHP pole väga riistvaralähedane arendusplatvorm, võib selles realiseeritud krüptograafilistel algoritmidel olla probleeme kõrvalkanalikindlusega.

6.3.7 .NET

.NET platvormil (mis praktikas tähendab enamaltjaolt C#-keelt) on süsteemne nimeruum *System.Security.Cryptography*, milles sisalduvad põhiliste krüptograafiliste primitiivide teostused. Eksisteerivad ka süsteemsed vahendid SSL/TLS ja

¹<https://cryptography.rs/>

²<https://blog.logrocket.com/rust-cryptography-libraries-a-comprehensive-list/>

³<https://crates.io/crates/openssl>

⁴<https://github.com/rustls/rustls>

⁵<https://github.com/project-oak/tink-rust/>

⁶<https://www.php.net/manual/en/refs.crypto.php>

⁷<https://phpseclib.com/>

HTTPS ühenduste loomiseks, näiteks kasutades `System.Net.Http.HttpClient`, `System.Net.HttpWebRequest`, `System.Net.Security.SslStream` jt klasse. Kui vaja on midagi enam, tasub ennekoike uurida, kas see on toetatud vabavaralises BouncyCastle teegis (vt jaotis 6.1.10).

6.3.8 Python

Python keele standardteek sisaldab mitmeid krüptograafilisi funktsioone pakkuvaid mooduleid.

- Moodul `hashlib` nimeline moodul pakub räsifunktsioone ning PBKDF2 (vt jaotis 2.4.2) ja `scrypt` (vt jaotis 2.4.3) võtmetuletusfunktsioone.
- Moodul `hmac` pakub räsipõhiste sõnumiautentimiskoodide [109] funktsioone.
- Moodul `secrets` pakub vahendeid krüptograafiliselt tugevate juhuarvude genereerimiseks.
- Moodul `ssl` pakub TLS/SSL funktsionaalsusega sokleid.

Kuna Python standardteegi pakutud vahendid on piiratud, on alates aastast 2001 arendatud ka eraldiseisvat `pyOpenSSL`¹ teeki, mis on OpenSSL mähis. Samal põhjusel lõi grupp Python arendajaid 2013. aastal eraldiseisva `cryptography`² teegi, mis pakub mitmete krüptoprimitiivide teostusi, ning koondus organisatsiooniks nimega *Python Cryptographic Authority* (PyCA),³ mille alla on aruande kirjutamise ajaks liikunud ka `pyOpenSSL` teegi arendus. Kui Python standardteegi pakutavatest krüptograafilistest vahenditest ei piisa, soovitame kasutada eelkõige just `cryptography` teeki, eelistades seda ka `pyOpenSSL` teegile. Viimast tuleks selle arendajate tugeval soovitusel kasutada ainult TLS ühenduste jaoks.

¹<https://pyopenssl.org/>

²<https://cryptography.io/>

³<https://github.com/pyca/>

7 Eesti ID-kaardi NFC-tugi

NFC (Near Field Communication) ehk lähiväljatehnoloogia võimaldab kiipkaartidel teha kontaktivabaid toiminguid kuni 10cm kauguselt. On näidatud, et seda kaugust on võimalik võimendada kuni 50 sentimeetrini [106]. Kuna kaardi lugeja võib suhtlust omakorda edastada, siis peab NFC'd käsitlema kui avalikku turvamata kanalit. Kaardi ja rakenduse vahelise turvalise kanali loomine sõltub kaardi poolt toetatud tehnoloogiast.

Eesti ID-kaardid, mis on väljastatud pärast 2018. aasta novembrikuud, toetavad ISO/IEC 14443 Type A lähiväljatehnoloogiat. Kaardil oleva IAS-ECC appletiga krüpteeritud suhtluse loomiseks on võimalik kasutada PACE protokollide raamistikku, mis on parooliga autenditud Diffie-Hellmani võtmevahetusskeem [2]. Parooliks on kaardil olev 6-kohaline Card Access Number (CAN), millega autenditakse mõlemad osapooled. Krüpteeritud kanali loomiseks genereeritakse ühine võti, mille konfidentsiaalsus ei sõltu paroolist.

Eesti ID-kaardiga suudab läbi PACE turvalist kanalit luua JMRTD¹ teek. JMRTD on Javas kirjutatud LGPL litsentsiga teek, mis sisaldab ICAO masinloetavate reisidokumentide standardite implementatsiooni.

Turvaanalüüs näitab, et PACE rahuldab Abdalla jt. *real-or-random* turvanõuet [1] kus ründav osapool, ligipääsuga mitmele samaaegsele protokollisessioonile, ei suuda eristada päris võtmeid juhuslikust müra [16]. Tänu sellele ei ole võimalik ründajal varem nähtud protokollisessiooni põhjal parooli murda. Protokollianalüüs kasutab mõningaid harilikke abstraktsioone: eeldatakse ideaalset käitumist räsifunktsioonilt (*random oracle model*) ning plokkšifrit (*ideal cipher model*).

Turvaanalüüs näitab ka, et ID-kaart kasutatuna mobiiltelefonist üle NFC liidese, kusjuures teated ID-kaardi ja telefoni vahel liiguvad läbi turvalise kanali, on sobilik vahend ID-kaardi omaniku autentimiseks ja digiallkirjade loomiseks [6, Ptk. 3–5]. Analüüs eeldab, et kaardiga suhtlev mobiilirakendus teostab teatud turvameetmed. Riskianalüüs suudab sellisel juhul tuvastada vaid riske, mille realiseerumistõenäosus on väike või mis on olemas ka ID-kaardi kasutamisel üle USB-liidese. Juriidiline analüüs [6, Ptk. 6] näitab, et NFC-liidese abil loodud digiallkirjad on euroallkirjad (**QES** – *qualified electronic signature*).

¹<https://jmrtid.org/>

Summary in English

This is an update to the seventh report in the series of reports on cryptographic algorithms and their life cycle. The report covers six topics. Its intended audience are managers and practitioners both in IT and in software development. As usual, the report describes several algorithms, protocols, formats and their secure usage. It takes a closer look at some of the protocols and the development tools for applications that use cryptography. The most important point is still that for practical evaluation of use of cryptography in any single context, one should hire a security engineer.

The second chapter focusses on cryptographic primitives and contains current recommendations on key length and other relevant parameters that need consideration to keep systems secure. It also contains an overview of the main types of cryptographic primitives and relevant algorithms that one could use. The last part of that chapter is an overview of the standardization of post-quantum cryptographic algorithms, including high-level descriptions of the mathematical constructions used by these algorithms.

The third chapter describes several families of widely used communication protocols. Where relevant, it gives recommendations about using them securely. Section on TLS also describes other measures that are relevant when protecting one's website or other services with TLS. Other protocols covered are SSH, IPsec and WireGuard.

The fourth chapter describes several container formats used to cryptographically secure one-time messages between parties or documents at rest. Covered formats include CMS, ASiC-E and its variants, JOSE formats, PASETO, PGP and Estonian local format CDOC.

The fifth chapter, on authentication and authorization protocols describes OAuth 2.0 and several potential problems that might occur while implementing it. It also mentions newer specifications such as PKCE and profile for native application usage. Upcoming standards such as OAuth 2.1 and profile for browser-based applications are also covered. The chapter goes on to describe OpenID Connect and UMA as well as refer to some other profiles and extensions to OAuth/OpenID Connect framework. The chapter also describes specifications produced by FIDO Alliance, such as FIDO U2F, FIDO UAF and WebAuthn.

The sixth chapter describes tools to implement cryptographic measures in software. It begins with an overview of popular libraries that implement cryptographic primitives or protocols, such as OpenSSL. The rest of the chapter describes support for cryptography in popular development environments and programming languages, such as Java, Go, Rust and PHP. Due to renewed interest, the chapter specifically looks into the support of the RSA PSS signature / padding scheme in different libraries.

The last chapter discusses the support of the NFC interface in Estonian digital identity cards. It points out the lack of significant obstacles for a more widespread usage of this interface.

Kirjandus

- [1] Michel Abdalla, Pierre-Alain Fouque ja David Pointcheval. „Password-Based Authenticated Key Exchange in the Three-Party Setting“. Teoses: *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings*. Toim. Serge Vaudenay. Köide 3386. Lecture Notes in Computer Science. Springer, 2005, lk. 65–84. DOI: 10.1007/978-3-540-30580-4_6.
- [2] *Advanced Security Mechanisms for Machine Readable Travel Documents – Part 1 - Version 2.20*. BSI – Technical Guideline TR-03110-1, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI-TR-03110-Part-1_V2-2.pdf. Mai 2015.
- [3] *Agreed Cryptographic Mechanisms*. SOG-IS Crypto Evaluation Scheme, <https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.2.pdf>. Jaanuar 2020.
- [4] Liliya R. Akhmetzyanova *et al.* „On post-handshake authentication and external PSKs in TLS 1.3“. *J. Comput. Virol. Hacking Tech.* 16.4 (2020), lk. 269–274. DOI: 10.1007/s11416-020-00352-0.
- [5] *Analysis of planned architectural changes in Open-eID*. <https://web-eid.github.io/web-eid-cybernetica-analysis/webextensions-main.pdf>. Cybernetica report no. T-176-1. 2020.
- [6] *Analysis of the Possibility to Use ID1 Card's NFC Interface for Authentication and Electronic Signing*. https://www.ria.ee/sites/default/files/content-editors/SF/analysis_of_the_possibility_to_use_id1_cards_nfc_interface_for_authentication_and_electronic_signing_d-26-7_1.1_0.pdf. Cybernetica report no. D-26-7. 2022.
- [7] Ross J. Anderson ja Roger M. Needham. „Robustness Principles for Public Key Protocols“. Teoses: *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*. Toim. Don Coppersmith. Köide 963. Lecture Notes in Computer Science. Springer, 1995, lk. 236–247.
- [8] *ANSSI views on the Post-Quantum Cryptography transition*. <https://www.ssi.gouv.fr/en/publication/anssi-views-on-the-post-quantum-cryptography-transition/>. Märts 2022.
- [9] William Baker, William Polk ja Murugiah Souppaya. *Getting Ready for Post-Quantum Cryptography: Exploring Challenges Associated with Adopting and Using Post-Quantum Cryptographic Algorithms*. NIST Cybersecurity White Paper, <https://doi.org/10.6028/NIST.CSWP.04282021>. Aprill 2021.

- [10] Elaine Barker. *Recommendation for Key Management. Part 1: General*. NIST Special Publication 800-57 Part 1, Revision 5, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>. Mai 2020.
- [11] Elaine Barker, Quynh Dang ja Sheila Frankel. *Guide to IPsec VPNs*. NIST Special Publication 800-77, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-77r1.pdf>. Juuni 2020.
- [12] Richard Barnes. *Use Cases and Requirements for JSON Object Signing and Encryption (JOSE)*. RFC 7165. Aprill 2014. URL: <https://rfc-editor.org/rfc/rfc7165.txt>.
- [13] *BDOC – digiallkirja vorming*. <https://www.id.ee/wp-content/uploads/2020/01/bdoc-spec212-est.pdf>. Versioon 2.1.2. OID: 1.3.6.1.4.1.10015.1000.3.2.3. 2014.
- [14] Mihir Bellare ja Phillip Rogaway. „The Exact Security of Digital Signatures-How to Sign with RSA and Rabin“. Teoses: *Advances in Cryptology — EUROCRYPT '96*. Toim. Ueli Maurer. Springer Berlin Heidelberg, 1996, lk. 399–416. ISBN: 978-3-540-68339-1.
- [15] Mike Belshe, Roberto Peon ja Martin Thomson. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC 7540. Mai 2015. URL: <https://rfc-editor.org/rfc/rfc7540.txt>.
- [16] Jens Bender, Marc Fischlin ja Dennis Kügler. „Security Analysis of the PACE Key-Agreement Protocol“. Teoses: *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*. Toim. Pierangela Samarati *et al.* Köide 5735. Lecture Notes in Computer Science. Springer, 2009, lk. 33–48. DOI: 10.1007/978-3-642-04474-8_3.
- [17] D. Benjamin ja C. A. Wood. *Importing External Pre-Shared Keys (PSKs) for TLS 1.3*. RFC 9258. Juuli 2022. URL: <https://rfc-editor.org/rfc/rfc9258.txt>.
- [18] David Benjamin. *Using TLS 1.3 with HTTP/2*. RFC 8740. Veebruar 2020. URL: <https://rfc-editor.org/rfc/rfc8740.txt>.
- [19] Daniel J. Bernstein. *ChaCha, a variant of Salsa20*. <https://cr.yp.to/chacha/chacha-20080128.pdf>. Jaanuar 2008.
- [20] Daniel J. Bernstein. „Curve25519: New Diffie-Hellman Speed Records“. Teoses: *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*. Toim. Moti Yung *et al.* Köide 3958. Lecture Notes in Computer Science. Springer, 2006, lk. 207–228. DOI: 10.1007/11745853_14.
- [21] Daniel J. Bernstein. *The Salsa20 family of stream ciphers*. <https://cr.yp.to/snuffle/salsafamily-20071225.pdf>. Detsember 2007.
- [22] Ward Beullens. „Breaking Rainbow Takes a Weekend on a Laptop“. Teoses: *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*. Toim. Yevgeniy Dodis ja Thomas Shrimpton. Köide 13508. Lecture Notes in Computer Science. Springer, 2022, lk. 464–479. DOI: 10.1007/978-3-031-15979-4_16.

- [23] Nina Bindel *et al.* „Transitioning to a Quantum-Resistant Public Key Infrastructure“. Teoses: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*. Toim. Tanja Lange ja Tsuyoshi Takagi. Kõide 10346. Lecture Notes in Computer Science. Springer, 2017, lk. 384–405. DOI: 10.1007/978-3-319-59879-6_22.
- [24] Alex Biryukov *et al.* *Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications*. RFC 9106. September 2021. URL: <https://rfc-editor.org/rfc/rfc9106.txt>.
- [25] A. Jerman Blazic, S. Saljic ja T. Gondrom. *Extensible Markup Language Evidence Record Syntax (XMLERS)*. RFC 6283. Juuli 2011. URL: <https://rfc-editor.org/rfc/rfc6283.txt>.
- [26] Daniel Bleichenbacher. „Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1“. Teoses: *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*. Toim. Hugo Krawczyk. Kõide 1462. Lecture Notes in Computer Science. Springer, 1998, lk. 1–12.
- [27] Sharon Boeyen *et al.* *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. Mai 2008. URL: <https://rfc-editor.org/rfc/rfc5280.txt>.
- [28] Dan Boneh, Henry Corrigan-Gibbs ja Stuart Schechter. *Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks*. Cryptology ePrint Archive, Report 2016/027. <https://eprint.iacr.org/2016/027>. 2016.
- [29] C. Bormann ja P. Hoffman. *Concise Binary Object Representation (CBOR)*. RFC 8949. Detsember 2020. URL: <https://rfc-editor.org/rfc/rfc8949.txt>.
- [30] Joppe W. Bos, Joost Renes ja Amber Sprenkels. „Dilithium for Memory Constrained Devices“. Teoses: *Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18–20, 2022, Proceedings*. Fes, Morocco: Springer-Verlag, 2022, lk. 217–235. ISBN: 978-3-031-17432-2. DOI: 10.1007/978-3-031-17433-9_10.
- [31] Colin Boyd, Anish Mathuria ja Douglas Stebila. *Protocols for Authentication and Key Establishment, Second Edition*. Information Security and Cryptography. Springer, 2020. ISBN: 978-3-662-58145-2.
- [32] John Bradley *et al.* *Web Authentication: An API for accessing Public Key Credentials*. Level 2 Candidate Snapshot <https://www.w3.org/TR/webauthn-2/>. Aprill 2021.
- [33] Christiaan Brand *et al.* *Client to Authenticator Protocol (CTAP)*. Jaanuar 2019. URL: <https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html>.
- [34] Christiaan Brand *et al.* *Web Authentication: An API for accessing Public Key Credentials*. Level 1 <https://www.w3.org/TR/webauthn-1/>. Märts 2019.

- [35] Marcus Brinkmann *et al.* „ALPACA: Application Layer Protocol Confusion - Analyzing and Mitigating Cracks in TLS Authentication“. Teoses: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. Toim. Michael Bailey ja Rachel Greenstadt. USENIX Association, 2021, lk. 4293–4310. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/brinkmann>.
- [36] Chris Brzuska *et al.* „Key-Schedule Security for the TLS 1.3 Standard“. Teoses: *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I*. Toim. Shweta Agrawal ja Dongdai Lin. Kõide 13791. Lecture Notes in Computer Science. Springer, 2022, lk. 621–650. DOI: 10.1007/978-3-031-22963-3_21.
- [37] Johannes Buchmann *et al.* „On the security of the Winternitz one-time signature scheme“. *International Journal of Applied Cryptography* 3.1 (2013), lk. 84–96. DOI: 10.1504/IJACT.2013.053435. URL: <https://doi.org/10.1504/IJACT.2013.053435>.
- [38] Ahto Buldas *et al.* „Server-Supported RSA Signatures for Mobile Devices“. Teoses: *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I*. Toim. Simon N. Foley, Dieter Gollmann ja Einar Snekkenes. Lecture Notes in Computer Science. <https://home.cyber.ee/~peeter/research/esorics2017.pdf>. 2017.
- [39] Brian Campbell, Chuck Mortimore ja Michael Jones. *Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants*. RFC 7522. Mai 2015. URL: <https://rfc-editor.org/rfc/rfc7522.txt>.
- [40] Brian Campbell ja Hannes Tschofenig. *An IETF URN Sub-Namespace for OAuth*. RFC 6755. Oktoober 2012. URL: <https://rfc-editor.org/rfc/rfc6755.txt>.
- [41] Wouter Castryck ja Thomas Decru. *An efficient key recovery attack on SIDH (preliminary version)*. Cryptology ePrint Archive, Paper 2022/975. <https://eprint.iacr.org/2022/975>. 2022.
- [42] *CMS Advanced Electronic Signatures (CAAdES)*. Versioon 2.2.1. Electronic Signatures and Infrastructures (ESI) https://www.etsi.org/deliver/etsi_ts/101700_101799/101733/02.02.01_60/ts_101733v020201p.pdf. 2013.
- [43] David A. Cooper *et al.* *Recommendation for Stateful Hash-Based Signature Schemes*. Tehniline raport. Oktoober 2020. DOI: 10.6028/nist.sp.800-208.
- [44] Daniele Cozzo ja Nigel P. Smart. „Sharing the LUOV: Threshold Post-quantum Signatures“. Teoses: *Cryptography and Coding - 17th IMA International Conference, IMACC 2019, Oxford, UK, December 16-18, 2019, Proceedings*. Toim. Martin Albrecht. Kõide 11929. Lecture Notes in Computer Science. Springer, 2019, lk. 128–153. DOI: 10.1007/978-3-030-35199-1_7.

- [45] Juan-Carlos Cruellas Ibarz. „Bringing JSON signatures to ETSI AdES framework: Meet JAdES signatures“. *Computer Standards & Interfaces* 71 (2020), lk. 103434. ISSN: 0920-5489. DOI: <https://doi.org/10.1016/j.csi.2020.103434>.
- [46] *Cryptographic algorithms lifecycle report 2016*. https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/cryptographic_algorithms_lifecycle_report_2016.pdf. Cybernetica report no. A-101-3. 2016.
- [47] *Cryptographic Mechanisms: Recommendations and Key Lengths*. BSI – Technical Guideline TR-02102-1, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf>. November 2022.
- [48] *Cryptographic Mechanisms: Recommendations and Key Lengths – Use of Internet Protocol Security (IPsec) and Internet Key Exchange (IKEv2)*. BSI – Technical Guideline TR-02102-3, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-3.pdf>. Jaanuar 2020.
- [49] *Cryptographic Mechanisms: Recommendations and Key Lengths – Use of Secure Shell (SSH)*. BSI – Technical Guideline TR-02102-4, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-4.pdf>. Jaanuar 2023.
- [50] The European Union Agency for Cybersecurity. *Post-Quantum Cryptography - Integration study*. <https://www.enisa.europa.eu/publications/post-quantum-cryptography-integration-study>. Accessed: 2022-12-09. Oktoober 2022.
- [51] Ivan Damgård *et al.* „Two-Round n-out-of-n and Multi-Signatures and Trapdoor Commitment from Lattices“. *Journal of Cryptology* 35.2 (aprill 2022). DOI: 10.1007/s00145-022-09425-3.
- [52] Don Davis. „Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML“. Teoses: *Proceedings of the General Track: 2001 USENIX Annual Technical Conference, June 25-30, 2001, Boston, Massachusetts, USA*. Toim. Yoonho Park. USENIX, 2001, lk. 65–78. URL: <http://www.usenix.org/publications/library/proceedings/usenix01/davis.html>.
- [53] William Denniss ja John Bradley. *OAuth 2.0 for Native Apps*. RFC 8252. Oktoober 2017. URL: <https://rfc-editor.org/rfc/rfc8252.txt>.
- [54] L. Peter Deutsch ja Jean-loup Gailly. *ZLIB Compressed Data Format Specification version 3.3*. RFC 1950. Mai 1996. URL: <https://rfc-editor.org/rfc/rfc1950.txt>.
- [55] *Digital Signature Standard (DSS)*. FIPS PUB 186-4, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>. Juuli 2013.
- [56] Jintai Ding. *Cryptographic systems using pairing with errors*. U.S. Patent No. 9,246,675. Aprill 2012. URL: <https://patents.google.com/patent/US9246675B2/en>.

- [57] Jintai Ding ja Dieter Schmidt. „Rainbow, a New Multivariable Polynomial Signature Scheme“. Teoses: *Applied Cryptography and Network Security*. Toim. John Ioannidis, Angelos Keromytis ja Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, lk. 164–175. ISBN: 978-3-540-31542-1.
- [58] Benjamin Dowling *et al.* „A Cryptographic Analysis of the TLS 1.3 Handshake Protocol“. *J. Cryptol.* 34.4 (2021), lk. 37. DOI: 10.1007/s00145-021-09384-1.
- [59] Nir Drucker ja Shay Gueron. „Selfie: reflections on TLS 1.3 with PSK“. *J. Cryptol.* 34.3 (2021), lk. 27. DOI: 10.1007/s00145-021-09387-y.
- [60] Viktor Dukhovni ja Wes Hardaker. *SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)*. RFC 7672. Oktoober 2015. URL: <https://rfc-editor.org/rfc/rfc7672.txt>.
- [61] Donald E. Eastlake 3rd. *Transport Layer Security (TLS) Extensions: Extension Definitions*. RFC 6066. Jaanuar 2011. URL: <https://www.rfc-editor.org/info/rfc6066>.
- [62] *Eliminating Obsolete Transport Layer Security (TLS) Protocol Configurations*. Jaanuar 2021. URL: https://media.defense.gov/2021/Jan/05/2002560140/-1/-1/0/ELIMINATING_OBSOLETE_TLS_U00197443-20.PDF.
- [63] *ETSI EN 119 182-1. Electronic Signatures and Infrastructures (ESI); JAdES digital signatures; Part 1: Building blocks and JAdES baseline signatures*. Versioon 1.1.1. 2021. URL: https://www.etsi.org/deliver/etsi_ts/119100_119199/11918201/01.01.01_60/ts_11918201v010101p.pdf.
- [64] *ETSI EN 319 122-1. Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 1: Building blocks and CAdES baseline signatures*. Versioon 1.2.1. Oktoober 2021. URL: https://www.etsi.org/deliver/etsi_en/319100_319199/31912201/01.02.01_60/en_31912201v010201p.pdf.
- [65] *ETSI EN 319 132-1. Electronic Signatures and Infrastructures (ESI); XAdES digital signatures; Part 1: Building blocks and XAdES baseline signatures*. Versioon 1.1.1. Veebruar 2022. URL: https://www.etsi.org/deliver/etsi_en/319100_319199/31913201/01.02.01_60/en_31913201v010201p.pdf.
- [66] *ETSI EN 319 142-1. Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; Part 1: Building blocks and PAdES baseline signatures*. Versioon 1.1.1. 2016. URL: https://www.etsi.org/deliver/etsi_en/319100_319199/31914201/01.01.01_60/en_31914201v010101p.pdf.
- [67] *ETSI EN 319 162-1. Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC); Part 1: Building blocks and ASiC baseline containers*. Versioon 1.1.1. Aprill 2016. URL: https://www.etsi.org/deliver/etsi_en/319100_319199/31914201/01.01.01_60/en_31914201v010101p.pdf.
- [68] Chris Evans, Chris Palmer ja Ryan Sleevi. *Public Key Pinning Extension for HTTP*. RFC 7469. Aprill 2015. URL: <https://rfc-editor.org/rfc/rfc7469.txt>.

- [69] Tony Finch, Matthew A. Miller ja Peter Saint-Andre. *Using DNS-Based Authentication of Named Entities (DANE) TLSA Records with SRV Records*. RFC 7673. Oktoober 2015. URL: <https://rfc-editor.org/rfc/rfc7673.txt>.
- [70] Hal Finney *et al.* *OpenPGP Message Format*. RFC 4880. November 2007. URL: <https://rfc-editor.org/rfc/rfc4880.txt>.
- [71] Tako Boris Fouotsa. *SIDH with masked torsion point images*. Cryptology ePrint Archive, Paper 2022/1054. <https://eprint.iacr.org/2022/1054>. 2022.
- [72] Stephan Friedl *et al.* *Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*. RFC 7301. Juuli 2014. URL: <https://www.rfc-editor.org/info/rfc7301>.
- [73] Eiichiro Fujisaki ja Tatsuaki Okamoto. „Secure Integration of Asymmetric and Symmetric Encryption Schemes“. *Journal of Cryptology* 26.1 (detsember 2011), lk. 80–101. DOI: 10.1007/s00145-011-9114-1.
- [74] Taher El Gamal. „A public key cryptosystem and a signature scheme based on discrete logarithms“. *IEEE Trans. Inf. Theory* 31.4 (1985), lk. 469–472.
- [75] Craig Gidney ja Martin Ekerå. „How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits“. *Quantum* 5 (2021), lk. 433. DOI: 10.22331/q-2021-04-15-433.
- [76] Daniel Kahn Gillmor. *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)*. RFC 7919. August 2016. URL: <https://rfc-editor.org/rfc/rfc7919.txt>.
- [77] Yoel Gluck, Neal Harris ja Angelo Prado. *BREACH: reviving the CRIME attack*. 2013. URL: <http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf>.
- [78] T. Gondrom, R. Brandner ja U. Pordesch. *Evidence Record Syntax (ERS)*. RFC 4998. August 2007. URL: <https://rfc-editor.org/rfc/rfc4998.txt>.
- [79] Paul A. Grassi, James L. Fenton ja Elaine M. Newton *et al.* *Authentication and Lifecycle Management*. Digital Identity Guidelines, NIST Special Publication 800-63B, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf>. Juuni 2017.
- [80] Ólafur Guðmundsson. *Adding Acronyms to Simplify Conversations about DNS-Based Authentication of Named Entities (DANE)*. RFC 7218. Aprill 2014. URL: <https://rfc-editor.org/rfc/rfc7218.txt>.
- [81] Jiaxing Guo *et al.* „Automated State-Machine-Based Analysis of Hostname Verification in IPsec Implementations“. *Inf. Technol. Control*. 50.3 (2021), lk. 570–587. DOI: 10.5755/j01.itc.50.3.27844.
- [82] Phillip Hallam-Baker, Rob Stradling ja Jacob Hoffman-Andrews. *DNS Certification Authority Authorization (CAA) Resource Record*. RFC 8659. November 2019. URL: <https://rfc-editor.org/rfc/rfc8659.txt>.
- [83] Dick Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. Oktoober 2012. URL: <https://rfc-editor.org/rfc/rfc6749.txt>.

- [84] Jeff Hodges, Collin Jackson ja Adam Barth. *HTTP Strict Transport Security (HSTS)*. RFC 6797. November 2012. URL: <https://rfc-editor.org/rfc/rfc6797.txt>.
- [85] Paul E. Hoffman ja Jakob Schlyter. *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA*. RFC 6698. August 2012. URL: <https://rfc-editor.org/rfc/rfc6698.txt>.
- [86] R. Housley *et al.* *Guidance for External Pre-Shared Key (PSK) Usage in TLS*. RFC 9257. Juuli 2022. URL: <https://rfc-editor.org/rfc/rfc9257.txt>.
- [87] Russ Housley. *Cryptographic Message Syntax (CMS)*. RFC 5652. September 2009. URL: <https://rfc-editor.org/rfc/rfc5652.txt>.
- [88] Russ Housley. *Update to the Cryptographic Message Syntax (CMS) for Algorithm Identifier Protection*. RFC 8933. Oktoober 2020. URL: <https://rfc-editor.org/rfc/rfc8933.txt>.
- [89] Andreas Hulsing *et al.* *Sphincs+: Algorithm specifications and supporting documentation, Selected Algorithms of NIST's post-quantum cryptography standardization process*. <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. Accessed: 2022-12-08. November 2022.
- [90] Phil Hunt *et al.* *Security Event Token (SET)*. RFC 8417. Juuli 2018. URL: <https://rfc-editor.org/rfc/rfc8417.txt>.
- [91] *IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques*. IEEE Std 1363a-2004 (Amendment to IEEE Std 1363-2000). 2004. DOI: 10.1109/IEEESTD.2004.94612.
- [92] Takeshi Imamura *et al.* *XML Encryption Syntax and Processing Version 1.1*. <https://www.w3.org/TR/2013/REC-xmlenc-core1-20130411/>. Aprill 2013.
- [93] *Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program*. NIST and Canadian Centre for Cyber Security. <https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402IG.pdf>. Oktoober 2022.
- [94] *IVXV protokollide kirjeldus*. <https://www.valimised.ee/sites/default/files/uploads/eh/IVXV-protokollid.pdf>. Vabariigi Valimiskomisjon IVXV-PR-1.5.0 (in Estonian). 2019.
- [95] Tibor Jager, Saqib A. Kakvi ja Alexander May. „On the Security of the PKCS #1 v1.5 Signature Scheme“. Teoses: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. Toim. David Lie *et al.* ACM, 2018, lk. 1195–1208. DOI: 10.1145/3243734.3243798.
- [96] M. Jensen *et al.* „On the effectiveness of XML Schema validation for countering XML Signature Wrapping attacks“. Teoses: *1st International Workshop on Securing Services on the Cloud (IWSSC)*. 2011.
- [97] Michael Jones, John Bradley ja Nat Sakimura. *JSON Web Signature (JWS)*. RFC 7515. Mai 2015. URL: <https://rfc-editor.org/rfc/rfc7515.txt>.

- [98] Michael Jones, John Bradley ja Nat Sakimura. *JSON Web Token (JWT)*. RFC 7519. Mai 2015. URL: <https://rfc-editor.org/rfc/rfc7519.txt>.
- [99] Michael Jones ja Dick Hardt. *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. RFC 6750. Oktoober 2012. URL: <https://rfc-editor.org/rfc/rfc6750.txt>.
- [100] Michael Jones ja Joe Hildebrand. *JSON Web Encryption (JWE)*. RFC 7516. Mai 2015. URL: <https://rfc-editor.org/rfc/rfc7516.txt>.
- [101] Jakob Jonsson ja Burt Kaliski. *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*. RFC 3447. Veebruar 2003. URL: <https://rfc-editor.org/rfc/rfc3447.txt>.
- [102] Burt Kaliski. *PKCS #1: RSA Encryption Version 1.5*. RFC 2313. Märts 1998. URL: <https://rfc-editor.org/rfc/rfc2313.txt>.
- [103] Charlie Kaufman et al. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296. Oktoober 2014. URL: <https://rfc-editor.org/rfc/rfc7296.txt>.
- [104] Stephen Kent. *IP Authentication Header*. RFC 4302. Detsember 2005. URL: <https://rfc-editor.org/rfc/rfc4302.txt>.
- [105] Stephen Kent. *IP Encapsulating Security Payload (ESP)*. RFC 4303. Detsember 2005. URL: <https://rfc-editor.org/rfc/rfc4303.txt>.
- [106] Ziv Kfir ja Avishai Wool. „Picking Virtual Pockets using Relay Attacks on Contactless Smartcard“. Teoses: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm 2005, Athens, Greece, 5-9 September, 2005*. IEEE, 2005, lk. 47–58. DOI: 10.1109/SECURECOMM.2005.32.
- [107] Yoongu Kim et al. „Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors“. Teoses: *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*. IEEE Computer Society, 2014, lk. 361–372. DOI: 10.1109/ISCA.2014.6853210.
- [108] Hugo Krawczyk. „The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)“ Teoses: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. Toim. Joe Kilian. Köide 2139. Lecture Notes in Computer Science. Springer, 2001, lk. 310–331.
- [109] Hugo Krawczyk, Mihir Bellare ja Ran Canetti. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104. Veebruar 1997. URL: <https://rfc-editor.org/rfc/rfc2104.txt>.
- [110] *Krüptoalgoritmid ning nende tugi teekides ja infosüsteemides (Cryptographic algorithms and their support in libraries and information systems)*. <https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/cryptoreport2021.pdf>. Cybernetica report no. T-184-7, v1.0 (in Estonian). 2021.

- [111] *Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms)*. https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/kryptoalgoritmide_elutsykli_uuring_15-07-2011.pdf. Cybernetica report no. A-60-1 (in Estonian). 2011.
- [112] *Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms)*. https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/krüptograafiliste_algoritmide_elutsykli_uuring_ii.pdf. Cybernetica report no. A-77-5 (in Estonian). 2013.
- [113] *Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms)*. https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/krüptograafiliste_algoritmide_uuring_2015.pdf. Cybernetica report no. A-101-1 (in Estonian). 2015.
- [114] *Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms)*. https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/krüptograafiliste_algoritmide_elutsykli_uuring_2017.pdf. Cybernetica report no. A-101-9 (in Estonian). 2017.
- [115] Adam Langley, Mike Hamburg ja Sean Turner. *Elliptic Curves for Security*. RFC 7748. Jaanuar 2016. URL: <https://rfc-editor.org/rfc/rfc7748.txt>.
- [116] Peeter Laud ja Meelis Roos. „Formal Analysis of the Estonian Mobile-ID Protocol“. Teoses: *Identity and Privacy in the Internet Age, 14th Nordic Conference on Secure IT Systems, NordSec 2009, Oslo, Norway, 14-16 October 2009. Proceedings*. Toim. Audun Jøsang, Torleiv Maseng ja Svein J. Knapskog. Kõide 5838. Lecture Notes in Computer Science. Springer, 2009, lk. 271–286. URL: <https://home.cyber.ee/~peeter/research/nordsec09.pdf>.
- [117] Peeter Laud, Nikita Snetkov ja Jelizaveta Vakarjuk. *DiLizium 2.0: Revisiting Two-Party Crystals-Dilithium*. Cryptology ePrint Archive, Paper 2022/644. 2022. URL: <https://eprint.iacr.org/2022/644>.
- [118] Ben Laurie, Adam Langley ja Emilia Käsper. *Certificate Transparency*. RFC 6962. Juuni 2013. URL: <https://rfc-editor.org/rfc/rfc6962.txt>.
- [119] Bingyu Li *et al.* „Certificate Transparency in the Wild: Exploring the Reliability of Monitors“. Teoses: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. Toim. Lorenzo Cavallaro *et al.* ACM, 2019, lk. 2505–2520.
- [120] Torsten Lodderstedt, Mark McGloin ja Phil Hunt. *OAuth 2.0 Threat Model and Security Considerations*. RFC 6819. Jaanuar 2013. URL: <https://rfc-editor.org/rfc/rfc6819.txt>.
- [121] Chris M. Lonvick ja Tatu Ylonen. *The Secure Shell (SSH) Transport Layer Protocol*. RFC 4253. Jaanuar 2006. URL: <https://rfc-editor.org/rfc/rfc4253.txt>.

- [122] Vadim Lyubashevsky *et al.* *CRYSTALS-DILITHIUM: Algorithm specifications and supporting documentation, Selected Algorithms of NIST's post-quantum cryptography standardization process*. <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. Accessed: 2022-12-08. November 2022.
- [123] Marcel Maehren *et al.* „TLS-Anvil: Adapting Combinatorial Testing for TLS Libraries“. Teoses: *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*. Toim. Kevin R. B. Butler ja Kurt Thomas. USENIX Association, 2022, lk. 215–232. URL: <https://www.usenix.org/conference/usenixsecurity22/presentation/maehren>.
- [124] Daniel Margolis *et al.* *SMTP MTA Strict Transport Security (MTA-STS)*. RFC 8461. September 2018. URL: <https://rfc-editor.org/rfc/rfc8461.txt>.
- [125] Daniel Margolis *et al.* *SMTP TLS Reporting*. RFC 8460. September 2018. URL: <https://rfc-editor.org/rfc/rfc8460.txt>.
- [126] Johannes Merkle ja Manfred Lochter. *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*. RFC 5639. Märts 2010. URL: <https://rfc-editor.org/rfc/rfc5639.txt>.
- [127] Kai Mindermann, Philipp Keck ja Stefan Wagner. „How Usable Are Rust Cryptography APIs?“ Teoses: *2018 IEEE International Conference on Software Quality, Reliability and Security, QRS 2018, Lisbon, Portugal, July 16-20, 2018*. IEEE, 2018, lk. 143–154.
- [128] Dustin Moody. *Digital Signature Standard (DSS)*. FIPS PUB 186-5. 2023. DOI: 10.6028/nist.fips.186-5.
- [129] Dustin Moody. *Recommendations for Discrete Logarithm-based Cryptography*. NIST Special Publication 800-186. 2023. DOI: 10.6028/nist.sp.800-186.
- [130] Dustin Moody. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. Tehniline raport. National Institute of Standards ja Technology, 2022. DOI: 10.6028/nist.ir.8413-upd1.
- [131] Kathleen Moriarty, Burt Kaliski ja Andreas Rusch. *PKCS #5: Password-Based Cryptography Specification Version 2.1*. RFC 8018. Jaanuar 2017. URL: <https://rfc-editor.org/rfc/rfc8018.txt>.
- [132] Tomoki Moriya. *Masked-degree SIDH*. Cryptology ePrint Archive, Paper 2022/1019. <https://eprint.iacr.org/2022/1019>. 2022.
- [133] Michele Mosca ja Marco Piani. *2021 Quantum Threat Timeline Report*. <https://globalriskinstitute.org/publication/2021-quantum-threat-timeline-report-global-risk-institute-global-risk-institute/>. Accessed: 2022-12-09. Jaanuar 2022.
- [134] Koksai Mus *et al.* *Jolt: Recovering TLS Signing Keys via Rowhammer Faults*. Cryptology ePrint Archive, Paper 2022/1669. <https://eprint.iacr.org/2022/1669>. 2022.
- [135] Jens Müller *et al.* „Re: What's Up Johnny? - Covert Content Attacks on Email End-to-End Encryption“. Teoses: *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*. Toim. Robert H. Deng *et al.* Kõide 11464. Lecture Notes in Computer Science. Springer, 2019, lk. 24–42.

- [136] National Institute of Standards and Technology. *Announcing Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms*. <https://csrc.nist.gov/News/2016/Public-Key-Post-Quantum-Cryptographic-Algorithms>. Accessed: 2022-12-20. Detsember 2016.
- [137] National Institute of Standards and Technology. *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. <https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf>. Accessed: 2022-12-20. August 2022.
- [138] National Security Agency. *Announcing the Commercial National Security Algorithm Suite 2.0*. https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF. Accessed: 2022-12-09. September 2022.
- [139] Yoav Nir ja Adam Langley. *ChaCha20 and Poly1305 for IETF Protocols*. RFC 7539. Mai 2015. URL: <https://rfc-editor.org/rfc/rfc7539.txt>.
- [140] Mart Oruaas ja Jan Willemson. „Developing Requirements for the New Encryption Mechanisms in the Estonian eID Infrastructure“. Teoses: *Databases and Information Systems - 14th International Baltic Conference, DB&IS 2020, Tallinn, Estonia, June 16-19, 2020, Proceedings*. Toim. Tarmo Robal et al. Kõide 1243. Communications in Computer and Information Science. Springer, 2020, lk. 13–20.
- [141] *PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - a framework document for PAdES*. Electronic Signatures and Infrastructures (ESI) https://www.etsi.org/deliver/etsi_ts/102700_102799/10277801/01.01.01_60/ts_10277801v010101p.pdf. 2016.
- [142] Colin Percival. *Stronger key derivation via sequential memory-hard functions*. BSDCan 2009, The Technical BSD Conference. <https://www.bsdcn.org/2009/schedule/events/147.en.html>. Jaanuar 2009.
- [143] Colin Percival ja Simon Josefsson. *The scrypt Password-Based Key Derivation Function*. RFC 7914. August 2016. URL: <https://rfc-editor.org/rfc/rfc7914.txt>.
- [144] Thomas Pornin. *New Efficient, Constant-Time Implementations of Falcon*. Cryptology ePrint Archive, Paper 2019/893. <https://eprint.iacr.org/2019/893>. 2019.
- [145] *Postkvant-krüptograafia ülevaade (Report on the post-quantum cryptographic algorithms)*. <https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/postkvant-krüptograafia-ulevaade-2018.pdf>. Cybernetica report no. A-101-10 (in Estonian). 2018.
- [146] Thomas Prest et al. *Falcon: Algorithm specifications and supporting documentation, Selected Algorithms of NIST's post-quantum cryptography standardization process*. <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. Accessed: 2022-12-08. November 2022.

- [147] Open Quantum Safe Project. *OQS algorithm performance visualizations*. https://openquantumsafe.org/benchmarking/visualization/speed_sig.html. Accessed: 2022-12-20.
- [148] Niels Provos ja David Mazières. „A Future-Adaptable Password Scheme“. Teoses: *Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference, June 6-11, 1999, Monterey, California, USA*. USENIX, 1999, lk. 81–91. URL: <http://www.usenix.org/events/usenix99/provos.html>.
- [149] *Quantum-safe cryptography – fundamentals, current developments and recommendations*. German Federal Office for Information Security. <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.html>. Accessed: 2022-12-09. Mai 2022.
- [150] Aina Toky Rasoamanana, Olivier Levillain ja Hervé Debar. „Towards a Systematic and Automatic Use of State Machine Inference to Uncover Security Flaws and Fingerprint TLS Stacks“. Teoses: *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part III*. Toim. Vijayalakshmi Atluri et al. Kõide 13556. Lecture Notes in Computer Science. Springer, 2022, lk. 637–657. DOI: 10.1007/978-3-031-17143-7_31.
- [151] „Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC“. OJ L 257 (august 2014), lk. 73–114. URL: <http://data.europa.eu/eli/reg/2014/910/oj>.
- [152] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. August 2018. URL: <https://rfc-editor.org/rfc/rfc8446.txt>.
- [153] Justin Richer. *OAuth 2.0 Token Introspection*. RFC 7662. Oktoober 2015. URL: <https://rfc-editor.org/rfc/rfc7662.txt>.
- [154] Justin Richer et al. *OAuth 2.0 Dynamic Client Registration Management Protocol*. RFC 7592. Juuli 2015. URL: <https://rfc-editor.org/rfc/rfc7592.txt>.
- [155] Justin Richer et al. *OAuth 2.0 Dynamic Client Registration Protocol*. RFC 7591. Juuli 2015. URL: <https://rfc-editor.org/rfc/rfc7591.txt>.
- [156] Finn de Ridder et al. „SMASH: Synchronized Many-sided Rowhammer Attacks from JavaScript“. Teoses: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. Toim. Michael Bailey ja Rachel Greenstadt. USENIX Association, 2021, lk. 1001–1018. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/ridder>.
- [157] Phillip Rogaway ja Thomas Shrimpton. „Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance“. Teoses: *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*. Toim. Bimal K. Roy ja Willi Meier. Kõide 3017. Lecture Notes in Computer Science. Springer, 2004, lk. 371–388.

- [158] Eyal Ronen *et al.* „The 9 Lives of Bleichenbacher’s CAT: New Cache ATtacks on TLS Implementations“. Teoses: *IEEE Symposium on Security and Privacy*. San Francisco: IEEE, 20. mai 2019, lk. 966–983.
- [159] N. Sakimura *et al.* *OpenID Connect Core 1.0 incorporating errata set 1*. https://openid.net/specs/openid-connect-core-1_0.html. November 2014.
- [160] Nat Sakimura, John Bradley ja Naveen Agarwal. *Proof Key for Code Exchange by OAuth Public Clients*. RFC 7636. September 2015. URL: <https://rfc-editor.org/rfc/rfc7636.txt>.
- [161] S. Santesson ja N. Pope. *ESSCertIDv2 Update for RFC 3161*. RFC 5816. IETF, aprill 2010. URL: <http://tools.ietf.org/rfc/rfc5816.txt>.
- [162] Peter Schwabe *et al.* *CRYSTALS-Kyber Algorithm specifications and supporting documentation, Selected Algorithms of NIST’s post-quantum cryptography standardization process*. <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. Accessed: 2022-12-08. November 2022.
- [163] *Secure Hash Standard (SHS)*. FIPS PUB 180-4, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>. August 2015.
- [164] Karen Seo ja Stephen Kent. *Security Architecture for the Internet Protocol*. RFC 4301. Detsember 2005. URL: <https://rfc-editor.org/rfc/rfc4301.txt>.
- [165] *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. FIPS PUB 202, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>. August 2015.
- [166] Y. Sheffer, P. Saint-Andre ja T. Fossati. *Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. RFC 9325. November 2022. URL: <https://rfc-editor.org/rfc/rfc9325.txt>.
- [167] Nikita Snetkov ja Jelizaveta Vakarjuk. *Integrating post-quantum cryptography to UXP*. Cybernetica research report D-2-499, v1.0. https://cyber.ee/uploads/PQC_UXP_report_8c97d91552.pdf. Jaanuar 2023.
- [168] National Institute of Standards ja Technology. *NIST PQC License Summary and Excerpts*. <https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/selected-algos-2022/nist-pqc-license-summary-and-excerpts.pdf>. Accessed: 2022-12-08. Detsember 2022.
- [169] National Institute of Standards ja Technology. *Round 4 Submissions*. <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions>. Accessed: 2022-12-20. November 2022.
- [170] *Status of quantum computer development*. German Federal Office for Information Security. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Quantencomputer/P283_QC_Studie-V_1_2.pdf?__blob=publicationFile&v=1. Accessed: 2022-12-09. August 2020.

- [171] George Arnold Sullivan *et al.* „Open to a fault: On the passive compromise of TLS keys via transient errors“. Teoses: *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*. Toim. Kevin R. B. Butler ja Kurt Thomas. USENIX Association, 2022, lk. 233–250. URL: <https://www.usenix.org/conference/usenixsecurity22/presentation/sullivan>.
- [172] Bryant Zadegan ja Ryan Lester. *Abusing Bleeding Edge Web Standards for AppSec Glory*. DEF CON 24. 2016. URL: <https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/>.
- [173] The IEEE and The Open Group. *Portable archive interchange*. <https://pubs.opengroup.org/onlinepubs/9699919799/utilities/pax.html>. 2018.
- [174] Kai Tian *et al.* „Analysis of Vulnerability of IPsec Protocol Implementation Based on Differential Fuzzing“. Teoses: *Security and Privacy in New Computing Environments*. Toim. Wenbo Shi, Xiaofeng Chen ja Kim-Kwang Raymond Choo. Köide 423. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer International Publishing, 2022, lk. 282–293. ISBN: 978-3-030-96791-8.
- [175] Meltem Sönmez Turan *et al.* *Recommendation for Password-Based Key Derivation Part 1: Storage Applications*. NIST Special Publication 800-132, <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf>. Detsember 2010.
- [176] Jelizaveta Vakarjuk, Nikita Snetkov ja Jan Willemsen. „DiLizium: A Two-Party Lattice-Based Signature Scheme“. *Entropy* 23.8 (2021). ISSN: 1099-4300. DOI: 10.3390/e23080989.
- [177] *Web eID: electronic identity cards on the Web*. <https://github.com/web-eid/web-eid-system-architecture-doc/>. 2022.
- [178] Friedrich Wiemer ja Ralf Zimmermann. „High-speed implementation of bcrypt password search using special-purpose hardware“. Teoses: *2014 International Conference on ReConFigurable Computing and FPGAs, ReConFig14, Cancun, Mexico, December 8-10, 2014*. IEEE, 2014, lk. 1–6.