KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Karolis Šlipaitis

KELIŲ FAKTORIŲ AUTENTIFIKAVIMO MIKROVALDIKLIU ALGORITMO SUDARYMAS

Baigiamasis magistro darbas

Vadovas
Lekt. dr. Dangis Rimkus

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Kelių faktorių autentifikavimo mikrovaldikliu algoritmo sudarymas

Baigiamasis magistro darbas
Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas
(parašas) Lekt. dr. Dangis Rimkus
(data)
Recenzentas
(parašas) Doc. dr. Jevgenijus Toldinas
(data)

Projektą atliko
(parašas) Karolis Šlipaitis
(data)

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

(Fakultetas)

KAROLIS ŠLIPAITIS

(Studento vardas, pavardė)

INFORMACIJOS IR INFORMACINIŲ TECHNOLOGIJŲ SAUGA, 621E10003

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Kelių faktorių autentifikavimo mikrovaldikliu algoritmo sudarymas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 17 m. gegužės 21 d.

Kaunas

Patvirtinu, kad mano, **Karolio Šlipaičio,** baigiamasis projektas tema „Kelių faktorių autentifikavimo mikrovaldikliu algoritmo sudarymas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Šlipaitis, Karolis. „Kelių faktorių autentifikavimo mikrovaldikliu algoritmo sudarymas“. Magistro baigiamasis projektas / vadovas lekt. dr. Dangis Rimkus; Kauno technologijos universitetas, Informatikos fakultetas, Kompiuterių katedra.

Kaunas, 2017. 69 psl.

SANTRAUKA

Sparčiai vystantis technologijoms, vis daugiau informacijos perkeliama į elektroninę erdvę. Kartu su technologijomis tobulėja ir kibernetiniai nusikaltėliai, kurie nori pasisavinti net ir privačią informaciją, kuri priklauso eiliniams vartotojams. Eiliniai vartotojai nespėja įsisavinti technologijų ir dėl to tampa lengvu taikiniu piktavaliams. Slaptažodžiais paremta autentifikacija savo kompleksiškumu vis labiau slegia eilinius vartotojus. Siekiant palengvinti jų naštą, pradėta naudoti kelių žingsnių autentifikacija. Tačiau ar ji tikrai yra saugesnė ir patogesnė eiliniam vartotojui?..

Siekiant tai išsiaiškinti, darbe išanalizuoti kelių pakopų autentifikavimosi būdai, jų metodai ir taikymo galimybes, bei „FIDO aljanso“ siūlomi standartai ir sprendimai. Išsiaiškinta, jog nėra vieno saugiausio ir tinkamiausio metodo visiems atvejams – kiekvienas turi savo privalumų ir trūkumų. Taip pat, dauguma autentifikacijos metodų reikalauja skirtingų prietaisų, norint jais autentifikuotis. Tuo remiantis, nuspręsta sukurti universalų autentifikacijos mechanizmą, kuris leistų autentifikuotis keliais skirtingais metodais ir turėtų plačias pritaikymo galimybes. Suprojektuota keletas teorinių autentifikacijos modelių, kurie būtų tinkami apjungimui viename fiziniame įrenginyje. Realizacijai pasirinkta „Arduino“ platforma, kurią galima programuoti ir kuri leidžia prijungti įvairius papildomus fizinius komponentus, bei yra pritaikyta pradedantiesiems ir turi didelę bendruomenę. Ši platforma nepasižymi dideliais skaičiavimo ištekliais, tačiau jų užtenka paprastų užduočių atlikimui. Su „Arduino“ sėkmingai pavyko realizuoti modelių prototipus ir juos apjungti į vieną universalų mechanizmą, kuris leidžia rinktis iš kelių realizuotų autentifikacijos būdų. Ištyrus sukurtų prototipų greitaveiką paaiškėjo, jog jie geba autentifikuoti nuo 1 iki 100 kartų (priklausomai nuo pasirinkto autentifikavimo modelio) per 1 sekundę ir kad kai kurių modelių sklandžiam veikimui reikalingas priverstinis užvėlinimas programiniame kode, kuris daro didelę įtaką greitaveikos mažėjimui.

Šlipaitis, K. *Multi-factor authentication using microcontroller algorithm development*: Master's thesis / supervisor lect. dr. Dangis Rimkus. The Faculty of Informatics, Kaunas University of Technology.

Kaunas, 2017. 69 p.

SUMMARY

While technologies rapidly evolve, more and more information is being transferred to the cyberspace. Cyber criminals evolve together with the technologies and they want to steal all the private data, which belongs to the average users. Average users can't keep up with the technologies and so they become easy targets for the cyber criminals. Average users are being clogged more and more by the password-based authentication. To address this problem, multi-step authentication was introduced. But is it really more secure and easier to use for the average user?

To find that out, multi-step authentication ways, techniques and applicability are being analyzed in this work, including standards and solutions offered by the "Fido Alliance". It was clarified that no method alone is good for all-round solution and every each of them has its benefits and drawbacks. Also, there's a need for a lot of various devices to be able to authenticate by certain methods. On this basis, it was decided to create universal authenticate mechanism, which could allow to authenticate by several different methods and would have a wide variety of use possibilities. Designed several theoretical authentication models, which could be appropriate for joining in one physical device. "Arduino" platform was chosen for implementation, because it is programmable and it allows to connect many other physical components to it as is designed for beginners and has a huge community. This platform lacks computing power, however it's enough for basic tasks. "Arduino" allowed successfully implement designed models prototypes and unite them to form one universal mechanism, which allows to choose of many implemented authentication methods. After the research of prototypes, it was clear that they can authenticate from 1 to 100 times (depends on the chosen authentication model) each second and that for smooth working of some models it is required to add the artificial delay in the code, which has a significant impact on throughput decrease.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

FIDO (angl. „Fast Identity Online“) – aljanso pavadinimo akronimas.

PIN (angl. „Personal Identification number“) – ne mažiau keturių skaitmenų slaptažodis.

SIM (angl. „Subscriber Identity Module“) – maža lustinė kortelė naudojama mobiliuosiuose telefonuose saugoti informacijai apie tinklą, telefono numerį ir kt.

USB (angl. „Universal Serial Bus“) – kompiuteriuose naudojama universalioji jungtis. Prie jos galima prijungti atmintines ir kitus įrenginius.

SMS (angl. „Short Message Service“) – trumpoji sms žinutė.

„Žvejybos“ (angl. „Phishing“) ataka – elektroninės atakos forma, kai auką bandoma apgauti siunčiant klaidingą informaciją el. paštu arba SMS žinute. Tuo siekiama išvilioti iš aukos tam tikrą informaciją ar priversti jį atlikti tam tikrus veiksmus.

„Pasiklausymo“ (angl. „Man-in-the-middle“) ataka – elektroninės atakos forma, kai piktavalis slapta įsiterpia į komunikaciją tarp vartotojo ir serverio, su kuriuo pastarasis bendrauja. Tokiu būdu piktavalis gali lengvai suklastoti perduodamą informaciją abejomis kryptimis.

„Grubios jėgos“ (angl. „Brute force“) ataka – elektroninės atakos forma, kai piktavalis mėgina atspėti prisijungimo informaciją, bandydamas įvairias simbolių kombinacijas.

U2F (angl. „Universal Second Factor“) – FIDO aljanso specifikacija, kuri leidžia sustiprinti slaptažodžių autentifikaciją.

UAF (angl. „Universal Authentication Framework“) – FIDO aljanso specifikacija, kuri leidžia atsisakyti slaptažodžių naudojimo autentifikacijai.

„Bluetooth“ – bevielio ryšio technologija, leidžianti apsikeisti informacija per trumpą atstumą.

NFC (angl. „Near Field Communication“) – komunikacijos protokolų aibė, kuri leidžia dviem elektroniniam įrenginiam užmegzti tarpusavio ryšį iki 4 cm atstumu. „NFC“ plačiai naudojama bekontaktėse mokėjimų sistemose, kaip ir paprastam duomenų apsikeitimui tarp elektroninių įrenginių.

AES (angl. „Advanced Encrypted Standard“) – blokinis simetrinis (viešojo rakto) šifravimo algoritmas. Dar kitaip žinomas kaip Rijndael algoritmas.

ECB (angl. „Electronic CodeBook“) – šifravimo būdas, kai šifruojamas pranešimas išskaidomas į daug vienodo dydžio blokų ir kiekvienas blokas užšifruojamas atskirai, bet tuo pačiu slaptu raktu.

CBC (angl. „Cipher Block Chaining“) – šifravimo būdas, kai šifruojamas pranešimas išskaidomas į daug vienodo dydžio blokų ir kiekvienas blokas užšifruojamas atskirai. Pirmasis blokas šifruojamas pasinaudojant pradiniu vektoriumi, o sekantys – užšifruotu pirmuoju bloku.

XOR – loginė operacija, kuria lyginami dviejų informacijos bitų reikšmės (0 arba 1). Jei reikšmės nesutampa, operacijos rezultatas lygus 1. Jei sutampa – 0.

ĮVADAS

Gyvename pasaulyje, kuriame technologijos tobulėja kiekvieną dieną. Kasdieniam naudojimui sukuriama vis daugiau įvairių informacinių sistemų ir elektroninių paslaugų. Šios vis glaudžiau bendrauja tarpusavyje interneto pagalba, tad vis daugiau informacijos perkeliama į elektroninę erdvę. Tačiau ne visa sukaupta informacija yra skirta viešai prieigai. Kaip ji apsaugoma nuo paviešinimo? Kokie apsaugos mechanizmai naudojami? Kiek pats vartotojas turi būti atsakingas už savo informacijos saugą?

Norint suvaldyti vis didėjantį technologijų srautą, reikalingas ir vis didesnis žinių bagažas. Siekiama suteikti eiliniams vartotojams galimybę naudotis technologijomis, nereikalaujant šių technologijų veikimo išmanymo. Lengvas ir patogus sistemos naudojimas užtikrina didesnį sistemos patrauklumą vartotojui. Ne išimtis ir saugumas – prieigos kontrolė.

Piktavaliai nuolat kėsinasi į sistemas, o su technologijų evoliucija jiems tampa prieinama vis daugiau metodų ir priemonių lengvesniam įsilaužimui. Dėl to ir sistemų kūrėjai turi neatsilikti ir kurti sudėtingesnes apsaugas arba jų reikalauti iš vartotojų. Laikui bėgant, reikalavimai slaptažodžiams darėsi vis griežtesni – ilgėjo minimalus reikalaujamas simbolių kiekis, pradėta prašyti specialų ir skaitinių simbolių slaptažodžiuose. Tai vis labiau apsunkino eilinių vartotojų darbą, nes taip buvo reikalauja iš jų atsiminti vis sudėtingesnius slaptažodžius. Be sustojimo plečiantis įvairiausių sistemų ratui, vartotojui darėsi vis sudėtingiau laikytis saugumo reikalavimų. Vis dažniau buvo naudojami tie patys slaptažodžiai keliose sistemose, mažėjo jų sudėtingumas ir dėl to – bendras sistemų saugumas. Nors bendrųjų saugumo praktikų nesilaikymas yra vartotojo kaltė ir jie labiausiai nuo to nukenčia, tačiau ir pačių sistemų kūrėjai turėtų būti atsakingi už saugomą vartotojų informaciją ir padėti vartotojams apsisaugoti. Dėl to buvo pradėta ieškoti būdų, kaip sustiprinti slaptažodžių naudojimą arba juos pakeisti kitais, didesnį saugumą užtikrinančiais, autentifikavimo metodais.

Iš pradžių lėtai, bet užtikrintai, augęs kelių faktorių autentifikavimo metodas, pastaruoju metu ypač išpopuliarėjo. Kelių faktorių autentifikavimo pagalba užtikrinamas didesnis vartotojo saugumas, neapsunkinant jo sudėtingu autentifikavimo mechanizmo naudojimu. Tai pasiekiama išskaidant sudėtingą autentifikavimo mechanizmą į kelias mažesnes dalis, kurios suteikia vartotojui daugiau erdvės ir reikalauja mažiau atsakomybės iš jo (pvz., atsiminti ilgą ir sudėtingą slaptažodį). Taip plečiamas saugumas ir horizontalia kryptimi, o ne tik apsiribojant vertikalia kryptimi. Tačiau, daugėjant žingsnių autentifikacijos procese, daugėja ir reikalingos specialios įrangos kiekis, norint turėti galimybę autentifikuotis skirtingais metodais. Taip pat, naudojama speciali įranga apribojama gamintojo konkrečiam panaudojimo atvejui ir nebetinka niekam kitam (pvz., kriptografinis USB raktas gali būti naudojamas tik jį prijungus prie kompiuterio, taip nebetenkant galimybės juo pasinaudoti per „WiFi“, „BT“ ar kitas bevielio ryšio technologijas).

Šiuo metu, įprastai apsiribojama dviejų žingsnių autentifikacija ir dar ne visur ji yra prieinama. Tačiau, žvelgiant į ateitį, matoma problema, jog ateityje, norint autentifikuotis įvairiais būdais, bus reikalingas didelis skirtingų prietaisų kiekis. Autentifikavimo prietaisų gamintojai toliau susikoncentravę į vieno autentifikavimo būdo autentifikacijos prietaisų gaminimą ir, kol kas, nematyti jokio judėjimo link universalesnių priemonių gaminimo. Dėl to, šiuo darbu bus bandoma žengti žingsnį į priekį ir sukurti universalų autentifikacijos mechanizmą viename fiziniame įrenginyje, kuris leistų autentifikuotis keliais skirtingais metodais, bei būtų pritaikomas įvairiam naudojimui (pvz. tiek prijungus laidu prie kompiuterio, tiek ir bevieliu ryšiu).

Darbo tikslas – sukurti universalų autentifikavimo mechanizmą ir ištirti jo veikimą.

Norint pasiekti tikslą, būtina įgyvendinti šiuos **uždavinius**:

1. išanalizuoti kelių pakopų autentifikavimo metodus ir jų taikymo galimybes;
2. sukurti keletą autentifikavimo modelių ir suprojektuoti jų prototipus;
3. išanalizuoti ir išsirinkti tinkamiausią priemonę prototipų realizacijai;
4. sukurti kelis suprojektuotus prototipus ir juos apjungti viename autentifikavimo mechanizmo prototipe, kuris leistų autentifikuotis keliais skirtingais modeliais;
5. ištirti ir įvertinti sukurtų prototipų greitaveikos galimybes, bei tobulintinas vietas.

Darbo rezultatas bus atitikmens neturintis ir universalus autentifikavimo mechanizmas, kuris leis sujungti įvairius, autentifikavimui skirtus, įrenginius į vieną ir suteiks galimybę autentifikuotis bet kuriuo pritaikytu metodu. Taip bus palengvinama vartotojo autentifikacija, nereikalaujant turėti kelių skirtingų įrenginių. Taip pat, darbe bus pateikta informacija, kaip eilinis vartotojas, turintis tik programavimo pagrindus, gali susikurti ir keisti autentifikacijai naudojamus autentifikavimo metodus, pagal savo poreikius ir norimą pasiekti saugumo lygį. Darbe bus apsiribojama tik autentifikavimo mechanizmo sukūrimu iš vartotojo (kliento) pusės ir darbo rezultatas neapims autentifikacijos metodo integracijos su jau egzistuojančiomis paslaugų sistemomis.

Šį dokumentą sudaro 4 pagrindiniai skyriai (analizės, projekto, prototipo ir tyrimo dalys) su išvadomis ir literatūros sąrašu pabaigoje. Analizės skyriuje pateikiama išsami informacija apie šiuolaikinius autentifikacijos metodus ir FIDO aljanso standartus. Projekto skyriuje - suprojektuoti autentifikavimo modeliai, iš kurių keli realizuoti prototipo skyriuje. Tyrimo skyriuje ištirta sukurto prototipo greitaveika. Išvadose pateikiama informacija apie padarytas išvadas atlikus darbą. Literatūros sąraše pateikiamas panaudotų informacijos šaltinių sąrašas, iš kurių 13 yra moksliniai šaltiniai ir 1 patentas.

1. KELIŲ FAKTORIŲ AUTENTIFIKAVIMO ANALIZĖ

1.1. Analizės tikslas

Atliekant analizę bus siekiama susipažinti su kelių pakopų autentifikavimo technologija, autentifikavimo būdais, metodais, jų veikimu ir taikymu. Taip pat bus nagrinėjamas šios technologijos paplitimas, naudingumas ir aktualumas, bei ateities perspektyvos – kur link judama. FIDO aljansas („FIDO alliance“) vieninteliai siūlo porą dviejų pakopų autentifikavimo standartų, todėl analizės dalyje bus išsamiau apžvelgiamas jų siūlomų sprendimų veikimas.

1.2. Kelių faktorių autentifikavimas

Ilgą laiko tarpą vienu iš pagrindinių autentifikavimosi būdų buvusi vien tik slaptažodžiais paremta autentifikacija iš lėto praranda sistemų kūrėjų ir vartotojų pasitikėjimą. Piktavaliams vis lengviau pavyksta įvairiomis atakomis išgauti slaptažodžius ir dažniausiai silpnoji saugumo vieta pasirodo žmogus-vartotojas, kuris naudoja ne pakankamai stiprius slaptažodžius. Tačiau nuolatos daugėjant įvairių internetinių paslaugų, vartotojui darosi sudėtinga visoms joms naudoti vienodai stiprius ir sudėtingus slaptažodžius [1]. Vartotojai linkę palengvinti savo darbą ir naudoti paprastus, lengvai įsimenamus – nesudėtingus slaptažodžius. Dėl to, vien tik slaptažodžiais grįstas autentifikavimas vis labiau tapo sistemų saugumo silpnąja vieta ir buvo pradėta galvoti apie modernesnius metodus.

Vienas iš tokių metodų tapo kelių etapų autentifikavimas. 1984 metais buvo užpatentuotas dviejų faktorių autentifikavimo metodas, kurio populiarumas tik paskutiniu metu smarkiai išaugo [2]. Kelių skirtingų faktorių autentifikavimo metodo veikimas pagrįstas tuo, jog vartotojas, norėdamas gauti prieigą, turi pateikti mažiausiai du skirtingus informacijos „vienetus“ (faktorius). Tai gali būti kažkas, ką tik vartotojas žino, ką tik vartotojas turi arba kas vartotojas yra. Galima rasti įvairių mėginimų praplėsti šių faktorių sąrašą ir kitokiais būdais, pvz., ką tik vartotojas daro [3], ką vartotojas pažįsta arba ką vartotojas apskaičiuoja [4], bei kur vartotojas yra [5].

1.2.1. Kažkas, ką vartotojas žino (angl. „Something You Know“)

Šis autentifikavimo būdas naudojamas jau seniai, nes tai yra tie patys slaptažodžiai, taip pat ir PIN kodai, kurie plačiai naudojami ne tik internetinėms paslaugoms, bet ir autentifikacijai su SIM ar banko kortelėmis. Slapta informacija, skirta autentifikacijai, turi būti įsimenama vartotojo ir jos saugumas pagrįstas simbolių sekos sudėtingumu. Sudėtingumas priklauso nuo to, kokio ilgio yra seka (ar 4, ar 20 simbolių), kokie simboliai ir koks jų įvairumas naudojamas (ar tai yra tik skaičiai, ar ir didžiosios, mažosios raidės, bei specialūs simboliai), bei kiek prasminga (aiškus žodis ar neaiškių simbolių „kratinys“) ir kiek šią simbolių sekos reikšmę galima susieti su vartotoju (vartotojo vardas ar gimimo metai, sukeista tvarka), kuris ją naudoja autentifikacijai.

Sudėtingus ir ilgus slaptažodžius vartotojai linkę greitai pamiršti, todėl net ir stengiantis laikytis gerųjų saugumo praktikų, vis sugrįžtama prie trumpų ir nesudėtingų (blogiausiu atveju – pasikartojančių) slaptažodžių naudojimo visur, kurie neužtikrina pakankamo saugumo lygio, norint efektyviai apsisaugoti nuo piktavalių.

2001 metais „CentralNic“ atlikta apklausa, kurios metus buvo apklausta 1200 ofisuose dirbančių britų, atskleidė, jog beveik 50 % darbuotojų slaptažodžiams naudoja savo, savo augintinio arba šeimos nario vardą [6]. Taip pat, ne maža dalis naudoja įžymybės arba filmų personažo vardą.

Kažkas, ką vartotojas žino būdas ypač populiarus išmaniuosiuose telefonuose, užsklandos atrakinimo mechanizme. Tačiau vien tik jo jau neužtenka, siekiant aukšto saugumo lygio, ir būtina imtis papildomų apsaugos priemonių, norint sustiprinti prieigos prie mobiliojo telefono saugumą [6].

1.2.2. Kažkas, ką vartotojas turi (angl. „Something You Have“)

Šis būdas taip pat naudojamas jau seniai realiame gyvenime – tai spynos ir rakto principas. Autentifikacijai panaudojamas tam tikras nešiojamas daiktas ar prietaisas, pvz. USB atmintinė ar banko kortelė. Kriptografinius raktus naudojanti USB atmintinė suteikia aukštą saugumo lygį, nes kriptografiniai raktai paremti sudėtingais kriptografiniais algoritmais. Naudojantis šiuo autentifikavimosi būdu, vartotojui nereikia nieko papildomai žinoti.

Šio būdo pagrindinis trūkumas tas, jog vartotojas turi nešiotis autentifikavimui skirtą daiktą ir jį pametęs arba jam sulūžus, nebegali prisijungti, kol nepakeičia jo nauju [8]. Taip pat, daiktą, skirtas autentifikacijai, gali pavogti piktavalis, bei bandyti jį panaudoti autentifikacijai.

Kita populiari ir daug kur sutinkama šio būdo forma – SMS žinute atsiunčiamas vienkartinis prisijungimo kodas. Tačiau SMS žinutės atsiuntimas kartais gali užtrukti, ir tokias žinutes įmanoma perimti arba blokuoti jų gavimą specialia įranga. Taip pat, gautą kodą reikia perrašyti, ir vartotojas gali suklysti, jei kodas per ilgas ar per sudėtingas.

1.2.3. Kažkas, kas vartotojas yra (angl. „Something You Are“)

Šis būdas paremtas kiekvieno žmogaus unikalia sandara (biometrine informacija) – unikalūs veido bruožai, akies tinklainė, piršto antspaudai ir kt. Nors yra tikimybė, jog dviejų skirtingų asmenų biometrinė informacija gali sutapti, tačiau tai retas atvejis, ir naudojant kelių faktorių autentifikavimo metodą, tai ne ką mažiau saugu nei stiprių slaptažodžių naudojimas.

Plačiau šį būdą apžvelgsime skyriuje apie biometrines charakteristikas.

1.2.4. Kažkas, ką vartotojas daro (angl. „Something You Do“)

Šis būdas paremtas autentifikavimu remiantis vartotojui būdingu elgesiu. Tai gali būti naudojama ne tik vienkartiniam autentifikavimui, bet ir kaip pasyvi autentifikavimo mechanizmo dalis. Stebimi vartotojo veiksmai sistemoje, prie kurios jis prisijungęs, ir, pastebėjus įtartiną vartotojo elgesį,

jam gali būti arba visiškai užblokuojama prieiga prie sistemos arba paprašoma papildomai autentifikuotis vienu iš galimų metodu.

Pirmųjų kelių prisijungimų metu renkama informacija apie vartotojui būdingą elgseną, pvz., kaip jis naudojasi sistema (kokiu greičiu naviguoja sistemoje, kur dažniausiai paspaudžia) ir iš to bandoma sudaryti tam tikrą elgesio šabloną. Jau turint suformuotą šabloną, sekančių prisijungimų metu tikrinama ar vartotojo elgesys atitinka jį ir imamasi atitinkamų veiksmų, jei pastebimi ženklūs nukrypimai nuo sukurto šablono.

Šio būdo panaudojimą galima būtų išplėsti kiekvieno vartotojo prisijungimo metu pateikiant kokią nors užduotėlę (pvz., surasti tam tikrus objektus tarp daugybės kitų objektų paveiksliuke), kurią kiekvienas vartotojas vykdytų skirtingu greičiu ir skirtingu eiliškumu, dėl kiekvieno žmogaus unikalių savybių. Nors tokioms užduotims generuoti ir apdoroti būtų reikalingi ypatingai sudėtingi algoritmai, jog jie būtų tinkami vartotojų unikalumams identifikuoti pramoniniu lygiu, galbūt ateityje jie išplis.

1.2.5. Kažkas, ką vartotojas pažįsta (angl. „Somebody You Know“)

Šis būdas remiasi vartotojo socialiniu pažinčių tinklu, kuris taikomas realiame pasaulyje nuo pat senovės [9]. Šio būdo panaudojimas elektroninėje erdvėje leidžia realizuoti tokius mechanizmus, kaip teisių delegavimas kitiems vartotojams ar pasitikėjimo hierarchiją. Pavyzdžiui, jei du vartotojai pasitiki vienas kitu ir pirmasis „supažindina“ antrąjį su trečiuoju vartotoju, antrasis pasitikėdamas pirmuoju gali pasitikėti (kad ir mažesniu laipsniu) ir trečiuoju.

Tai nėra pats tinkamiausias būdas saugiai autentifikacijai, nes lengvai pažeidžiamas „socialinės inžinerijos“ (angl. „social engineering“) atakomis, kai piktavalis, turėdamas informacijos apie pirmąjį ir trečiąjį vartotoją, gali įtikinti antrąjį, jog jis pažįsta pirmąjį ir yra trečiasis vartotojas, tokiu apgaulės būdu išsireikalaujant, jog antrasis patvirtintų jo tapatybę, kaip trečiojo vartotojo, nors jis juo nėra.

1.2.6. Kažkas, ką vartotojas apskaičiuoja (angl. „Something You Proccess“)

Šis būdas pagrindžiamas tuo, jog vartotojas autentifikavimo metu gauna pradinę informaciją ir ja pasinaudojęs, slaptu (tik jam žinomu) algoritmu apskaičiuoja rezultatą, kuriuo patvirtina savo tapatybę. Pavyzdžiui, vartotojas gauna sąrašą skaičių ir žino, jog jo algoritmas yra iš nelyginių skaičių atimti lyginius ir gautą skaičių padauginti iš savo amžiaus. Vartotojas gautą rezultatą įveda į prisijungimo langą ir taip patvirtina savo tapatybę.

Kadangi patį algoritmą vartotojas vis tiek turi atsiminti, todėl šio būdo negalime vadinti visiškai nauju faktoriumi, o tik *kažko, ką vartotojas žino* išvestiniu būdu. Visgi, šitas būdas užtikrina didesnį atsparumą prieš „žiūrėjimo per petį“ (angl. „Shoulder surfing“) atakas, nei kad *kažkas, ką vartotojas žino* būdas. Taip yra dėl to, jog jei ir kas nors mato, kokią informaciją gauna ir išveda vartotojas, nežinant specialaus algoritmo, sėkmingai atkartoti autentifikacijos nepavyktų – priešingai nei naudojant slaptažodį ar PIN kodą.

1.2.7. Kažkur, kur vartotojas yra (angl. „Somewhere You Are“)

Šis būdas dažniausiai naudojamas internetiniuose kazino, siekiant apsidrausti, jog prisijungęs vartotojas yra ten, kur internetiniai lošimai yra legalūs. Nors šis būdas pats savaime nesuteikia jokios konkrečios informacijos apie vartotojo tapatybę, tačiau gali būti panaudojamas su kitais būdais, saugumui sustiprinti. Puikus pavyzdys būtų įmonė, prie kurios vidinių sistemų leidžiama prisijungti tik vartotojams, kurie yra įmonei priklausančių patalpų geografinėse vietovėse.

Tai pakankamai geras apsaugos būdas, jei prie įvairių paslaugų bandoma autentifikuotis paprastai tik iš namų, nes piktavaliui pavykus išgauti vieną autentifikacijos sudedamąją dalį, jam vis dar reikėtų sužinoti tikslią vietą, iš kurios pavyktų sėkmingai atlikti pilną autentifikaciją. Tačiau, jei piktavalis turi surinkęs daugiau informacijos apie vartotoją ar žino, kokiose vietose lankosi vartotojas, gali suklastoti geografinę lokaciją (pakeičiant į vartotojo dažniausiai lankomą) ir sėkmingai autentifikuotis.

Kelių faktorių autentifikavimo populiarėjimą labiausiai lemia tai, jog vis daugėja atvejų, kada atskleidžiamas didelis kiekis vartotojų pavogtų slaptažodžių iš socialinių tinklų ar kitokio tipo internetines paslaugas teikiančių organizacijų. 2012 metais iš „LinkedIn“ nutekinta virš 6 milijonų vartotojų slaptažodžių, bei tais pačiai metais įsibrauta ir į „IEEE“, bei „Dropbox“ serverius [10].

1.3. Biometrinės charakteristikos

Vis labiau populiarėja ir biometrinėmis charakteristikomis paremtas vartotojų autentifikavimas, tačiau dėl vieningų standartų nebuvimo, tai vyksta palyginti lėtai. Taip pat, tik keletas biometrinių charakteristikų laikomos tinkamomis pramoniniam naudojimui [11]. Didžiausias biometrinio autentifikavimo (*kažkas, kas esi*) būdo privalumas, jog nereikia nešiotis jokio specialaus asmeninio prietaiso ar atsiminti jokios slaptos informacijos. Autentifikacijai su biometrinėmis charakteristikomis tereikia, jog autentifikavimosi vietoje būtų norimos panaudoti biometrinės charakteristikos skaitytuvas. Biometrine charakteristika gali būti paprasčiausias piršto antspaudas arba veido forma ar net akies rainelė. Augant šio autentifikavimo metodo populiarumui, vis daugiau išmaniųjų įrenginių (telefonų, kompiuterių) gamintojų diegia biometrinių charakteristikų skaitytuvus savo gaminamuose įrenginiuose. Dėl mažos kainos ir paprasto naudojimo, labiausiai paplitęs piršto antspaudo skaitytuvas.

Biometrinės autentifikacijos mechanizmas susideda iš 5 komponentų: skaitytuvo, pagrindinių savybių atskyrimo, šablonų duomenų bazės, palyginimo ir sprendimo priėmimo modulių. Skaitytuvas nuskaito asmens biometrinę informaciją ir perduoda ją savybių atskyrimo moduliui, kuris atrenka tik reikiamą informaciją unikalios (pvz., piršto antspaudų linijų pokyčius) biometrinės charakteristikos atpažinimui – sukuria asmens biometrinės charakteristikos šabloną, kuris išsaugomas duomenų bazėje, vėlesniam vartotojo atpažinimui. Sekantį kartą vartotojui pasinaudojus skaitytuvu, sukuriamas naujas

šablonas ir, palyginimo modulio pagalba, palyginimas su saugomu duomenų bazėje. Palyginimo modulis grąžina šablonų atitikimo įvertį ir pagal šį įvertį, bei nustatytą reikalaujamą tikslumo lygį, sprendimo priėmimo modulis grąžina atsakymą ar šablonai sutampa.

Nors šis autentifikavimo metodas laikomas patikimesniu, nei įprastos autentifikavimo priemonės, tačiau jis taip pat gali būti pažeidžiamas keletu būdu. Pavyzdžiui, vartotojo biometrinės charakteristikos šablono pakeitimas paslaugos duomenų bazėje arba fizinis šablono „pavogimas“, panaudojant papildomas technines priemones nuskaitant perduodamą informaciją ir ją atkartojant. Nors teigiama, jog biometrinių charakteristikų šablonų sukūrimui ir saugojimui naudojami neiššifruojami vienkrypčiai šifravimo algoritmai, daugėja šaltinių, teigiančių priešingai [12].

Ranka pasirašytas parašas vis dar naudojamas kaip autentifikavimo priemonė, tačiau jis nėra itin tinkama autentifikavimo priemonė elektroninėje erdvėje, dėl galimai lengvo falsifikavimo, todėl detaliau jo neaptarsime.

Kiekvienas biometrinės charakteristikos išgavimo metodas turi savo privalumų ir trūkumų – nėra vieno gero metodo, kuris tiktų bet kurioje situacijoje [13]. Todėl, norint išsirinkti tinkamiausią, būtina žinoti apie kiekvieną detaliau.

1.3.1. Veido skenavimas

Paremtas kiekvieno žmogaus veido bruožų santykiniu unikalumu. Leidžia atlikti autentifikavimą nuotoliniu būdų ir didelėje žmonių minioje, nereikalaujant kontakto, tačiau galimas mažas tikslumas esant blogam apšvietimui arba veido pasukimui. Skaitytuvai kainuoja santykinai brangiai ir šablonų saugojimui reikia nemažų išteklių, nes jie užima daug atminties. Gimininginiai vartotojai (sesės, broliai, pusbroliai ir kt.) gali turėti daug bendrų ar net visiškai identiškų veido bruožų, kas, šiuo atveju, gerokai sumažina šio metodo patikimumą. Įvairūs veido sužalojimai ar pakitimai dėl senėjimo proceso stipriai įtakoja šios biometrinės charakteristikos patikimumą šablono ilgaamžiškumui.

1.3.2. Akies tinklainės skenavimas

Laikomas vienu iš didžiausią saugumą ir patikimumą užtikrinančiu metodu. Akies tinklainė mažai keičiasi per gyvenimą ir kiekvienas žmogus turi unikalią tinklainę. Atpažinimas atliekamas greitai ir tiksliai. Gali būti atliekamas nuo 10 centimetrų iki kelių metrų atstumu, bet didėjant atstumui mažėja tikslumas. Didžiausia problema, jog gali būti lengvai apgaunamas panaudojus aukštos kokybės paveiksliukus ir įrenginys yra ganėtinai brangus.

1.3.3. Piršto antspaudų skenavimas

Paremtas piršto antspaudo iškilimų ir griovelių nuskaitymu, kurie yra skirtingi kiekvienam asmeniui. Šis autentifikavimo būdas labiausiai paplitęs, nes jį lengva naudoti, jis yra greitas ir pasižymi žema įrangos kaina. Nors šią biometrinę charakteristiką labai sudėtinga suklastoti, tačiau kuriant tinkamą antspaudo šabloną gali kilti įvairių problemų. Tokiomis problemomis gali būti tiek

nešvarumai ant skaitytuvo, tiek ir ant piršto ar įvairios kitos kūno savybės (pvz. drėgni pirštai, žaizdos, cheminis poveikis pirštams). Kita vertus, piršto antspaudus gan lengva pavogti, nes asmuo palieka piršto antspaudus visur.

1.3.4. Piršto venų skenavimas

Veikimas panašus į piršto skaitytuvo, tačiau nuskaito piršte esančias venas, o ne piršto antspaudą. Net ir identiški dvyniai turi skirtingą venų išsidėstymą, todėl šis metodas garantuoja mažą klaidų skaičių. Sudėtinga nuskaityti ar pavogti, nes piršto venos yra žmogaus kūne. Taip pat šis metodas atsparesnis piršto išoriniams pakitimams, bet brangus, todėl naudojamas rečiau. Nuskaitymo prietaisas didesnis, nei piršto antspaudo skaitytuvas ir nėra skirtas pramoniniam naudojimui, todėl mažai paplitęs.

1.3.5. Balso atpažinimas

Autentifikavimo proceso metu garsas verčiamas į elektrinius signalus ir juos apdorojus sugeneruojamas šablonas. Šis metodas nereikalauja jokio pasiruošimo ir lengvai naudojamas turint kokią nors negalią. Labai jautrus trukdžiams (foniniams garsams), galima apgauti panaudojus įrašytą balsą, bei labai brangus. Balso atpažinimo metodu gali nebepavykti autentifikuotis, jei vartotojas užkimsta ar kitaip pakinta jo balsas dėl ligos ar kitų priežasčių.

1.3.6. Lūpų skenavimas

Pastaruoju metu labiausiai populiarėjantis metodas. Veikimas pagrįstas lūpų formos ir spalvos nuskaitymu, todėl šio metodo tobulinimas turi didelę įtaką ir veido skenavimo metodo tobulėjimui. Šis metodas pasižymi mažu sukuriamo šablono dydžiu, bei galimybe jį naudoti papildomai prie kitų metodų. Taip pat nereikalauja jokio kontakto autentifikavimui.

1 lentelė Biometrinių charakteristikų palyginimas

Biometrinė charakteristika	**Tikslumas**	**Kaina**	**Šablono dydis**	**Atsparumas pokyčiams**	**Saugumas**	**Taškai**
Veido atpažinimas	Mažas(1)	Didelė(1)	Didelis(1)	Mažas(1)	Mažas(1)	**5**
Akies tinklainė	Didelis(3)	Didelė(1)	Mažas(3)	Vidutinis(2)	Vidutinis(2)	**11**
Piršto antspaudas	Vidutinis(2)	Maža(3)	Mažas(3)	Mažas(1)	Mažas(1)	**10**
Piršto venos	Didelis(3)	Vidutinė(2)	Vidutinis(2)	Didelis(3)	Didelis(3)	**13**
Balso atpažinimas	Mažas(1)	Vidutinė(2)	Mažas(3)	Mažas(1)	Mažas(1)	**8**
Lūpų atpažinimas	Vidutinis(2)	Vidutinė(2)	Mažas(3)	Vidutinis(2)	Didelis(3)	**12**

1 lentelėje, rastoje [13] šaltinio tekste, galime matyti bendrą metodų vaizdą, kokius privalumus ir trūkumus kiekvienas iš jų turi. Pirmame stulpelyje pateikiamos biometrinės charakteristikos, kurias nuskaito skaitytuvas, atliekant autentifikavimą. Antrame stulpelyje pateikiamas kiekvienos charakteristikos suteikiamas tikslumas. Kuo didesnis tikslumas, tuo lengviau suklastoti biometrinę charakteristiką. Trečiame stulpelyje pateikiamas kainų santykinis įvertinimas. Ketvirtame – koks vieno sukurto šablono dydis atmintyje. Penktame stulpelyje pateikiama kiek biometrinė charakteristika išlieka aktuali keičiantis organizmui, t.y. kiek ilgai bus tinkamas sukurtas šablonas, kol reikės sukurti naują. Šeštame stulpelyje pateikiamas saugumo įvertinimas, o septintame – taškai, įvertinus visus aspektus.

1.4. FIDO aljansas

Panagrinėkime FIDO aljansą, kuris buvo įkurtas 2012 metų liepos mėnesį „PayPal“, „Lenovo“ ir dar kelių stambių įmonių [14] [15]. Šiandien jį sudaro daugiau nei 200 įmonių, tarp kurių yra „Google“, „Microsoft“, „Samsung“ ir „Intel“ [16]. Aljansas siekia atkreipti dėmesį į sąveikos trūkumą tarp saugaus autentifikavimo prietaisų ir vartotojų patiriamų problemų galvojant naujus ir mėginant atsiminti visus anksčiau sukurtus sudėtingus slaptažodžius [17].

FIDO taikosi sukurti technines specifikacijas, kurios apibrėžtų atvirą, lengvai pritaikomą ir sąveikaujančią mechanizmų aibę, kuri padėtų sumažinti priklausomybę nuo slaptažodžių, naudojamų vartotojam autentifikuoti. Taip pat jie vykdo įvairias programas, siekdami užtikrinti kuriamų specifikacijų sėkmingą pritaikymą visame pasaulyje.

1.5. FIDO standartų veikimas

1.5.1. FIDO registracija

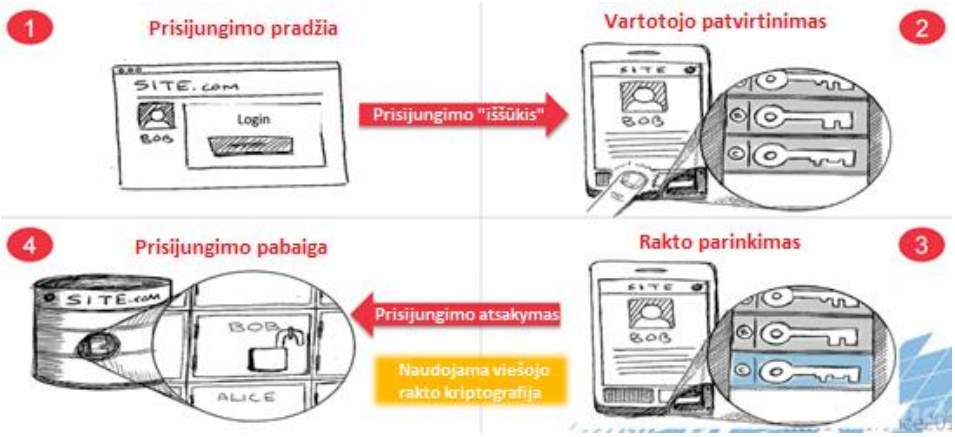
Norint prisijungti prie tam tikros paslaugos, kuri naudoja FIDO standartus, pirmiausia vartotojo paprašoma pasirinkti vieną iš paslaugos palaikomų FIDO autentifikavimo būdų. Vartotojas pasinaudoja vienu iš leidžiamu vietinės autentifikacijos būdu ir vartotojo naudojamas vietinis įrenginys (mobilusis telefonas) sukuria kriptografinių raktų porą (viešąjį ir privatųjį) ir siunčia vieną iš jų (viešąjį raktą) paslaugos serveriui. Paslaugos serveris susieja viešąjį raktą su vartotoju ir išsaugo raktą savo duomenų bazėje.

1 pav. FIDO registracijos procesas

1 pav. matyti registracijos, prie FIDO standartus palaikančios paslaugos, procesas. Vartotojui norint prisijungti prie sistemos, jis turi autentifikuotis prie savo mobilaus įrenginio, kuris po sėkmingos autentifikacijos sukuria kriptografinių raktų porą ir viešąjį raktą persiunčia paslaugos serveriui, prie kurio vartotojas mėgina prisiregistruoti.

1.5.2. FIDO prisijungimas

Vartotojui norint prisijungti prie FIDO standartus palaikančios paslaugos, paslauga atsiunčia „iššūkį“ (bet kokį tekstą), kurį vartotojas turi pasirašyti registracijos metu sugeneruotu privačiuoju raktu. Vartotojas gauna prieigą prie privataus rakto tik sėkmingai atlikus autentifikaciją prie lokalaus įrenginio (mobiliojo telefono). Vartotojo privačiu raktu pasirašytas „iššūkis“ siunčiamas paslaugos serveriui. Šis gali patikrinti parašo tikrumą pasinaudojant savo duomenų bazėje turimu vartotojo, iš kurio gavo pasirašytą „iššūkį“, viešuoju raktu. Jei pavyksta sėkmingai iššifruoti gautą užšifruotą „iššūkį“ ir jis atitinka pradinį, kuris buvo išsiųstas vartotojui – vartotojui suteikiama prieiga prie paslaugos.



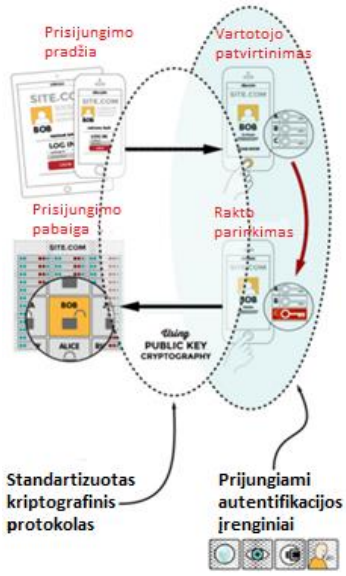
2 pav. FIDO prisijungimo procesas

2 pav. matyti prisijungimo procesas prie paslaugos, kuri palaiko FIDO standartus. Prisijungimo proceso pradžioje paslaugos serveris atsiunčia tam tikrą „iššūkį“, kurį vartotojas turi pasirašyti

registracijos metu sugeneruotu privačiuoju raktu. Vartotojas tai gali atlikti savo mobiliojo įrenginio pagalba, prieš tai atlikus vietinę autentifikaciją prie mobiliojo įrenginio. Atlikus vietinę autentifikaciją, pagal identifikuotą vartotoją, mobilusis įrenginys parenka atitinkamą vartotojo privatųjį raktą iš kelių turimų, juo pasirašo gautą „iššūkį“ ir siunčia atgal paslaugos serveriui. Paslaugos serveris patikrina gautą pasirašytą „iššūkį“ su nurodyto vartotojo viešuoju raktu, kurį saugo savo duomenų bazėje.

1.6. FIDO saugumas

Naudojant FIDO standartus galima būti ramiam, jog įsilaužėliai nepadarys rimtos žalos. Kadangi FIDO atskiriamas autentifikavimo procesas į vietinį autentifikavimą mobiliuoju telefonu, kuris atpažindamas vartotoją sukuria ar atrakina kriptografinius raktus ir pats toliau atsako už išorinę autentifikaciją su paslauga, jai siųsdamas tik vartotojo viešąjį raktą. Tokiu atveju „žvejybos“ (angl. „phishing“), ir slapto pasiklausymo (angl. „man-in-the-middle“) atakų metu viskas, ką pavyksta išgauti, yra tik viešasis vartotojo raktas, kurį, dėl kriptografinių algoritmų sudėtingumo, nulaužti maža tikimybė [18]. Šie algoritmai generuoja viešuosius ir privačiuosius raktus taip, jog jie būtų matematiškai susiję, bet turint vieną iš jų, būtų beveik neįmanoma išgauti kito. Tai apima saugumą tiek vartotojo vietiniame įrenginyje, tiek paslaugos serveryje. Piktavalis taip pat negalėtų susieti išgautos informacijos su konkrečiu vartotoju. Jei naudojamas UAF standartas ir vartotojas autentifikacijai per vietinį įrenginį naudoja ne slaptažodžius, tai dar labiau sustiprina apsaugą prieš „žvejybos“ tipo atakas, nes tada ir pats vartotojas nebesaugo jokios paslapties, kurią galėtų kas nors išgauti. Silpna vieta galime laikyti tik naudojamo išorinio įrenginio fizinę vagystę arba įrenginio pametimą. Tačiau ir šiuo atveju užtektų tik pranešti paslaugos tiekėjui apie prarastą įrenginį, vos tik jo pasigedus, jog būtų užblokuojama prieiga prie paslaugos iš to įrenginio.

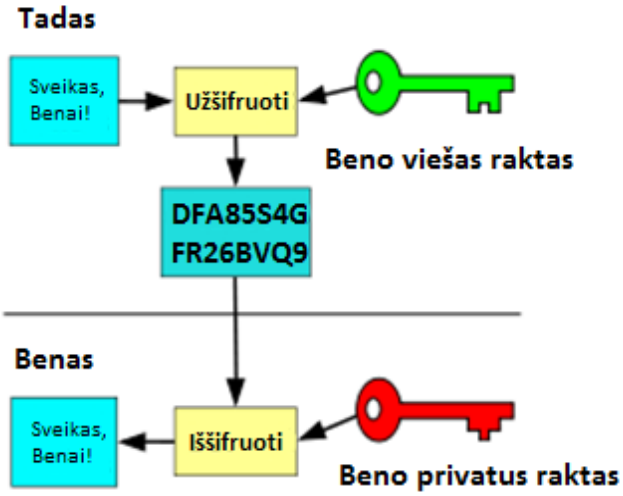


3 pav. FIDO komponentai

Iš 3 pav. matyti, kaip FIDO atskiria vietinį ir viešą autentifikavimą. Vietiniam autentifikavimui gali būti panaudojami įvairūs prijungiami autentifikavimo prietaisai, kuriuose saugoma vartotojo identifikacinė informacija. Viešam autentifikavimui tarp vietinio prietaiso ir paslaugos serverio naudojami kriptografiniai mechanizmai, siekiant užtikrinti duomenų saugumą.

1.7. Kriptografiniai protokolai

FIDO protokolai komunikacijai tarp vietinio įrenginio ir internetinės paslaugos naudoja asimetrinę kriptografiją, todėl reikėtų šiek tiek panagrinėti kas tai yra ir kaip tai veikia, norint įsitikinti, jog tai tikrai saugu [19]. Asimetrinė kriptografija (dar vadinama viešojo rakto kriptografija) yra kriptografijos priemonių aibė, kurioje naudojama raktų pora [20]. Raktų porą sudaro viešasis ir privatusis raktas. Vienas raktas gali būti panaudojamas informacijai užšifruoti, o kitas – iššifruoti [21]. Privatusis raktas skirtas vienam konkrečiam asmeniui ir jis turi jį saugoti nuo atskleidimo ir su niekuo juo nesidalinti. Tuo tarpu viešasis raktas gali būti atskleidžiamas visiems, kurie siekia saugios komunikacijos su rakto šeimininku. Viešasis ir privatusis raktai yra matematiškai susiję taip, kad bet kokią informaciją užšifravus viešuoju raktu, ši informacija gali būti iššifruota tik tos pačios poros privačiuoju raktu.



4 pav. Viešojo ir privataus raktų poros panaudojimas

4 pav. pateiktas asimetrinės kriptografijos naudojimo pavyzdys. Tadas nori nusiųsti slaptą žinutę Benui, jog niekas kitas negalėtų jos perskaityti. Tadas užšifruoja slaptą žinutę Beno viešuoju raktu ir siunčia ją Benui nesaugiu kanalu. Jei piktavalis perimtų užšifruotą pranešimą, jis jo negalėtų iššifruoti ir perskaityti, nes neturi Beno privataus rakto. Tik Benas gavęs užšifruotą pranešimą gali jį iššifruoti savo privačiuoju raktu ir perskaityti, ką Tadas jam parašė.

Taip pat asimetrinę kriptografiją galima panaudoti norint identifikuoti asmenį. Jei asmuo užšifruoja tam tikrą pranešimą su savo privačiuoju raktu ir jį atsiunčia, tai turint to asmens viešąjį raktą ir sėkmingai juo iššifravus pranešimą, galima neabejoti, jog tai tikrai to asmens siųstas pranešimas.

Panagrinėkime asimetrinės kriptografijos veikimą iš matematinės pusės, pasinaudojant RSA kriptografine sistema [22]. Pirmiausia reikia raktų poros, kuria remiasi visa kriptografinė sistema. Ji generuojama štai taip:

1) pasirenkami bet kokie du tarpusavyje pirminiai skaičiai p ir q;
2) sudauginame juos, jog gautume sandaugą $n = p \times q = 53 \times 61 = 3233$;
3) toliau apskaičiuojame $\varphi(3233) = (p-1)(q-1) = 52 \times 60 = 3120$;
4) pasirenkame bet kokį skaičių e, kuris yra tarp 1 ir 3120, bet būtų tarpusavyje pirminis su pasirinktų pirminių skaičių sandauga (skaičiumi 3120), pvz., skaičius 17;
5) naudojantis Euklido algoritmu, surandame skaičių d, kuris šiuo atveju lygus skaičiui 2753.

Įvykdę visus žingsnius gauname viešąjį raktą (e, n) = (17, 3233) ir privatųjį raktą d = 2753. Norint užšifruoti slaptą pranešimą siuntimui nesaugiu kanalu, pasinaudojame šifravimo formule $c = m^e mod\ n$, kur m yra norimas užšifruoti pranešimas, o c – užšifruotas pranešimas. Tokiam pranešimui iššifruoti turėtume pasinaudoti formule $m = c^d\ mod\ n$ ir jei viskas gerai – gautume pradinį pranešimą, kurį turėjome prieš užšifravimą.

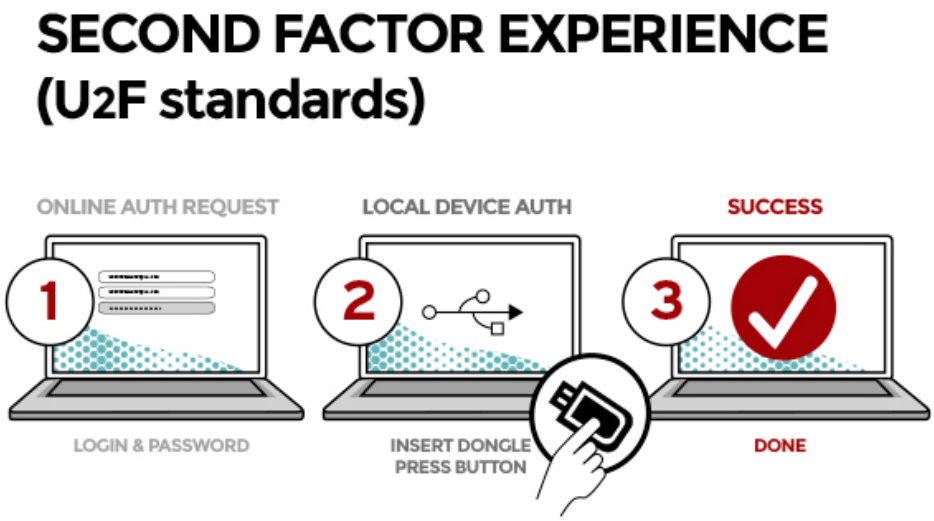
Jau turėdami sugeneruotą raktų porą, pamėginkime užšifruoti ir iššifruoti slaptą pranešimą, jog įsitikintume praktiniu algoritmo veikimu. Pranešimu pasirenkame skaičių 33, kuris, panaudojus formule, pavirsta į $c = m^e mod\ n = 33^{17} mod\ 3233 = 1853$. Norint iššifruoti tikrąjį skaičių, kuris slepiasi po 1853, panaudojame iššifravimo formulę $m = c^d\ mod\ n = 1853^{2753} mod\ 3233 = 33$. Iššifruoto pranešimo reikšmė atitinka pradinę pasirinktą reikšmę, todėl užšifravimas ir iššifravimas pavyko sėkmingai. Verta paminėti, jog realiose sistemose naudojamų raktų simbolių kiekis siekia 2048 bitus, tad čia pateikti skaičiavimai žymiai supaprastinti ir neatitinka vykdomų tikrųjų, kurie yra sunkiai įveikiami ir galingiems kompiuteriams, kas ir garantuoja aukštą saugumo lygį. Taip pat šie skaičiavimai vykdomi žinant raktų poros abu raktus, tad įsibrovėliui net ir turint vieną raktą, reikalingi atlikti skaičiavimai yra žymiai sudėtingesni.

1.8. FIDO protokolai

Siekiant saugumo ir paprasto naudojimo vartotojams, FIDO vysto du skirtingus protokolus – „Universal Second Factor“ (U2F) ir „Universal Authentication Framework“ (UAF) [23]. Abu protokolai paremti viešojo rakto kriptografija ir dėl to yra atsparūs „žvejybos“ tipo atakoms [24].

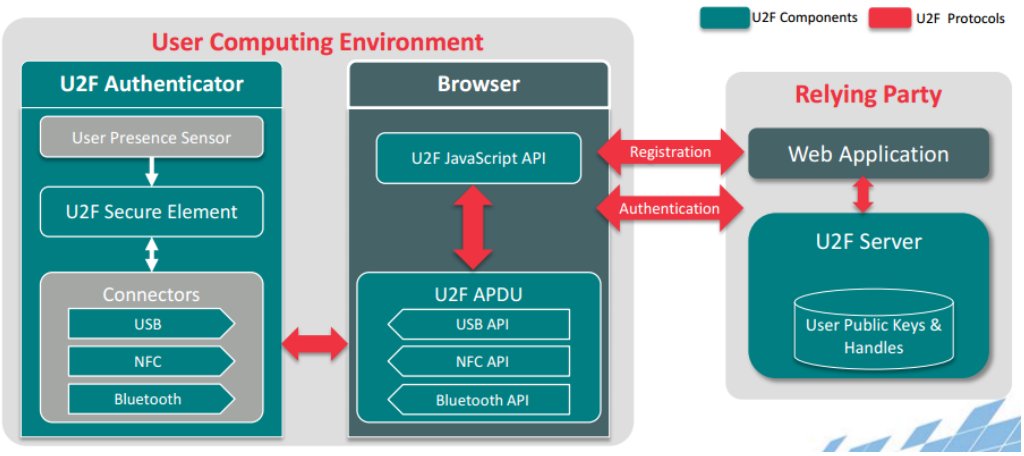
1.8.1. U2F protokolas

Šis protokolas koncentruojasi į slaptažodžiais paremtos autentifikacijos sustiprinimą panaudojant papildomą faktorių vartotojo autentifikavimui. Paprasčiausias pavyzdys gali būti USB atmintinės panaudojimas, kurią vartotojas turi prijungti prie įrenginio, iš kurio nori autentifikuotis prie internetinės paslaugos. Įprastai naudojamas papildomas faktorius yra koks nors fizinis įrenginys arba fizinio įrenginio pagalba išgaunama biometrinė charakteristika. Tai yra neparanku nuolatos keliaujantiems vartotojams dėl poreikio visada su savimi turėti autentifikacijai skirtą prietaisą. Norint to išvengti, siūlomi įvairūs sprendimai pakeisti fizinius įrenginius į „debesų“ paslaugų sprendimus [25]. Taip pat, 2015 metais FIDO aljansas praplėtė savo U2F specifikaciją „Bluetooth“ ir „NFC“ palaikymu [26].



5 pav. U2F protokolas

Šio protokolo panaudojimui reikalingas prietaisas, turintis U2F palaikymą interneto naršyklėje. Prisijungimui prie paslaugos serverio naudojamas slaptažodis (kaip ir įprastai), tačiau papildomai paslaugos serveris gali paprašyti antro žingsnio vietinės autentifikacijos.

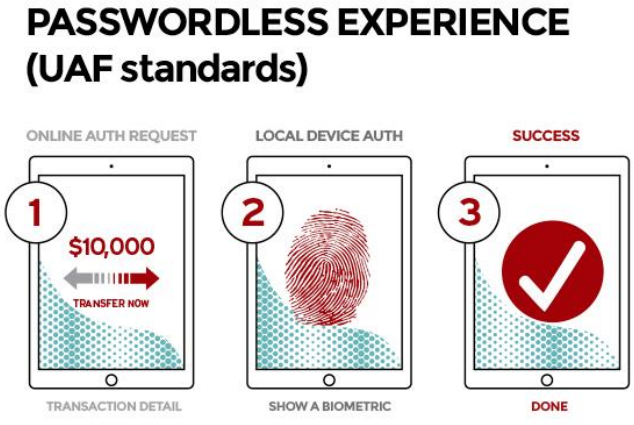


6 pav. FDO U2F komponentai ir protokolai

6 pav. matomi U2F protokole naudojami komponentai ir kaip jie komunikuoja tarpusavyje. Pagrindė dalis susideda iš vartotojo įrenginių ir paslaugos serverio, su kuriuo bendraujama. Vartotojo dalį apima interneto naršyklė, palaikanti U2F protokolą ir vietinis autentifikavimo prietaisas. Paslaugos serveris turi savo aplikaciją ir U2F serverį su duomenų baze, kurioje saugomi autentifikavimo duomenys – vartotojų viešieji raktai.

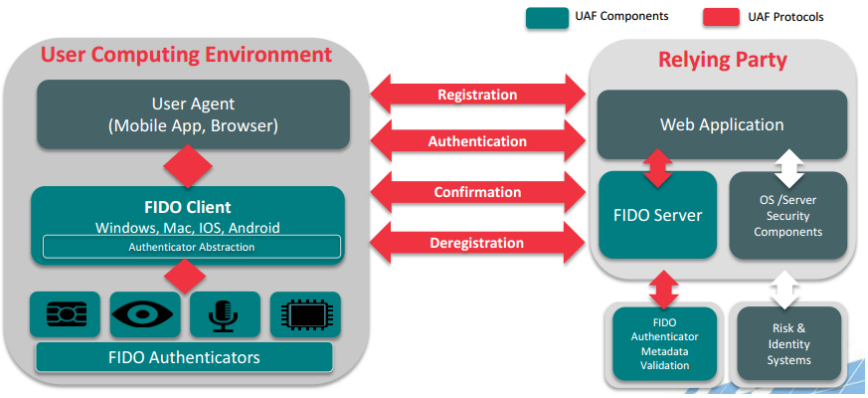
1.8.2. UAF protokolas

UAF leidžia visiškai atsikratyti slaptažodžių ir pakeisti juos biometrine arba kitokia vietine autentifikacija. Šiuo atveju vartotojas autentifikuojasi prie vietinio įrenginio (pvz., mobiliojo telefono), o įvykus sėkmingai autentifikacijai – jungiasi prie paslaugos. Taip gaunamas tarpinis komunikavimas ir vartotojas išvengia tiesioginio komunikavimo su paslaugos serveriu, kas leidžia sustiprinti saugumą apsaugant vartotojo konfidencialumą.



7 pav. UAF protokolas

Šio protokolo naudojimui reikalingas prietaisas su paruošta UAF įranga. Nebelieka slaptažodžių autentifikavimosi, tad galimi keli skirtingų tipų autentifikavimo metodai, pvz. piršto antspaudas ir USB atmintinė.



8 pav. UAF komponentai ir protokolai

Nors iš 8 pav. matyti, jog UAF atrodo gan panašiai į U2F, tačiau UAF turi papildomai patvirtinimo ir išregistravimo protokolus. Taip pat šiuo atveju palaikoma ne tik interneto naršyklė, bet apskritai bet kokia programėlė, nes standartas įdiegtas pačiame įrenginyje, o ne tik palaikomoje naršyklėje. Taip pat tarp autentifikavimo prietaiso ir programėlės patalpintas klientas, kuris leidžia patogiau naudoti skirtingus elementus skirtinguose įrenginio galuose (programėles su autentifikavimo prietaisais). Paslaugos serverio pusėje taip pat padaugėjo komponentų saugumo sustiprinimui.

1.9. Analizės išvados

Atlikus analizę buvo susipažinta su kelių faktorių autentifikavimo principu, nauda ir taikomais 3 pagrindiniais (ką žinai, ką turi ir kas esi), bei 4 mažiau žinomais (ką darau, ką pažįstu, ką apskaičiuoju ir kur esu), būdų tipais. Apžvelgti biometriniai autentifikavimo metodai ir jų charakteristikos. Prieita prie išvados, jog kiekvienas autentifikavimo būdas, ir netgi kiekvieno būdo kiekvienas metodas, turi savo privalumų ir trūkumų, todėl nėra vieno geriausio būdo viskam ir reikia rinktis tinkamiausią konkrečiam poreikiui užtikrinti. Tačiau, taikant kelių faktorių autentifikavimą, kelių skirtingų autentifikavimo būdų kombinacija puikiai gali padengti viena kitos silpnas vietas ir taip leisti sukurti pakankamai saugų mechanizmą, neinvestuojant daug piniginių resursų. Dėl šios priežasties kelių faktorių autentifikavimas populiarės ir toliau, vis labiau atsisakant autentifikacijos paremtos vienu faktoriumi.

Išanalizuoti FIDO aljanso siūlomi šiuolaikiniai autentifikavimo sprendimai (UAF ir U2F), turintys plačias galimybes ir suteikiantys papildomą saugumą jau prie egzistuojančių sistemų. Visa tai pasiekta tuo pačiu palengvinus ir pagerinus eilinių vartotojų patirtį naudojantis internetinėmis autentifikavimo paslaugomis. Pateiktas FIDO aljanso naudojamos asimetrinės kriptografijos supaprastintas pavyzdys. Išsiaiškintas asimetrinės kriptografijos panaudojimas komunikacijai su internetinėmis paslaugomis ir abstraktus jos veikimas. FIDO žada daug, toliau siekdami tobulėjimo ir augimo, siūlant vis daugiau galimybių ir gerinant vartotojų patirtį.

2. KELIŲ FAKTORIŲ AUTENTIFIKAVIMO PROJEKTAS

Analizės dalyje, nagrinėjant autentifikavimo metodus, buvo prieita prie išvados, jog kiekvienas autentifikavimo mechanizmas turi savo privalumus ir trūkumus, lyginant juos tarpusavyje. Norint pasiekti didesnį saugumo lygį, būtina kombinuoti kartu tokius autentifikavimo metodus, kurie padengia vienas kitą (saugumo prasme).

Įsivaizduokime organizaciją, kurioje taikoma kelių lygių saugumo patikra. Visi, į darbą atvykę, darbuotojai tokioje organizacijoje pirmiausia galėtų būti identifikuojami tokiu biometrinio autentifikavimo metodu, kurio šablono dydis nėra didelis, tačiau ir nesuteikiamas aukštas saugumo lygis, pvz. piršto antspaudo skaitytuvas. Šis autentifikavimo metodas tarnautų kaip pirminis identifikavimo šaltinis, kuris leistų užtikrinti, jog konkretus darbuotojas pateko į darbines patalpas ir jose yra. Esant zonoms, kuriose būtų reikalingas papildomas, griežtesnis saugumas, prie patekimo į jas galėtų būti naudojamas, pvz., piršto venų skaitytuvas, kuris pasižymi tiek tikslumu, tiek aukštu saugumu, tiek ir atsparumu piršto pokyčiams. Nors piršto antspaudo skaitytuvas nepasižymi didele sauga, piršto venų skaitytuvas garantuotų patikimumą, jog į sugriežtintos apsaugos patalpas galėtų patekti tik tinkami asmenys. Organizacijos atveju naudojami bendri, visiems vartotojams skirti, autentifikavimo prietaisai skirtingose vietose, todėl nekyla didelių problemų nei dėl kainos, nei vietos.

Žvelgiant iš eilinio vartotojo perspektyvos, kuris naudojasi autentifikavimo priemonėmis savo asmeniniams tikslams (prisijungimui prie banko ar kitų internetinių paslaugų), susiduriame su tam tikromis problemomis. Kai kurių autentifikavimo metodų naudojimas reikalauja turėti specialius prietaisus, o tai reiškia, jog gali prireikti ne mažos aibės šių prietaisų, jei kombinuojame juos, siekiant didesnio saugumo. Norint juos visus įsigyti, reikės skirti nemažai piniginių resursų ir juos reikės nešiotis visur kartu, kas nėra patogu. Tai tam tikras trūkumas, kuriam reikalingas sprendimas. Sprendimu gali būti kelių autentifikavimo prietaisų apjungimas į vieną, taip sutaupant ir užimamos vietos ir pinigų prasme.

2.1. Projektui keliami reikalavimai

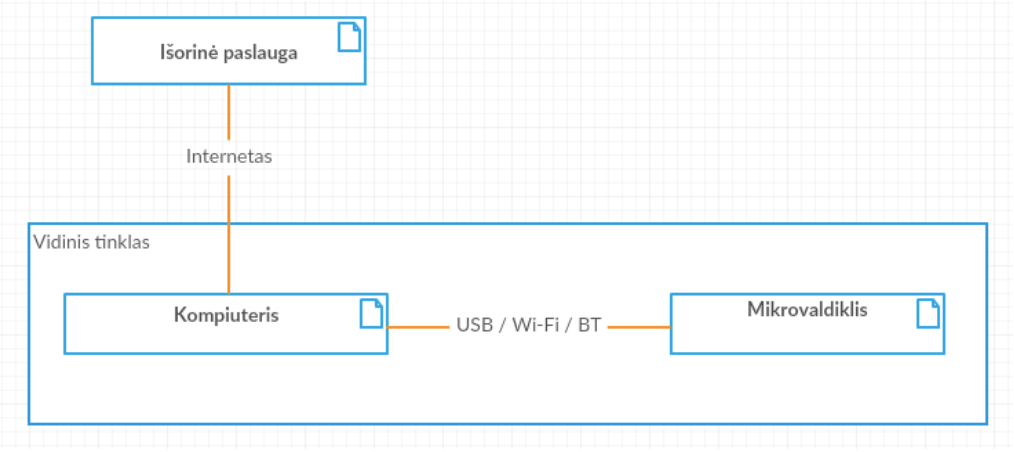
Siekiama sukurti universalų autentifikavimo mechanizmą, kuris leistų vartotojui naudojant vienintelį įrenginį, autentifikuotis keliais skirtingais metodais. Šiam mechanizmui sukurti panaudota techninė įranga turėtų būti patogi naudojimui, užimanti mažai vietos, pigi ir lengvai pritaikoma įvairiems kelių žingsnių autentifikavimo mechanizmams. Pats sprendimas turėtų leisti atlikti kelių žingsnių autentifikavimą panaudojant paprasčiausius autentifikavimo metodus. Turėtų būti galimybė pritaikyti sudėtingesnius ar daugiau nei dviejų žingsnių autentifikavimo mechanizmus.

2.2. Techninės dalies pasirinkimas

Norint, jog kuriamas sprendimas būtų kiek galima universalesnis ir pritaikytas naudojimui buityje ir neapsiribojant vien komerciniu lygiu, turėtų būti naudojamas nedidelis techninis įrenginys. Jis turėtų pasižymėtų kompiuterio savybėmis ir būti programuojamas, jog tiek jį patį būtų galima naudoti autentifikavimui, tiek ir prie jo būtų galima prijungti įvairias, autentifikavimui skirtas, priemones ir jas panaudoti. Geriausiai tam tinka programuojamas mikrovaldiklis, nes jis yra kompaktiškas, pigus ir jį galima prijungtas prie kompiuterio, bei galima prijungti kitus elektronikos komponentus (pvz., skaitytuvus) prie jo. Kai kurie mikrovaldikliai turi specialius prijungiamus modulius, tokius kaip piršto skaitytuvas, kuriuo pasinaudojus galima realizuoti biometrinį autentifikavimą kaip vieną iš kelių autentifikacijos žingsnių. Taip pat, naudojant mikrovaldiklį poroje su kompiuteriu, yra galimybė apsikeisti informacija ne tik laidu, bet ir kitomis priemonės, tokiomis kaip belaidžiu „Bluetooth“ ryšiu arba per tinklą, naudojantis „WiFi“ technologija.

2.3. Autentifikavimo modeliai

Pasinaudojus aukščiau paminėtomis techninėmis priemonės, toliau peržvelgsime su jomis galimus realizacijos modelius. Pateikti modeliai bus pakankamai paprasti, pateikiant juos tik kaip galimus realaus gyvenimo autentifikacijos panaudojimo pavyzdžius. Turint omenyje pasirinktų techninių priemonių galimybes, realizuoti galima itin sudėtingas autentifikavimo sistemas, panaudojant daugiau nei kelis metodus.



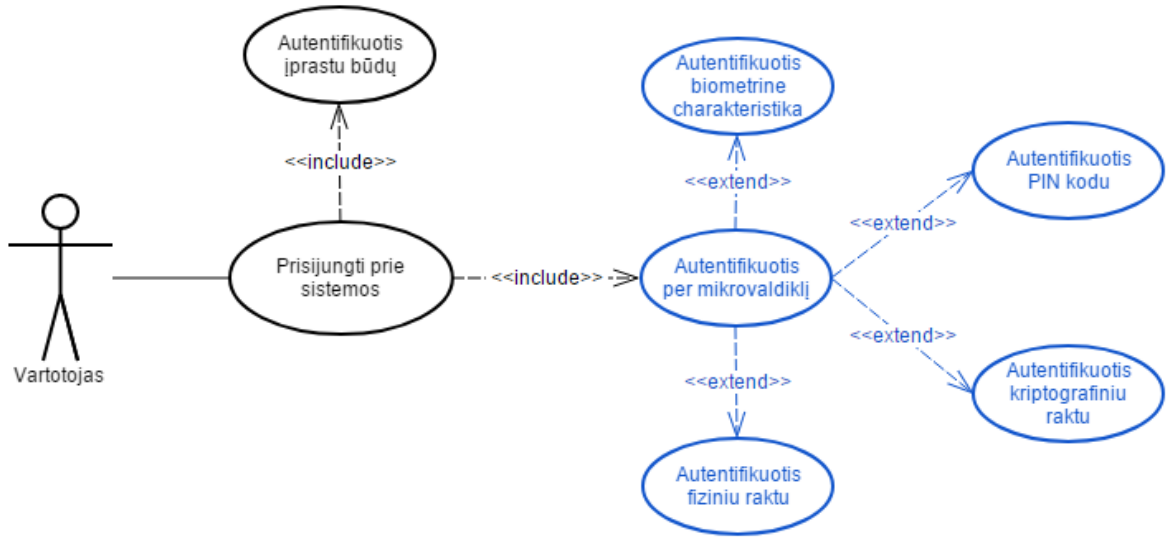
9 pav. Išdėstymo diagrama

Aukščiau pateiktame paveiksliuke atvaizduota bendra, visiems modeliams, komponentų architektūra. Išorine paslauga vadinama bet kokia paslauga, kuria norint pasinaudoti, reikalinga pirmiausia būtina prie jos autentifikuotis. Tai gali būti tiek bankinė sistema, tiek ir diskusijų forumas. Komunikacija su šia paslauga vyksta internetu iš vidiniame tinkle esančio kompiuterio (tai gali būti bet koks techninis įrenginys, kuris sugebėtų apsikeisti informacija tiek su išorine paslauga, tiek ir su

mikrovaldikliu). Jame įdiegtą programinė įranga skirta komunikacijai norimu kanalu su mikrovaldikliu ir ankščiau minėta paslauga.

Priklausomai nuo mikrovaldiklio modelio ir gamintojo, komunikacija su kompiuteriu gali būti vykdoma tiek USB laidu, internetu per interneto laidą ar „Wi-Fi“ tinklu, bei „Bluetooth“ ryšiu. Visa tai gali būti pasiekiama panaudojant papildomus komponentus. Autentifikacijai mikrovaldiklis gali saugoti slaptas reikšmes savo viduje, kurios būtų perduodamos gavus užklausą iš kompiuterio. Prie mikrovaldiklio prijungus papildomus komponentus, galima jį panaudoti kaip papildomą tarpinę stotelę informacijos perdavimui ar papildomą apsaugos mechanizmą. Pavyzdžiui, prijungus biometrinį skaitytuvą, tokį kaip pirštų antspaudų skaitytuvas, juo gautą biometrinę charakteristiką būtų galima perduoti išorinei paslaugai autentifikavimui. Be to, ta pati biometrinė charakteristika gali būti panaudota vartotojo identifikavimui pačioje mikrovaldiklio programinėje įrangoje. Taip papildomai identifikavus patį vartotoją, būtų atrakinama slapta, mikrovaldiklyje saugoma, reikšmė ir tik tada ji būtų perduodama išorinei paslaugai, vartotojo autentifikavimui.

Autentifikavimo mechanizmas gali būti įvairių lygių ir panaudojant įvairius autentifikavimo faktorius. Paprasčiausiu atveju, panaudojant standartinį prisijungimą su slapyvardžiu ir slaptažodžiu, o sudėtingiausiu – pareikalaujant visų trijų faktorių ir panaudojant tiek mikrovaldiklio programinės, tiek ir fizinės įrangos galimybes, kas reiškia unikalių schemų sukūrimą fiziniame lygyje, kurias galėtų panaudoti mikrovaldiklis vartotojo autentifikavimui.



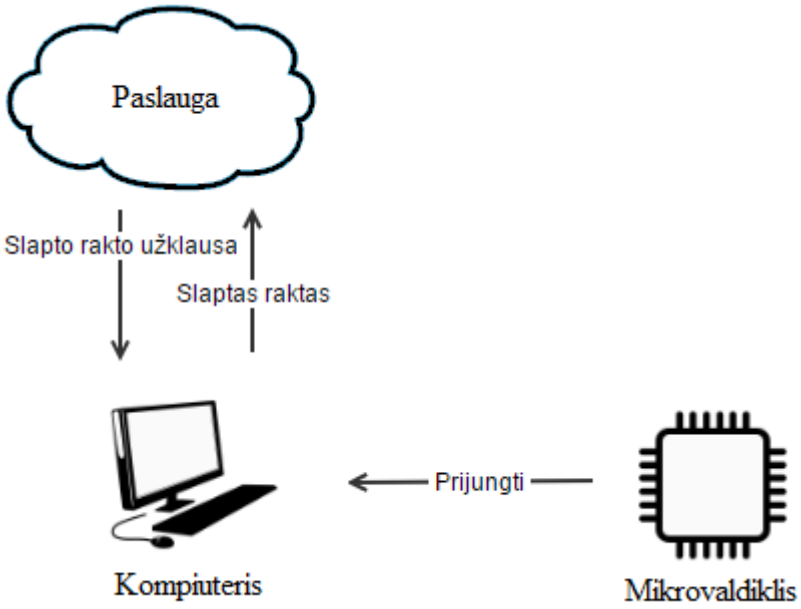
10 pav. Preliminari panaudos atvejų diagrama

Iš 10 pav. pateiktos preliminarios abstrakčios modelių panaudos atvejų diagramos matome, jog vartotoją, norintį prisijungti prie sistemos, pirmiausia galėtume identifikuoti įprastomis priemonėmis – prisijungimo vardu ir slaptažodžiu. Prototipo realizavimas apsiribotų mėlyna spalva pažymėtomis dalimis – antro ir kitų autentifikacijos žingsnių reikalavimo per mikrovaldiklį. Jo pagalba autentifikuotis būtų galima tiek PIN kodu, tiek prijungus fizinį ar kriptografinį raktus. Turint

papildomą įrangą, galima panaudoti ir biometrines charakteristikas autentifikacijos procesui. Informacija tarp mikrovaldiklio ir sistemos (ar tarpinio vartotojo kompiuterio), prie kurios norima prisijungti, gali būti perduodama tiek paprastu USB laidu, tiek ir internetu – interneto laidu arba „WiFi“ belaidžio ryšio technologija.

Mikrovaldiklio universalumas atveria labai plačias galimybes kiekvienam galimam realizacijos žingsniui ir galutinis viso norimo sukurti sprendimo sudėtingumas ir saugumo lygis priklauso tik nuo pasiriktų priemonių realizacijai įgyvendinti. Mikrovaldikliu galima pakeisti visus įprastinius autentifikavimo būdus, kada užtenka turėti prijungtą mikrovaldiklį prie namų tinklo ir norint prisijungti prie sistemos, nebereikia įvedinėti jokios informacijos. Sistema pati kreipiasi į mikrovaldiklį internetu ir gauna iš jo reikalingą informaciją vartotojo autentifikavimui, o vartotojui nereikia nieko papildomai atlikti. Taip pat, jei mikrovaldiklis panaudojamas ir tam tikrų duomenų surinkimui ir perdavimui (pvz. vandens ir elektros skaitiklių parodymams), galima realizuoti ir slapto kanalo su konkrečia sistema užmezgimą saugiam informacijos perdavimui, kai sistema kreipiasi norėdama gauti informaciją. Tokiu būdu galima iš karto perduodama ir autentifikacijos informacija, taip nereikalaujant visiško žmogiškojo faktoriaus įsikišimo į autentifikacijos procesą.

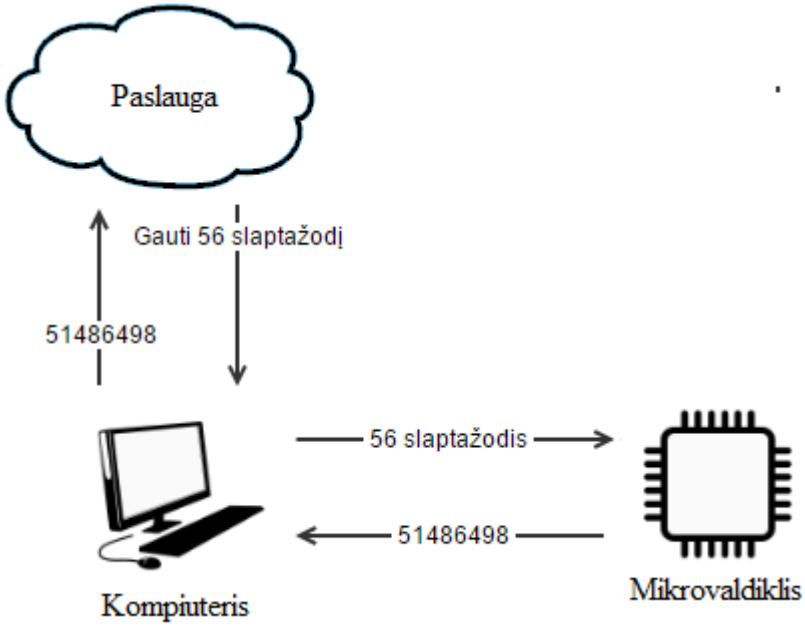
2.3.1. Fiziniai raktai



11 pav. Autentifikacija – fizinis raktas

Pats paprasčiausias ir lengviausiai suprantamas variantas. Mikrovaldiklis naudojamas kaip vartotojo, norinčio prisijungti prie paslaugos, identifikatorius. Kai paslauga, prie kurios jungiamasi, paprašo papildomos autentifikacijos iš vartotojo, vartotojas prijungia specialiai paruoštą autentifikavimui mikrovaldiklį prie kompiuterio per USB ar kitą palaikomą jungtį. Paprasčiausiu variantu paslaugos sistemai užtenka to fakto, jog mikrovaldiklis yra prijungtas ir vartotojas prijungiamas prie paslaugos. Siekiant didesnio saugumo, galima reikalauti kokios nors specifinės informacijos apie vartotoją, kurią turi mikrovaldiklis ir tik jį prijungus, ji bus perduodama paslaugos sistemai. Perduodama informacija gali būti paprastas tekstas (ar kodas), pagal kurį paslaugos sistema identifikuos vartotoją. Ši informacija sistemai gali būti perduodama registracijos prie paslaugos metu, jog vėliau sistema turėtų kaip patikrinti ar tai tas pats vartotojas bando jungtis, kuris ir užsiregistravo. Kitu atveju į mikrovaldiklį turėtų įprogramuoti atitinkamą informaciją paslaugos atstovai ir tik tada mikrovaldiklį gautų vartotojas iš paslaugos tiekėjo. Dar vienas galimas variantas, jei mikrovaldiklis turi savyje kokią nors identifikacinę informaciją, kuri saugoma ten, kur sistema galėtų pasiekti ir pasitikrinti ją gavus iš mikrovaldiklio.

2.3.2. Kodų kortelės

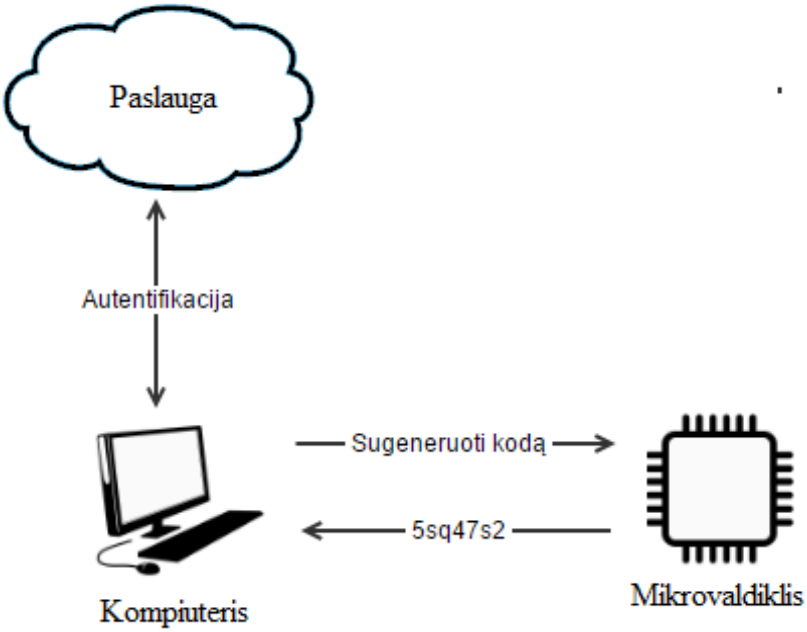


12 pav. Autentifikacija – kodų kortelė

Dar vienas iš paprastesnių metodų – kodų kortelių imitacija. Veikimo principas panašus į banko kortelės su slaptažodžiais (skaičiais). Mikrovaldiklyje įprogramuotas baigtinis skaičius slaptažodžių, kurie pasiekiami pagal indeksą. Norint gauti tam tikrą paslaugą, prašoma prisijungti prie paslaugos serverio. Viename iš prisijungimo žingsnių būtų prašoma įvesti atsitiktinai parinktą indeksą atitinkantį slaptą kodą. Atsitiktinai parinktas indeksas perduodamas vartotojo kompiuteriui arba išvedamas į ekraną vartotojui. Pirmuoju atveju kompiuteris kreipiasi į mikrovaldiklį prašydamas nurodyto indekso slaptažodžio ir jį gavęs iš jo, perduoda paslaugos serveriui. Antruoju atveju, vartotojas, pasinaudodamas komunikacijos su mikrovaldikliu priemonėmis (pvz., mygtukais ant mikrovaldiklio), įveda norimo gauti slaptažodžio indeksą ir mikrovaldiklis arba pats perduoda slaptažodį paslaugos sistemai, arba išveda vartotojui slaptažodį per vartotojo sąsają ir vartotojas pats įveda gautą slaptažodį į prisijungimo prie paslaugos formą.

Saugumo lygį galima koreguoti keičiant slaptažodžių sąrašo dydį arba keičiant pačių slaptažodžių ilgį ir sudėtingumą (pvz. slaptažodžiams naudoti ne tik skaičius, bet ir raides ar net specialius simbolius). Naudojant ilgus slaptažodžius, kuriuos perkopijuoti vartotojui į formą gali kilti problemų (pvz. 10 ir daugiau įvairių simbolių sekos), galima leisti mikrovaldikliui perduoti informaciją tiesiai į kompiuterį. Taip būtų prarandama šiek tiek saugumo, jei piktavalis įsilaužtų į kompiuterį ir gautų tiesioginę prieigą prie mikrovaldiklio, tačiau nenaudojamas mikrovaldiklis galėtų būti atjungtas nuo kompiuteriu, taip sumažinant tokios atakos atveju sukeltą pavojų.

2.3.3. Kodų generatoriai

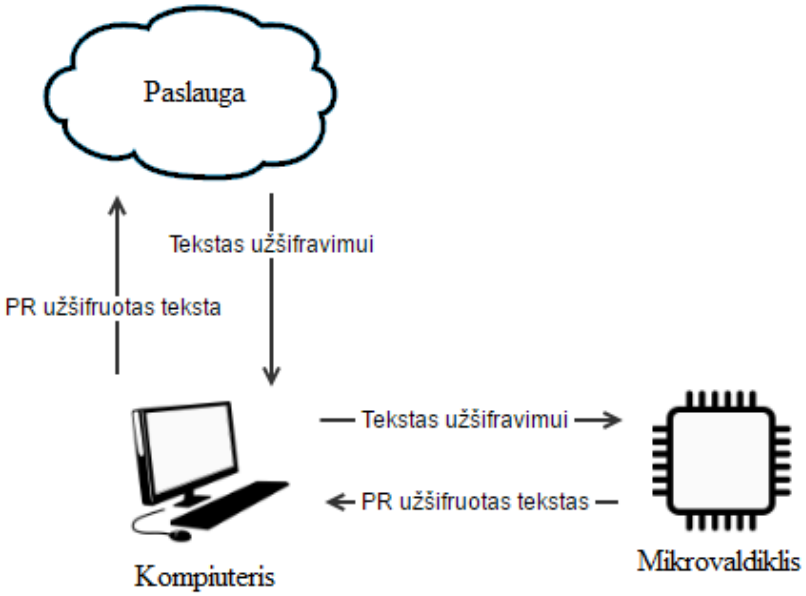


13 pav. Autentifikacija – kodų generatorius

Dar vienos, naudojamos bankinėse sistemose, autentifikacijos supaprastintas pavyzdys. Mikrovaldiklyje gali būti programiškai arba fiziškai įprogramuotas specialus kodų generatorius. Paprasčiausiu atveju toks generatorius gali atsitiktinai „sugeneruoti“ vieną iš kelių viduje numatytų reikšmių, kurios sistemoj būtų siejamos su konkrečiu vartotoju. Kadangi šis būdas būtų neefektyvus resursų atžvilgiu, sudėtingesnė alternatyva galėtų būti generavimo funkcija, kuri pagal tam tikrus duomenis (pvz. vartotojo vardą ir laiko žymą) gebėtų sugeneruoti reikšmę – vienkartinį slaptažodį. Sugeneruoto slaptažodžio surišimas su laiko žyma reiškia, jog jo panaudojimą būtų galima apriboti laike. Taip galima atmesti visus slaptažodžius, kurie buvo sugeneruoti seniau, nei prieš vieną ar kelias minutes. Susiejimui su paslauga, galima inicijuoti papildomo parametro iš paslaugos sistemos siuntimą milkrovaldikliui, kai vartotojas inicijuoja autentifikacijos procesą, kuris panaudotų gautą parametrą slaptažodžio generavimui. Tokiu atveju papildomai būtų galima pastebėti grubios jėgos (angl. „brute-force“) atakas, jei konkretus vartotojas nesikreipė autentifikacijai ir jam nebuvo išsiųstas parametras slaptažodžio generavimui, tačiau buvo gauta daug netinkamų slaptažodžių.

Šis būdas gali būti itin saugus, nes laiko žymos panaudojimas leistų iš karto pastebėti, jei koks piktavalis bandytų suklastosi slaptažodį atsitiktinai ar išgavus stebint realų vartotoją (pvz. sistema gautų kelių mėnesių senumo slaptažodžius arba su tuo pačiu slaptažodžiu būtų bandoma prisijungti prie sistemos antrą kartą).

2.3.4. Kriptografiniai raktai

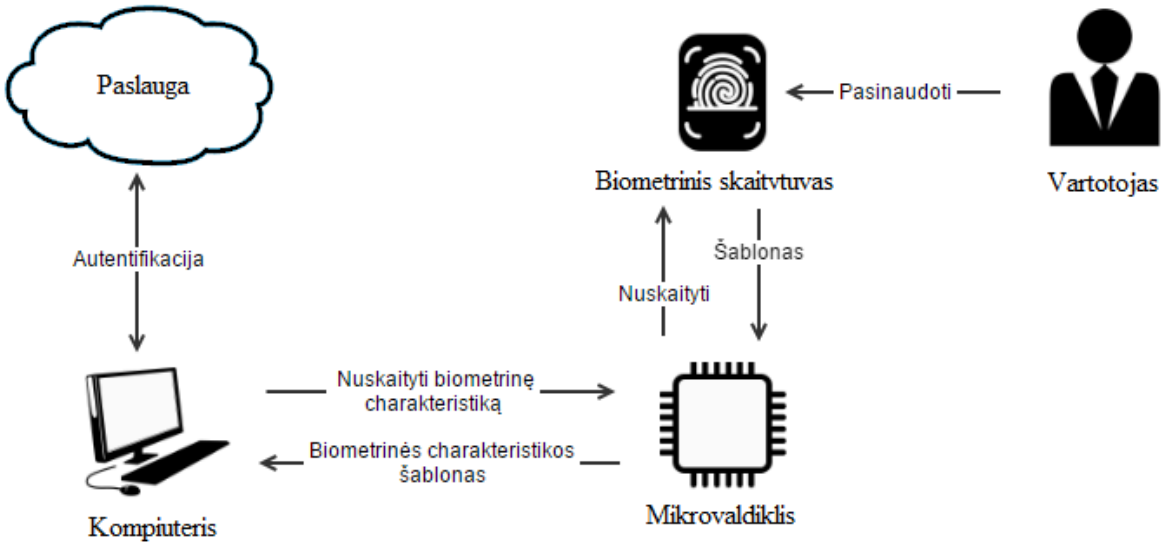


14 pav. Autentifikacija – šifravimas

Šio modelio autentifikavimo principas būtų panašus į FIDO, tačiau vietoje FIDO teikiamos autentifikavimo paslaugos, būtų naudojamas mikrovaldiklis. Mikrovaldiklis savyje turi vartotojui priklausančią kriptografinių raktų porą ir geba užšifruoti gautą informaciją. Kaip ir fizinio rakto modelyje, mikrovaldiklis prijungiamas prie kompiuterio. Skirtumas nuo fizinio rakto modelio yra tai, kad registracijos prie paslaugos etape vartotojas pateikia paslaugos sistemai savo viešąjį raktą, kuris sugeneruojamas ir išsaugomas mikrovaldiklyje, kartu su privačiu raktu. Vėliau, norint prisijungti prie paslaugos, autentifikacijai atsiunčiamas koks nors tekstas, kurį vartotojas, pasinaudojęs mikrovaldiklio turimu privačiu raktu, pasirašo ir siunčia paslaugos sistemai atgal. Paslaugos sistemai gavus privačiu raktu pasirašytą pranešimą, jis patikrinamas registracijos etape gautu vartotojo viešuoju raktu. Jei pavyksta sėkmingai pasinaudoti viešuoju raktu, vadinasi privatusis raktas yra tinkamo vartotojo (to, kuris bando prisijungti) ir jam suteikiama prieiga prie paslaugos.

Šiuo atveju suteikiama daugiau saugumo, nes autentifikavimui neužtenka fakto, jog mikrovaldiklis prijungtas (tai nesudėtinga suklastoti). Norint apgauti paslaugos sistemą, reikėtų papildomų pastangų sėkmingam vartotojo raktų poros išgavimui iš vartotojo mikrovaldiklio arba mėginti suklastoti siunčiamą pranešimą. Tokio modelio realizacijai papildomai reikėtų, jog sugeneruota raktų pora vartotojui būtų patalpinta į mikrovaldiklį, prieš jį pradedant naudoti vartotojui, tačiau galima būtų pažengusiam vartotojui leisti pačiam susidiegti raktus į mikrovaldiklį. Visiškai patikimam naudojimui reikalingas trečiosios šalies įsikišimas, kuri generuotų ir teiktų privatų raktą vartotojui, o vartotojų viešuosius raktus – sistemoms, jog šios galėtų patikrinti vartotojo tapatybę. Sistema, šiuo atveju, turėtų pasitikėti trečiąja šalimi.

2.3.5. Biometrinės charakteristikos



15 pav. Autentifikacija – biometrinė charakteristika

Vienas iš pažangesnių galimų būdų modelis – specialaus įrenginio (biometrinio skaitytuvo) panaudojimas autentifikavimui per mikrovaldiklį. Jungiantis prie paslaugos, vartotojas gali pasirinkti vieną iš šio autentifikavimo būdo metodų (piršto, akies tinklainės ar kitos charakteristikos nuskaitymas). Šio būdo esmė – vartotojas pasinaudoja biometriniu įrenginiu (kuris prijungiamas prie mikrovaldiklio) norėdamas išgauti biometrinės charakteristikos šabloną ir prisijungti prie paslaugos serverio. Jei naudojamu biometriniu įrenginiu pasirenkamas piršto skaitytuvas, vartotojas pasinaudoja juo gauti savo piršto antspaudo šabloną. Šis šablonas siunčiamas per mikrovaldiklį į kompiuterį ir iš jo į paslaugos sistemą, prie kurios norima prisijungti. Sistema palygintų gautą piršto antspaudo šabloną su anksčiau išsaugotu (registracijos metu) taip autentifikuodama vartotoją. Registracijos metu biometrinės charakteristikos šablono kūrimas gali užtrukti ir gali būti prašoma kelių pakartotinių nuskaitymų, tačiau prisijungimo procesas išskirtinai paprastas ir nereikalaujantis jokių papildomų vartotojo veiksmų.

Tokios autentifikacijos tipas vadinamas „kažkas, kuo esi“ autentifikacija. Šis tipas išskirtinis tuo, jog vartotojui nereikia atsiminti jokios slaptos informacijos (slaptažodžio ar PIN kodo) ir vartotojui nereikia nešiotis jokio unikalaus prietaiso, jog galėtų būti autentifikuotas. Tai didžiulis privalumas prieš kitus autentifikacijos tipus, nes slaptažodžiai ilgainiui pasimiršta, o nešiotis specialų prietaisą visada su savimi nėra praktiška ir patogu. Taip pat yra nemažas biometrinių skaitytuvų pasirinkimas ir kiekvienas jų turi privalumų ir trūkumų, kurie buvo aptarti analizės dalyje.

2.4. Modelių apibendrinimas

Visi, aukščiau pateikti, modeliai gali suteikti papildomos apsaugos autentifikacijai prie bet kokios paslaugos sistemos. Įmanomi įvairūs kiekvieno modelio realizacijos variantai, tiek panaudojant įvairią techninę įrangą, tiek algoritmus (kriptografinio rakto atveju). Be to, galima įvairių modelių kombinacija, kai tas pats mikrovaldiklis naudojamas ne tik kaip fizinis raktas, bet dar, pavyzdžiui, kaip ir kodų kortelė. Neapribojamas ir daugiau nei dviejų modelių panaudojimas autentifikacijai – pvz. fizinis raktas, kriptografinis raktas ir biometrine charakteristika paremta autentifikacija. Tokiu būdu gaunamas kelių žingsnių autentifikavimas atitinkamai padidina saugumą. Pridėjus pačios sistemos pirminį autentifikavimą prisijungimo vardu ir slaptažodžiu, gaunamas kompleksinis autentifikacijos mechanizmas, galintis apimti visus autentifikacijos faktorius – „kažkas, ką žinai“, „kažkas, ką turi“, „kažkas, kas esi“.

Mikrovaldiklio privalumai nesibaigia ties galimybę panaudoti įvairius autentifikacijos metodus, bei sujungti juos viename įrenginyje. Dar vienas iš jo panaudojimo privalumų – galimybė perduoti informaciją ne tik laidu į kompiuterį, bet ir belaidžiu „Bluetooth“ ryšiu ar internetu (tiek laidu, tiek naudojanti belaidžio ryšio technologijas). Tai leidžia mikrovaldiklį panaudoti nepriklausomai nuo kompiuterio vietos, prie kurio jį norima prijungti. Galima panaudoti vieną mikrovaldiklį visiems vietinio tinklo vartotojams (pvz. organizacijos atveju arba buitiniam naudojimui, keliems šeimos nariams). Tokiu atveju sutaupoma nemažai finansų, nes nereikia pirkti atskiro mikrovaldiklio ir jo dalių kiekvienam asmeniui, bei taupomi kiti resursai (pvz., elektros energija) ir vieta.

2.5. Reikalavimai realizacijai

Pasirinktų modelių realizacija turėtų būti kuo paprastesnė ir patogesnė vartotojui, tačiau neapriboti universalumo. Tai apima slaptažodžių perdavimą tiek tiesiai į kompiuterį, tiek paliekant galimybę parodyti slaptažodį vartotojui per prijungtą ekraną. Pastarasis variantas būtų reikalingas, jei nebūtų techninių galimybių prijungti mikrovaldiklį prie kompiuterio nei laidu, nei belaidžiu „Bluetooth“ ryšiu, nei per internetą (atvejai, kai mikrovaldiklis nešiojamas). Taip pat, esant galimybei, autentifikaciją turėtų būti galima atlikti tiek be jokio vartotojo įsikišimo (turint tik prijungtą mikrovaldiklį prie kompiuterio), tiek tik gavus vartotojo patvirtinimą mygtuko paspaudimu ar panašiomis priemonėmis. Tokiu atveju vartotojas pats valdys, kada atliekama autentifikacija ir bus išvengta pavojaus, jog kažkas įsilaužęs į vartotojo kompiuterį per tinklą galės prisijungti prie bet kurios paslaugos, kuriose vartotojas registruotas. Tas pats galioja ir piktavalių mėginimams išgauti autentifikacijos duomenis pakartotinai daug kartų, kol bus surinkta visa reikalinga informacija (pvz. visi slaptažodžiai saugomi mikrovaldiklyje), vartotojui apie tai net nežinant.

Kadangi pagrindinis autentifikavimo procesas vyksta naudojantis mikrovaldikliu, turi būti užtikrinama jo ir kitų sudedamųjų proceso dalių sauga. Tai apima griežtesnės saugos laikymasis identifikuojant vartotoją pirmajame autentifikacijos žingsnyje – sistemoje, prie kurios jungiamasi. Tik po sėkmingos autentifikacijos prie sistemos dalies (pirmojo žingsnio, kelių dalių autentifikacijos procese), turi būti vykdoma tolimesnė autentifikacija, susijusi su mikrovaldiklio panaudojimu.

Kai kuriais atvejais, turėtų būti leidžiama vartotojui pačiam užpildyti su autentifikacija susijusius duomenis. Paprastas pavyzdys būtų galimybė papildyti slaptažodžių sąrašą, saugomą mikrovaldiklyje arba skaičių generavimui reikalingų parametrų įvedimas, kurie yra laikini ir reikalingi teisingo rezultato suskaičiavimui – slaptažodžiui.

3. KELIŲ FAKTORIŲ AUTENTIFIKAVIMO PROJEKTO PROTOTIPAS

3.1. Prototipo platformos pasirinkimas

Žengiant pirmuosius žingsnius link projekto realizacijos, būtina išsirinkti tinkamiausią techninę įrangą. Nors pačių mikrovaldiklių yra didelis pasirinkimas, tačiau siekiamam projekto įgyvendinimo patogumui galime rinktis iš egzistuojančių atvirojo kodo platformų, skirtų darbui su perprogramuojamais mikrovaldikliais ir kitais elektroniniais komponentais. Kelios iš populiariausių tokių platformų: „Arduino“, „Raspberry Pi“ ir „BeagleBone“. Toliau apžvelgsime kiekvieną iš jų detaliau, siekiant išsiaiškinti esminius skirtumus ir kiekvienos platformos privalumus.

3.1.1. „Arduino“ platforma

Šiai platformai naudojami įvairūs mikrovaldikliai, turintys analogines ir skaitmenines įvesties/išvesties jungtis. Pasinaudojant šiomis jungtis, galima kurti įvairias elektronines schemas, taip pat ir panaudojant „Arduino“ platformos išplėstinius komponentus (angl. „shields“). Jų dėka galima susikurti įvairių naudingų prietaisų, tokių kaip termometras ar laikrodis. „Arduino“ mikrovaldiklis gali turėti USB jungtį, per kurią galima komunikuoti su kompiuteriu, kuriame įdiegta „Arduino“ programavimo sąsaja. Šioje programavimo sąsajoje parašytas programinis kodas gali būti įrašytas į mikrovaldiklio atmintį per USB sąsają ir pradėtas vykdyti per kelias sekundes po programinio kodo parašymo. „Arduino“ programavimo sąsaja palaiko C ir C++ programavimo kalbas, nors galima naudoti ir kitas, papildomai įsidiegus tam reikalingas programavimo bibliotekas. Pirmasis „Arduino“ mikrovaldiklis buvo išleistas 2005 metais. Keletas iš „Arduino“ prietaisų: „Uno“, „Leonardo“, „Mega“, „Nano“, „Due“, „Yun“.

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
101	Intel® Curie	3.3 V/ 7-12V	32MHz	6/0	14/4	-	24	196	Regular	-
Gemma	ATtiny85	3.3 V / 4-16 V	8 MHz	1/0	3/2	0.5	0.5	8	Micro	0
LilyPad	ATmega168V ATmega328P	2.7-5.5 V / 2.7-5.5 V	8MHz	6/0	14/6	0.512	1	16	-	-
LilyPad SimpleSnap	ATmega328P	2.7-5.5 V / 2.7-5.5 V	8 MHz	4/0	9/4	1	2	32	-	-
LilyPad USB	ATmega32U4	3.3 V / 3.8-5 V	8 MHz	4/0	9/4	1	2.5	32	Micro	-
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Micro	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
MKR1000	SAMD21 Cortex-M0+	3.3 V/ 5V	48MHz	7/1	8/4	-	32	256	Micro	1
Pro	ATmega168 ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	0.512 1	1 2	16 32	-	1
Pro Mini	ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	1	1	32	-	1
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	-	32	256	2 Micro	2

16 pav. „Arduino“ modelių palyginimas

Kaip matyti iš 16 pav., rasto [27] šaltinyje, įvairios „Arduino“ versijos gali skirtis procesoriumi ir jo greičiu, naudojama elektros įtampa, įvesčių ir išvesčių kiekiu, bei atmintimi. Didžioji dalis „Arduino“ mikrovaldiklių naudoja „ATmega“ procesorius ir veikia esant 3,3 arba 5 voltų elektros įtampai. Procesoriaus greitis, priklausomai nuo versijos, gali siekti nuo 8 MHz iki 48 MHz. Taip pat tarp versijų smarkiai variuoja analoginių ir skaitmeninių įvesčių kiekis. „EEPROM“ (angl. „Electrically Erasable Programmable Read-Only Memory“) stulpelis nurodo atmintį, kurioje galima saugoti informaciją ilgesnį laiko tarpą. „SRAM“ (angl. „Static Random Access Memory“) saugomi kintamieji su savo reikšmėmis, kurie sukuriami programos veikimo metu. Flash atmintyje saugoma visa įkelta programa vykdymui. Flash ir EEPROM atmintis išlieka atjungus „Arduino“ įrenginį nuo elektros srovės, o SRAM atmintis – prarandama. „Arduino“ įrenginiai neturi daug atminties, bet prie jų galima prijungti SD kortelėms skirtus priedus, leidžiančius įrašyti ir skaityti informaciją iš SD kortelių.

3.1.2. „Raspberry Pi“ platforma

Ši platforma labai panaši į „Arduino“, tačiau pati „Raspberry Pi“ plokštelė atstoja visą kompiuterį, o ne tik mikrovaldiklį, kaip „Arduino“ [28]. Tuo galima įsitikinti palyginus jų charakteristikas. „Raspberry Pi“ atminties talpa skiriasi nuo kelių šimtų megabaitų, kai tuo tarpu „Arduino“ nesiekia net vieno megabaito. Taip pat “Raspberry Pi” procesius keliasdešimt kartų greitesnis nei „Arduino“. Į „Raspberry Pi“ galima įdiegti „Linux“ operacinę sistemą, tačiau nors tai ir skamba viliojančiai, „Arduino“ yra žymiai paprastesnė ir lengviau išmokstama naudoti platforma. „Arduino“ platforma yra draugiškesnė pradedantiesiems, nes nereikalauja išskirtinių žinių (tokių kaip naudojimusi „Linux“ operacine sistema) ir patys įrenginiai nereikalauja ypatingos priežiūros – prietaisus galima prijungti ir atjungti nuo elektros, be rizikos sugadinti įrenginį. „Arduino“ platforma suteikia daugiau dinamiškumo iš techninės įrangos pusės, o tuo tarpu „Raspberry Pi“ iš programinės įrangos pusės.

				
Raspberry Pi:	Model A+	Model B	Model B+	2, Model B
Price:	$19.99	$39.99	$29.99	$39.99
Availability:	Add to Cart	Add to Cart	Add to Cart	Add to Cart
Quick summary:	Cheapest, smallest single board computer.	The original Raspberry Pi.	More USB and GPIO than the B. Ideal choice for schools	Newest, most advanced Raspberry Pi.
Chip:	Broadcom BCM2835			Broadcom BCM2836
Processor:	ARMv6 single core			ARMv7 quad core
Processor Speed:	700 MHz			900 MHz
Voltage and Power Draw:	600mA @ 5V			650mA @ 5V
GPU:	Dual Core VideoCore IV Multimedia Co-Processor			
Size:	65x56mm	85x56mm		
Memory:	256 MB SDRAM @ 400 MHz	512 MB SDRAM @ 400 MHz		1 GB SDRAM @ 400 MHz
Storage:	Micro SD Card (not included)	SD Card (not included)	Micro SD Card (**included**)	Micro SD Card (not included)
GPIO:	40	26	40	
USB 2.0:	1	2	4	
Ethernet:	None	10/100mb Ethernet RJ45 Jack		
Audio:	Multi-Channel HD Audio over HDMI, Analog Stereo from 3.5mm Headphone Jack			

17 pav. „Raspberry Pi“ modelių palyginimas

17 pav. pateikti naujesni „Raspberry Pi“ modeliai ir jų techninės charakteristikos. Matome, jog visi modeliai turi tą pačia vaizdo plokštę ir garso jungtį. „Model A+” yra šiek tiek mažesnis už kitus ir kainuoja mažiausiai, bet tuo pačiu turi ir mažiausiai atminties, bei visai neturi tinklo plokštės. Naujausias „2, Model B“ turi daugiausiai atminties ir USB jungčių, bei galingesnį procesorių, lyginant su ankstesnėmis versijomis.

3.1.3. „BeagleBone“ platforma

„BeagleBone“ labai artima „Raspberry Pi“ platformai, tačiau yra brangesnė kainos atžvilgiu. „BeagleBone“ įrenginiai turi daugiau jungčių ir yra šiek tiek galingesni, bei jais lengviau pradėti naudotis, kitaip nei „Raspberry Pi“.

	BeagleBoard.org BeagleBone Black	BeagleBoard.org BeagleBone (original)	Seeed Studio BeagleBone Green	SanCloud BeagleBone Enhanced
Processor	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8
Maximum Processor Speed	1GHz	720MHz (1GHz on latest)	1GHz	1GHz
Analog Pins	7	7	7	7
Digital Pins	65 (3.3V)	65 (3.3V)	65 (3.3V)	65 (3.3V)
Memory	512MB DDR3 (800MHz x 16), 2GB (4GB on Rev C) on-board storage using eMMC, microSD card slot	256MB DDR2 (400MHz x 16), microSD card slot	512MB DDR3 (800MHz x 16), 4GB on-board storage using eMMC, microSD card slot	1GB DDR3 (800MHz x 16), 4GB on-board storage using eMMC, microSD card slot
USB	miniUSB 2.0 client port, USB 2.0 host port	miniUSB 2.0 client port, USB 2.0 host port	microUSB 2.0 client port, USB 2.0 host port	miniUSB 2.0 client port, 4 USB 2.0 Ports (2 A-type connectors, 2 on pin headers)
Video	microHDMI, cape add-ons	cape add-ons	cape add-ons	microHDMI, cape add-ons
Audio	microHDMI, cape add-ons	cape add-ons	cape add-ons	microHDMI, cape add-ons
Supported Interfaces	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers, FTDI USB to Serial, JTAG via USB	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers, 2 Grove (I2C, UART)	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers
Sensors	n/a	n/a	n/a	Barometer, Accelerometer, Gyro, Temperature
MSRP	$49	$89	$39	$69

18 pav. „BeagleBone“ modelių palyginimas

Iš 18 pav., rasto [29] šaltinyje, matyti, jog „BeagleBone“ modeliai gali kainuoti iki 90 dolerių, kas yra pakankamai didelė suma, norint tiesiog įsigyti tokį įrenginį pabandymui. Techninės charakteristikos gana panašios, išskyrus atmintį, kuri gali būti nuo kelių šimtų megabaitų iki vieno gigabaito. „Enhanced“ modelis vienintelis savyje turi barometrą, akcelerometrą, giroskopą ir termometrą.

3.1.4. Platformų palyginimas

Apžvelgus kelias populiariausias platformas ir jų siūlomus mikrovaldiklių modelius, galima lengviau palyginti jas kartu. Nors platformos ir gana panašios, tačiau kiekviena turi savo privalumų ir trūkumų, todėl tinkamiausios pasirinkimą lemia kam tiksliai ji bus skirta ir ką norima su ja atlikti.

Name	Arduino Uno	Raspberry Pi	BeagleBone
Model Tested	R3	Model B	Rev A5
Price	$29.95	$35	$89
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"
Processor	ATMega 328	ARM11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD Card)	4GB(microSD)
EEPROM	1KB		
Input Voltage	7-12v	5v	5v
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)
Digital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6		8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Dev IDE	Arduino Tool	IDLE, Scratch, Squeak/Linux	Python, Scratch, Squeak, Cloud9/Linux
Ethernet	N/A	10/100	10/100
USB Master	N/A	2 USB 2.0	1 USB 2.0
Video Out	N/A	HDMI, Composite	N/A
Audio Output	N/A	HDMI, Analog	Analog

19 pav. Pagrindinių platformų modelių palyginimas

Iš 19 pav., rasto [30] šaltinyje, pateiktos informacijos lengviausia pastebėti, jog „Arduino“ savo techninėmis charakteristikomis sunkiai prilygsta „Raspberry Pi“ ir „BeagleBone“ platformoms, kurios yra pakankamai panašios tarpusavyje. Tačiau atitinkamai „Arduino“ yra mažesnis ir šiek tiek pigesnis, bei nereikalauja atminties kortelės jo naudojimui. Taip pat „Arduino“ skirtas pradedantiesiems, todėl jis ir nėra toks galingas, palyginus su kitų platformų modeliais. Nors jis pats savyje neturi nei interneto, nei vaizdo, nei garso jungčių, tačiau yra galimybė papildomai prisidėti norimus komponentus iš nemažos papildomų komponentų (angl. „shields“) aibės, kurią sudaro prie „Arduino“ prijungiami komponentai, tokie kaip „Wi-Fi“ plokštelė ar piršto antspaudo skaitytuvas. Kadangi „Arduino“ platforma labiau skirta pradedantiesiems, tad jos bendruomenė didesnė ir apie pačią „Arduino“ platformą galima rasti kur kas daugiau informacijos ir mokomųjų vaizdo įrašų, nei apie kitas platformas. Naudojimasis ir pritaikomumas techninės įrangos praplėtimui taip pat yra paprastesnis ir lengviau įgyvendinamas, ypač turint omenyje tam skirtus papildomus komponentus.

Atitinkamai „Raspberry Pi“ ir „BeagleBone“ modeliai yra galingesni ir gali būti panaudoti sudėtingoms užduotims atlikti, kurios reikalauja kompiuterio lygio resursų kiekio. Dėl to ir naudojimasis šiomis platformomis yra šiek tiek sudėtingesnis ir reikalauja papildomų žinių norint jas

išnaudoti kaip tikrus kompiuterius ar praplėsti jų funkcionalumą techninę įranga. „BeagleBone“ naudojimas ir konfigūravimas šiek tiek paprastesnis nei „Raspberry Pi“, tačiau ir kaina atitinkamai didesnė. Prie to pačio galima paminėti ir “BeagleBone” paprastumo pranašumą norint prijungti prie jo įvairius sensorius. Tuo pačiu aspektu ne ką mažiau nusileidžia ir “Arduino”, kuris, be kita ko, dar ir gali būti maitinamas iš baterijos, ko negali kitos platformos.

Kadangi projektas bus kuriamas norint išbandyti mikrovaldiklių pritaikomumą autentifikacijai, svarbiausi kriterijai šiuo atveju yra:

- tinkamiausia platforma pradedančiajam;
- naudojimo paprastumas;
- universalumas ir lankstumas.

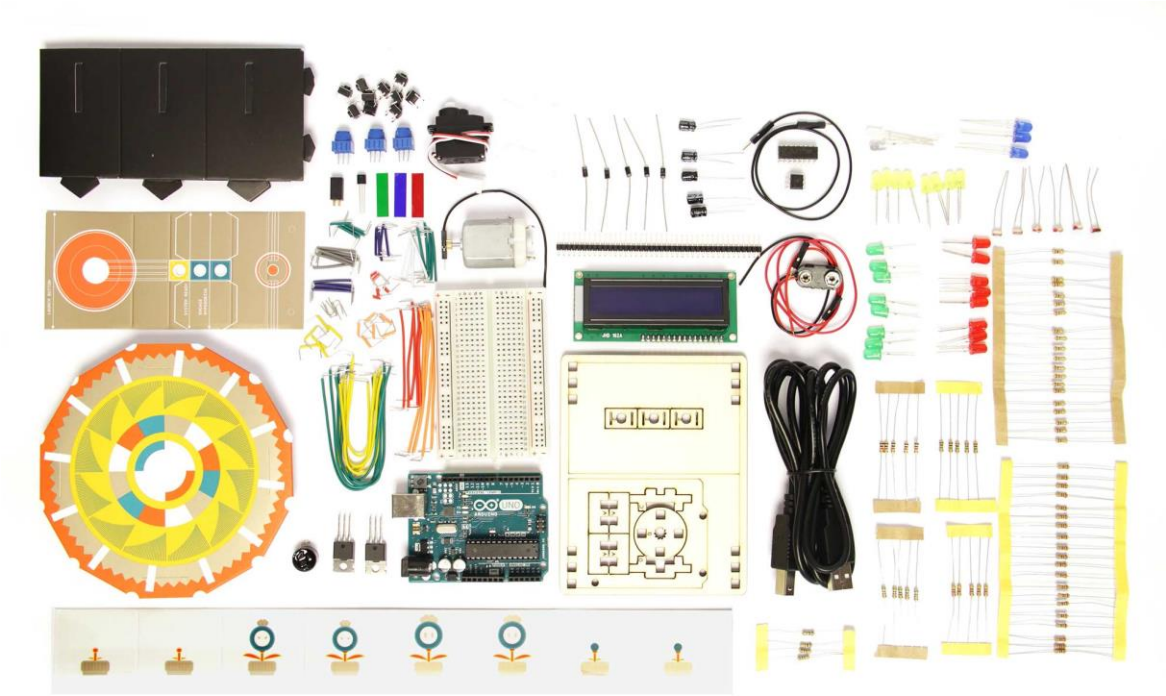
Šiuos kriterijus labiausiai atitinka „Arduino“ dėl savo mažos kainos, didelės naudotojų bendruomenės ir susikoncentravimo į pradedančiuosius. „Arduino“ platforma naudotis yra nesudėtinga, nes ji skirta paprastam naudojimui ir galima rasti daug informacijos internete apie pačią platformą, bei įvairių mokymosi šaltinių. Taip pat, papildomų komponentų (angl. „Shields“) pagalba galima lengvai išplėsti sukurtą sistemą, pridedant LED ekraną informacijos išvedimui ar prijungiant temperatūros sensorių. „Arduino“ suteikia ne mažą modelių pasirinkimą, kas leidžia pasirinkti artimiausią siekiamam tikslui, neišleidžiant daugiau pinigų, nei reikia. Speciali programavimo aplinka leidžia vieno mygtuko paspaudimu perkelti parašytą kodą į, per USB jungtį prijungtą, „Arduino“ ir šis kodas pradedamas vykdyti iš karto. „Arduino“ nereikalauja jokios papildomos priežiūros ar konfigūracijos. Jį galima prijungti tiesiai prie baterijos specialių komponentų pagalba ir nešiojantis kartu su savimi naudoti bet kur, net kur nėra elektros lizdų. Mažesnis modelių dydis taip pat yra vienas iš esminių faktorių ieškant tinkamiausio sprendimo, kuris būtų tinkamas patogiam naudojimui keliaujant.

Nors „Arduino“ modeliai iš techninės pusės nėra galingi, tačiau šios platformos užtenka siekiamam tikslui – paprastos sukurti ir naudoti prototipo platformos pasirinkimas. Pakankamai maža atmintis apribos praplėtimo galimybes, tačiau leis sutaupyti nemažai pinigų. Esant poreikiui, prie modelių galima prijungti specialų komponentą, kuris leistų naudoti SD kortelę kaip atmintį, taip šiek tiek praplečiant atmintį. Kita vertus, platformos yra pakankamai panašios, tad sėkmingai realizavus projektą vienoje, galima lengvai perkelti ją į kitą, turinčią galingesnius, skaičiavimo resursų prasme, modelius.

3.1.5. „Arduino“ platformos modelių palyginimas

Pasirinkus priimtiniausią ir tinkamiausią platformą, lieka klausimas, kokį konkretų modelį pasirinkti? „Arduino“ šiuo atveju siūlo „Uno“ modelį kaip geriausią pradedantiesiems dėl didelio informacijos kiekio internete ir palaikymo. Be to, galima rasti ir keletą komplektų su įvairiais elektroniniais komponentais, kuriuose bus ir pati „Arduino Uno“ plokštelė. Šie komplektai puikiai

tinka susipažinimui tiek su elektronika, tiek su programavimu, nes jų dėka galima kurti elektronines grandines iš įvairių komponentų ir jas programuoti. To dėka neapsiribojama vien tik mikrovaldiklio galimybėmis, bet ir galima išnaudoti kitų elektroninių komponentų galimybes. Didžiausiame komplekte galima rasti specialią mokomąją knygą, kurioje aprašyta, kaip sukurti daugybę skirtingų projektų su skirtingai komplekte pateikiamais komponentais.

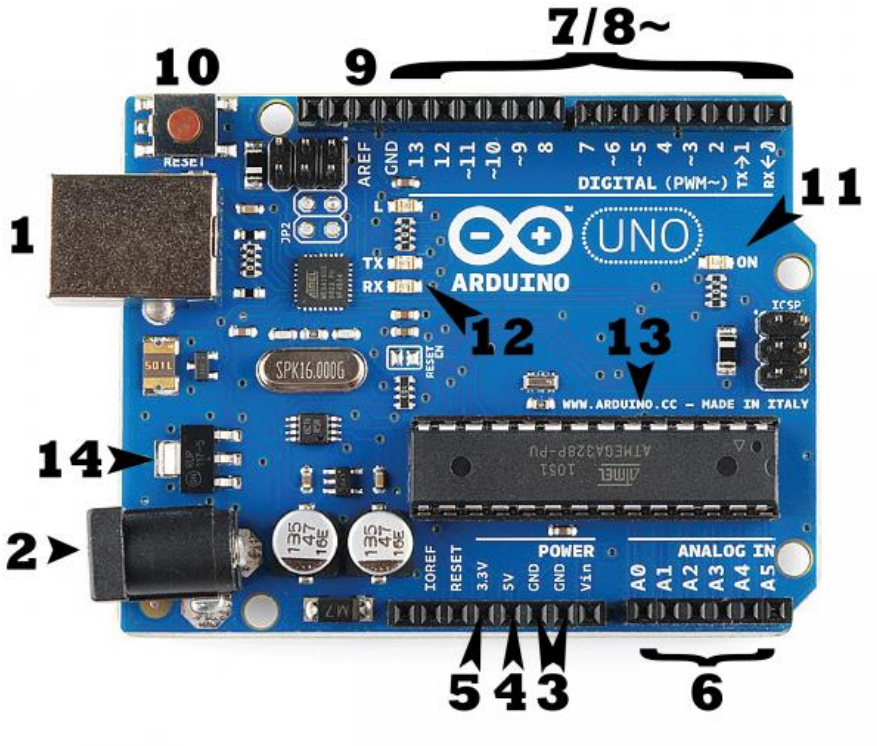


20 pav. „Arduino Starter Kit“ komplektas

20 pav. atvaizduoti komplekto sudėtyje esantys komponentai. Pagrindiniai komponentai yra pats „Arduino Uno“ mikrovaldiklis ir bandomoji lenta (angl. „breadboard“) į kurią jungiami visi kiti komponentai ir jungtys iš mikrovaldiklio. Komplektą sudaro ir tokie komponentai, kaip LED lemputės, įvairių varžų rezistoriai, įvairūs jutikliai ir kiti. Pridedami ir popieriniai piešiniai, kuriais galima papuošti knygoje aprašytus projektus, taip siekiant dar labiau sudominti pradedančiuosius.

3.2. „Arduino Uno“ modelis

3.2.1. „Arduino Uno“ sandara



21 pav. „Arduino Uno“ mikrovaldiklis

21 pav., rastame [31] šaltinyje, matome, kas sudaro ir kaip atrodo pats „Arduino Uno“ mikrovaldiklis. Skaičiumi 1 pažymėta USB jungtis, kuria mikrovaldiklis prijungiamas prie kompiuterio ir per kurią sukurtas programinis kodas įrašomas į mikrovaldiklį. Taip pat, per šią jungtį teikiamas elektros maitinimas mikrovaldikliui. Mikrovaldiklis gali būti maitinamas ir per skaičiumi 2 pažymėtą jungtį, tačiau komplekte nėra laido šiai jungčiai. Skaičiumi 3 pažymėtos įžeminimo įvestys. 4 ir 5 žymi 5 V ir 3,3 V išvestis, kuriomis gali būti maitinama sukurta elektros grandinė. Skaičius 6 žymi analogines įvestis, kurios yra skirtos nuskaityti signalą iš analoginių sensorių ir paversti šį signalą į skaitmeninę reikšmę. 7 ir 8 žymi skaitmenines įvestis. Skaičiumi 9 pažymėta įvestis skirta įtampos viršutinės ribos reguliavimui analoginėse išvestyse. 10 pažymėtas perkrovimo mygtukas, kurį paspaudus perkraunamas mikrovaldiklis ir jame įrašytas kodas pradedamas vykdyti iš naujo. Mikrovaldikliui prijungus maitinimą, užsidega 11 skaičiumi pažymėta LED lemputė. Skaičiumi 12 pažymėtos LED lemputės TX (informacijos perdavimas) ir RX (informacijos gavimas). Paprastai jų mirksėjimas pastebimas, kai į mikrovaldiklį įrašinėjama nauja programa. 13 pažymėta pagrindinė mikrovaldiklio dalis – procesorius. Mikrovaldiklio įtampos reguliatorius, apsaugantis mikrovaldiklį nuo per didelės elektros įtampos (iki 20 V), pažymėtas skaičiumi 14.

3.2.2. „Arduino Uno“ programavimo pagrindai

Norint parašyti programinį kodą, kuris veiktų „Arduino Uno“ mikrovaldiklyje, reikia parsisiųsti programavimo aplinką iš oficialios „Arduino“ svetainės. Programavimui naudojama Arduino programavimo kalba, naudojanti C/C++ funkcijas, kurias galima kviesti. Kompiliuojant parašytą kodą, pirmiausiai sugeneruojami reikalingi kodo fragmentai (funkcijų aprašymai), o tada visas kodas perduodamas C/C++ kompiliatoriui, kuris ir paverčia parašytą programinį kodą į kompiuteriui suprantamą kodą.

Tiek “Arduino” modeliai, tiek ir programinė įranga, naudojama juose, yra atviro kodo, kas leidžia nemokamai ir laisvai pasiekti visą informaciją (su licencijos apribojimais). Dėka to, internete galima rasti nemažai įvairiausių „Arduino“ platforma paremtų modelių ir labai daug bibliotekų, kuriomis pasinaudojus galima atlikti sudėtingus veiksmus žymiai paprasčiau, nes jų nereikia programuoti pačiam. Keletas pagrindinių bibliotekų jau būna įdiegta į „Arduino“ programinę įrangą, jog būtų galima iš karto pradėti kurti nesudėtingas programas ir nereikėtų aiškintis kaip pridėti papildomas bibliotekas norint paleisti pirmąją programą.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Done compiling.

Sketch uses 450 bytes (1%) of program storage space. Maximum is 32,256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2,039 bytes for local variables. Maximum is 2,048 bytes.
```

22 pav. Sukompiliuota tuščia „Arduino“ programa

22 pav. pateikta pati paprasčiausia galima programa. Dvi pagrindinės dalys sudarančios visą kodą yra „setup“ ir „loop“ funkcijos. Prieš jas esantis kodas gali būti skirtas bibliotekų prijungimui arba konstantų ir paprastų kintamųjų aprašymui. „Setup“ funkcijoje aprašomas programos konfigūracinis kodas, kuris bus suvykdomas tik vieną kartą programos veikimo pradžioje. Čia paprastai talpinama informacija susijusi su mikrovaldiklio jungčių konfigūracija, nurodant kokia jungtis (pagal jos numerį) išves ar priims informaciją/signalą. Taip pat, gali būti ir nurodomas pradinis teksto išvedimas į monitorių, siekiant pasižymėti, kada prasidėjo programa ar atlikti konfigūracinius veiksmus naudojant tam tikrus fizinius komponentus, prijungtus prie mikrovaldiklio. „Loop“ funkcijoje pateikiamas visas kitas kodas, kuris bus vykdomas po „setup“ funkcijos vis iš naujo, kol „Arduino“ maitinamas elektra.

Norint, jog parašytas kodas būtų vykdomas mikrovaldiklyje, pirmiausia būtina sukompiliuoti kodą. Kompiliavimo metu patikrinama ar kodas teisingai suformuotas ir ar nėra padaryta programavimo kalbos sintaksės klaidų. Jei klaidų nerandama, kodas sėkmingai paverčiamas į kompiuteriui suprantamą instrukcijų seką ir programavimo aplinkos lango apačioje matome, kiek ir kokio tipo atminties užima sukurtas programinis kodas. Kadangi „Arduino“ platformos modeliai yra ganėtinai primityvūs ir neturi daug skaičiavimo resursų, bei atminties, nes skirti paprastam ir lengvam naudojimui, informacija apie programos užimamą atmintį naudinga planuojant programinio kodo plėtimą.

Keletą pagrindinių „Arduino Uno“ komandų sudaro:

- serial.begin(9600) – funkcija, skirta pradėti komunikacijai tarp mikrovaldiklio ir kompiuterio (skaičius nurodo perduodamos informacijos kiekį bitais per sekundę) per serijinį prievadą;
- serial.print() – išveda tarp skliaustų pateiktą informaciją (nurodyto kintamojo saugomą reikšmę arba tekstą kabutėse) į serijinį prievadą. Informacija pateikiama žmogui suprantamu ASCII formatu.
- analogRead(pin) – skaito elektrinės įtampos reikšmę iš nurodytos analoginės jungties esančios ant mikrovaldiklio. „Arduino Uno“ turi 6 tokias jungtis. Mikrovaldiklio viduje esantis ADC (angl. „analog-to-digital converter“) verčia gaunamą įtampą voltais (nuo 0V iki 5V) į proporcingas skaitines reikšmes nuo 0 iki 1023;
- analogWrite(pin, value) – rašo nurodytą reikšmę į pasirinktą analoginę jungtį (reikšmių intervalas 0-255). Kviečiant šią funkciją, neprivaloma prieš tai nurodyti naudojamos jungties rėžimo;
- pinMode(pin, mode) – nustato nurodytą mikrovaldiklio skaitmeninę jungtį skaitymo arba rašymo rėžimui. Visų jungčių numatytoji reikšmė yra skaitymo rėžimas, tad naudojant jas kaip tokias, neprivaloma tai nurodyti su šia funkcija;
- map(value, 0, 1023, 0, 255) – leidžia konvertuoti vieno intervalo reikšmę į atitinkamą reikšmę kitame intervale. Pavyzdyje pateikta, kaip reikšmė, esanti intervale 0-1023 konvertuojama į reikšmę intervale 0-255;
- digitalRead(pin) – nuskaito gaunamą reikšmę iš nurodytos skaitmeninės jungties kaip HIGH, jei teka elektros srovė ir kaip LOW, jei elektros srovės nėra.
- digitalWrite(pin, value) – esant nustatytam jungties rėžimui kaip OUTPUT ir nurodžius šią jungtį kaip parametrą, antrojo parametro reikšmė nurodo ar bus paduodama elektros stovė (HIGH - 5 voltai) ar ne (Low – 0 voltų).

3.3. Autentifikacijos modelių prototipų realizacijos su „Arduino Uno“

Siekiant paprastumo ir aiškumo, bei atsižvelgiant į „Arduino“ techninių charakteristikų apribojimus, bus pasirinkti paprastesni ankščiau apžvelgti autentifikacijos modeliai ir jų galimos realizacijos variantai. Tai leis nenukrypti į žemą techninį lygį, bet išmėginti ir pateikti modelio veikimą abstrakčiame lygyje realizuojant tik esmines, su „Arduino“ veikimu susijusias, dalis. Esant poreikiui, „Arduino“ leidžia įvairaus lygio ir sudėtingumo pasirinktų modelių realizacijas, žymiai sudėtingesnes, nei bus pateikiamos toliau.

Komunikacijai tarp kompiuterio ir „Arduino Uno“ puikiai tinka „Processing“ programavimo kalba. Informacijos apsikeitimui naudojamas tas pats mechanizmas, kaip ir su „Arduino“ grafine sąsaja, tad šiuo atveju apsiribojame ja. Dėka to, realizacijos etape papildomai nereikės programuoti paslaugos kliento imitacinės paprogramės „Processing“ kalba, kuri tik priiminėtų „Arduino“ pranešimus ir išvedinėtų informaciją, ką galima pasiekti ir panaudojus jau pilnai funkcionuojančią „Arduino“ serijinio kanalo grafinę sąsają, be papildomų pastangų.

3.3.1. Fizinio rakto prototipas

Vienas iš paprasčiausių ir lengviausiai įgyvendinamų modelių – fizinio rakto imitacija autentifikacijai. Mikrovaldiklio atmintyje saugoma iš anksto įprogramuota tik tam konkrečiam mikrovaldikliui būdinga reikšmė, pagal kurią galima identifikuoti, jog tai yra konkretaus žmogaus mikrovaldiklis. Prijungus mikrovaldiklį prie kompiuterio, mikrovaldiklis laukia, kol bus atsiųsta užklausa. Šiuo atveju teisinga užklausa laikomas skaičius 1 (vienas) simbolis. Mikrovaldikliui perskaičius šią reikšmę iš serijinio prievado, jis grąžins slaptą PIN kodą. Visos kitos perduodamos reikšmės (užklausos) bus ignoruojamos ir mikrovaldiklis negrąžins nieko.

```
1  int pinCode = 1234;
2
3  void setup() {
4    Serial.begin(9600);
5  }
6
7  void loop() {
8    if(Serial.available() > 0) {
9      char i = Serial.read();
10     if(i == '1') {
11       Serial.println(pinCode);
12     }
13   }
14 }
```

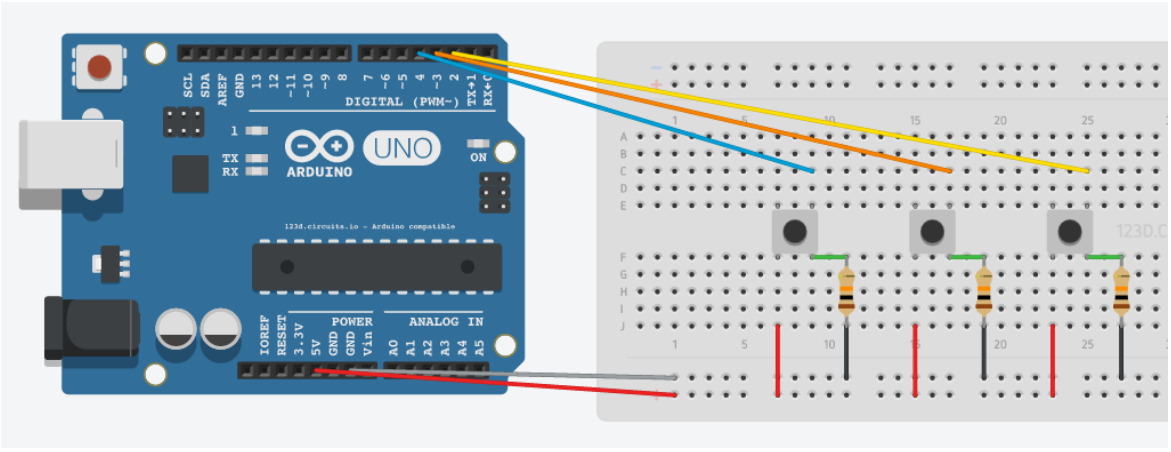
23 pav. Fizinio rakto modelio prototipo kodas

23 pav. pateiktas kodas, kaip gali atrodyti realizacija pačiame „Arduino Uno“. Pirmoje eilutėje matome iš anksto įrašytą specifinę reikšmę (gali būti ne tik skaičiai), kuri saugoma kintamajame. Funkcijoje „setup“, kuri vykdoma tik vieną kartą, kai paleidžiama programa, nurodome kokiu greičiu bus vykdoma komunikacija su kompiuteriu per serijinį prievadą. Toliau, „loop“ funkcijoje, kuri kartojama nuolat, kiekvieną iteraciją patikriname, ar jau atsirado naujos informacijos komunikacijos kanale. Jei taip, nuskaitome rastą informaciją kaip simbolį ir jei jo reikšmė lygi vienetui – išvedame PIN kodą. Iš karto matome didelį potencialą sukurti papildomas apsaugos priemones, vien tik panaudojus sudėtingesnę atpažinimo reikšmę (užklausą) vietoje paprasto simbolio.

3.3.2. Kodų kortelės prototipas

Kodų kortelės imitacija panaši į fizinio rakto, tik šiuo atveju mikrovaldiklyje turime įprogramuotą slaptų kodų sąrašą. Jungiantis prie paslaugos, paslaugos sistemos paprašo atskleisti vieną nurodytą slaptą kodą, pagal kurį galėtų papildomai identifikuoti, jog tikrai tas vartotojas bando prisijungti. Galimas supaprastintas realizacijos variantas, kada mikrovaldikliui tiesiog perduodame norimo gauti slapto kodo indeksą ir jis jį grąžina.

Įvairumo dėlei (tuo pačiu ir saugumo), atsisakysime automatinio užklausos persiuntimo mikrovaldikliu ir įvesime norimos gauti reikšmės indeksą pasinaudojant sukurta fizine schema, mygtukų pagalba. Kadangi esame apriboti bandomosios lentos dydžiu, neprijungsime atskirų mygtukų kiekvienos reikšmės indeksui pasirinkti, bet panaudosime paprastą algoritmą indekso išskaičiavimui iš kelių mygtukų paspaudimo. Grąžinamos reikšmės indeksas bus nustatomas pagal paspaustų mygtukų reikšmių sumą. Prijungti iš eilės trys mygtukai, kurių kiekvienas turi savo paspaudimo vertę (2, 3 ir 4). Pvz., nuspaudus pirmąjį mygtuką, išvedama 2-uoju indeksu sąraše saugoma reikšmė. Kitokiu atveju, pvz., nuspaudus pirmąjį ir trečiąjį mygtukus – 6-uoju indeksu (2 už pirmąjį mygtuką + 4 už trečiąjį mygtuką) sąraše saugoma reikšmė.



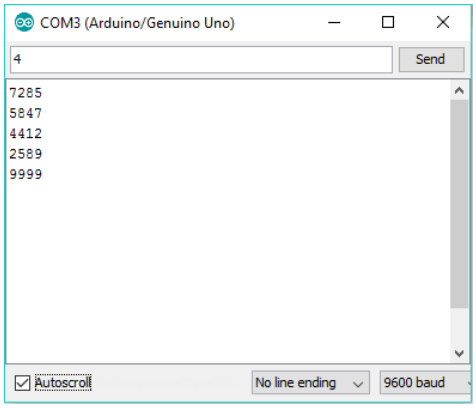
24 pav. Kodų kortelės modelio prototipo schema

24 pav. matyti, jog fizinę schemą sudaro 3 lygiagrečiai sujungti mygtukai ir prie kiekvieno prijungtos jungtys (geltona, oranžinė ir mėlyna) iš mikrovaldiklio, kurios tikrina ar mygtukas paspaustas (pagal tai, ar elektros grandinė uždaryta - ar ja teka elektros srovė). Raudonos spalvos jungtimis žymimas elektros grandinės „pliuso“ pusė, o juodos – „minuso“. Perėjimas į įžeminimą pažymėtas pilka spalva dėl matomumo.

```
int pinCodes[] = {594, 1485, 2589, 3879, 4412, 5847, 6895, 7285, 8771, 9999};
int firstButtonPin = 2;
int secondButtonPin = 5;
int thirdButtonPin = 8;
int firstButton, secondButton, thirdButton;

void setup() {
  Serial.begin(9600);
  pinMode(firstButtonPin, INPUT);
  pinMode(secondButtonPin, INPUT);
  pinMode(thirdButtonPin, INPUT);
}

void loop() {
  if(Serial.available() > 0) {
    char i = Serial.read();
    if(i == '4') {
      int indexSum = 0;
      do {
        firstButton = digitalRead(firstButtonPin);
        secondButton = digitalRead(secondButtonPin);
        thirdButton = digitalRead(thirdButtonPin);
        delay(1000);
        indexSum = firstButton * 2 + secondButton * 3 + thirdButton * 4;
      } while (indexSum < 1);
      Serial.println(pinCodes[indexSum]);
    }
  }
}
```

25 pav. Kodų kortelės modelio prototipo kodas

25 pav. pateiktas prototipo programinis kodas, kurio pradžioje aprašomi įprogramuoti kodai autentifikacijai ir jungtys, prie kurių prijungti mygtukai, bei paruošiami kintamieji mygtukų reikšmių (ar jie yra paspausti) saugojimui. „setup“ funkcijoje užmezgama komunikacija su kompiuteriu ir nurodomi aukščiau aprašytų jungčių rėžimai. „Loop“ funkcijoje, kiekvienoje iteracijoje patikrinama ar yra perduodamų duomenų. Jei taip, patikrinama ar gauti duomenys atitinka šio modelio identifikacinę reikšmę (simbolį 4). Jei atitinka, laukiama, kol vartotojas paspaus bent vieną mygtuką ir bus gauta didesnė už nulį suminė indekso reikšmė (nuspaudus mygtuką, jo kintamojo reikšmė pasikeičia iš 0 į 1 ir padauginama iš mygtuko koeficiento – 2, 3 arba 4). Tai pasiekiama nuskaitant visų mygtukų reikšmes ir jei nė vienas iš jų nėra paspaustas, mygtukų reikšmių suma lygi 0. Tokiu atveju iš naujo bandoma nuskaityti mygtukų reikšmes, kol bent vienas paspaudžiamas ir gaunamas didesnis už 0 suminis indeksas. Prieš sudedant mygtukų reikšmes, padaroma sekundės pauzė, jog užtektų laiko visoms mygtukų reikšmėms nuskaityti ir priskirti kintamiesiems.



26 pav. Kodų kortelės modelio prototipo rezultatai

Iš 26 pav. išvestų kodų matome, kokia eilės tvarka buvo paspausti mygtukai. Kiekvienos operacijos pradžioje suvedamas skaičius 4, jog mikrovaldiklis atpažintų, jog norime kodų kortelės autentifikacijos metodo. Pirmąjį kartą buvo paspausta antrasis ir trečiasis mygtukai (antro mygtuko koeficientas 3 + ketvirto mygtuko koeficientas 4) ir išvesta 7 indekso reikšmė iš kodų sąrašo (pirmas kodo skaičius skaitmuo specialiai pakeistas, jog nurodytų indeksą). Antrąjį kartą – pirmasis ir antrasis mygtukai (pirmo mygtuko koeficientas 2 + antro mygtuko koeficientas 3), kurių reikšmių suma yra 5. Trečiąjį kartą – tik trečiasis mygtukas (mygtuko koeficientas – 4), kurio reikšmė lygi 4. Ketvirtąjį kartą – tik pirmasis mygtukas (koeficientas 2), kurio reikšmė lygi 2. Paskutinį kartą buvo paspausti visi 3 mygtukai (koeficientai atitinkamai 2, 3 ir 4), kurių reikšmių suma lygi 9.

3.3.3. Kriptografinio rakto prototipas

Kriptografinio rakto prototipo realizacijai panaudosime oficialiame „Arduino“ forume rastą AES (angl. „Advanced Encryption Standard“) biblioteką, kuri sukurta vartotojo, slapyvardžiu „MarkT“ [31]. Ši biblioteka palaiko 128, 192 ir 256 bitų rakto ilgius bei EBC (angl. „Electronic Codebook“) ir CBC (angl. „Cipher Block Chaining“) rėžimus.

```
byte set_key (byte key[], int keylen) ;
void clean () ;  // delete key schedule after use
void copy_n_bytes (byte * dest, byte * src, byte n) ;

byte encrypt (byte plain [N_BLOCK], byte cipher [N_BLOCK]) ;
byte cbc_encrypt (byte * plain, byte * cipher, int n_block, byte iv [N_BLOCK]) ;

byte decrypt (byte cipher [N_BLOCK], byte plain [N_BLOCK]) ;
byte cbc_decrypt (byte * cipher, byte * plain, int n_block, byte iv [N_BLOCK]) ;
```

27 pav. „MarkT“ AES bibliotekos pagrindinės funkcijos

27 pav. pateiktos pagrindinės funkcijos yra *set_key*, kurios pagalba nurodomas slaptas šifravimo raktas ir jo ilgis. Trumpam tekstui užšifruoti ir iššifruoti skirtos *encrypt* ir *decrypt* funkcijos. Šios funkcijos priima kintamųjų parametrus, kurie saugo tekstus užšifravimui ir iššifravimui. Panašiai veikia ir *cbc_encrypt* / *cbc_decrypt* funkcijos, kurios skirtos saugiems veiksmams su ilgais tekstais. Abejos funkcijų poros išskaido ilgą tekstą į vienodo dydžio mažesnius blokus.

Pirmoji funkcijų pora kiekvienam blokui užšifruoti naudoją tą patį slaptą raktą, tad jei sutampa kelių pranešimų tekstai – sutaps ir jų užšifruotas tekstas. Tai nėra gerai iš saugumo perspektyvos, nes piktavalis perėmęs užšifruotus pranešimus, gali identifikuoti, jog buvo siunčiamas toks pats arba panašus pranešimas. Antroji šifravimo / iššifravimo funkcijų pora skirta išvengti būtent tokių šifruoto teksto sutapimų, papildant šifravimo procesą didesniu kiekiu atsitiktinių duomenų. Būtent todėl pirmasis blokas šifruojamas pasinaudojant pradiniu vektoriumi (angl. „Inicialization Vector“), kuris sugeneruojamas atsitiktinai, ir slapto rakto kombinacija XOR logine operacija. Sekantys blokai šifruojami ta pačia XOR logine operacija, tik vietoje pradinio vektoriaus, toliau naudojamas ankstesnio bloko užšifruotas pranešimas. Kiekvienam pranešimui keičiant pradinį vektorių, net ir siunčiant tą patį pranešimą gaunamas skirtingas šifruotas pranešimas. Nors taip užtikrinamas didesnis saugumas, tačiau kiekvieno sekančio bloko šifravimas tampa priklausomu nuo ankstesniojo bloko. Tai reiškia, jog prarandama lygiagretaus šifravimo galimybė, kada keli skirtingi blokai gali būti šifruojami tuo pačiu metu, nelaukiant, kol kuris nors kitas blokas bus užšifruotas ankščiau.

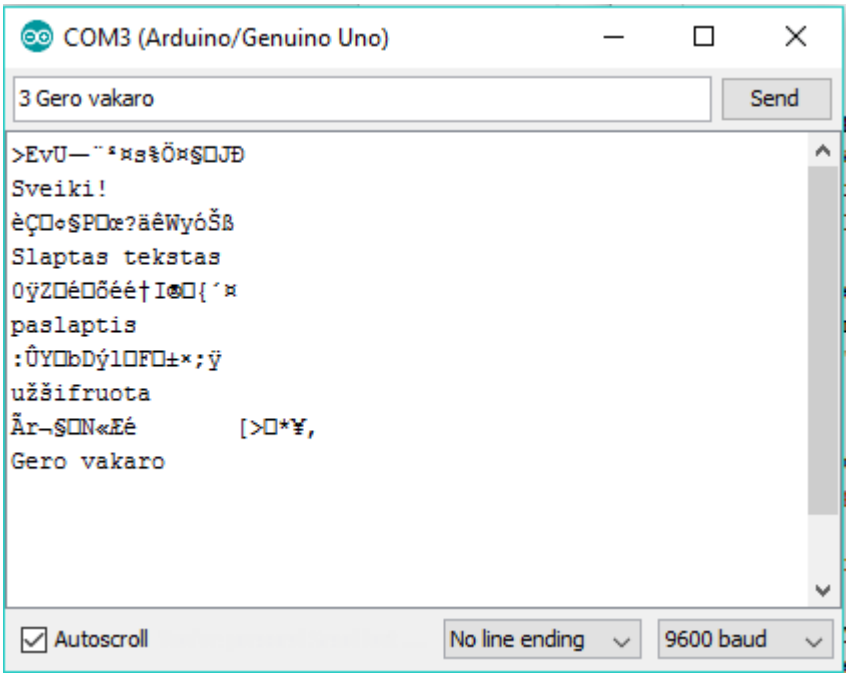
Žemiau pateikiamas sukurtos „Arduino“ programos kodas, kuriuo pasinaudojus užšifruojamas ir iššifruojamas (norint įsitikinti programos tinkamu veikimu) norimas „slaptas“ tekstas. Šio metodo pasirinkimui, į serijinio prievado grafinę sąsają pirmiausia įvedame skaičiaus 3 simbolį, jog mikrovaldiklis žinotų, jog tikrai šį metodą norime panaudoti. Po identifikacinio metodo simbolio paliekame tarpą ir įvedame norimą užšifruoti tekstą. Paprastumo dėlei, neskaidysime teksto į blokus ir šifruosime tik vieną bloką, todėl šifruojamo teksto maksimalus dydis bus tik 15 simbolių.

```
#include <AES.h>

#define KEYLENGTH 256

AES aes;

void setup() {
  Serial.begin(9600);
  byte key[KEYLENGTH] = "Ypač slaptas raktas";

  if (aes.set_key(key, KEYLENGTH) != 0)
    Serial.println("Failed to set key");
}

void loop() {
  if(Serial.available() > 0) {
    String input = Serial.readString();
    char type = input[0];

    if(type == '3') {
      String message = input.substring(2);
      if(message.length() > 0 && message.length() < 16) {
        Serial.println(encrypt(message));
      } else {
        Serial.println("Turi buti maziau nei 16 simboliu.");
      }
    }
  }
}
```

```
char* encrypt(String message) {
  byte messageInBytes[N_BLOCK] = "";
  for (int i = 0; i < N_BLOCK; i++)
    messageInBytes[i] = message[i];

  byte cipher[N_BLOCK];
  if (aes.encrypt (messageInBytes, cipher) != 0)
    return "Užšifruoti nepavyko.";

  for(int i = 0; i < sizeof(cipher); i++) {
    char asc2char = cipher[i];
    Serial.print(asc2char);
  }
  Serial.println();

  byte decryptedInBytes[N_BLOCK];
  if (aes.decrypt(cipher, decryptedInBytes))
    return "Iššifruoti nepavyko.";

  char decryptedMessage[sizeof(message)-1];
  for (int i = 0; i < N_BLOCK; i++)
    decryptedMessage[i] = decryptedInBytes[i];

  return decryptedMessage;
}
```

28 pav. Kriptografinio rakto modelio prototipo kodas

Iš 28 pav. pateikto programinio kodo matome, jog programos pradžioje prijungiame šifravimui naudojamą AES biblioteką ir nustatome naudojamo slapto rakto ilgį. „Setup“ funkcijoje nustatome kokiu greičiu bendrausime su „Arduino Uno“ ir nurodę slaptą raktą, jį nustatome kaip naudojamą šifravimui AES bibliotekai. Tolimesniame cikle, kas 5 sekundes bandome nuskaityti per grafinę sąsają įvestą tekstą. Jei kažkas buvo įvesta, verčiame įvestą tekstą į baitų seką ir perduodame ją užšifravimo funkcijai. Jei šifravimo metu įvyksta klaida – parodome tai. Priešingu atveju bandome iššifruoti užšifruotą pranešimą. Iššifruotą pranešimą gauname baitų seka, todėl papildomai verčiame kiekvieną baitą į simbolį, jog galėtume perskaityti žmogui suprantama kalba iššifruotą reikšmę. Ciklo pabaigoje palaukiame 5 sekundes ir bandome nuskaityti ar per grafinę sąsają gavome naują pranešimą užšifravimui.



29 pav. Kriptografinio rakto prototipo panaudojimo pavyzdys

29 pav. pateikta grafinė vartotojo sąsaja, per kurią buvo vykdoma komunikacija su Arduino mikrovaldikliu. Matome, jog trumpus pranešimus pavyko sėkmingai užšifruoti ir iššifruoti pasinaudojant raktu. Taip pat parodomo ir šifruoto pranešimo reikšmė, norint įsitikinti, jog tekstas tikrai buvo užšifruotas.

3.3.4. Jungtinis prototipas

Siekiant sukurti universalų mechanizmą, pasinaudojant vienu „Arduino“ įrenginiu, apjungiame aukščiau aprašytus prototipus į vieną jungtinį. Šis jungtinis prototipas leis autentifikuotis visais trimis modeliais pasirinktinai. Prie galimų autentifikacijos būdų pridėtas ir tyrimui realizuotas kodų kortelės automatinis modelis, kuris leidžia automatiškai gauti informaciją ir ją išsiųsti, nereikalaujant vartotojo įsikišimo, kaip šio modelio rankinio varianto realizacijoje su mygtukais. Toliau pateikiamas paveiksliukas su kodu ir jo aprašymu.

```
#include <AES.h>

#define KEYLENGTH 256

AES aes;

char physicalKeyType = '1';
char pinCodesType = '2';
char cryptographyType = '3';
char pinCodesManualType = '4';

int firstButtonPin = 2;
int secondButtonPin = 5;
int thirdButtonPin = 8;

int firstButton, secondButton, thirdButton;

int secretCode = 1234;
int pinCodes[] = {594, 1485, 2589, 3879, 4412,
                  5847, 6895, 7285, 8771, 9999};
int pinCodesSize = sizeof(pinCodes) / sizeof(int);

char* encrypt(String message);

void setup() {
  Serial.begin(9600);
  byte key[KEYLENGTH] = "Ypač slaptas raktas";

  if (aes.set_key(key, KEYLENGTH) != 0)
    Serial.println("Failed to set key");

    pinMode(firstButtonPin, INPUT);
    pinMode(secondButtonPin, INPUT);
    pinMode(thirdButtonPin, INPUT);
}

void loop() {
  authenticate();
}

void authenticate() {
  if(Serial.available() > 0) {
    String input = Serial.readString();
    char type = input[0];

    if(type == physicalKeyType) {
      Serial.println(secretCode);
    } else if(type == pinCodesType) {
      int index = input[2] - '0';
      if (index < pinCodesSize && index >= 0) {
        Serial.println(pinCodes[index]);
      }
    } else if(type == cryptographyType) {
      String message = input.substring(2);
      if(message.length() > 0 && message.length() < 16) {
        Serial.println(encrypt(message));
      } else {
        Serial.println("Ne daugiau 15 simboliu!");
      }
    } else if(type == pinCodesManualType) {
      Serial.println(getPinCodeByButtonPress());
    } else {
      Serial.print("Bad type: ");
      Serial.println(type);
    }
  }
}

char* encrypt(String message) {
  byte messageInBytes[N_BLOCK] = "";
    for (int i = 0; i < N_BLOCK; i++)
      messageInBytes[i] = message[i];

    byte cipher[N_BLOCK];
    if (aes.encrypt (messageInBytes, cipher) != 0)
      return "Užšifruoti nepavyko.";

    for(int i = 0; i < sizeof(cipher); i++) {
      char asc2char = cipher[i];
      Serial.print(asc2char);
    }
    Serial.println();

    byte decryptedInBytes[N_BLOCK];
    if (aes.decrypt(cipher, decryptedInBytes))
      return "Iššifruoti nepavyko.";

    char decryptedMessage[sizeof(message)-1];
    for (int i = 0; i < N_BLOCK; i++)
      decryptedMessage[i] = decryptedInBytes[i];

    return decryptedMessage;
}

int getPinCodeByButtonPress() {
  int indexSum = 0;
  do {
    firstButton = digitalRead(firstButtonPin) * 2;
    secondButton = digitalRead(secondButtonPin) * 3;
    thirdButton = digitalRead(thirdButtonPin) * 4;
    delay(1000);
    indexSum = firstButton + secondButton + thirdButton;
  } while (indexSum < 1);
  return pinCodes[indexSum];
}
```

30 pav. Jungtinio prototipo programinis kodas

30 pav. didžioji dalis pateikto kodo sudaryta iš ankščiau aprašytų modelių prototipo realizacijų programinio kodo fragmentų. Siekiant aiškumo, kiekvieno modelio identifikaciniai simboliai pateikti programos viršuje (šiuo atveju, skaičių simboliai 1, 2, 3 ir 4). Šie identifikaciniai simboliai naudojami atpažinti norimą panaudoti autentifikacijos metodą.

„Arduino“ nuolat tikrina ar nėra naujos informacijos komunikacijos kanale. Jei yra, perskaito visą informaciją iš kanalo ir ją išsaugo į kintamąjį. Remiantis perskaitytos informacijos pirmuoju simboliu, kuris laikomas autentifikacijos modelio pasirinkimo identifikaciniu simboliu, parenkamas autentifikacijos modelis. Priklausomai nuo pasirinkto modelio, išsaugota informacija iš komunikacijos kanalo, gali būti panaudota toliau. Kodų kortelės modelio automatizuotos realizacijos atveju, trečiasis simbolis iš gautos informacijos laikomas norimo gauti PIN kodo indeksu. Kriptografinio rakto realizacijos atveju – viskas, kas seka po antrojo simbolio, laikoma slaptu pranešimu, kuris bus užšifruojamas.

Gaunamai informacijai taikomi tam tikri apribojimai. Pavyzdžiui, kodų kortelės modelio realizacijos dalyje gautas trečiasis simbolis, kuris identifikuota PIN kodo indeksą, bandomas versti į skaitinę reikšmę. Jei pavyksta, papildomai patikrinama ar gauta reikšmė nėra neigiama arba didesnė, nei kad saugoma PIN kodų reikšmių sąraše. Kriptografinio rakto modelio prototipo atveju, šifruojamo teksto ilgis negali būti didesnis nei 15 simbolių, nes pasirinkta šifravimo realizacija nėra skirta dideliam informacijos kiekiui šifruoti.

Sėkmingai komunikacijai su „Arduino“ jungtiniu prototipu būtina laikytis tam tikrų taisyklių, formuojant siunčiamą informaciją. Pavyzdžiui, fizinio rakto ir kodų kortelės rankinio naudojimo realizacijos ignoruoja po pirmojo simbolio (kuris identifikuoja modelį) pateiktą informaciją. Kodų kortelės modelio automatizuotos realizacijos atveju ignoruojamas antrasis perduodamos informacijos simbolis ir nuskaitomas trečiasis. Visa kita informacija, einanti po trečiojo simbolio taip pat ignoruojama. Kriptografinio rakto modelio realizacijos atveju ignoruojamas tik antrasis simbolis, tačiau, po antrojo simbolio esantis tekstas neturėtų būti ilgas, dėl ankščiau paminėtos priežasties. Jei pirmasis, gautos informacijos, simbolis neatitinka nė vienos įprogramuotos identifikacinės modelio reikšmės, parodomas klaidos pranešimas, jog nurodytas blogas identifikacinis modelio tipas.

Realizuotas jungtinis prototipas veikia pasirinkus bet kurį norimą modelį ir visada grąžina teisingą atsakymą, tad galima daryti išvadą, jog sėkmingai pavyko sukurti mechanizmą, kuris leidžia autentifikuotis daugiau nei vienu metodu, viename fiziniame įrenginyje. Modelių realizacijos nebuvo orientuotos į maksimalų saugumo užtikrinimą, tačiau jų sėkmingas veikimas suteikia vilties, jog „Arduino“ sugebėtų apdoroti ir sudėtingesnius autentifikavimo modelius ir jų įvairias realizacijas.

4. KELIŲ FAKTORIŲ AUTENTIFIKAVIMO PROJEKTO TYRIMAS

4.1. Pasiruošimas modelių greitaveikos tyrimui

Modelių prototipų greitaveikai ištirti reikalinga tarpinė „stotelė“, kuri imituotų gyvenimišką situaciją, kada vartotojas, pavyzdžiui, prisijungia prie sistemos pirmu žingsniu ir sekančiame žingsnyje reikalinga mikrovaldikliu paremta autentifikacija. Šiai galinei komunikacijai su mikrovaldikliu panaudosime „Processing“ programavimo aplinką, kuri puikiai tinka informacijos apsikeitimui su „Arduino“. “Processing” programavimas beveik identiškas “Arduino”, todėl nereikia nieko papildomai mokytis ir galioja tie patys veikimo principai. Pagrindines komandas, kaip ir “Arduino”, sudaro komunikacijos kanalo sukonfigūravimas ir duomenų įrašymas ar nuskaitymas iš jo. Visas kitas programinis kodas priklauso nuo pasirinkto sprendimo realizacijos. Pasirinktos modelių prototipų realizacijos gali neužtikrinti geriausio galimo veikimo greičio ar optimalaus veikimo, tad tyrimas pilnai neatspindės visų realizacijos galimybių. Greičio faktorių gali įtakoti tiek pasirinktas komunikacijos metodas, tiek ir taikomų apsaugų kiekis, tad tyrimui bus naudojamos paprasčiausios technologijos, imituojant paprasto vartotojo gebėjimus.

Visą pagrindinę informaciją apie „Processing“ programavimo aplinką galima rasti šios platformos kūrėjų puslapyje [33]. Jame galima rasti tiek tekstinio formato pamokų, tiek galimų naudoti funkcijų dokumentaciją. Programavimo aplinkos diegimas ypač paprastas, užtenka iš internetinio puslapio parsisiųsti suarchyvuotą katalogą, kurį išarchyvavus ir atsidarius, paleidžiamas viduje esantis „processing.exe“ failas.

Skirtingai nuo „Arduino“ programavimo aplinkos, „Processing“ aplinka leidžia išvesti informaciją į tekstinę sąsają ir leidžia įjungti derinimo rėžimą. Šis funkcionalumas labai padeda tiek pradedantiesiems, tiek pažengusiems, kuriant programas, nes leidžia lengvai pasižiūrėti eilutės tikslumu kur kokia informacija gaunama ir perduodama, taip sutaupant daug laiko, nei aiškinantis kodėl galutinis rezultatas gaunamas blogas.

4.2. Fizinio rakto modelio tyrimas

Norint ištirti šio modelio veikimą, “Processing” programoje pasiruošiame komunikacijai tą patį informacijos perdavimo kanalą, kaip ir “Arduino”, bei realizuojame PIN kodo metodo pasirinkimo užklausą (skaičiaus vienas) siuntimą šiuo kanalu. Po autentifikacijos prašymo PIN kodo metodu išsiuntimo, laukiame, kol informacijos kanale atsiras naujos informacijos. Naujai informacijai atsiradus, perskaitome ją ir patikriname ar gauta informacija atitinka tikrąjį žinomą PIN kodą. Kadangi “Arduino” kiekvieną kartą spausdina PIN kodą naujoje eilutėje, papildomai praleidžiame naujos eilutės simbolį gautame pranešime (pasinaudodami komanda “trim”).

```
import processing.serial.*;

Serial port;
String s;

long totalTime;
long failureCounter;
int delayTime = 0;
long repeat = 10000;

void setup() {
  port = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  delay(1000);
  totalTime = 0;
  failureCounter = 0;
  getPinCode(repeat);
  print("Run time: " + totalTime + " ms. ");
  print("Delay time: " + delayTime + " ms. ");
  println("Failed " + failureCounter + " times.");
  delayTime++;
}

void getPinCode(long times)
  for (int i=0; i<times;) {
    int start = millis();
    port.write('1');
    delay(delayTime);
    if (port.available() > 0) {
        s = trim(port.readString());
        int finish = millis();
        i++;
        totalTime += (finish - start);
        if (!s.equals("1234")) {
          failureCounter++;
        }
    }
  }
}
```

31 pav. Fizinio rakto autentifikacijos imitacija “Processing” aplinkoje

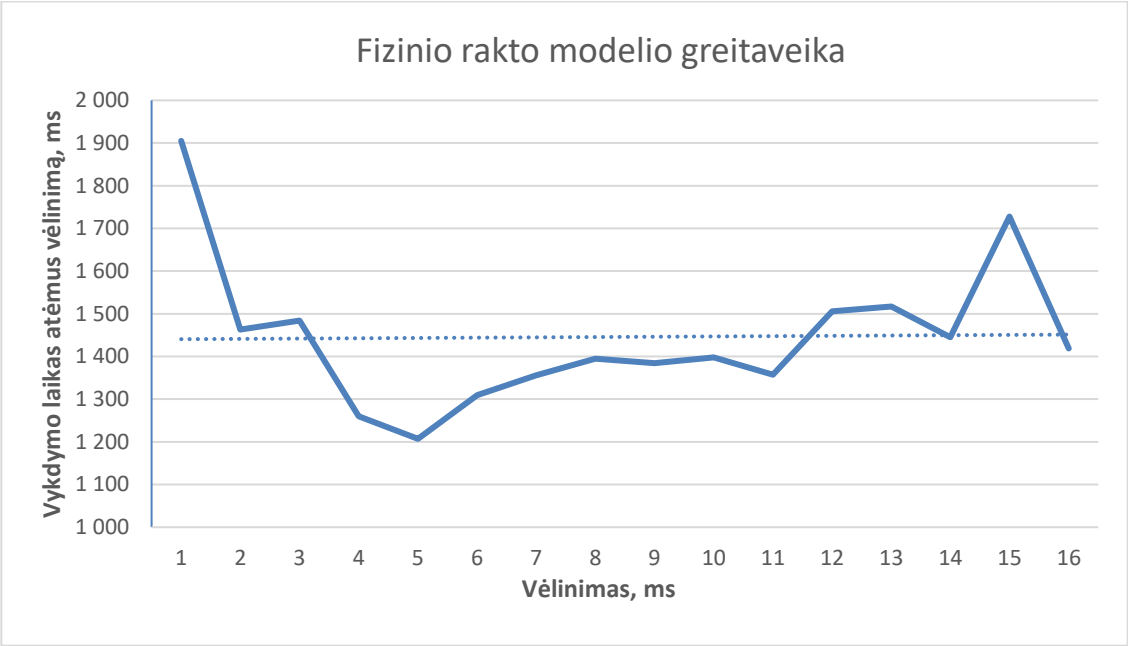
Funkcionalumą, nuo PIN kodo užklausos išsiuntimo iki atsakymo gavimo, kartojame 10 000 kartų, kiekvienoje „draw“ funkcijos iteracijoje. Atitinkamai kiekvienoje „draw“ funkcijos iteracijoje didiname ir kodo vykdymo vėlinimą (angl. „delay“) po informacijos įrašymo į serijinį kanalą. Tokiu būdu tikriname kokiam užvėlinimui esant, „Arduino“ spėja apdoroti užklausas ir dažniausiai grąžinti teisingą PIN kodą. Kiekvienos iteracijos pabaigoje (po 10 000 kreipinių įvykdymo) pateikiama, kiek laiko užtruko visi kreipiniai iki atsakymo gavimo, kokio ilgio užvėlinimas panaudotas ir kiek klaidingų PIN kodų gauta.

2 lentelė Fizinio rakto modelio tyrimo rezultatai

Vykdymo laikas, ms	**Vėlinimas, ms**	**Vykdymo laikas atėmus vėlinimą, ms**	**Klaidų skaičius, vnt.**	**Vykdymų skaičius, vnt**
11 905	1	1 905	8 435	10 000
21 463	2	1 463	8 436	
31 484	3	1 484	8 429	
41 260	4	1 260	8 451	
51 207	5	1 207	8 383	
61 309	6	1 309	8 312	
71 356	7	1 356	8 716	
81 395	8	1 395	4 291	
91 384	9	1 384	2 465	
101 398	10	1 398	0	
111 357	11	1 357	0	
121 506	12	1 506	1	
131 517	13	1 517	0	
141 445	14	1 445	0	
151 728	15	1 728	0	
161 419	16	1 419	0	

Iš 2 lentelės matome, jog kiek milisekundžių priverstinai užvėliname kiekvieną iteracijos vykdymą, tiek gauname pridėtinį vėlinimą padaugintą iš 10 000. Atmetus šio priverstinio vėlinimo laiką, bendras vykdymo laikas varijuoja nuo 1 207 milisekundžių iki 1 905 milisekundžių. Vidutinis vykdymo laikas atėmus vėlinimą – 1 446 milisekundės.

Dėmesį labiausiai verta atkreipti į klaidų skaičiaus stulpelį, kuris parodo, kiek teisingų PIN kodų grąžina „Arduino“ iš 10 000 bandymų. Matome, jog esant iki 8 milisekundžių vėlinimui, „Arduino“ nespėja apdoroti ir perduoti atgal visos informacijos, tad tik 15 % visų bandymų gaunamas teisingas atsakymas. Esant 8 milisekundžių vėlinimui, klaidų skaičius sumažėja dvigubai, nei esant mažesniam vėlinimui. Atitinkamai pastebimas dvigubas klaidų skaičiaus sumažėjimas esant 9 milisekundžių vėlinimui, lyginant su 8 milisekundžių vėlinimu. Esant 10 ir daugiau milisekundžių vėlinimui, klaidų skaičius beveik išnyksta – pasitaikė tik 1 klaida iš 10 000 bandymų, esant 12 milisekundžių vėlinimui.



32 pav. Fizinio rakto modelio greitaveika

Iš grafiko matome, jog vykdymo laikui, atėmus vėlinimo faktorių, vėlinimas neturi įtakos. Pasitaiko tam tikri pokyčių šuoliai (pvz., esant 1 milisekundės ir 15 milisekundžių vėlinimui), tačiau jokios ryškios tendencijos iš to negalime pastebėti. Taškuota linija pažymėtas vidutinis vykdymo laikas be vėlinimo (1,446 sekundės), remiantis visomis iteracijomis. Tai yra ypač didelis greitis, turint omenyje, jog atliekama 10 000 nuoseklių autentifikacijos imitacijų per mažiau nei pusantros sekundės. Žinoma, visiškai tuo remtis negalime, nes klaidų skaičius tampa priimtinu tik esant 10 ir daugiau milisekundžių vėlinimui. Tačiau, kadangi jau iš grafiko padarėme išvadą, jog priverstinis vėlinimas nedaro įtakos vykdymo greičiui, tad ir atmetus vykdymo laikus, esant mažesniam nei 10 milisekundžių vėlinimui, gauname gan panašų vykdymo greičių vidurkį be vėlinimo – 1,481 sekundės.

Kadangi funkcionalumas paprastas – tik patikrinti gautą reikšmę ir perduoti atgal PIN kodą – toks vykdymų kiekis per tokį trumpą laiko tarpą nestebina. Tačiau iš to galima daryti išvadą, jog „Arduino“ pakankamai gerai ir greitai susidorotų su tokio tipo autentifikacija, net esant ypač didelei apkrovai (daug vartotojų, mėginančių vienu metu autentifikuotis).

4.3. Kodų kortelės modelio tyrimas

Norint ištirti šį modelį, reikia pakeisti jo realizaciją, jog ji būtų pritaikyta automatiniam, o ne rankiniam, autentifikavimui. Kadangi šis modelis panašus į fizinio rakto, pagrindinis programinio kodo skirtumas bus tas, jog vietoje vieno PIN kodo, bus renkamasi iš PIN kodų masyvo, pagal gautą indeksą iš „Processing“ programos pusės. Pakeistas „Arduino“ programinis kodas pateikiamas žemiau.

```
int pinCodes[] = {594, 1485, 2589, 3879, 4412, 5847, 6895, 7285, 8771, 9999};
int pinCodesSize = sizeof(pinCodes) / sizeof(int);

void setup() {
  Serial.begin(9600);
}

void loop() {
  if(Serial.available() > 0) {
    String input = Serial.readString();
    char type = input[0];
    if(type == '2') {
      int index = input[2] - '0';
      if (index < pinCodesSize && index >= 0) {
        Serial.println(pinCodes[index]);
      }
    }
  }
}
```

33 pav. Automatizavimui pritaikyta kodų kortelės modelio „Arduino“ programa

33 pav. pateiktame programiniame kode matyti, jog viršuje apsirašome naudojamus PIN kodus ir dinamiškai apskaičiuojamą jų kiekį. „Setup“ funkcijoje sukuriamas komunikacijos kanalas. „Loop“ funkcijoje, kiekvienoje iteracijoje tikriname ar yra perduodamų duomenų. Jei yra, nuskaitome visą eilutę. Iš nuskaitytos eilutės pasiimame pirmąjį simbolį ir tikriname ar jis atitinka šio autentifikavimo metodo identifikacinę reikšmę (skaičių 2). Jei taip, iš tos pačios duomenų eilutės pasiimame trečiąjį simbolį, kuris nurodo norimo PIN kodo indeksą (šio modelio realizacijoje). Jei nurodytas indeksas yra teigiamas skaičius ir ne didesnis nei saugomų PIN kodų sąrašo dydis, į komunikacijos kanalą grąžinamas PIN kodas, kuris susietas su šiuo indeksu.

Šio modelio „Processing“ kodas taip pat labai panašus į fizinio rakto „Processing“ kodą, tačiau kodų kortelės imitavimo testavimo tikrinimui pridedame tą patį PIN kodų masyvą, kaip ir „Arduino“ programoje. Taip pat papildomai reikia ir sveikų skaičių generatoriaus, jog kiekvieną kartą perduotų skirtingas reikšmes komunikacijos kanalu ir gautų skirtingus PIN kodus, kurių teisingumą vėliau būtų galima patikrinti. Taip pat, šio modeliui sėkmingam testavimui reikia pridėti 1000 milisekundžių vėlinimą, jog „Arduino“ spėtų apdoroti gautą informaciją ir grąžinti atgal rezultatą, prieš „Processing“ paimant netinkamas reikšmes (šiukšles), „Arduino“ nespėjus perduoti teisingų. „Processing“ programinis kodas pateiktas žemiau.

```
import processing.serial.*;

Serial port;
String s;

long totalTime;
long failureCounter;
int delayTime = 1000;
long repeat = 100;

int pinCodes[] = {594, 1485, 2589, 3879, 4412, 5847, 6895, 7285, 8771, 9999};

void setup() {
  port = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  delay(1000);
  totalTime = 0;
  failureCounter = 0;
  getPinCode(repeat);
  print("Run time: " + totalTime + " ms. ");
  print("Delay time: " + delayTime + " ms. ");
  println("Failed " + failureCounter + " times.");
  delayTime++;
}
```

```
void getPinCode(long times) {
  for (int i=0; i<times;) {
    int start = millis();
    int index = generateIndex();
    String request = "2 " + index;
    port.write(request);
    delay(delayTime);
    if (port.available() > 0) {
        s = trim(port.readString());
        int finish = millis();
        i++;
        totalTime += (finish - start);
        if (!s.equals(str(pinCodes[index]))) {
          failureCounter++;
        }
    }
  }
}

int generateIndex() {
  return (int) random(10);
}
```

34 pav. Kodų kortelės autentifikacijos imitacija “Processing” aplinkoje

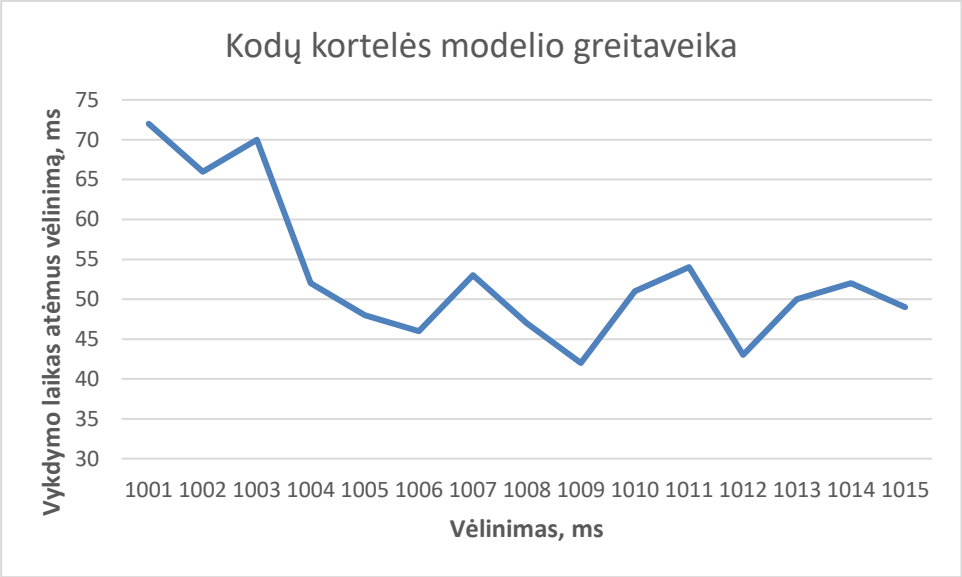
34 pav. matyti, jog kodų kortelės modelio „Processing“ programinis kodas nuo fizinio rakto modelio „Processing“ programinio kodo skiriasi tik tuo, jog turimas ne be vienas PIN kodas, o PIN kodų sąrašas. Taip pat, kiekvieną kartą kreipiantis į „Arduino“, papildomai generuojamas atsitiktinis skaičius nuo 0 iki 9, kurio indeksą norimą gauti ir pagal kurį bus tikrinama su PIN kodų sąrašu, ar teisingas PIN kodas gautas iš „Arduino“. Kiekviena iteracija kartojama tik po 100 kartų, nes šis modelis papildomai reikalauja bent jau 1 sekundės užvėlinimo, kitaip „Arduino“ nespėja apdoroti gautų reikšmių ir atsiųsti atsakymo atgal. Norint atrasti tinkamiausią vėlinimą, kiekvienoje iteracijoje vėlinimas padidinamas 1 milisekunde iki kol klaidų skaičius pasiekia priimtiną lygį.

3 lentelė Kodų kortelės modelio tyrimo rezultatai

Vykdymo laikas, ms	**Vėlinimas, ms**	**Vykdymo laikas atėmus vėlinimą, ms**	**Klaidų skaičius, vnt.**	**Vykdymų skaičius, vnt**
100 172	1001	72	88	100
100 266	1002	66	93	
100 370	1003	70	88	
100 452	1004	52	94	
100 548	1005	48	99	
100 646	1006	46	99	
100 753	1007	53	88	
100 847	1008	47	78	
100 942	1009	42	70	
101 051	1010	51	25	
101 154	1011	54	0	
101 243	1012	43	0	
101 350	1013	50	0	
101 452	1014	52	0	
101 549	1015	49	0	

Iš 3 lentelės matyti, jog, kaip ir fizinio rakto modelio atveju, priverstinio vėlinimo didinimas nedaro įtakos vykdymo laikui, atėmus vėlinimą. Vykdymo laikas, be vėlinimo, varijuoja nuo 42 iki 72 milisekundžių, o gaunamas bendras iteracijų vidurkis – 53 milisekundės.

Klaidų skaičiaus mažėjimas pastebimas nuo 1008 milisekundžių vėlinimo (8 milisekundžių, atmetus šiam metodui reikalingą priverstinį, 1000 milisekundžių, vėlinimą). Esant 1010 milisekundžių vėlinimui klaidų skaičius sumažėja iki 25 %, o dar padidinus vėlinimą iki 1011 milisekundžių – klaidų nebeaptinkama.



35 pav. Kodų kortelės modelio greitaveika

35 pav. matome, jog esant mažam užvėlinimui, vykdymo laikas šiek tiek ilgesnis, tačiau po truputį didinant vėlinimą – nusistovi ir nebepastebima tokių aukštų grafiko šuolių. Nors vidutinis visų iteracijų vykdymo laikas yra 53 milisekundės, tačiau skaičiuojant tik nuo 1011 milisekundžių vėlinimo, kada klaidų skaičius tampa priimtinas, vidutinis vykdymo laikas gaunamas šiek tiek mažesnis – 50 milisekundžių.

Norint sulyginti šio modelio greitį su fizinio rakto, padauginame 50 milisekundžių iš 100 (skirtumo tarp šio modelio iteracijos bandymų kiekio ir fizinio rakto modelio iteracijų bandymų kiekio, t.y. 100 prieš 10 000). Rezultatas būtų 5 sekundės ir tai yra virš 3 kartų lėčiau, nei fizinio rakto modelio greitis. Nors tai atrodo tinkamas rezultatas, paaukojus šiek tiek greičio, išgauti aukštesnį saugumo lygį, tačiau reikia nepamiršti, jog kodų kortelės modeliui reikalingas papildomas 1 sekundės užvėlinimas, jog „Arduino“ spėtų apdoroti gautą reikšmę, surasti tinkamą PIN kodą, pagal indeksą, ir perduoti atsakymą atgal. Dėl to, šis palyginimas nėra teisingas ir gali būti laikomas tik orientaciniu, dėl „Arduino“ platformos apribojimų. Kita vertus, kodų kortelės modelis suteikia platesnes galimybes ir nebūtina apsiriboti tokiais trumpais PIN kodais ir tokiu mažu kiekiu jų sąrašu. Gal būt tai leistų atskleisti kitas „Arduino“ savybes.

4.4. Kriptografinio rakto modelio tyrimas

Kriptografinio modelio realizacija sudėtingesnė, nei ankstesniųjų, nes reikalauja kriptografinių algoritmų panaudojimo ir komunikacija tarp abiejų pusių vykdoma esant didesniems pranešimų dydžiams. Šifravimas ir iššifravimas vykdomas tik „Arduino“ pusėje, o „Processing“ programa tik perduos pranešimą iššifravimui ir grąžins iššifruotą pranešimą, kuris turi būti nepakitęs. Tai leis išvengti papildomo darbo mėginant realizuoti lygiavertę kriptografinę biblioteką „Processing“ aplinkoje, nes naudojamos „Arduino“ kriptografinės bibliotekos panaudoti „Processing“ aplinkoje nepavyko.

```
import processing.serial.*;

Serial port;
String s;

long totalTime;
long failureCounter;
int delayTime = 1010;
long repeat = 100;

void setup() {
  port = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  delay(1000);
  totalTime = 0;
  failureCounter = 0;
  getEncryptedText(repeat);
  print("Run time: " + totalTime + " ms. ");
  print("Delay time: " + delayTime + " ms. ");
  println("Failed " + failureCounter + " times.");
  delayTime+=5;
}

void getEncryptedText(long times) {
  for (int i=0; i<times;) {
    String secretText = "Mr.Robot" + generateIndex();
    int start = millis();
    String request = "3 " + secretText;
    port.write(request);
    delay(delayTime);
    if (port.available() > 0) {
        s = port.readString();
        int finish = millis();
        i++;
        totalTime += (finish - start);
        if (!s.contains(secretText)) {
          failureCounter++;
        }
    }
  }
}

int generateIndex() {
  return (int) random(999);
}
```

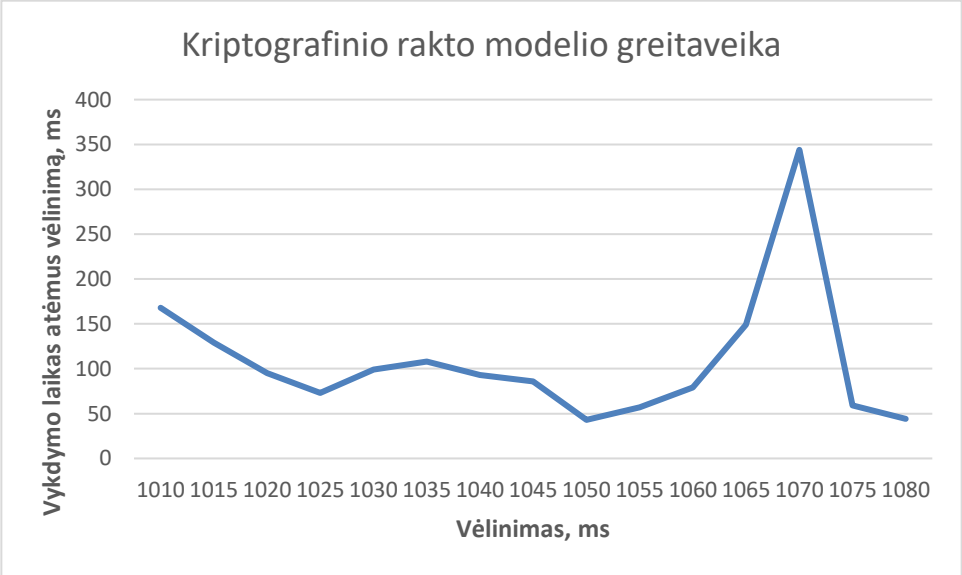
36 pav. Kriptografinio rakto autentifikacijos imitacija “Processing” aplinkoje

„Processing“ programa modifikuota taip, jog galėtų perduoti ir priimti atgal slaptą tekstą, kurio gale atsitiktinai generuojamas skaičius nuo 0 iki 998. Atsitiktinai generuojamas skaičius leis užtikrinti, jog atgal gaunamas tikrai tas pranešimas, kuris buvo išsiųstas. Pradiniu vėlinimu parinkta 1010 milisekundžių, nes mažesnis vėlinimas nesuteikia „Arduino“ pakankamai laiko perskaityti ir užšifruoti gautą pranešimą, bei atsiųsti atgal iššifruotą, dėl ko jis užstringa nuolatinio informacijos skaitymo cikle. Atsižvelgus į kelis ankstesnius neaprašytus bandymus, vėlinimo laiko iteraciniu žingsniu parinktos 5 milisekundės, jog geriau padengtų rezultatų spektrą. Dėl reikalingo privalomo 1010 milisekundžių vėlinimo, kiekviena iteracija bus kartojama tik po 100 kartų.

4 lentelė Kriptografinio rakto modelio tyrimo rezultatai

Vykdymo laikas, ms	Vėlinimas, ms	Vykdymo laikas atėmus vėlinimą, ms	Klaidų skaičius, vnt.	Vykdymų skaičius, vnt
101 168	1010	168	100	100
101 629	1015	129	98	
102 095	1020	95	100	
102 573	1025	73	100	
103 099	1030	99	99	
103 608	1035	108	99	
104 093	1040	93	99	
104 586	1045	86	87	
105 043	1050	43	4	
105 607	1055	57	0	
106 079	1060	79	0	
106 649	1065	149	0	
107 344	1070	344	0	
107 559	1075	59	0	
108 044	1080	44	0	

4 lentelėje pateikti kriptografinio rakto autentifikacijos modelio rezultatai. Iš jų matyti, jog vykdymo laikas, atėmus vėlinimą, varijuoja nuo 43 iki 344 milisekundžių. Klaidų skaičius pradeda mažėti esant apie 1045 milisekundžių vėlinimui ir ties 1050 milisekundžių pasiekia priimtiną lygį. Nuo 1055 ir didesnio milisekundžių vėlinimo, klaidų daugiau aptikta nebuvo. Vidutinis visų iteracijų vykdymo laikas atėmus vėlinimą – apie 108 milisekundes. Skaičiuojant vidutinį vykdymo laiką tik tų iteracijų, kurių klaidų skaičius nulinis, gauname 122 milisekundes.



37 pav. Kriptografinio rakto modelio greitaveika

37 pav. matome gan nuoseklų grafiką, išskyrus gan žymų nukrypimą nuo normos. Ties 1070 milisekundžių vėlinimu, dėl nežinomos priežasties, vykdymo laikas pakilo iki 344 milisekundžių. Pakartotinių specifinių bandymų metu šios situacijos atkartoti nepavyko, tad galima teigti, jog tai nesusiję su jokiu tendencingumo, o tik galimais netikėtais trikdžiais, kurie privertė „Arduino“ vykdyti operacijas lėčiau.

Atmetus pastebėta 344 milisekundžių nukrypimą nuo normos skaičiuojant vidutinį vykdymo laiką esant nuliniam klaidų skaičiui, gauname veikimo greitį artimą 77 milisekundėms. Padauginus šį greitį iš 100 (teoriškai imituojant 10 000 vykdymų skaičių per iteraciją), gauname teorinį 7,7 sekundės greitį. Kaip ir kodų kortelės atveju, taip ir šiuo, nereikėtų pamiršti, jog reikalingas privalomas 1 sekundės vėlinimas, jog šis metodas veiktų teisingai. Dėl šios priežasties, metodai lyginami tik teoriniame lygyje.

Nepaisant to, 100 sėkmingų kriptografinio rakto modelio autentifikacijų per 10,5 sekundės (be vėlinimo) yra pakankamai geras rezultatas, asmeniniam vartojimui. Siekiant geresnių rezultatų, galima pakeisti tiek naudojamą kriptografinę biblioteką, tiek ir pačią realizaciją, komunikacijos kanalu keliaujančios informacijos valdymą. Reikalingi sudėtingesni informacijos perdavimo koordinavimo mechanizmai, norint išvengti informacijos užsikimšimo komunikacijos kanale, jog „Arduino“ galėtų sėkmingai vykdyti informacijos ir nuskaitymą, ir įrašymą į komunikacijos kanalą, esant dideliems greičiams, taip reikalaujant mažesnio vėlinimo.

4.5. Tiriamų prototipų saugumo apžvalga

Sukurti modeliai ir „Arduino“ platforma leidžia įvairiapusę prototipų realizaciją. Tai apima nuo komunikacijos pasirinkimo tarp laido ir belaidžio ryšio ar USB jungties ir optinio kabelio iki visiškai automatizuoto arba pilnai rankinio (pvz., indekso gavimo iš paslaugos sistemos, įvedimo į mikrovaldiklį ir atsakymo gavimo iš mikrovaldiklio per specialią sąsają, bei atsakymo įvedimo į paslaugos sistemą autentifikacijai) mechanizmo sukūrimo. Tačiau, norint gebėti efektyviai ištirti greitaveiką, buvo pasirinktos pilnai automatizuotos prototipų realizacijos (išskyrus kodų kortelės prototipą, kuris buvo specialiai automatizuotas tyrimui – prieš tai buvo pusiau rankinis). Tai atitinkamai atveria platesnį atakos paviršių piktavaliams. Taip pat, sukurti prototipai naudoja paprasčiausius apsaugos mechanizmus, siekiant tik pavaizduoti galimus modelių realizacijos atvejus ir dėl to nėra pakankamai saugūs realiam naudojimui.

Aukštesnį saugumo lygį būtų galima pasiekti panaudojant sudėtingesnes modelių identifikacines reikšmes arba visiškai nutraukus tiesioginę komunikaciją tarp mikrovaldiklio ir kompiuterio, vykdant informacijos perdavimą per žmogų (informacija ranka pervedama iš mikrovaldiklio į kompiuterį ir atvirkščiai). Taip iš karto būtų sumažinta rizika patirti grubios jėgos ataką, jei piktavalis užvaldytų kompiuterį.

4.6. Tyrimo išvados

5 lentelė Autentifikavimo modelių tyrimo rezultatų suvestinė

Metodas	Vidutinis vykdymo laikas, ms	Priverstinis vėlinimas, ms	Papildomas vėlinimas, ms	Vykdymų skaičius, vnt
Fizinio rakto	101 481	0	10	10 000
Kodų kortelės	101 153	1 000	11	100
Kriptografinio rakto	105 627	1 000	55	100

Iš atlikto tyrimo pastebime, jog fizinio rakto modelio pasirinktos prototipo realizacijos gautas vidutinis greitis, skaičiuojant tik tas iteracijos, kurių klaidų skaičius artimas 0, yra 101,481 sekundės atliekant 10 000 autentifikacijų iš karto. Kodų kortelės modelio pasirinktos prototipo realizacijos atitinkamas vidurkis – 101,153 sekundės atliekant 100 autentifikacijų. O kriptografinio rakto vidutinis greitis atliekant 100 autentifikacijų be klaidų – 105,577 sekundės.

Pastebėta tendencija, jog kuo sudėtingesnis taikomas autentifikacijos mechanizmas, tuo ilgesnio vėlinimo reikia, norint užtikrinti korektišką mechanizmo veikimą. Fizinio rakto atveju reikalingas vėlinimas yra 10 milisekundžių, kodų kortelės – 11 milisekundžių, o kriptografinio rakto – net 55 milisekundės.

Fizinio rakto modelio realizacija per panašų laiką, kaip kiti metodai, atlieka šimtą kartų daugiau autentifikavimo operacijų, tačiau fizinio rakto modelis yra mažiausiai saugus iš visų trijų. Kodų kortelės ir kriptografinio rakto modeliai reikalauja priverstinio sekundės ilgio vėlinimo, jog apskritai veiktų, tačiau suteikia gerokai didesnį saugumo lygi, nei fizinio rakto modelis. Atitinkamai kriptografinio rakto modelio prototipas veikia lėčiausiai, tačiau vykdymo laiko skirtumas per menkas, jog turėtų įtakos naudojant autentifikacijai buitinėmis sąlygomis.

100 autentifikacijos operacijų (kodų kortelės ir kriptografinio rakto modeliuose) per 100 sekundžių (beveik 2 minutės) nėra daug – vidutiniškai 1 autentifikacijos operacija per sekundę. Toks vykdymo greitis nėra tinkamas pramoniniam vartojimui, kada tuo pačiu įrenginiu autentifikacijai naudotųsi keli žmonės vienu metu, tačiau yra pakankamas buitiniam naudojimui.

Ištirti prototipai pritaikyti automatizuotam greitaveikos tyrimui, todėl jų realizacija yra orientuota į greitaveiką ir kuo paprastesnį funkcionalumą, o ne aukšto saugumo lygio užtikrinimą. Tačiau, patys modeliai kurti remiantis plačiomis realizacijos galimybėmis, tad, esant poreikiui, sukurti prototipai gali būti modifikuojami orientacijai į maksimalų saugumo lygio užtikrinimą.

5. IŠVADOS

1. Yra daug autentifikavimo metodų, bet kiekvienas metodas turi savo privalumų ir trūkumų, tad tik jų kombinavimas gali užtikrinti efektyvią apsaugą.
2. Suprojektuoti 5 universalūs modeliai, leidžiantys įvairias praplėtimo ir realizacijos galimybes, bei tinkami apjungimui viename fiziniame įrenginyje.
3. „Arduino“ platforma suteikia plačias realizacijos galimybes, tiek programiniame, tiek ir fiziniame lygyje, tačiau pasižymi ribotais skaičiavimo ištekliais, todėl, esant poreikiui atlikti sudėtingesnes užduotis, patartina rinktis iš gerokai brangesnių platformų.
4. Sukurtas universalaus autentifikavimo mechanizmo prototipas, kuris apjungia 3 autentifikavimo modelių realizacijas, naudojimui viename fiziniame įrenginyje.
5. „Arduino“ platformoje sukurtas prototipas geba atlikti 100 autentifikacijų per sekundę fizinio rakto modeliu, o kodų kortelės ir kriptografinio rakto modeliais – 1 autentifikaciją per sekundę.
6. Kodų kortelės ir kriptografinio rakto modelių prototipų greitaveiką „Arduino“ platformoje stipriai apriboja priverstinis 1 sekundės vėlinimas, kuris reikalingas sklandžiam informacijos perdavimui komunikacijos kanalu.

6. LITERATŪRA

[1] E. Stobert, „The agony of passwords: can we learn from user coping strategies?,“ įtraukta *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, Toronto, 2014.

[2] K. P. Weiss, „Method and apparatus for positively identifying an individual“. United States of America Patentas US4720860 A, 30 lapkričio 1984.

[3] Y. Shah, V. Choyi ir L. Subramanian, „Multi-factor Authentication as a Service,“ įtraukta *3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, San Francisco, 2015.

[4] S. U. Shah, F.-e.-H. ir A. A. Minhas, „New Factor of Authentication: Something You Process,“ įtraukta *2009 International Conference on Future Computer and Communication*, Kuala Lumpar, 2009.

[5] R. V. Yampolskiy, „User Authentication via Behavior Based Passwords,“ įtraukta *2007 IEEE Long Island Systems, Applications and Technology Conference*, Farmingdale, 2007.

[6] H. Al-Assam, H. Sellahewa ir S. Jassim, „Multi-factor biometrics for authentication: a false sense of security,“ įtraukta *12th ACM workshop on Multimedia and security*, Roma, 2010.

[7] A. Yohan, N.-W. Lo ir H. R. Lie, „Dynamic multi-factor authentication for smartphone,“ įtraukta *27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Valencia, 2016.

[8] R. L. de Souza, L. C. Lung ir R. F. Custodio, „Multi-Factor Authentication in Key Management Systems,“ įtraukta *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Melbourne, 2013.

[9] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo ir M. Yung, „Fourth-factor authentication: somebody you know,“ įtraukta *13th ACM conference on Computer and communications security*, Alexandria, 2006.

[10] S. H. Khan ir M. A. Akbar, „Multi-Factor Authentication on Cloud,“ įtraukta *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Adelaide, 2015.

[11] D. Hema ir S. Bhanumathi, „Mouse behaviour based multi-factor authentication using neural networks,“ įtraukta *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, Kuramacoil, 2016.

[12] A. K. Jain, A. Ross ir U. Uludag, „Biometric template security: Challenges and solutions,“ įtraukta *13th European Signal Processing Conference*, Antalya, 2005.

[13] R. Saini ir N. Rana, „Comparison of various biometric methods,“ įtraukta *International Journal of Advances in Science and Technology (IJAST)*, 2014.

[14] D. Strom, „Password-free authentication: Figuring out FIDO,“ [Tinkle]. Available: http://searchsecurity.techtarget.com/feature/Password-free-authentication-Figuring-out-FIDO. [Kreiptasi 25 spalio 2016].

[15] C. Oak, „Passwordless Experience – The FIDO Standards behind this,“ 28 sausio 2015. [Tinkle]. Available: http://digitalmoney.shiftthought.co.uk/passwordless-experience-the-fido-standards-behind-this/. [Kreiptasi 13 balandžio 2016].

[16] Wikipedia, „FIDO Alliance,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/FIDO_Alliance. [Kreiptasi 06 birželio 2016].

[17] „FIDO Alliance,“ [Tinkle]. Available: https://fidoalliance.org/. [Kreiptasi 01 lapkričio 2015].

[18] S. Machani ir K. Alikhani, „Does FIDO really usher in the Death of Passwords?,“ įtraukta *RSA Conference 2015*, Singapore, 2015.

[19] H. Tschofenig, „W3,“ [Tinkle]. Available: https://www.w3.org/2012/webcrypto/webcrypto-next-workshop/papers/webcrypto2014_submission_1.pdf. [Kreiptasi 15 sausio 2016].

[20] Wikipedia, „Public-key cryptography,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/Public-key_cryptography. [Kreiptasi 06 spalio 2015].

[21] M. Rouse, „Asymmetric cryptography (public key cryptography),“ [Tinkle]. Available: http://searchsecurity.techtarget.com/definition/asymmetric-cryptography. [Kreiptasi 28 gruodžio 2015].

[22] Wikipedia, „RSA (cryptosystem),“ [Tinkle]. Available: https://en.wikipedia.org/wiki/RSA_(cryptosystem). [Kreiptasi 23 lapkričio 2016].

[23] Quantag Munich and SecureScrypt Germany, „Secures Crypt,“ 2015. [Tinkle]. Available: http://securescrypt.com/FIDO/FIDOWP4-4-15.pdf. [Kreiptasi 13 sausio 2017].

[24] „FIDO U2F (UNIVERSAL 2ND FACTOR),“ [Tinkle]. Available: https://www.yubico.com/about/background/fido/. [Kreiptasi 10 gruodžio 2015].

[25] F. Reimair, C. Kollmann ir A. Marsalek, „Emulating U2F authenticator devices,“ įtraukta *2016 IEEE Conference on Communications and Network Security (CNS)*, Philadelphia, 2016.

[26] S. M. Kerner, „FIDO Alliance Extends Two-Factor Security Standards to Bluetooth, NFC,“ 01 liepos 2015. [Tinkle]. Available: http://www.eweek.com/security/fido-alliance-extends-two-factor-security-standards-to-bluetooth-nfc. [Kreiptasi 15 lapkričio 2016].

[27] „Compare board specs,“ [Tinkle]. Available: https://www.arduino.cc/en/Products/Compare. [Kreiptasi 20 rugsėjo 2016].

[28] L. Orsini, „Arduino Vs. Raspberry Pi: Which Is The Right DIY Platform For You?,“ 07 gegužės 2014. [Tinkle]. Available: http://readwrite.com/2014/05/07/arduino-vs-raspberry-pi-projects-diy-platform/. [Kreiptasi 19 lapkričio 2016].

[29] „BeagleBone,“ [Tinkle]. Available: http://beagleboard.org/bone. [Kreiptasi 21 lapkričio 2016].

[30] A. Allan, „Arduino Uno vs BeagleBone vs Raspberry Pi,“ 15 balandžio 2013. [Tinkle]. Available: http://makezine.com/2013/04/15/arduino-uno-vs-beaglebone-vs-raspberry-pi/. [Kreiptasi 19 lapkričio 2016].

[31] „What is an Arduino?,“ [Tinkle]. Available: https://learn.sparkfun.com/tutorials/what-is-an-arduino. [Kreiptasi 12 gruodžio 2016].

[32] MarkT, „Arduino Forum,“ 25 sausio 2012. [Tinkle]. Available: https://forum.arduino.cc/index.php?topic=88890. [Kreiptasi 02 gruodžio 2016].

[33] „Processing,“ [Tinkle]. Available: https://processing.org/. [Kreiptasi 21 gruodžio 2016].