TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl Udu 213422IACB

STM32 turvalise käivitamise süsteemi integreerimine ja arendus

Bakalaureusetöö

Juhendaja: Terry London
MSc
Kaasjuhendaja: Peeter Ellervee
PhD

Tallinn 2024

Autorideklaratsioon

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

Autor: Karl Udu

16.05.2024

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks oli lisada firma Proekspert poolt loodud STM32 mikrokontrolleril põhinevale turvalise püsivarauuendaja süsteemi prototüübile selle turvalisuse tõstmiseks turvalise käivitamise (*secure boot*) tugi. Selle jaoks kasutati STMicroelectronics'u SBSFU (*Secure Boot and Secure Firmware Update*) valmislahendust, mis koos prototüübi olemasoleva uuendaja rakendusega integreeriti, ning mida lisaks täiendati turvalisuse ja kasutajasõbralikkuse tõstmiseks. Peale arenduse ülevaate anti töös ülevaade valminud lahenduse kohta turvalisuse aspektist, ning analüüsiti selle võimalikke kitsaskohti.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 5 peatükki, 6 joonist, 1 tabelit.

Abstract

STM32 Secure Boot Integration and Development

The following bachelor's thesis describes the development of a secure boot solution for an existing STM32-based secure firmware updater prototype by Proekspert. STMicroelectronics provides a ready-made secure boot solution known as SBSFU (Secure Boot and Secure Firmware Update), which was chosen as the base for developing the secure boot system. The first stage of development was integrating SBSFU with the prototype's existing secure flasher application, which was followed by some enhancements made to SBSFU to improve both security and UX. The thesis also includes an overview of the developed SBSFU solution from a security perspective, and analysis of its security objectives with possible vulnerabilities being described.

First, an overview is provided of the updater prototype's hardware, secure boot as a concept, and SBSFU's operation and inner workings. This is followed by a description of the development process from start to finish, describing new features added to SBSFU along the way. After that, an overview is given of STM32 hardware security features that are utilized by SBSFU, such as TrustZone, various memory protections and more, with some of their potential drawbacks described. The finished solution's resilience to various kinds of attacks is analyzed, ranging from conventional software vulnerabilities to sophisticated hardware fault injection attacks.

The thesis is in Estonian and contains 31 pages of text, 5 chapters, 6 figures, 1 table.

Lühendite ja mõistete sõnastik

AES	*Advanced Encryption Standard*
CBC	*Cipher Block Chaining*
CCM	*Counter with CBC-MAC*
CTR	*Counter Mode*
DFA	*Differential Fault Analysis*
DH	Diffie-Hellman
DPA	*Differential Power Analysis*
ECB	*Electronic Codebook*
ECC	*Elliptic Curve Cryptography*
ECDSA	*Elliptic Curve Digital Signature Algorithm*
eFuse	*Electronic Fuse*
FI	*Fault Injection*
GCM	*Galois/Counter Mode*
HSM	*Hardware Security Module*
I^2C	*Inter-Integrated Circuit*
IDE	*Integrated Development Environment*
GMAC	*Galois Message Authentication Code*
HDP	*Hide Protection*
LCD	*Liquid Crystal Display*
MFI	*Multiple Fault Injection*
MPU	*Memory Protection Unit*
OTP	*One-Time Programmable*
PKA	*Public Key Accelerator*
RDP	*Readout Protection*
ROP	*Return-Oriented Programming*
RSA	Rivest-Shamir-Adleman
SBSFU	*Secure Boot and Secure Firmware Upgrade*
SHA	*Secure Hash Algorithm*

TF-M	*Trusted Firmware-M*
UART	*Universal Asynchronous Receiver-Transmitter*
USB	*Universal Serial Bus*
WRP	*Write Protection*

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva bakalaureusetöö aluseks on firma Proekspert poolt välja töötamisel olev STM32 mikrokontrolleril põhinev turvalise sardtarkvara uuendaja prototüüp (edaspidi *uuendaja*) [1]. Uuendaja abil on võimalik teostada selle külge ühendatud seadmele püsivarauuendusi, tehes seda turvalisel viisil. Digitaalsete allkirjade kasutamise abil saab süsteem veenduda selles, et uuendused on autentsed, ning lisaks sellele on uuenduste sisu krüpteeritud. Süsteem on suunatud tööstuslike seadmete kasutajatele ja tootjatele, kuna varsti jõustuv Euroopa Liidu küberkerksuse õigusakt (*Cyber Resilience Act*) [2] tekitab sektoris märkimisväärselt suurema vajaduse seadmete turvalisuse järele. Joonisel 1 on näha uuendaja prototüüp selle praeguses seisus.

Joonis 1: Uuendaja prototüüp

Töö eesmärgiks on uuendaja üleüldist turvalisust täiustada, võttes kasutusele turvalise käivitamise (ingl k *secure boot*). Turvalise käivitamise kasutamine tagab süsteemi poolt täidetava koodi usaldusväärsuse. Seda teostatakse turvalise alglaaduri (*bootloader*) abil, mis süsteemi käivitamise käigus teostab selle poolt laaditavale rakendusprogrammile turvakontrollid, kontrollide ebaõnnestumisel rakendus ei käivitu [3].

Eesmärgi teostamiseks otsustati kasutada STMicroelectronics'u poolt pakutavat turvalise käivitamise valmislahendust nimega SBSFU (*Secure Boot and Secure Firmware Update*) [4]. Lisaks vajaminevale turvalisele alglaadurile, sisaldab see seadmel oleva tarkvara arvuti kaudu uuendamise võimekust. SBSFU on ainuke tootja poolt ametlikult pakutud turvalise käivitamise lahendus, mis on testitud ja nende poolt toetatud. Kõik soovitud nõuded turvalise käivitamise süsteemi jaoks olid SBSFU poolt täidetud, mistõttu otsustati kiiresti selle valimise kasuks, võrreldes isetehtud lahenduse või mõne muu saadavaloleva

lahendusega. Töö kaheks põhietapiks oli SBSFU lahenduse integreerimine olemasoleva Proeksperdi turvalise uuendaja rakendusega, ning lisaks SBSFU komponentidele mõningate täienduste tegemine.

Töö koosneb kolmest osast. Esimene osa annab ülevaate uuendaja prototüübi riistvarast, seletab lahti turvalise käivitamise üldised põhimõtted ja mõisted, ning kirjeldab SBSFU lahenduse ülesehitust ja toimimist. Teine osa kirjeldab arenduse protsessi ning selle käigus lisatud funktsionaalsust. Kolmas osa annab ülevaate SBSFU poolt kasutatud mikrokontrolleri riistvaralistest turvamehhanismidest, ning analüüsib SBSFU lahendust selle turvalisuse poolest.

2 Süsteemi ülevaade

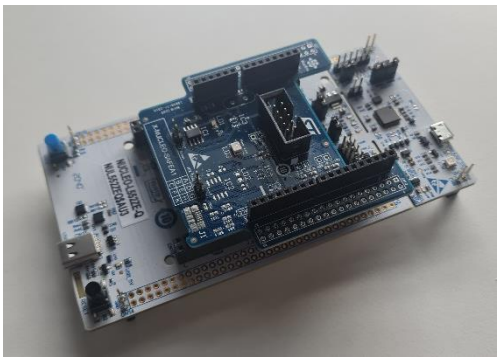
Siinsetes peatükkides kirjeldatakse uuendaja süsteemi riistvara ja antakse ülevaade SBSFU lahendusest. Kirjeldatud on uuendaja prototüübi riistvara osad, mis on relevantsed turvalise käivitamise jaoks, ning antud on ülevaade SBSFU lahenduse ülesehitusest ja toimimise põhimõtetest ning turvalisest käivitamisest üldisemalt.

2.1 Riistvara

Uuendaja prototüübi tuumaks on STMicroelectronics'u NUCLEO-L552ZE arendusplaat (vt joonis 2) [5]. Plaadil on STM32L5 seeria mikrokontroller [6], mille täpsem mudel on STM32L552ZE [7]. STM32L5 seeria mikrokontrollerid on madala võimsustarbega kontrollerid, mille juures on lisaks rõhku pandud ka turvalisusele. Kiibil on saadaval erinevad turvalisusega seotud võimekused, nt räsialgoritmide kiirendi ja Arm TrustZone. Enamus nendest võimekustest on süsteemi sees kasutusel, ning nende olemust ja kasutamise põhimõtet seletatakse lahti hilisemates peatükkides.

Lisaks STM32L552 kiibile, on samas alaseerias olemas tugevam STM32L562. Võrreldes eelmisega on sellel olemas just turvalisuse poolest kasulikud AES krüpteerimise ja avaliku võtme krüptograafia riistvaralised kiirendid. Töö raames oleks need kiirendid olnud kasulikud, aga L552 kiipi sisaldavad arendusplaadid on võrreldes L562'ga palju odavamad, mistõttu oli L552 kiip juba varem prototüübis kasutamiseks ära valitud.

Väga tähtsaks süsteemi osaks on STMicroelectronics'u X-NUCLEO-SAFEA1 arendusplaat (vt joonis 2) [8]. See sisaldab STSAFE-A110 turvaelementi [9], mis on kiip, mis pakub erinevate krüptograafiliste algoritmide rakendamise võimekust, turvalist krüptograafiliste võtmete ja muude andmete hoiustamist, ning muud turvalisusega seotud funktsionaalsust. Turvaelemendil on kogu süsteemi turvalisuse osas väga tähtis roll, ning uuendaja rakenduse poolt on selle funktsionaalsus olnud juba kasutusel. Turvalise käivitamise otstarbeks on kasutusel turvaelemendis olevad ühesuunalised loendurid, mida kasutatakse uuendaja rakenduse vanemale versioonile tagasi uuendamise ära hoidmiseks.



Joonis 2: NUCLEO-L552ZE arendusplaat koos turvaelemendi arendusplaadiga

LCD ekraan (vt joonis 3) sai algselt süsteemile lisatud kui kiire messieelse täiendusena, millest on nüüdseks saanud süsteemi püsiv osa. Uuendaja rakendus kasutab ekraani kasutajale ühendatud seadme uuendamise protsessi edenemise info ja selle kohta erinevate info- ja veateadete kuvamiseks. Turvalise käivitamise mastaabis kasutab alglaadur ekraani endapoolsete teadete näitamiseks. Ekraaniks on tavaline I^2C protokolli kaudu juhitav 16x2 tekstiekraan [10], mis on tuntud oma *Arduino*'ga ühilduvuse poolest. See ekraan sai valitud oma robustsuse ja lihtsa ühilduvuse pärast.

Joonis 3: I^2C 16x2 LCD ekraani moodul

Peale eelnevalt kirjeldatud komponentide on süsteemil olemas USB ühenduvust pakkuv arendusplaat. Seda kasutab uuendaja rakendus selle poolt uuendatava seadmega ühenduseks, aga turvalise käivitamise kontekstis USB'd ei kasutata.

2.2 SBSFU lahenduse ülevaade

Järgnevad peatükid annavad lihtsustatud ülevaate turvalisest käivitamisest ja sellega seonduvatest mõistetest, ning kirjeldavad SBSFU lahenduse olemust ja ülesehitust.

2.2.1 Turvaline käivitamine (secure boot)

Turvaline käivitamine on turvamehhanism, mille abil saab tagada, et kood, mida arvutisüsteem täitma asub, on usaldusväärne [3]. Usalduse tagamine toimub läbi üksteisele järgnevate käivitusetappide järjestikuse kontrollimise, kuni lõpuni rakendusprogrammi koodini välja. Kõige lihtsam turvalise käivitamise süsteem koosneb

alglaadurist, ja rakendusest, mida alglaadur käivitab. Keerukamate süsteemide puhul võib alglaadureid olla üksteise järel mitu tükki, mis võib olla tehtud nt alglaaduri väiksemaks muutmatuks ning suuremaks muudetavaks osaks tükeldamise eesmärgil. Teise näidisena võib leiduda eraldi kiibitootja alglaadur, mis laeb süsteemi koostaja-poolse alglaaduri, jagades sedasi ära nende vahel süsteemi käivitamise vastutused. Samuti võib rakenduseks olla üksik muutumatu programm lihtsa sardsüsteemi puhul, või keerukas operatsioonisüsteem, mis omakorda käivitab turvaliselt kasutaja poolt soovitud rakendusi.

Usaldusväärsuse kontrollimiseks turvalise käivitamise süsteemis kasutatakse tihti digitaalseid allkirju, mis põhinevad avaliku võtme krüptograafial. Tavapärased krüpteerimisalgoritmid kasutavad nii krüpteerimisel kui ka lahti krüpteerimisel ühte ja sama võtit, sellised algoritme nimetatakse sümmeetrilisteks krüpteerimisalgoritmideks. Asümmeetrilises ehk avaliku võtme krüptosüsteemis on kaks võtit: avalik ja salajane võti, mis moodustavad omavahel võtmepaari. Kontrollitaval käivitusetapil, nt rakendusel, on lisaks sellele endale kaasas digitaalne allkiri, millega saab käivitusetapi autentsust kontrollida avaliku võtme abil. Allkirja loomiseks on vaja avaliku võtmega seonduvat salajast võtit, mis on kättesaadav vaid tootjale vm partiile, kes tohib valiidseid allkirju luua. Salajasele võtmele vastupidiselt saab avalikku võtit levitada kuhu iganes.

Allkirjastamiseks arvutatakse kogu käivitusetapi sisu põhjal räsifunktsiooni lühend, tüüpiliselt kasutatakse selleks räsifunktsioone SHA-2 või SHA-3 perekonnast [11]. Järgmisena luuakse allkiri avaliku võtme krüptograafia algoritmi abil, levinud algoritmid selleks on RSA või ECDSA [12]. Mistahes sellisel krüptograafilisel algoritmil leidub digitaalse allkirja loomise funktsioon [13], mis võtab sisendiks lühikese pikkusega allkirjastatava sõnumi (siinpuhul räsi lühendi) ja salajase võtme, ning annab väljundiks allkirja, mis lisatakse käivitusetapile. Alglaaduri sees allkirja kontrollimiseks arvutatakse kontrollitava käivitusetapi pealt lühend, ning teostatakse kontroll krüptograafilise algoritmi digitaalse allkirja kontrollimise funktsiooni abil [13], mis võtab sisendiks sõnumi (lühendi), allkirja ja avaliku võtme, ning annab tulemusena kas kontrolli õnnestumise või ebaõnnestumise. Kui sõnum või avalik võti algselt allkirjastamise käigus kasutatust erinevad, siis kontroll ei õnnestu.

Allkirjade kasutamise abil saab turvalist käivitamist edukalt läbi viia, kuna turvakontroll õnnestub vaid õige allkirjaga käivitusetapi peal, ning allkirjastatud käivitusetapis

mistahes baidi väärtuse muutmisel selle allkirja kontroll enam ei õnnestu. Peale allkirja kontrollimist võib mistahes käivitusetapi turvakontroll endas sisaldada veel tingimusi, nt käivitusetapi versiooni kontrollimine, et takistada selle olemasolevast vanema versiooni paigaldamist, mis võib tagasi tuua uue versiooni käigus parandatud turvaauke.

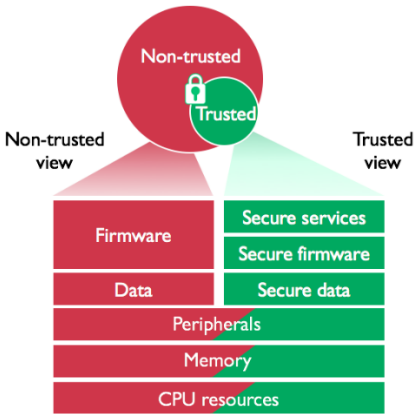
Turvalise käivitamise puhul omab tähtsust see, et kõige esimene süsteemis käivitatav alglaadur oleks muutmatu koodiga, kuna vaja on absoluutselt usaldusväärset tõe allikat, mida tuleb usaldada sellest edasiste etappide kontrolliks, aga enne mida pole kuskil eelnevat etappi, mis seda kontrolliks. Samuti peab olema muutmatu avalik võti, mille abil allkirju kontrollitakse, vastasel juhul poleks nendest kasu. Muutmatu esimese-etapi alglaaduri ja avaliku võtme kooslust nimetatakse tihti usaldusjuureks (ingl k *root of trust*), mis on kogu turvalise käivitamise süsteemi usaldatavuse vundamendiks [3]. Tüüpiliselt hoitakse mõlemat kiibi sees asuvas püsimälus e *ROM*-mälus, paindlikuma süsteemi jaoks võib avalik võti olla ka ühekordselt programmeeritavas mälus [14] (tihti levinud terminid on OTP e *one-time programmable* mälu, või *eFuse*'id [15]). Usaldusjuures algavat üksteist järjest kontrollivate etappide jada, kuni rakenduseni välja, nimetatakse usaldusjadaks [3] (ingl k *chain of trust*). Esimese-etapi alglaaduri maht ja vastutusala peaks olema nii väike kui võimalik, et selle sees potentsiaalsete turvaaukude esinemise risk oleks minimaalne. Tegemist on turvalisuse poolest väga tundliku komponendiga, kuna muutmatus alglaaduris olev turvaauk tähendab kogu turvalise käivitamise süsteemi usalduse varisemist.

Tasub märkida, et turvaline käivitamine tagab usaldusväärsuse ainult koodi **käivitamise** hetkel. Üksinda ei hoia see ära usaldusväärsuse rikkumist peale käivitamist, nt rakenduse koodi sees oleva turvaaugu läbi iseenda mälus ülekirjutamist või uude kohta kirjutatud pahatahtliku koodi täitmist. Samuti ei saa turvaline alglaadur tagada turvalist käivitamist enam siis, kui see ise sisaldab turvaauku, mis võimaldab turvalisusest mööda saada.

2.2.2 TrustZone

Kuigi turvalise käivitamise kui üldise ideena pole TrustZone seotud, siis töö aluseks oleva uuendaja süsteemi ja SBSFU lahendusega see on. TrustZone on osade Arm protsessorite poolt toetatud turvatehnoloogia, mis jagab nende poolt täidetava koodi turvaliseks ja mitteturvaliseks osaks. Mitteturvalises maailmas (*non-secure world*) asub tavapärane rakenduse kood, mis võib välja kutsuda turvalises maailmas (*secure world*) asuvaid funktsioone, kus olev kood on turvalisuse poolest tundlik. Joonisel 4 on skeem, mis

selgitab TrustZone'i ülesehitust. Muutmälu, välkmälu, MCU perifeeriaseadmed ning muud riistvaralised ressursid jaotatakse kahe maailma vahel ära, teatud juhtudel saab neid jagada. Turvalisel maailmal on ligipääs kõigele mitteturvalises maailmas olevale, vastupidi see ei kehti. Olemas on kaks TrustZone varianti: keerukamates rakendusprotsessorites esinev TrustZone-A, ning selles töös kehtiv mikrokontrollerites esinev TrustZone-M [16]. TrustZone oli enne töö algust kasutusel uuendaja rakenduse sees, ning seda kasutab ka SBSFU lahendus.



Joonis 4: TrustZone ülevaade [17]

2.2.3 SBSFU ülevaade

SBSFU e *Secure Boot and Secure Firmware Update* on STMicroelectronics'u poolt pakutav turvalise käivitamise valmislahendus oma erinevate STM32 seeria mikrokontrollerite jaoks. SBSFU koosneb kahest eraldiseisvast osast: alglaadur ja taasterežiim. Alglaaduri eesmärgiks on välkmälus hoitava rakenduse kontrollimine ja laadimine, ning osaliselt selle uuendamise läbiviimine. Alglaadur on ainuke turvalise käivitamise protsessi etapp enne rakendust, ning ühtlasi ka süsteemi usaldusjuure osa, seda hoitakse muutmatus välkmälu alas. Taasterežiim on programm, mis viib läbi arvuti vm ajutiselt ühendatud seadme kaudu rakenduse uuendamist. Alglaadur käivitab automaatselt taasterežiimi, kui sellel peaks rakenduse laadimine ebaõnnestuma, lisaks saab kasutaja vajadusel ise taasterežiimi siseneda. Sarnaselt alglaadurile, asub taasterežiimi programm muutmatus välkmälu alas.

2.2.4 Alglaadur

Alglaadur on STMicroelectronics'u poolt loodud komponent, mis põhineb kahel avatud lähtekoodiga komponendil: Trusted Firmware-M ja MCUboot.

Trusted Firmware-M e TF-M on Arm'i poolt välja töötatud turvalise tarkvarakeskkonna spetsifikatsioon nende Cortex-M seeria mikrokontrollerite jaoks mõeldud protsessoritele [18]. See sätestab standardid turvalise käivitamise jaoks, tarkvara turvaliseks ja mitteturvaliseks kihiks eraldamise ja nende kihtide suhtluse jaoks, ning turvalise kihi poolt pakutavad teenused, nt turvaline andmete hoiustamine ja süsteemi tarkvara turvaline uuendamine. Enamikku nendest standarditest SBSFU ei paku. STM32L552 kiibi-spetsiifiline SBSFU väljalase teostab TF-M standardite järgi vaid turvalist käivitamist, kuigi STM32L562 kiibile tehtud väljalaskes nimega „TF-M" (nimi ST poolt, mitte segi ajada Arm'i standardiga) on lisaks olemas turvalise hoiustamise ja krüptograafia teenused [19]. SBSFU'l on turvaline tarkvara uuendamise võimekus olemas, aga see on STMicroelectronics'u enda tehtud lahendus, mis pole TF-M standardite järgi implementeeritud. TF-M spetsifitseerib turvalise käivitamise jaoks kaks eraldi alglaadurit, BL1 ja BL2. BL1 on kiibi püsimälus hoitav esimese-etapi alglaadur, mis laeb ja kontrollib selle järgnevat BL2 alglaadurit [20]. BL2 asub välkmälus ja laeb ning kontrollib rakendust ennast [21]. Kuna STM32L5 mikrokontrolleritel pole olemas alglaaduri jaoks sisemist püsimälu, siis TF-M mõistes BL1 alglaadurit sellel pole. On vaid üks ja ainus alglaadur, BL2, mis on seega küll esimeseks etapiks, aga BL1 see olla ei saa, kuna kindlalt muutmatu püsimälu asemel asub see kirjutamise eest kaitstud välkmälus (täpsemad detailid hiljem), mis võib olla küll näiliselt muutmatu, aga 100% garanteeritult seda muutmatuks nimetada ei saa. Seega on SBSFU'l BL2 alglaadur, mis on veidi vastuoluliselt hoopis esimene etapp ja seega usaldusjuure osa. BL2 jaoks on Arm teinud kättesaadavaks avatud lähtekoodiga referentsimplementatsiooni, mille põhjal on kõik tootjate poolt loodud TF-M implementatsioonid mõeldud põhinema. Valdavalt seab BL2 ise paika vaid kõrgel tasemel juhtimisvoo ja liidesed, neid liideseid täidab omakorda MCUboot, mis on BL2'e lähtekoodiga kaasa pandud.

MCUboot on mikrokontrollerite jaoks loodud avatud lähtekoodiga turvaline alglaadur [22]. MCUboot sisaldab endas ühist alglaaduri arhitektuuri ja defineerib süsteemi poolt kindla välkmälu kasutuse jaotuse, ning on alglaadurina disainitud süsteemi tarkvara uuendamist hõlbustama. Lisaks alglaaduri enda pakkumisele, defineerib MCUboot ühise vormingu uuendusteks kasutatavate failide ehk uuendusfailide jaoks, koos skriptidega uuendusfailide koostamiseks. MCUboot, mis pakub liideseid selle ülal olevale TF-M BL2 koodile, sisaldab omakorda riistvaralisi portimisliideseid, mis tuleb implementeerida

kindla platvormi jaoks. SBSFU puhul on STMicroelectronics need liidesed ära implementeerinud.

Turvalise käivitamise jaoks teostab MCUboot rakenduse allkirja kontrolli RSA algoritmi abil, uuendusfailide krüpteerimist ja seadme sees nende lahti krüpteerimist peale uuendamist, ning vanemale versioonile tagasi uuendamise vältimise kontrolli. Varem mainitud TrustZone tehnoloogia on süsteemis kasutusel, seda nii käivitusprotsessi ajal kui ka uuendaja rakenduse töös. TrustZone mastaabis asub kogu alglaadur turvalise maailma sees, käivitamise alguses mitteturvaline maailm veel kasutusel ei ole.

Kasutaja näeb käivitamise kohta infot LCD ekraanil. Eduka käivitamise korral pole see midagi muud, kui momentaarne „Booting…“ tekst, peale mida juba rakendus käivitub. Kui tekib viga, mis pole just nii katastroofne, et kogu süsteemi töö koheselt seiskub, siis kuvatakse selle kohta ekraanile veateade, ning peale nupuvajutust siseneb süsteem taasterežiimi. Alglaadur toetab lisaks UART jadaliidest, millele saab arvuti vm välise seadmega ligi läbi plaadil oleva USB pesa, arendusplaadil leiduva USB-UART konverteri abil. Alglaadur saadab jadaliidese kaudu arenduse ja silumise otstarbeks detailsemat infot ja veateateid, aga sisendit sealt vastu ei võta. Taasterežiim kasutab jadaliidest kasutajale info näitamise ja talt sisendi saamise jaoks.

2.2.5 Taasterežiim (recovery)

Taasterežiimile algne STMicroelectronics'u poolt antud nimi oli *local loader* e *loader*, mis oli autori arvates kohmakas, kuna eelneva alglaaduri komponendi kohta kasutatakse inglise keeles tihti sõna *bootloader*. Seetõttu otsustati panna sellele uueks nimeks *recovery*. *Recovery* e taasterežiim on STMicroelectronics'u poolt loodud lihtne programm, mis käivitatakse alglaaduri poolt, kui süsteemi tavapärane käivitamine peaks ebaõnnestuma. Taasterežiimi saab kasutaja omal soovil siseneda käivitamise ajal arendusplaadil oleva nupu käivitamise ajal hoidmise abil. Üle jadaterminali kuvatakse kasutajale taasterežiimis joonisel 5 kujutatud menüü, kus ta saab läbi viia toiminguid püsivara haldamiseks.

Joonis 5: Taasterežiimi menüü

Peale taaskäivitamise on menüüs saadaval kaks valikut, millest SBSFU algses kujus oli olemas ainult esimene. Esimeseks valikuks on süsteemi püsivarauuenduse läbiviimine, teise valiku abil saab muuta süsteemis hoitavaid STSAFE turvaelemendiga sidumise võtmeid. Selle valiku otstarvet kirjeldatakse hiljem lähemalt, aga praegusel kujul on see olemas vaid arenduse ja silumise lihtsustamise eesmärgil, ning kindlasti ei esine see sellisel kujul tulevikus prototüübist välja kujunenud lõpliku lahenduse sees. Uuendamise valiku valimisel tehakse selle protsessiga algust, mille esimene samm on välkmälus rakenduse ala kustutamine. Seejärel laeb taasterežiimi programm teisel pool olevalt seadmelt alla MCUboot vormingus uuendusfaili. Uuendusfailid võivad esineda kas krüpteerimata või krüpteeritud kujul, turvalisuse huvides on ilmselt enamus kasutajate poolt eelistatud teine valik. Uuendusfailide saatmiseks kasutatakse YMODEM protokolli, mis on algselt modemitele mõeldud protokoll üle jadaliidese plokkide kaupa andmete saatmiseks [23]. Taasterežiim võtab ühe ploki vastu, kirjutab selle välkmällu, ning kordab sama ülejäänud plokkidega, kuni kogu püsivara on vastu võetud. Kui uuendusfaili vastu võtmine läks edukalt, siis on järgmiseks sammuks taaskäivitamine, peale mida asub alglaadur värskelt paigaldatud uuendusfaili kontrollima. Taasterežiimi enda sees ühtegi turvakontrolli ei toimu, nende eest vastutab alglaadur.

Võrreldes alglaaduriga, kasutab taasterežiim TrustZone'i puhul nii selle turvalist kui ka mitteturvalist maailma. Enamus taasterežiimi koodist asub mitteturvalises maailmas. Turvalises maailmas asub mitteturvalisele koodile saadaval väike kiht funktsioone, mis võimaldavad välkmälule ligipääsu. Välkmälu on jagatud turvalisteks ja mitteturvalisteks osadeks, ning mitteturvalisel koodil turvalistele osadele ligipääsu ei ole. Uuendamist läbi viiv kood asub mitteturvalises maailmas, ning uuendamise jaoks on sellel vaja ligipääsu rakendust hoidvale välkmälu alale. Rakendusel endal on omakorda turvaline ja mitteturvaline osa, ning mitteturvalisele osale on ligipääs olemas, aga turvalist osa hoidvale välkmälu alale mitte. Sellele alale saab uuendamist läbi viiv kood turvalise maailma funktsioonide abil ligi, mis lasevad sealset välkmälu kustutada ja kirjutada.

Muudele turvalistele aladele ei tohi mitteturvaline kood ligi pääseda, ning selle vältimiseks kontrollivad funktsioonid, et küsitud välkmälu aadressid oleks kindlalt lubatud alade sees.

Kui uuendamise käigus paigaldatud uuendusfail ei ole krüpteeritud, siis peale taaskäivitamist on uuendamise protsess sisuliselt lõppenud, kuna uuele rakendusele tehakse täpselt samad turvakontrollid, mida tehakse iga tavapärase käivitamise puhul. Kui uuendus on aga krüpteeritud, siis tuleb alglaaduril see lahti krüpteerida. Uuenduste lahti krüpteerimise otstarbeks sisaldab iga uuendaja seade RSA salajast võtit, mida hoitakse turvatud muutmatus välkmälu alas. Uuenduse algselt krüpteerimise protsessis kasutatakse sellele salajasele võtmele vastavat avalikku võtit. Selle võtmepaariga uuendusfailid ise otseselt krüpteeritud ei ole, kuna mitte-tillukese andmehulga krüpteerimine on reaalselt teostatav ainult sümmeetrilist krüptograafiat kasutades. Selleks otstarbeks kasutatakse AES-128 algoritmi [24], CTR režiimis [25]. Uuendusfaili loomisel genereeritakse suvaline sümmeetriline võti, millega seejärel uuenduse sisu krüpteeritakse. See võti omakorda krüpteeritakse ära eelneva võtmepaari avaliku võtmega, kasutades RSA-OAEP-2048 algoritmi [26], ning pakitakse uuendusfailiga kaasa. Õige salajase võtme seadmes olemasolul saab see krüpteeritud võtme lahti krüpteerida, ning seejärel lahti krüpteerida uuenduse enda.

Erinevaid krüpteerimise võtmepaare kasutades saab reguleerida, mis seadmetel on võimalik kindlaid uuendusi lahti krüpteerida. Kõige lihtsam on kasutada kõigi seadmete jaoks ühte ja sama võtmepaari. See tähendab aga seda, et võtmepaari salajase võtme lekkimisel on võimalik kõigi nende seadmete uuendusi lahti krüpteerida, ning selle lekkimine on reaalne risk. Selle riski mõningaks maandamiseks saab kasutada eri seadmete mudelite või perekondade uuenduste jaoks erinevaid võtmepaare. Teoreetiliselt oleks võimalik kasutada iga seadme jaoks unikaalset võtmepaari, kuigi ilmselt oleks see praktikas liiga keeruline. Seadmes hoitava uuenduste dekrüpteerimise salajase võtme turvamist ja sellega seonduvaid riske kirjeldatakse lähemalt hilisemates peatükkides, aga lühidalt on ründajal võtme välja lugemiseks vaja leida turvaauk alglaaduri sees.

3 Arendus

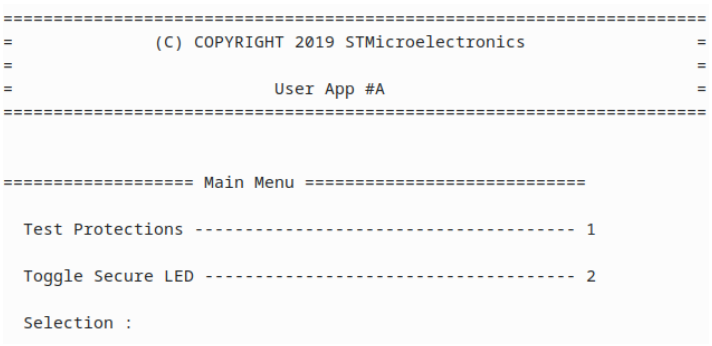
Järgnevate peatükkide teemaks on töö eesmärgiks olev tarkvaraarendus. Kirjeldatakse arenduse keskkonda ja selleks kasutatud tööriistu, ning arenduse erinevaid samme ja süsteemile lisatud funktsionaalsust.

3.1 Tööriistad

Peamiseks arenduskeskkonnaks oli STM32CubeIDE [27], STMicroelectronics'u ametlik vabavaraline IDE, mis põhineb Eclipse'il. Peale IDE oli lisaks kasutusel nende pakutav rakendus STM32CubeProgrammer [28], mille abil saab muuta teatud mikrokontrolleri seadistusi, ning mida kasutavad sisemiselt osad SBSFU *build*-skriptid mikrokontrollerile tarkvara üleslaadimiseks. SBSFU kasutab kasutajaliidesena peamiselt jadaterminali, mistõttu oli vaja ka terminaliprogrammi, selle jaoks kasutati Tera Term'i [29].

3.2 Projekti üles seadmine

SBSFU on üks suurest hulgast STMicroelectronics'u poolt pakutavatest näidisprojektidest, mida saab STM32CubeIDE-siseselt alla laadida. Oma algsel kujul koosneb SBSFU kolmest eraldiseisvast projektist, mis tuleb eraldi alla laadida ja üksteisega tööle saada. Esimesed kaks nendest on juba varasemalt kirjeldatud alglaadur ja taasterežiimi programm. Kolmas projekt on lihtne alglaaduri katsetamiseks mõeldud näidisrakendus, mis laseb kasutajal arendusplaadil LEDi vilgutada. Joonisel 6 on näha ekraanitõmmist näidisrakenduse menüüst.



Joonis 6: SBSFU näidisrakendus

Esimeseks ülesandeks oli mainitud kolm projekti saada kokku ühise IDE projekti alla. Selle jaoks liigutati need olemasoleva uuendaja rakenduse projekti sisse, mille käigus said neist rakenduse alamprojektid. TrustZone'i kasutavad projektid jagunevad veel omakorda turvaliseks ja mitteturvaliseks alamprojektiks. Kõigi kolme projekti puhul on TrustZone kasutusel, aga alglaadur on erand, kuna sellel mitteturvaline osa puudub, seega tuli lõpuks kokku 5 alamprojekti. Peale nende ühisesse kohta saamist oli SBSFU projekte võimalik kompileerida ja katsetada, aga SBSFU ja uuendaja rakendus veel üksteisega koos ei toiminud.

Peale IDE projektide tööle saamist tuli kiiresti ilmsiks üks parandatav aspekt. Kuigi kolm projekti olid eraldi kompileeritavad, siis eraldi plaadile üleslaadida neid läbi IDE polnud võimalik, kuna seda tuli teha läbi spetsiaalse skripti. Lisaks ebavajalikult pikale üleslaadimise ajale eemaldas skripti kasutamise vajadus ka läbi IDE koodi silumise võimaluse, mis on päris suur asi, mida kaotada. Skripti vajaduse põhjuseks oli see, et enne igat plaadile programmi üleslaadimist tuli selle välkmälu läbi STM32CubeProgrammer'i ära lähtestada, kuna vastasel juhul takistasid mikrokontrolleris sisse lülitatud välkmällu kirjutamise kaitse ja muud turvaseaded uue koodi üleslaadimist, peale selle esimest korda. IDE üksinda vajaminevat lähtestamist teha ei suuda. Vajalikud turvaseaded lülitab alglaadur automaatselt ise sisse siis, kui see avastab, et need veel sisse lülitatud ei ole, mis on vaikimisi olek peale välkmälu lähtestamist. Selle jaoks, et IDE-sisene üleslaadimine ja silumine tööle saada, tuli nende turvaseadete automaatne sisselülitamine lihtsalt eemaldada. Selle jaoks tehti alglaaduri IDE alamprojekti jaoks kaks konfiguratsiooni: väljalaske ja silumise konfiguratsioon. Silumise konfiguratsiooni korral on alglaaduris olev turvaseadete muutmisega seonduv kood läbi makrode eemaldatud, ning seda kasutades saab edukalt kõiki projekte eraldi otse läbi IDE üles laadida ja siluda.

3.3 Build-skriptide täiustamine

Arenduse esimeseks suureks eesmärgiks oli SBSFU näidisrakenduse asemel saada rakendusena tööle päris uuendaja rakendus. Enne seda tuli aga SBSFU projekti esialgsele *build*-süsteemile teha mõningaid parandusi. IDE'l üksinda päris kõikide vajaminevate sammude teostamiseks piisavalt võimekust ei ole, sellepärast on protsessi käigus vaja välja kutsuda skripte. Skriptid on kasutusel kõigi kolme alamkomponendi puhul. Alglaaduri alamprojektis on skript, mis käivitatakse peale selle kompileerimist, ning see

skript on sellele järgnevate skriptide jaoks tähtis. Skript loeb alglaaduri konfiguratsiooniga seotud päisefailidest seaded, ning kirjutab teiste skriptide tööks vajaminevate seadete väärtused nende skriptide enda sisse, muutes sealsete ridade teksti. Selline tegutsemisviis autori arvates päris sobilik ei olnud, ning alglaaduri skripti muudeti selliselt, et kõigi skriptide eraldi muutmise asemel muudetakse vaid ühte ühist skripti, mille ainukeseks otstarbeks on seadete hoidmine, ning mida saavad ülejäänud skriptid seadistuste väärtuste kättesaamiseks välja kutsuda.

Taasterežiimi alamprojekti skript käivitatakse samuti peale kompileerimist, selle eesmärgiks on kompileeritud mitteturvaline ja turvaline osa üheks binaarfailiks kokku kombineerida. Rakenduse alamprojekti skriptil on sama otstarve, lisaks kutsub see välja MCUboot'i uuendusfaili koostamise skripti, mis allkirjastab ja krüpteerib toore rakenduse binaarfaili, tehes sellest seadmele paigaldatava uuendusfaili. Kõik skriptid tehti arendajasõbralikumaks. Eraldi igas skriptis olevad väliprogrammide asukohad liigutati ühisesse faili, ning rakenduse loomise skripti puhul tehti rakenduse versiooninumbri muutmine lihtsamaks. Üks täiendus tehti ka eelnevalt mainitud plaadile koodi üleslaadimise skriptile. Kuigi IDE kaudu koodi üleslaadimine toimib, siis vahel on ka skripti kaudu selle tegemine kasulik, kuna skripti kasutades tehakse plaadile täielik lähtestamine. Skripti kaudu üleslaadimine oli aga aeglasem, kui IDE kaudu laadimine, mis tulenes sellest, et skript ei kutsunud STM32CubeProgrammer'it välja kiires režiimis, vaid tavalise kiirusega režiimis. Kiire režiimi kasutamisega muutus skript sama kiireks, kui IDE kaudu laadimine. Kõik ülalkirjeldatud muudatused tehti nii Windows'i kui ka Linux'i operatsioonisüsteemidega toimivate skriptide jaoks, et projektiga saaks hõlpsalt tööd teha mõlemat platvormi kasutavad arendajad. Üleüldiselt võttis kogu *build*-keskkonna tööle saamine ja täiustamine oodatust kauem aega. Enne alustamist eraldi sammuna see mõttes ei olnud, vaid pigem kiire kõrvalmärkusena, mille peale pidi vähe aega kuluma. Saadud mugavuse pärast oli aga kulutatud aeg ennast vägagi väärt.

3.4 SBSFU integreerimine

Peale skriptide korrastamist sai keskenduda SBSFU ja uuendaja rakenduse integreerimisele. Näidisrakenduse osad võeti projektist välja, peale selle *build*-skripti, mis kohandati uuendaja rakenduse jaoks. Peale uuendaja rakenduse kompileerimise koos uue skriptiga tööle saamist, oli nüüd olemas viis genereerida uuendaja rakenduse

uuendusfaile, mida sai läbi taasterežiimi paigaldada ning seeläbi testida. Uuendaja rakendus SBSFU'ga koheselt tööle ei hakanud, ning jooksis kiiresti kokku. Põhjuseks olid riistvara initsialiseerimisega seonduvad vajalikud muudatused, ja uuendaja rakenduse sees kindlate mäluaadresside kasutamine, mis olid peale SBSFU kasutusele võtmist nihkunud. Peale vajalike muudatuste tegemist hakkas uuendaja rakendus lõpuks sujuvalt tööle, ning seeläbi oli turvalise käivitamise tugi süsteemile edukalt lisatud.

3.5 Turvaelemendi tugi

Peale edukat turvalise käivitamise integreerimist, oli järgmiseks etapiks selle funktsionaalsuse täiustamine. Selle esimeseks sammuks oli alglaadurile STSAFE turvaelemendi toe lisamine. Turvaelement oli juba varasemalt olnud uuendaja rakenduse sees kasutusel. Plaaniks oli alglaaduris kasutada turvaelementi selle jaoks, et pakkuda turvalisemat vanemale versioonile tagasi uuendamise vältimise kontrolli. Oma algsel kujul on SBSFU alglaaduris versiooni kontrollimine olemas, aga turvalisuse mõistes jätab sealne implementatsioon soovida selle poolest, et see hoiab parasjagu paigaldatud versiooni väärtust välkmälu sees. Välkmälu ala, kus seda hoitakse, on turvatud ja ligipääsetav ainult alglaadurile (see on osa alast, mis lülitatakse välja peale alglaaduri lõpetamist). Siiski on see ala vabalt muudetav, kui sellele ligipääs peaks olemas olema, ning seeläbi on alglaaduri sees turvaaugu leidnud ründajal võimalik versiooni väärtust muuda ja vanema versiooni kontroll kahjutuks teha.

STSAFE-A110 turvaelemendil on olemas ühesuunalised loendurid [30], mille väärtust on võimalik lugeda, ja muuta ainult olemasolevast väiksemaks väärtuseks. Turvaelemendi näiteprofiilis, millega on seadistatud iga X-NUCLEO-SAFEA1 arendusplaadil olev turvaelement, on olemas kaks ühesuunalist loendurit, igaühel 500 000 sammu [31]. Üks nendest loenduritest võeti kasutusele versiooni väärtuse hoidmiseks, teine on vaba tulevikus kasutamiseks. Võrreldes versiooni hoidmisega välkmälus, kus saab seda potentsiaalselt ründaja ära muuta, hoiab ühesuunalise loenduri kasutamine kindlalt ära versiooni 'tagasi keeramise' võimaluse.

Turvaelemendi kasutusele võtmiseks tuli kõigepealt alglaaduri koodis sisse lülitada I^2C tugi ja tööle saada STMicroelectronics'u turvaelemendi draiver [32], mis uuendaja rakenduse sees juba toimis. I^2C tööle saamine võttis oodatust veidi kauem aega, aga lõpuks hakkas suhtlus turvaelemendiga toimima. Peale seda kirjutati ümber alglaaduris

turvaliste loendurite kasutamise liidest pakkuvad funktsioonid. Turvaliste loendurite liides toetab kuni 5 eraldi loenduri kasutamist, aga alglaadur kasutab neist ainult ühte, mis on kasutusel versiooni väärtuse hoidmiseks. Turvaliste loendurite jaoks tehti kaks implementatsiooni: esimene loeb ja uuendab päriselt turvaelemendis asuvaid loendureid, ning teine on silumise jaoks loendureid simuleeriv implementatsioon, mis hoiab nö võlts-loendureid välkmälus. Turvaelemendil arenduse jaoks ühesuunaliste loendurite väärtuste tagasi muutmise funktsionaalsust ei eksisteeri, loendureid simuleerides saab neid ohutult testida.

SBSFU sees turvaelemendi kasutuselevõtuga seoses tulid vajalikuks ka muudatused uuendaja rakendusele. Iga turvaelemendi sees on kaks unikaalselt genereeritud ja seda kasutava seadmega jagatud võtit, mida turvaelemendiga suhtlev seade kasutab turvaelemendilt saadud andmete autentimiseks ja lahti krüpteerimiseks, ning turvaelemendile saadetavate käskluste signeerimiseks. Seadmel on neid võtmeid vaja, et turvaelementi usaldada, ja vastupidi. Ründaja poolt nende võtmete kätte saamine võib potentsiaalselt tähendada seda, et ründaja saab seadme I^2C siini külge ühendatud riistvaraseadme abil, mis ei ole päris turvaelement, turvaelementi teeselda ning seeläbi turvalisust negatiivselt mõjutada. Varasemalt hoidis uuendaja rakendus neid võtmeid lihtsalt välkmälu sees, kiire ajutise lahendusena. SBSFU puhul asuvad võtmed nüüd turvatud välkmälu alas. Uuendaja rakendusel sellele alale igasugune ligipääs puudub, kuid turvaelemendi kasutamiseks on sellel võtmeid ikkagi vaja. Võtmed on vaja alglaadurilt rakendusele üle anda, aga alglaadur puhastab enne rakendusse sisenemist kogu muutmälu, seega läbi selle võtmeid üle anda ei saa. Lahendusena kirjutati alglaaduri lõppu assemblerkood, mis enne rakendusse sisenemist mõlemad võtmed protsessori registritesse kirjutab, ning rakenduse algusesse vastandiks assemblerkood, mis registrites olevad võtmete väärtused muutmällu kopeerib.

3.6 LCD ekraani tugi

Viimaseks täienduseks SBSFU'le oli LCD ekraani toe lisamine. Uuendaja rakenduse jaoks oli sama ekraani toe lisamine juba autori poolt varem läbi tehtud, seega selle peale väga palju aega ei kulunud. Ekraani tuge pakub GitHub'ist leitud Arduino I^2C ekraani teegi põhjal STM32 peale porditud teek [33], mis oli juba uuendaja rakenduses varem kasutusel. Uuendaja rakenduses toimub ekraani jaoks kasutatava I^2C kontrolleri

initsialiseerimine programmi main.c faili sees. SBSFU jaoks tehti teegile mõned muudatused, et kogu riistvara initsialiseerimisega seonduv kood asuks hoopis teegi enda sees. Seega on teek iseseisev ja mahu poolest ainult ühe lähtekoodi- ja päisefaili suurune, ning ei vaja enam enne kasutamist eelnevat initsialiseerimist, peale lihtsalt selle initsialiseerimisfunktsiooni välja kutsumise. Kui ekraan oli edukalt tööle saadud, lisati alglaaduri sisse uuenduste edenemise ja veateadete näitamine, ning taasterežiimi samuti uuenduste allalaadimise edenemise ja kasutajale juhiste näitamine.

4 Turvalisuse analüüs

Järgnevates peatükkides analüüsitakse SBSFU lahenduse poolt mikrokontrolleris saadaval olevate turvamehhanismide kasutust, ning võimalikke puudujääke ja turvariske nende kasutamisega seoses. Lisaks analüüsitakse SBSFU turvaeesmärkide täitmist.

4.1 Kasutatud turvamehhanismid

Edasised peatükid kirjeldavad igas peatükis ühte kasutusel olevat STM32 turvamehhanismi, selle kasutust SBSFU poolt, ning osade mehhanismide puhul nendega seonduvaid turvariske.

4.1.1 TrustZone

Nagu varasemalt kirjeldatud, on TrustZone protsessorisisene turvatehnoloogia, mis tükeldab programmi turvaliseks ja mitteturvaliseks osaks, jagades nende vahel ära välkmälu, muutmälu ja muud riistvaralised ressursid. TrustZone on kasutusel kogu süsteemi ulatuses, seda ka nii SBSFU komponentide kui ka uuendaja rakenduse puhul.

Tabelis 1 on kirjeldatud süsteemis kogu välkmälu kasutus, kogusuuruselt 512 kB. Mäluala võib olla TrustZone mõistes kas turvaline või mitteturvaline. Protsessor käivitub kindlalt seadistatud aadressil ning see aadress peab asuma turvalises alas. Ainult muudetavaid välkmälu alasid on võimalik kustutada ja kirjutada, vastasel juhul on võimalik ainult alast lugemine. Seda võimaldav WRP mehhanism on kirjeldatud hilisemas peatükis. Lisaks on olemas välkmälu alade peitmise mehhanism HDP, mida kirjeldatakse hilisemalt.

Tabel 1: Välkmälu alade kasutus ja konfiguratsioon

Aadress	**Kirjeldus**	**Muudetav**	**Turvaline/mitte**	**Peidus**
0x00000	STSAFE võtmed (endine versiooniloenduri ala)	Jah (ajutine)	Turvaline	Jah
0x01000	Allkirjade kontrolli, dekrüpteerimise võtmed	Ei	Turvaline	Jah
0x01800	Alglaadur	Ei	Turvaline	Jah
0x0E800	Alglaaduri nähtav osa	Ei	Turvaline	Ei
0x0F000	Rakendus+päis, turvaline	Jah	Turvaline	Ei
0x2E000	Rakendus+päis, mitteturvaline	Jah	Mitteturvaline	Ei
0x78000	Taasterežiim, turvaline	Jah	Turvaline[1]	Ei
0x7B000	Taasterežiim, mitteturvaline	Jah	Mitteturvaline	Ei

Alglaaduri programm asub tervenisti TrustZone turvalises maailmas. Taasterežiim jaguneb mitteturvaliseks põhiosaks ja väikseks turvaliseks osaks, mille abil saab rakenduse uuendamist läbi viiv mitteturvaline osa ligi rakenduse turvalisele osale. Samuti on rakendusel turvaline ja mitteturvaline osa. Kõik turvalises osas asuv on mitteturvalise osa eest kaitstud, küll aga asub mõnes turvalises osas tundlikke andmeid, millele päris kogu turvalise osa koodil ligipääsu ei tohiks olla. Välkmälu alguses on alglaaduris kasutatavad delikaatsed võtmed, mille lugemist teiste turvalise ala programmide poolt välditakse välkmälu peitemehhanismi abil.

4.1.2 Väljalugemise kaitse

Readout Protection, ehk RDP, on turvamehhanism, mis on STM32 mikrokontrollerite turvalisuse tuumaks [34]. See keelab ära mikrokontrolleri külge silumisseadme ühendamise ning seeläbi välkmälu koodi väljalugemise ja ülekirjutamise võimaluse, ja veelgi tähtsamalt keelab ära mikrokontrolleri turvaseadete edasise muutmise. RDP jaguneb tasemeteks, millest kõige tähtsamad on tasemed 0 ja 2. Tase 0 on vaikimisi olek, mis on igal värskel kiibil, selle taseme puhul on kogu mehhanismi poolt pakutud kaitse välja lülitatud. Silumine on võimalik täies mahus, mispuhul saab koodi lihtsasti välja lugeda ja muuta. Kasutada tohib kõiki võimalikke protsessori käivitusrežiime, sealhulgas

[1] Mitteturvaline, aga muudetakse alglaaduris turvaliseks enne taasterežiimi käivitamist

välisest mälust käivitamist. Tase 2 tähendab täielikku turvet, ning on sobilik seadmete jaoks, mille tootja tahab vältida mikrokontrolleris oleva koodi väljalugemist teiste poolt. Peale selle taseme aktiveerimist pole enam võimalik mikrokontrolleri turvaseadeid muuta ega seda muul viisil seadistada, aktiveerimise hetkel olevad seadistused jäävad lõplikuks. Kiibi silumise võimalus lülitatakse täielikult välja. Turvalisuse osas nõudlike seadmete puhul peaks nende lõppkasutajani jõudval kujul olema tase 2 aktiveeritud. Kui RDP tase 2 aktiveeritakse, tähendab see seda, et madalamale tasemele tagasiminek pole enam võimalik.

Siinkohal tasub mainida, et varasemate STM32 kiipide puhul on avastatud rünnakuid, mille abil saab ajutiselt RDP taset madalamaks muuta. Tegemist on riistvaraliste rünnakutega, mis ei ole triviaalsed, aga teostamiseks mitte ülemäära palju raha ega elektroonika vallas eksperttasemel teadmisi nõudvad. Näitena on toodud üks nendest rünnakutest STM32F2 ja F4 seeria mikrokontrollerite vastu, mis võimaldab kogu välkmälu sisu lugemist isegi RDP taseme 2 puhul [35]. See on pinge-*glitch*'imise tüüpi rünnak, mis on üks näide riistvaralistest rünnakutest üldisemas FI rünnakute (*fault injection* e veasüstimine) kategoorias. Lihtsamate FI rünnakute teostamiseks on loodud isegi valmistooteid, näiteks ChipWhisperer [36]. Pinge-*glitch*'imise põhimõtteks on mikrokontrolleri toiteviigu kiire pulseerimine toitepinge ja maa vahel, tekitades seeläbi toites häireid, mis kiibi toimimist täielikult ei seiska, aga kanduvad edasi protsessorile, kus tekivad seetõttu koodi täitmises vead. Kui pulseerimine ajastatakse täpselt õigeks ajaks nii, et vead tekivad turvakontrollide eest vastutava koodi täitmise ajal, kus tagajärjena täidetakse nt hargnemiskäske valesti, on võimalik nende turvakontrollide tulemust muuta. Sedasi saabki mikrokontrolleri panna uskuma, et RDP tase on madalam, kui see tegelikult on (RDP taset ei loe mitte riistvara otse, vaid kiibi tootmisel STMicroelectronics'u poolt eraldi välkmälualasse kõrvetatud käivituskood, mis on kõige esimene peale *reset*'i täidetav kood. Seetõttu on võimalik just tarkvaraliste häirete kaudu RDP taset 'muuta').

STM32F2 ja F4 seeriad on STMicroelectronics'u kontrollerite hulgas vanemad seeriad, kus turvalisusele on tähelepanu vähem pööratud. STM32L5 seeria puhul väidab STMicroelectronics, et nad on FI rünnakute vastu L5 seeria kiipe mingisugusel määral siseselt testinud, aga testimise tulemusi nad avaldanud ei ole [37]. Sellest hoolimata on Saß jt [38] L5 seeria kiipe edukalt rünnanud MFI rünnaku (*multiple fault injection* e mitmikveasüstimine) abil. Rünnaku sihtmärgiks oli TrustZone, mitte RDP, ja võrreldes

tavalise FI rünnakuga on MFI rünnakud palju rohkem ressursse ja aega nõudvamad, aga sellest hoolimata on turvalisuse murdmisest tõestus olemas.

4.1.3 Välkmälu kirjutamise kaitse

Write Protection, ehk WRP, on osades STM32 mikrokontrollerites olev turvamehhanism [34], mille abil saab valitud välkmälu aladel nende kirjutamise ja kustutamise ära keelata, tehes nendest seeläbi justkui muutmatud püsimälu alad. WRP poolt kaitstud välkmälu vahemikke saab olla kuni neli, täpsemalt kaks tükki vastavalt välkmälu esimesel ja teisel poolel. Vahemik ei saa katta mõlemat välkmälu poolt korraga.

WRP mängib olulist rolli turvalise käivitamise tagamisel. Just selle abil on STM32 mikrokontrollerites võimalik tekitada usaldusjuur, kuna see on ainuke viis mäluala tekitamiseks, milles hoitud andmeid pole võimalik pärast muuta. Muutmatud peavad olema nii alglaaduri kood, kui ka selle poolt rakenduse verifitseerimiseks kasutatav avalik võti, WRP kannab hoolt mõlema eest. Riskina tasub võtta seda, et WRP vahemikke on RDP tasemega madalam kui 2 võimalik muuta, seega on teoreetiliselt võimalik RDP taset madaldava rünnaku abil WRP vahemikelt kaitse eemaldada, seades WRP kui mehhanismi usaldusjuure pakkumiseks ohu alla.

4.1.4 Välkmälu peitmine

Hide Protection, ehk HDP, on osades STM32 mikrokontrollerites olev turvamehhanism [39], mis võimaldab välkmälu alasid protsessori jaoks välja lülitada peale kindla registri kirjutamist, muutudes uuesti nähtavaks vaid protsessori *reset*'i puhul. Seega on HDP abil võimalik tundlikke välkmälu alasid programmi täitmise käigus jäädavalt ära peita. Peidetavaid HDP alasid on saadaval kaks, üks välkmälu esimesel poolel ja teine teisel poolel. HDP vahemik ei saa katta mõlemat poolt korraga.

SBSFU raames kasutatakse HDP'd välkmälus oleva alglaaduri koodi ja selle tundlike andmete peitmiseks. HDP aktiveeritakse nii rakendusse kui ka taasterežiimi sisenedes, tehes mõlemale alglaaduri alad nähtamatuks. Alglaadur puhastab oma tööd lõpetades ka kogu muutmälu, seega on selle olemasolu järgnevatele programmidele sisuliselt nähtamatu. Väike osa alglaadurist peab jääma nähtavaks, et alglaadur HDP aktiveerimisel iseenda koodi ära ei peidaks, aga nähtava osa kood on seotud vaid järgmise programmi koodi hüppamisega ning ei ole turvalisuse poolest tundlik. HDP peidab ära enamus

alglaaduri koodist, uuenduste lahti krüpteerimiseks kasutatava salajase võtme, ning turvaelemendi unikaalsed võtmed.

HDP puhul on võimalik turvarisk see, et kuna HDP lülitatakse sisse just registrisse kirjutamise teel, siis ründajal mingisugusel viisil sinna kirjutamise ära hoidmine tähendab seda, et HDP ei aktiveeru, kuigi see peaks. Üks võimalik viis seda teha on eelnevalt mainitud riistvaraliste FI rünnakute teel, mille abil ründaja HDP registrist sisse lülitavas koodis häireid tekitab. FI rünnakuid saab raskendada, kirjutades nende osas tundlikku koodi just neid silmas pidades, nt HDP registri näite puhul seda sisse lülitada mitu korda, mitte ainult üks kord. Veel üks viis FI rünnakuid raskendada on programmi suvalise pikkusega viivituste sisestamine selleks, et rünnaku ajastamist raskendada, see aga nõuab tõeliselt juhusliku juhuarvude generaatori olemasolu (tarkvaraline juhuarvude generaator siinkohal ei sobi). STMicroelectronics on SBSFU koodi lisanud mõned lihtsamad vastumeetmed FI rünnakute jaoks: HDP sisse lülitamise funktsiooni kutsutakse välja mitmes eraldi kohas, ja funktsioon kontrollib peale kirjutamist registri väärtust, et see tõesti sisse lülitatud oleks.

4.1.5 MPU

Memory Protection Unit, ehk MPU, on osadel STM32 mikrokontrolleritel leiduv riistvaraplokk, mis võimaldab kogu protsessori aadressiruumi raames mälualade lugemise, kirjutamise, ja koodi täitmise volitusi reguleerida [40]. MPU'l on saadaval 16 regiooni, 8 nendest TrustZone jaoks turvalised ja 8 mitteturvalised. Igas regioonis saab defineerida selle alg- ja lõppaadressi, ja regiooni volitused. Volituste kaudu saab regioon olla nii loetav kui ka kirjutatav, ainult loetav, või üldsegi ligipääsmatu. Lisaks saab regioonis olla protsessori poolt seal koodi täitmine kas lubatud, või mitte. Peale selle on võimalik reguleerida regiooni vahemälustrateegiat ja muid täpsemaid seadeid. Mitte ühegi regiooni poolt kaetud mälualadel on lubatud kõik volitused.

Oma töö käigus piirab alglaadur koodi täitmise volituse enamikel aladel. Nendeks aladeks on rakenduse välkmälualad, välkmälus võtmeid hoidvad alad, kogu muutmälu, ja kõik registrid. Täitmise volitus jääb ainult alglaaduri enda koodile, ja taasterežiimi koodile, aga alglaaduris enamus ajast vahetult enne lõppu on taasterežiimi ala veel mitteturvaline ja alglaadur ise turvaline, nii et alglaadur taasterežiimi koodi niisama täita ei saa. Rakenduse käivitamisel antakse rakenduse aladele täitmise volitused.

MPU abil protsessori poolt koodi täitmise volituste piiramine raskendab oluliselt turvaaukude leidmist. Paljude turvaaukude ärakasutamiseks on vaja, et ründajal oleks võimalus muutmälust sinna enda poolt kirjutatud koodi täita, kogu muutmälu raames koodi täitmise keelamine välistab selle täielikult. Registrite alas täitmise keelamine välistab ka seal potentsiaalselt leiduvate seadmete, kus võib äkki olla hulk registreid, mida ründaja saab ajutiselt väikese mälualana 'taaskasutada', ründe eesmärgil kasutamise. Tasub märkida, et kuigi muutmälu koodi täitmise keelamine raskendab turvaaukude leidmist, on see kaugel nende leidmise võimatuks tegemisest. ROP-tüüpi rünnakuid [41] (*return-oriented programming*) kasutades on võimalik turvaauke ära kasutada ka juhul, kui ründajal puudub võimalus omalt poolt mällu kirjutatud koodi täita. Selle asemel seisneb ROP rünnatavas programmis olemasolevate väikeste koodijuppide ründe eesmärgil ära kasutamises. ROP turvaaukude leidmine on keerulisem, kui tavaliste puhul, aga sellest hoolimata on nendest tulenev turvaoht reaalne. ROP rünnakute raskendamiseks on olemas nii tarkvaralisi kui ka riistvaralisi vastumeetmeid, aga väheste ressurssidega sardsüsteemides, milleks on ka tööalune uuendaja, on neid keeruline rakendada. See-eest on potentsiaalselt rünnatavat 'pinda' vähem, kui tüüpilise süsteemi puhul, ja selle ründamine on nt varem mainitud teiste rakenduste täitmisvolituste piiramise tõttu keerulisem, aga oht püsib.

4.1.6 Krüptograafiakiirendid

STM32L5 seeria mikrokontrolleritel on olemas krüptograafiliste algoritmide rakendamiseks riistvaralised kiirendid, saadaval olevad kiirendid sõltuvad täpsemalt kasutusel olevast alamudelist. Kõigil seeria kiipidel on olemas HASH kiirendi, mis võimaldab SHA-1, SHA-2 (SHA-224 ja 256) ja MD5 räsialgoritmide kiirendamist [42]. STM32L562 kiipidel on olemas lisaks HASH kiirendile ka AES ja PKA kiirendid. AES kiirendi toetab AES sümmeetrilise krüpteerimisalgoritmi kiirendamist, seda nii 128 ja 256-bitiste võtmetega, ja peaaegu kõikide algoritmi kasutusrežiimidega (ECB, CBC, CTR, GCM, GMAC ja CCM) [43]. PKA, ehk *Public Key Accelerator*, on avaliku võtme krüptograafia kiirendi, mille abil on võimalik kiirendada RSA, DH ja ECC algoritmide arvutuslikult intensiivseid osi [44].

Hetkel uuendaja poolt kasutataval STM32L552 kiibil on ülal mainitud kiirenditest olemas ainult HASH kiirendi. See kiirendi on kasutusel nii SBSFU poolt, mis kasutab selle SHA-2 võimekust allkirjade kontrollimisel, kui ka uuendaja rakenduse poolt, mis samuti

kasutab SHA-2'te. Turvalisuse mõistes HASH kiirendi kasutamine või mitte kasutamine erinevust ei anna, põhiline võit tuleb väiksema koodi mahu näol, võrreldes puhtalt tarkvaralise teegi kasutamisega. Jõudlus on ilmselt halvem, aga kui pole tegemist just rakendusega, millel on vaja suures koguses räsialgoritmide arvutusi teha, siis pole see üldse märgatav, ning siin pole sellega tegemist. Veel üks eelis on oluliselt madalam võimsustarve, aga hetkel prototüübi jaoks see tähtis ei ole.

AES ja PKA kiirendeid hetkel kasutada ei saa, kuna kiip neid ei toeta, aga nende kasutusele võtmisel oleks mitmeid eeliseid. AES'i kasutab nii alglaadur kui ka uuendaja rakendus. PKA saaks kasutusele võtta alglaaduris, seevastu uuendaja rakenduses toimub enamus avaliku võtme krüptograafiat juba turvaelemendi kaudu. Nagu HASH kiirendi puhul, paraneks mõlema kiirendiga nii koodi maht, kui ka võimsustarve. PKA kiirendit kasutatakse vähe ja jõudlust see ei mõjutaks. AES kiirendit kasutatakse veidi intensiivsemalt erinevate andmehulkade lahti krüpteerimiseks, aga jõudlus ikkagi halveneda ei tohiks, kuna mõlemal kasutusjuhul esinevad juba enne lahti krüpteerimise kiirust teised pudelikaelad, alglaaduri puhul välkmälu kirjutamise kiirus.

Turvalisuse mõistes esineks ka eeliseid, eelkõige parem kindlus kõrvalkanalirünnete (ingl k *side-channel attack*) [45] vastu. Krüptograafia mõistes tähendab see võtmete kätte saamise eesmärgil tehtud rünnakut, mis seisneb valitud, või mõnel juhul juhuslike andmete krüpteerimise ajal mõne kõrvalkanali, nt krüpteerimiseks kulunud aja või võimsustarbe mõõtmises, ja seejärel kogutud andmete põhjal võtme tuletamises. Tarkvaraliste krüptograafiateekide vastu on tihti levinud ajastusrünnakud (ingl k *timing attack*) [46], mis toimivad krüpteerimise ajakulu mõõtmise abil. Ajastusrünnakuid on praktiliselt tehtud nii tarkvaraliste AES implementatsioonide vastu [47], kui ka RSA implementatsioonide vastu [48]. Nii alglaaduris kui ka uuendaja rakenduses kasutatav Mbed TLS krüptograafiateek [49] ei luba hetkel täielikku kaitset ajastusrünnakute vastu [50]. Riistvaralise kiirendi kasutamine hoiaks ära lihtsamad kõrvalkanaliründed nagu eelmainitud ajastusrünnakud, aga selle kasutamine ohtu täielikult ei eemalda. Kõrvalkanaliründeid on kasutatud ka riistvaraliselt teostatud krüptograafia vastu. Üks näide sellistest rünnetest on DPA (*differential power analysis*, diferentsiaalne võimsusanalüüs) [45]. DPA rünnaku põhimõtteks on teostada suurel hulgal krüpteerimis- või dekrüpteerimisoperatsioone kasutades võtit, mida ründaja soovib kätte saada, ning mõõta riistvara võimsustarvet kõigi nende operatsioonide ajal. Kui andmeid on kogutud piisavalt, on võimalik neid statistiliselt analüüsida ja seeläbi kasutatud võti tuletada. DPA

rünnakutest veelgi efektiivsemad on DFA rünnakud (*differential fault analysis*, diferentsiaalne veaanalüüs) [51]. DFA rünnak on sarnane varem mainitud üldisematele FI rünnakutele selle poolest, et tähtsaks osaks on tahtlik riistvaraliste vigade tekitamine, nt toitepinge *glitch*'imise või elektromagnetiliste häirete tekitamise kaudu. Eesmärgiks on eelnevalt teada olevate andmete (de)krüpteerimisoperatsioonide ajal tekitada häireid, mille mõjul operatsiooni tulemus õige tulemusega võrreldes muutub. Kogudes piisavalt vigaseid tulemusi, mille puhul on täpselt õigel ajal häired tekitatud, on võimalik nende tulemuste põhjal krüpteerimisalgoritmi sisemist olekut analüüsida, ja seeläbi võti kätte saada. Riistvaralised kõrvalkanaliründed on ohuks, aga nõuavad füüsilist ligipääsu riistvarale ja nende teostamine on tarkvaraliste rünnakutega võrreldes oluliselt kallim ja keerukam.

4.2 Turvaeesmärgid

Järgnevad peatükid analüüsivad SBSFU koodi selle ründekindluse poolest, ning analüüsivad lisaks kolme SBSFU lahenduse turvaeesmärki ja nendega seonduvaid turvariske.

4.2.1 Koodi ründekindlus

Selle alapeatüki fookuseks on vaadelda SBSFU koodi turvalisuse vaatepunktist, ja välja tuua turvaaukude leidmise mõistes selles kõige kriitilisemad kohad. Kriitiline kood on selline, mida on võimalik ründajal otse või kaudselt proovida ära kasutada turvalisuse õõnestamise eesmärgil. Nt mõne pildiformaadi dekodeerimise kood, mis hakkab vigase pildifaili parseerimisel hoopis kõrvalist mälu üle kirjutama, andes halvimal juhul ründajale viisi panna süsteem täitma tema poolt sinna lisatud pahatahtlikku koodi. Koodi, millele kasutaja otse sisendeid anda saab ja seeläbi ka ründaja, on SBSFU's suhteliselt vähe, aga siiski sellist keerukat ja selletõttu kriitilist koodi leidub. Peamine riskantne koht on alglaaduris rakenduse uuendusfailide parseerimist läbi viiv kood, mis loeb igas uuendusfailis olevat päist, kus hoitakse mitut keerukat andmestruktuuri, mille sisu saab ründaja vabalt valida. Veel üks koht on taasterežiimis YMODEM protokolli kaudu andmete vastuvõtmise kood, mis juba oma olemuselt peab ründajalt andmeid vastu võtma, ning nende põhjal kergelt keeruka protokolli järgi suhtlust läbi viima. Kriitilisi kohti leidub veel, nt STSAFE turvaelemendiga suhtlust läbi viiv kood, aga kuna selle võimalik ärakasutamine nõuab juba riistvaralisi muudatusi, siis nii kriitiline see ei ole.

Ideaalselt tuleks kogu mainitud kriitilisele koodile läbi viia põhjalik analüüs turvalisuse vaatepunktist, ja seda testida vigade leidmiseks, aga aja nappuse tõttu tuleb hetkel piirduda lihtsalt koodi turvalisuse pealiskaudse hindamisega. Alglaaduri koodi puhul on selle kirjutamisel vägagi arusaadavalt just turvalisusele tähelepanu pööratud, ning seetõttu on autor selle koodi turvalisuses päris kindel. YMODEM'i kood, mis tuleb STMicroelectronics'ult, sellega võrreldes nii kindel välja ei näe, aga tegemist on ikkagi suhteliselt lihtsa protokolliga, nii et selle koodi üle kontrollimine väga aeganõudev olla ei tohiks. Lisaks asub see kood TrustZone'i mitteturvalises maailmas, mis tähendab, et ainuüksi sealse turvaaugu leidmisest ei piisa, vaid ründajal tuleb ka TrustZone'i turvalisusest mööda saada. Korralik turvalisusele keskenduv koodibaasi kontroll kuluks ära küll, aga hetkeseisuga on autor enamus osade arvatava turvalisusega rahul.

4.2.2 Allkirjade kontroll

Allkirjade kontrolli turvalisus on turvalise käivitamise tagamiseks kõige tähtsam osa, kuna sellest mööda saamine tähendaks võimalust käivitada mistahes tarkvara. Peamised riskid on vigane krüptograafiliste algoritmide implementeerimine, mis võib oluliselt nende poolt pakutavat turvataset vähendada. Teine risk on loogikavead allkirjade kontrolli käigus või nende ümber asuvas koodis, mis võimaldavad kontrolli läbida isegi siis, kui see tegelikult läbitud olla ei tohiks. Mõlema kategooria puhul on autor kindel, et kasutatav Mbed TLS krüptograafiateek ja kontrollidega tegelev MCUboot ja TF-M on oma otstarveteks enam kui kõlbulikud. Peale kontrollide enda murdmise, on võimalik risk allkirjastamiseks kasutatava salajase võtme lekkimine, mille toimumine tähendaks kogu turvalisuse kokku kukkumist. Võtmete turvalisena hoidmine on keeruline ülesanne, aga selle ülesande lahendamiseks on olemas riistvaralised turvamoodulid ehk HSM'id (*Hardware Security Module*) [52], mis on väga kõrge turvatasemega seadmed, mille ülesandeks on võtmeid jm salajast materjali enda sees hoida, ja nende võtmetega allkirjastamist vm tegevusi läbi viia, aga võtmed jätta seejuures välisele maailmale täiesti kättesaamatuks. HSM'i kasutamise puhul genereeritakse salajane võti HSM'i sees, ning võti sealt seest ei välju kunagi. Vähemalt ühel korral on toimunud päris süsteemi puhul juhtum, kus allkirjastamiseks kasutatud salajased võtmed on lekkinud, seda kurikuulsa Sony PlayStation 3 võtmete lekkimise intsidendi näol [53], aga peale selle on sarnased juhtumid olnud praktiliselt olematud.

4.2.3 Vanema versiooni paigalduse takistamine

Vanemate versioonide paigaldamine ei ole päris otsene süsteemi turvalisuse murdmine, aga on selle jaoks päris suur turvarisk, kuna selle kaudu saab potentsiaalselt ära kasutada vanemates versioonides leiduvaid turvaauke. STSAFE turvaelemendi kasutamine versiooniloenduri hoidmiseks on päris robustne lahendus, aga 100% riskivaba see ei ole, kuna leidub ikkagi üks riistvaraline ja keerukas teoreetiline rünnak, mille abil versioonikontrolli petta saaks. Turvaelemendilt sellele saadetud käskude vastusena saadud andmeid on võimalik autentida, et veenduda, et need andmed võltsitud ei oleks, ning see autentimine on muidugimõista versioonikontrolli jaoks kasutusel. Paraku aga tehakse seda autentimist kahe võtme abil, mis on mõlemad olemas nii turvaelemendil kui ka seda kasutaval seadmel. See tähendab seda, et võtmed peavad seadmel alati kusagil kättesaadaval olema, ning kui need saab seadmest endale kätte ka ründaja, siis on tal võimalik turvaelement süsteemile füüsilise ligipääsu ja piisavate elektroonikaoskustega asendada omaenda võlts-turvaelemendiga, mis saadud võtmete abil saab seadmele anda autentsena näivaid vastuseid, ning seeläbi versiooniloenduri väärtust võltsida. Isegi kui võtmeid proovida peita nii hästi kui võimalik, pole neid võimalik täielikult ära peita, kuna turvaelemendiga suhtlemiseks peavad võtmed alati muutmälus saadaval olema, kust saab neid ka ründaja piisava ligipääsuga välja lugeda. Rünnak on vägagi keerukas, aga ikkagi üks asi, mida tasub silmas pidada. Alternatiivse variandina turvaelemendis versiooniloenduri hoidmisele, on kasutada selle jaoks mikrokontrolleri-sisest OTP mälu. Mälu on ühekordselt programmeeritav ja peale seda muutmatu, aga miinuseks on see, et STM32L5 puhul OTP mälu omapära tõttu saab seda kasutades versiooniloenduri maksimaalne pikkus olla ainult 32 sammu. Sellise lahenduse puhul tuleb versiooniloendurit konservatiivsemalt kasutada, tõstes selle väärtust ainult turvalisuse poolest kriitiliste uuenduste puhul.

4.2.4 Uuenduste krüpteerimine

Uuenduste sisu krüpteerituna hoidmine on keeruline ülesanne. Selle eesmärgi all mõeldakse krüpteeritud kujul uuendusfailide lahti krüpteerimiseks kasutatava võtme turvalisena hoidmist, mitte mikrokontrolleri sees lahti krüpteeritud kujul oleva rakenduse välja lugemist. Raskeks teeb selle asjaolu, et kõigi krüpteeritud uuenduste lahti krüpteerimiseks on vaja ründajal kätte saada ainult üks võti, mille abil need kõik on krüpteeritud. Töös on varem mainitud erinevate krüpteerimisvõtmete kasutamist sõltuvalt nt tootemudelist või mõnest muust kategooriast, mis puhul on kõigi uuenduste katmiseks

vaja rohkem võtmeid kätte saada, aga probleem jääb ikkagi alles selles mõttes, et ründajal on ühe soovitud uuendustegrupi võtme saamiseks vaja süsteem murda ainult ühel korral. Selle asjaolu tõttu on riskina sobilikumad ka mõnevõrra kallimad ja keerukamad rünnakud. Kaitsealune võti on uuenduste ühekordsete võtmete lahti krüpteerimiseks kasutatav salajane võti, mida hoitakse turvatud välkmälu alas. Kuigi see võti on nii TrustZone turvalises alas kui ka peitealas olemise tõttu saadaval ainuüksi alglaadurile, siis põhimõtteliseks probleemiks jääb ikkagi see, et võti on loetav tarkvara poolt, ning kui ründaja leiab turvaaugu piisavalt varakult käivitusprotsessi sees, on sellega ka võti paljastatud. Kui FI rünnaku abil on ründajal võimalik HDP kaitse kahjutuks teha, nagu varasemalt mainitud, on võimalik ka hilisemal ajal võtit välja lugeda. Uuenduste krüpteerimise turvamudel on piisavalt tugev selleks, et nende sisu hoida salajasena keskmise huvilise eest, aga tähtis on meeles hoida seda, et piisavalt motiveeritud ründaja eest nende sisu ilmselt salajaseks lõpuks ei jää. Krüpteerimise turvalisust suurendaks oluliselt puhtalt riistvaralise krüpteerimise kasutamine. Mitmetes kiipides on levinud riistvaralised krüptograafiamootorid, mis ei teosta krüpteerimist mitte protsessori poolt antud võtmetega, vaid turvaliselt mootori enda sees hoitud võtmetega, mis on tootmise ajal vm ajal sinna eelnevalt sisse kirjutatud. STM32L5 ja teistes STMicroelectronics'u seeriates olevates krüptograafiakiirendites sellist võimekust ei ole. Sellise lahenduse kasutamine teeb võtmete kätte saamise oluliselt raskemaks, aga mitte võimatuks, kuna liiga nõrgad krüptomootorid on ikkagi haavatavad varasemalt mainitud DPA ja DFA rünnakutele sealsete võtmete välja lugemiseks.

5 Tulemused

Lõpetuseks on siinsetes peatükkides ära kirjeldatud töö tulemus, valminud lahenduse testimine ja selle tulemused, ning tööga seotud arengukohad tuleviku jaoks.

5.1 Saavutatud eesmärgid

Autori arvates võib öelda, et arendustöö tulemusena said töö eesmärgid edukalt täidetud. Turvalise uuendaja prototüübile ja sealsele uuendaja rakendusele sai juurde integreeritud turvalise käivitamise lahendus, ning lahendusele endale lisati turvaelemendi ühesuunaliste loendurite kasutamise tugi turvalisuse suurendamiseks, ja LCD ekraani tugi kasutajasõbralikkuse suurendamiseks. Lisaks sai süsteem turvalisuse poolest ära analüüsitud. Selle analüüsi tulemusi lühidalt kokku võttes võib öelda, et süsteem on lihtsamate rünnakute vastu, mis on teostatud väheste oskuste ja/või ressurssidega ründaja poolt, igati turvaline. Seevastu on aga keerukamate riistvarapõhiste rünnakute vastu kaitsmise alal veel palju arengumaad.

5.2 Arengukohad

Töö käigus valminud turvalise käivitamise lahenduse näol said sellelt nõutud eesmärgid rahuldatud. Siiski on mitmeid kohti, mida oleks saanud käsitleda põhjalikumalt, või mida annab tulevikus edasi arendada või uurida. Siinpool on mõned sellised arengukohad välja toodud.

5.2.1 Testimine

Just kõrgele turvalisusele keskenduva tarkvara puhul on väga tähtsal kohal selle testimine. Muidugimõista on vajalikud tavalised üksustestid ja funktsionaalsed testid. Lisaks sellele on tähtis koodibaasi staatiline defektianalüüs selleks mõeldud tööriistade, nt Coverity [54] abil. Peale selle on tähtsal kohal programmide dünaamiline analüüs nendes vigade leidmiseks, nt meetodite nagu *fuzzing* abil. Kuna aga hetkeseisul on autor projekti meeskonnas ainuke arendaja ja lisaks sellele veel poole kohaga, siis prioriteetsemate ülesannete tõttu korralikuks testimiseks aega ei olnud, ja tuli piirduda vaid käsitsi läbi

viidud lihtsa pinnapealse testimisega veendumaks, et vähemalt nõutud funktsionaalsus toimib ja selgetes veaolukordades tarkvara ootamatult ei käitu. Testiti uuendamise läbiviimist toimivate kui ka mittetoimivate stsenaariumitega, mittetoimivatest nt vale allkiri või vanem versioon, kui lubatud. Lisaks sellele testiti süsteemi veakindlust vigaste uuendusfailidega, ning lõpuks testiti uuendamise veakindlust arvutiga ühenduse katkestamise, või järsu toite eemaldamise korral. Autori jaoks olid selle testimise tulemused rahuldavad ning süsteem toimis, nagu pidi nii tava- kui ka veaolukordades. Muidugi on testimine kogu uuendaja projekti juures ala, millele tuleks tulevikus palju rohkem tähelepanu pöörata.

5.2.2 Riistvaraliste krüptograafiakiirendite kasutamine

Varasemalt said mainitud nii sümmeetrilise kui ka avaliku võtme krüptograafia otstarbeks tarkvaralise teegi asemel mikrokontrolleri riistvaraliste krüptograafiakiirendite kasutamine, ning nende kasutamisest tulenevad eelised. Hetkel neid kiirendeid kasutada ei saa, kuna kasutataval STM32L552 mikrokontrolleril neid ei ole, seevastu sama seeria STM32L562 mikrokontrolleril on need olemas. Kui tulevikus peaks see kontroller kasutusele tulema, või mõni muu sarnase seeria oma, kus on kiirendid olemas, siis nende tööle saamine peaks olema suhteliselt kiire protsess. STMicroelectronics'u poolt on saadaval Mbed TLS teegile riistvaraliste kiirendite tuge pakkuvad lähtekoodifailid, millega olemasolevate lähtekoodifailide asendamisel ja koodi sees lisaks mõne muu väikese muudatuse tegemise järel peaks need toimima.

5.2.3 Versiooniloenduri hoidmine OTP mälus

Varasemalt toodi välja idee, et hoida alglaaduri versiooniloendurit mikrokontrolleri ühekordselt programmeeritavas OTP mälus. Võrreldes hetkel kasutatava turvaelemendi ühesuunalise loenduri lahendusega on eeliseks see, et selline lahendus ei oleks haavatav varasemalt mainitud teoreetilisele riistvaralisele turvaelemendi võltsimise rünnakule. Miinuseks on aga see, et OTP mälu loendurina kasutamise puhul on kogu OTP mälu selleks kasutades loenduri maksimaalne pikkus ainult 32 sammu. Seega kõigi tehtavate uuenduste jaoks loendurit kasutada ei saa, vaid tuleb seda tõsta ainult süsteemi turvalisuse poolest tähtsate uuenduste puhul. Teoreetilise rünnaku päriselt teostatavus, sellise lühikese loenduri kasutamise praktilisus ja loenduri võimalik pikkus teistel STM32 mikrokontrolleritel on kõik teemad, mida tasuks lähemalt uurida.

5.2.4 SBSFU lisaseaded

Töö jaoks rakendati SBSFU lahendust selle vaikekonfiguratsioonis, aga peale selle on veel olemas erinevaid konfigureerimise võimalusi [55], millest osad lisavad teatud määral uut funktsionaalsust. Üks väga kasulik seade, mida saab sisse lülitada, on kahe rakendust hoidva partitsiooni kasutamine. Vaikimisi on kasutusel ainult üks partitsioon, mis toimib nagu täiesti tavaline süsteem. Kahe partitsiooni kasutamine annab aga oluliselt töökindlust juurde, kasutades teist partitsiooni esimese varupartitsioonina. Üks partitsioon hoiab rakenduse tavalist koopiat, mis igapäevasel kasutamisel käivitatakse. Kui kasutaja asub süsteemi uuendama, siis uuendamisel paigaldatakse uus koopia teisele partitsioonile. Kui mistahes põhjusel peaks uuendamisel tekkima viga ning uut koopiat käivitada ei saa, siis saab lihtsalt tagasi minna partitsioonile, kus on vana koopia alles, jättes süsteemi ebaõnnestunud uuendusest hoolimata töökorda. Kahe partitsiooni kasutamist töö käigus ei rakendatud, kuna uuendaja rakendus sel hetkel mahu poolest põhjalikult optimeeritud ei olnud, ning kaks rakenduse koopiat ei oleks välkmällu mahtunud. Kui tulevikus peaks maht piisavalt vähenema, saab ka kahe partitsiooni süsteemi kasutusele võtta.

Lisaks kahe partitsiooni kasutamisele, on veel üks valik vajadusel rakenduse eraldi (TrustZone) turvaliseks ja mitteturvaliseks osaks eraldamine, mispuhul on võimalik neid uuendada eraldi. Uuendaja rakenduse puhul on turvaline ja mitteturvaline osa üks tervik, mistõttu seda seadet kasutada polnud tarvis. Viimane tähelepanu vääriv valik on allkirjastamiseks kasutava krüptograafiaalgoritmi valik. Kolm saadavalolevat valikut on RSA-2048, RSA-3072 ja ECDSA-256. Vaikimisi on kasutusel RSA-2048, mis on autori arvates hetkeseisuga täiesti sobilik.

6 Kokkuvõte

Lõputöö eesmärgiks oli STM32 mikrokontrolleril põhinevale turvalise uuendaja süsteemile lisada turvalise käivitamise tugi SBSFU valmislahenduse põhjal, ning lisaks SBSFU turvalisust ja funktsionaalsust täiustada. Tulemusena said süsteemi olemasolev uuendaja rakendus ja SBSFU turvaline alglaadur üksteisega integreeritud, lisaks sai SBSFU alglaadurile lisatud STSAFE-A110 turvaelemendi ühesuunaliste loendurite kasutamise tugi, ning LCD ekraani kaudu kasutajale info näitamine.

Töö koosnes nii praktilistest kui ka teoreetilistest osadest. Teoreetilistes osades tutvustati ja kirjeldati turvalise uuendaja süsteemi riistvara, turvalise käivitamise põhimõtteid ja mõisteid, SBSFU valmislahenduse olemust ja tööpõhimõtteid, võimaliku rünnakuid SBSFU lahenduse vastu ning analüüsiti valminud turvalise käivitamise lahendust turvalisuse vaatepunktist. Praktilistes osades kirjeldati töö eesmärgiks oleva arenduse käiku, tulemusi ja lõpptulemuse testimist. Arendus viidi edukalt läbi ja selle eesmärgid said täidetud, ning turvalisuse analüüsi tulemusena järeldati, et süsteem on nõrgemate rünnete vastu küllaltki turvaline, aga keerukamate ja kallimate rünnete mõistes kogenud ründaja poolt on veel palju maad minna.

Tulemusena on valminud toimiv turvalise alglaadimise süsteem, mida saab edasi portida teistele STM32 seeria mikrokontrolleritele. Süsteemi annab edasi täiustada lisaks hetkel kasutusel olevatele täiendavate STM32 riistvarafunktsioonide kasutusele võtmisega, ning SBSFU’s kahe uuenduspartitsiooni kasutusele võtmisega.

Kasutatud kirjandus

[1] Proekspert AS, „Secure Firmware Updater technology,“ 20.03.2024. [Võrgumaterjal]. Saadaval: https://proekspert.com/blog/connected-products/secure-firmware-updater-technology/.

[2] Euroopa Liit, „The European Cyber Resilience Act (CRA),“ 2024. [Võrgumaterjal]. Saadaval: https://www.european-cyber-resilience-act.com/.

[3] Riscure BV, „Embedded secure boot,“ [Võrgumaterjal]. Saadaval: https://www.riscure.com/embedded-secure-boot/.

[4] STMicroelectronics, „Secure boot & secure firmware update software expansion for STM32Cube,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/embedded-software/x-cube-sbsfu.html.

[5] STMicroelectronics, „STM32 Nucleo-144 development board with STM32L552ZE MCU, SMPS, supports Arduino, ST Zio and morpho connectivity,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/evaluation-tools/nucleo-l552ze-q.html.

[6] STMicroelectronics, „STM32L5 Series,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/microcontrollers-microprocessors/stm32l5-series.html.

[7] STMicroelectronics, „STM32L5x2,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/microcontrollers-microprocessors/stm32l5x2.html.

[8] STMicroelectronics, „Secure element expansion board based on STSAFE-A110,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/ecosystems/x-nucleo-safea1.html.

[9] STMicroelectronics, „Authentication, state-of-the-art security for peripherals and IoT devices,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/secure-mcus/stsafe-a110.html.

[10] LastMinuteEngineers.com, „Interface an I2C LCD with Arduino,“ [Võrgumaterjal]. Saadaval: https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/.

[11] Analog Devices, „Back to Basics: Secure Hash Algorithms,“ 13.06.2019. [Võrgumaterjal]. Saadaval: https://www.analog.com/en/resources/technical-articles/back-to-basics-secure-hash-algorithms.html.

[12] Analog Devices, „Implementing Secure Authentication Without Being a Cryptography Expert,“ [Võrgumaterjal]. Saadaval: https://www.analog.com/en/resources/design-notes/implementing-secure-authentication-without-being-a-cryptography-expert.html.

[13] Wikipedia, „Digital signature,“ [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/Digital_signature.

[14] Analog Devices, „Secure the IoT: Part 2, A Secure Boot, the "Root of Trust" for Embedded Devices,“ [Võrgumaterjal]. Saadaval: https://www.analog.com/en/resources/app-notes/secure-the-iot-part-2-a-secure-boot-the-root-of-trust-for-embedded-devices.html.

[15] J. Chou, „Solving Chip Security's Weakest Link,“ [Võrgumaterjal]. Saadaval: https://www.design-reuse.com/articles/51232/solving-chip-security-s-weakest-link.html.

[16] Arm, „TrustZone for Cortex-M,“ [Võrgumaterjal]. Saadaval: https://www.arm.com/technologies/trustzone-for-cortex-m.

[17] Arm Community, pilt foorumipostitusest, [Võrgumaterjal]. Saadaval: https://community.arm.com/cfs-file/__key/communityserver-discussions-components-files/468/6886.2-worlds2.png.

[18] Arm, „Trusted Firmware-M Introduction,“ [Võrgumaterjal]. Saadaval: https://trustedfirmware-m.readthedocs.io/en/latest/introduction/index.html.

[19] STMicroelectronics, „Overview of Secure Boot and Secure Firmware Update solution on Arm® TrustZone® STM32 microcontrollers (AN5447, Rev 3),“ 16.08.2021. [Võrgumaterjal]. Saadaval: https://www.st.com/resource/en/application_note/an5447-overview-of-secure-boot-and-secure-firmware-update-solution-on-arm-trustzone-stm32-microcontrollers-stmicroelectronics.pdf.

[20] Arm, „BL1 Immutable bootloader,“ [Võrgumaterjal]. Saadaval: https://trustedfirmware-m.readthedocs.io/en/latest/design_docs/booting/bl1.html.

[21] Trusted Firmware, „Trusted Firmware-M Secure boot,“ [Võrgumaterjal]. Saadaval: https://trustedfirmware-m.readthedocs.io/en/latest/design_docs/booting/tfm_secure_boot.html.

[22] Trusted Firmware, „MCUboot,“ [Võrgumaterjal]. Saadaval: https://www.trustedfirmware.org/projects/mcuboot/.

[23] Wikipedia, „YMODEM,“ [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/YMODEM.

[24] Wikipedia, „Advanced Encryption Standard,“ [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[25] Wikipedia, „Block cipher mode of operation - Counter (CTR),“ [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Counter_(CTR).

[26] Wikipedia, „Optimal asymmetric encryption padding,“ [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding.

[27] STMicroelectronics, „Integrated Development Environment for STM32,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/development-tools/stm32cubeide.html.

[28] STMicroelectronics, „STM32CubeProgrammer software for all STM32,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/development-tools/stm32cubeprog.html.

[29] TeraTerm Project, „Tera Term Home Page,“ [Võrgumaterjal]. Saadaval: https://teratermproject.github.io/index-en.html.

[30] STMicroelectronics, „STSAFE-A110 Datasheet (DS13039, Rev 1),“ 19.12.2019. [Võrgumaterjal]. Saadaval: https://www.st.com/resource/en/datasheet/stsafe-a110.pdf.

[31] STMicroelectronics, „STSAFE-A110 SPL03 generic sample profile description (AN5762, Rev 1),“ 07.02.2022. [Võrgumaterjal]. Saadaval: https://www.st.com/resource/en/application_note/an5762-stsafea110-spl03-generic-sample-profile-description-stmicroelectronics.pdf.

[32] STMicroelectronics, „STSAFE-A complete set of API built on STM32Cube,“ [Võrgumaterjal]. Saadaval: https://www.st.com/en/embedded-software/stsw-safea1-mw.html.

[33] J. Lee, „HD44780 I2C Library for STM32,“ [Võrgumaterjal]. Saadaval: https://github.com/eziya/STM32_HAL_I2C_HD44780/tree/master.

[34] STMicroelectronics, „STM32L5 MEMPROTECT,“ [Võrgumaterjal]. Saadaval: https://www.st.com/content/ccc/resource/training/technical/product_training/group1/72/e7/22/68/9d/e3/43/f7/STM32L5-Security-Memories_Protections_MEMPROTECT/files/STM32L5-Security-Memories_Protections_MEMPROTECT.pdf/_jcr_content/translations/en.STM32L5-Security-Memories_Protections_MEMPROTECT.pdf.

[35] Kraken Security Labs, „Kraken Identifies Critical Flaw in Trezor Hardware Wallets,“ 31.01.2020. [Võrgumaterjal]. Saadaval: https://blog.kraken.com/product/security/kraken-identifies-critical-flaw-in-trezor-hardware-wallets.

[36] NewAE Technology, „ChipWhisperer,“ [Võrgumaterjal]. Saadaval: https://www.newae.com/chipwhisperer.

[37] STMicroelectronics, „STM32L5 Series safety manual (UM2752, Rev 2),“ 13.01.2021. [Võrgumaterjal]. Saadaval: https://www.st.com/resource/en/user_manual/um2752-stm32l5-series-safety-manual-stmicroelectronics.pdf.

[38] X. M. Saß, R. Mitev ja A.-R. Sadeghi, „Oops..! I Glitched It Again! How to Multi-Glitch the Glitching-Protections on ARM TrustZone-M,“ 01.03.2023. [Võrgumaterjal]. Saadaval: https://arxiv.org/pdf/2302.06932.

[39] STMicroelectronics, „STM32L5 - Flash,“ [Võrgumaterjal]. Saadaval: https://www.st.com/content/ccc/resource/training/technical/product_training/group0/5c/77/af/1c/86/07/44/ce/STM32L5-Memory-Flash_FLASH/files/STM32L5-Memory-Flash_FLASH.pdf/_jcr_content/translations/en.STM32L5-Memory-Flash_FLASH.pdf.

[40] STMicroelectronics, „Introduction to memory protection unit management on STM32 MCUs (AN4838, Rev 8),“ 04.04.2024. [Võrgumaterjal]. Saadaval: https://www.st.com/resource/en/application_note/an4838-introduction-to-memory-protection-unit-management-on-stm32-mcus-stmicroelectronics.pdf.

[41] Arm, „Return-oriented programming,“ [Võrgumaterjal]. Saadaval: https://developer.arm.com/documentation/102433/0200/Return-oriented-programming.

[42] STMicroelectronics, „STM32L5 - HASH,“ [Võrgumaterjal]. Saadaval: https://www.st.com/content/ccc/resource/training/technical/product_training/group1/f1/27/db/3d/cc/3d/49/8e/STM32L5-Security-Hash_processor_HASH/files/STM32L5-Security-Hash_processor_HASH.pdf/_jcr_content/translations/en.STM32L5-Security-Hash_processor_HASH.pdf.

[43] STMicroelectronics, „STM32L5 - AES,“ [Võrgumaterjal]. Saadaval: https://www.st.com/content/ccc/resource/training/technical/product_training/group1/69/a8/5f/41/37/af/49/74/STM32L5-Security-Advanced_Encryption_Standard_HW_Accelerator_AES/files/STM32L5-Security-

Advanced_Encryption_Standard_HW_Accelerator_AES.pdf/_jcr_content/translations/en.STM32L5-Security-Advanced_Encryption_Standard_HW_Accelerator_AES.pdf.

[44] STMicroelectronics, „STM32L5 - PKA,“ [Võrgumaterjal]. Saadaval: https://www.st.com/content/ccc/resource/training/technical/product_training/group1/b8/bc/a1/3e/75/66/46/00/STM32L5-Security-PublicKeyAccelerator_HW_Accelerator_PKA/files/STM32L5-Security-PublicKeyAccelerator_HW_Accelerator_PKA.pdf/_jcr_content/translations/en.STM32L5-Security-PublicKeyAccelerator_HW_Accelerator_PKA.pdf.

[45] Rambus, „Side-channel attacks explained: everything you need to know,“ 14.10.2021. [Võrgumaterjal]. Saadaval: https://www.rambus.com/blogs/side-channel-attacks/.

[46] Wikipedia, „Timing attack,“ [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/Timing_attack.

[47] M. Wienecke, „Cache based Timing Attacks on Embedded Systems,“ 20.07.2009. [Võrgumaterjal]. Saadaval: https://informatik.rub.de/wp-content/uploads/2021/11/ms_wienecke.pdf.

[48] A. Murari, „A Timing Attack on Software Implementations of RSA,“ 2003. [Võrgumaterjal]. Saadaval: https://ir.library.oregonstate.edu/downloads/xd07h207n.

[49] TrustedFirmware, „Mbed TLS,“ [Võrgumaterjal]. Saadaval: https://github.com/Mbed-TLS/mbedtls.

[50] TrustedFirmware, „Mbed TLS turvapoliitika,“ [Võrgumaterjal]. Saadaval: https://github.com/Mbed-TLS/mbedtls/security/policy.

[51] Wikipedia, „Differential fault analysis,“ [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/Differential_fault_analysis.

[52] Entrust, „What Is A Hardware Security Module (HSM)?,“ [Võrgumaterjal]. Saadaval: https://www.entrust.com/resources/learn/what-are-hardware-security-modules.

[53] BBC, „iPhone hacker publishes secret Sony PlayStation 3 key,“ 06.01.2011. [Võrgumaterjal]. Saadaval: https://www.bbc.com/news/technology-12116051.

[54] Synopsys, „Coverity Static Analysis,“ [Võrgumaterjal]. Saadaval: https://www.synopsys.com/software-integrity/static-analysis-tools-sast/coverity.html.

[55] STMicroelectronics, „NUCLEO-L552ZE-Q SBSFU readme,“ [Võrgumaterjal]. Saadaval: https://github.com/STMicroelectronics/STM32CubeL5/blob/master/Projects/NUCLEO-L552ZE-Q/Applications/SBSFU/readme.txt.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Karl Udu

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „STM32 turvalise käivitamise süsteemi integreerimine ja arendus“, mille juhendaja on Terry London
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

16.05.2024

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.