

Универзитет у Београду
Електротехнички факултет
Електротехника и рачунарство



ДИПЛОМСКИ РАД

Имплементација аутентификације коришћењем
ФИДО2 механизма

Ментор

проф. др. Павле Вулетић

Студент

Млађан Михајловић 0423/2015

Београд, 2019.

Садржај

1	Увод	1
2	ФИДО 2.....	6
2.1	Преглед	6
2.2	СТАР (<i>Client to Authenticator protocol</i>) протокол	6
2.2.1	Аутентификатор АПИ	6
2.2.2	Енковање порука	8
2.2.3	Форматирање порука у зависности од начина транспорта.....	8
2.3	<i>Webauthn (W3C Web Authentication)</i> спецификација	8
2.3.1	Церемонија регистрације креденцијала.....	9
2.3.2	Церемонија аутентификације корисника.....	10
2.3.3	Повлачење креденцијала	11
2.3.4	<i>Web Authentication API</i>	12
2.4	Аутентификатори	15
2.4.1	<i>Yubikey</i>	16
2.4.2	<i>Windows Hello</i>	16
2.4.3	<i>Android Key</i>	17
3	JWS – JSON веб потписи (JSON Web Signatures)	19
3.1	Преглед	19
3.2	JWS типови серијализације.....	19
3.2.1	JWS компакта серијализација.....	20
3.2.2	JWS JSON серијализација	20
3.3	JOSE заглавље.....	20
3.4	Генерисање и коришћење JWS-а.....	22

3.5	Предности <i>JWS</i> -а у односу на <i>XMLDSig</i>	23
4	Спецификација имплементације	24
4.1	Архитектура система	24
4.2	Апликација за потписивање помоћу паметних картица	24
4.2.1	Интерфејс апликације	25
4.2.2	Функционалности апликације	25
4.3	ФИДО2 менаџер идентитета	27
4.3.1	Почетна страна апликације	28
4.3.2	Управљање регистрованим ФИДО2 аутентификаторима	30
4.3.3	Управљање регистрованим паметним картицама	32
4.3.4	Ауторизација ресурса	33
4.4	Тестна клијентска апликација	34
5	Имплементациони детаљи	36
5.1	Коришћене технологије	36
5.2	Дизајн базе података	36
6	Закључак	40
7	Коришћени термини и скраћенице	41
8	Литература	42
9	Додатак	44
9.1	Модели	44
9.1.1	Модел <i>PublicKeyCredentialCreationOptions</i>	44
9.1.2	Модел <i>PublicKeyCredentialRpEntity</i>	45
9.1.3	Модел <i>PublicKeyCredentialUserEntity</i>	46
9.1.4	Модел <i>PublicKeyCredentialDescriptor</i>	46

9.1.5	Модел <i>PublicKeyCredentialParameters</i>	46
9.1.6	Модел <i>PublicKeyCredentialRequestOptions</i>	46
9.1.7	Модел <i>AuthenticatorAttestationResponse</i>	48
9.1.8	Модел <i>AuthenticatorAssertionResponse</i>	48
9.2	Примери кода	49

1 Увод

У паралели са развојем друштвених мрежа, платформа за интернет банкарство, и другим интернет платформама које садрже осетљиве корисничке податке су се развијали механизми за крађу тих података.

До крађе корисничких података од стране нападача може доћи тако што програмери нису довољно добро осигурали систем, тако да су омогућили нападачу да дође до корисничких креденцијала (корисничких имена и лозинки). Овакви случајеви се ређе дешавају, зато што се избегава чување корисничке лозинке у изворној форми, као и слање корисничке лозинке кроз мрежу у изворној форми, већ се уместо тога на клијентској страни над лозинком примени нека хеш функција и затим се лозинка у таквом облику шаље кроз мрежу и чува у систему. У овој ситуацији, нападач, чак и да дође до копије базе података из система, не може доћи до корисничке лозинке уколико је примењена јака хеш функција.

У већини случајева, до хаковања корисничких налога долази због непажње корисника. Данас је већина корисничких налога и даље осигурана само лозинком. Проблеми са коришћењем лозинки су многобројни:

- Корисници често користе слабе лозинке, уколико веб сајт не захтева експлицитно коришћење јаких лозинки што нападачима олакшава посао да хакују кориснички налог тако што ће покушати да се улогују са неком од најчешће коришћених лозинки;
- Уколико веб сајт захтева коришћење јаких лозинки, корисници често прибегавају записивању лозинки на папир или у своје мобилне телефоне што је огроман ризик јер нападач који познаје корисника може врло лако да дође до записаних података;
- Већина корисника користи исту лозинку па чак и исто корисничко за све веб сајтове на које се региструју, што представља огромни сигурносни ризик јер

уколико нападач успе да компромитује систем само једног од тих сајтова, доћи ће до корисничких креденцијала за све сајтове на које је корисник регистрован;

- Велики број корисника на интернету није у потпуности дигитално образован, тако да нападачи неким од метода обмане могу доћи до њихових лозинки, једна од таквих метода је фишинг (енг. *Phishing*), где нападач кориснику уместо правог веб сајта презентује копију која личи или изгледа потпуно исто као веб сајт на који корисник жели да се улогује, али када корисник покуша да се улогује он своје креденцијале пошаље нападачу;
- Лењост корисника да често мењају лозинке – корисници у великом броју случајева исту лозинку користе у периоду и по пар година што драстично повећава вероватноћу да нападач хакује корисничке налоге;
- Појава која је честа је да корисници своје креденцијале деле са другим особама. Те особе су у већини случајева особе од поверења међутим, те особе могу ненамерно, или у неким случајевима, намерно те креденцијале открити потенцијалном нападачу.

Поред ових наведених проблема, постоји велики број оних који су изостављени. Анализом наведених проблема и њихових решења се може доћи до закључка да решења неких проблема доводе до појаве неких других. На пример: форсирање корисника да често мења лозинку или да користи јаку лозинку, доводи до тога да корисник лако може заборавити лозинку, па зато прибегава њеном записивању.

Последице крађе корисничких креденцијала могу бити разне:

- Нападач може доћи до приватних података корисника које касније може искористити за уцену корисника;
- Нападач може украсти новац од корисника;
- Нападач може украсти идентитет корисника итд.

У случају компромитовања корисничких креденцијала, корисник често није свестан да је то последица његове непажње и прибегава нападу на власнике веб сајтова на

којима су им креденцијали компромитовани. Ово може довести до негативног рејтинга тих сајтова и до одлива корисника, чак и поред тога што веб сајтови у том случају нису директно одговорни за крађу корисничких креденцијала.

Из свега наведеног се долази до закључка да коришћење само лозинки одавно није довољна сигурност за корисника на интернету. Оно чему се данас прибегава да би се корисничко искуство на интернету додатно осигурало јесте коришћење других метода аутентификације или додавање додавање додатних фактора аутентификације корисника.

Једна од распрострањених метода јесте изазов/одговор (енг. *Challenge/Response*). У овој методи сваки од клијената добија своју приватну хеш функцију. Сервер садржи копију ове хеш функције. Када корисник изда захтев серверу за аутентификацију, сервер изгенерише изазов који пошаље кориснику. Корисник над овим изазовом примени хеш функцију која му је издата и пошаље резултат серверу. Сервер над оригиналним изазовом примени исту хеш функцију и ако је резултат исти као онај који је добијен од корисника, успешно аутентификује корисника. Ипак овај метод није практичан за свакодневно коришћење од стране корисника и најчешће се користи у дигиталним телевизијским пријемницима који садрже ТВ картице.

Када је у питању двофакторска аутентификација, начини за њену реализацију могу варирати. Вероватно најзаступљеније методе су: слање СМС поруке од стране веб сајта са верификационим кодом на број телефона који је корисник повезао са својим налогом или слање пуш нотификације на паметни телефон корисника. Мана методе слања СМС поруке је потреба да корисник достави веб сајту свој број телефона. Добар део корисника не жели да остави свој број телефона, чак и на највећим сајтовима попут Фејсбука, Гугла или Твитера из страха од злоупотребе. Мана приступа слања пуш нотификације јесте потреба да у већини случајева корисник има на свом паметном телефону инсталирану апликацију веб сајта на који жели да се улогује.

Још једна популарна метода за аутентификацију корисника јесте коришћењем криптографије помоћу приватног и јавног кључа. Код ове методе, у највећем броју

случајева, корисник поседује хардверски уређај који садржи сертификат са јавним кључем чијем издавачу сервер верује, као и приватни кључ заштићен лозинком која је позната само кориснику. Прво је потребно да се корисник на веб апликацији региструје са својим јавним кључем. Сервер тада уз кориснички налог сачува кориснички сертификат са јавним кључем који ће касније користити за аутентификацију корисника. Када корисник пожели да се улогује, он од сервера добије поруку која је најчешће случајно генерисана и коју је потребно да потпише својим приватним кључем. Апликација на клијентској страни затим серверу пошаље потписану поруку. Сервер добијени потпис дешифрује јавним кључем са којим се корисник регистровао и упоређује га са поруком која је послата кориснику. Уколико се поруке поклапају корисник бива успешно аутентификован. Мана ове методе је то што корисник мора да поседује хардверски уређај који садржи сертификат (најчешће у виду паметне картице (Smart card) или у виду дигиталног сертификата издатог од стране државног тела на биометријском документу).

ФИДО2 спецификација представља резултат удруженог напора ФИДО Алијансе [1] чији су чланови водеће светске компаније попут Гугла, Фејсбука, Интела, Мајкрософта итд. и Светског интернет конзорцијума [2] чији је циљ креирање јаке аутентикације на интернету.

Појавом ФИДО2 спецификације која је заснована на криптографији помоћу јавног и приватног кључа је овај начин аутентификације знатно олакшан јер омогућава корисницима да користе овај метод као додатни (или једини) фактор аутентификације а да им притом нису потребни било какви додатни хардверски уређаји поред оних које већ поседују (рачунар или паметни телефон).

ФИДО2 спецификација поред овога, омогућава комплетно уклањање лозинки као вида аутентификације корисника, а у неким затвореним системима попут компанија може елиминисати потребу и за коришћењем корисничких имена.

У поглављу 2 ће бити разматрани ФИДО2 механизми и саставни делови ФИДО2: *CTAP (Client to Authenticator protocol)* протокол и *WebAuthn (W3C Web Authentication*

specification) спецификација, као и најдосупнији типови аутентификатора. У поглављу три ћемо се упознати са *JWS* стандардом за генерисање потписа док ће. У поглављу четири ће бити дат приказ практичне употребе и случајева коришћења ФИДО2 и *JWS*-а кроз конкретну апликацију.

2 ФИДО 2

2.1 Преглед

ФИДО2 се састоји од *WebAuthn* (*W3C Web Authentication*) стандарда [3] и *CTAP* (*FIDO Client to Authenticator Protocol*) протокола [4].

2.2 *CTAP* (*Client to Authenticator protocol*) протокол

CTAP протокол је протокол апликативног слоја који служи за комуникацију између аутентификатора и клијента (интернет претраживач, оперативни систем). *CTAP* протокол је сачињен од три целине:

- Аутентификатор АПИ (*Authenticator API*): садржи апстраховане дефиниције операција аутентификатора. Ове дефиниције садрже информацију о томе које параметре свака од операција прима, као и резултате тих операција са дефинисаним кодовима грешака.
- Енковање порука: свака метода садржи опис формата у ком очекује да подаци које операције АПИ-ја примају буду енкодovани као и опис формата у коме враћа енкодovани резултат.
- Форматирање порука у зависности од начина транспорта (*Transport-specific Binding*): специфицира начине паковања порука у зависности од вида транспорта (нпр. *USB, NFC, Bluetooth*).

2.2.1 Аутентификатор АПИ

Свака од операција АПИ-ја може бити независно извршена и све операције аутентификатора су асинхроне. Протокол не гарантује поуздан транспорт као ни да ће одговори долазити у редоследу у ком су слани захтеви. У наставку ће бити описане методе и структуре података које АПИ садржи. Детаљан опис параметара сваке од метода дат је у поглављу 9.

Ентитети који учествују у протоколу су:

- Клијент – веб претраживач или оперативни систем

- Аутентификатор – екстерни уређај попут USB кључа, или рачунар (PC, лаптоп, паметни уређај)
- Поуздана страна – веб апликација за коју корисник жели да региструје креденцијале са аутентификатора

authenticatorMakeCredential – ова метода се позива од стране клијента када је потребно послати захтев за генерисање новог креденцијала у аутентификатору.

authenticatorGetAssertion – ова метода се позива од стране клијента када је потребан доказ корисничке аутентификације или одобрење корисника за одређену трансакцију, а све то користећи претходно генерисане креденцијале у аутентификатору за одређену поуздану страну.

authenticatorGetNextAssertion – клијент позива ову методу када одговор на позив методе `authenticationGetAssertion` садржи поље `numberOfCredentials` и његова вредност прелази вредност 1. Ова метода се користи зарад добављања потписа за друге креденцијале за одређени `authenticatorGetAssertion` захтев.

authenticatorGetInfo – помоћу ове методе је могуће добити информацију о аутентификатору. Следеће информације се могу придобити: *AAGUID* – јединствени идентификатор аутентификатора, подржане верзије протокола и подржане екстензије.

authenticatorClientPIN – аутентификатор може поседовати и пин код који се пре коришћења аутентификатора мора унети од стране корисника чиме он ауторизује захтевану акцију на аутентификатору. Метода се може користити за постављање сигурносног ПИН кода на аутентификатору, промену постојећег пин кода или добављање ПИН токена од аутентификатора који се касније може користити у позивима метода `authenticatorMakeCredential` и `authenticatorGetAssertion`.

authenticatorReset – користи се од стране клијента да се ресетује аутентификатор назад на фабричка подешавања и притом инвалидирајући све генерисане креденцијале.

2.2.2 Енковање порука

Потреба за енковањем порука долази од тога што неки видови транспорта порука (нпр. *Bluetooth*) имају ограничен пропусни опсег (*Bandwidth*) па је пожељно да се формати серијализације попут *JSON*-а и поготову *XML*-а избегавају у овим ситуацијама. Из овог разлога се сво енковање ради помоћу *Концизног бинарног енковања* [5].

Да би се смањила комплексност порука и количина ресурса који су потребни да се поруке валидирају, поруке морају поштовати *CTAP2* каноничну *CBOR* форму за енковање дефинисану од стране ФИДО алијансе. Сви енкодери морају серијализовати *CBOR* у ову каноничну форму без дуплицирања мапа кључева. Сви декодери морају да одбију *CBOR* који није у овој каноничној форми и требало би да одбију поруке који садрже дупликате кључева.

2.2.3 Форматирање порука у зависности од начина транспорта

У зависности од начина транспорта порука између поуздане стране и аутентификатора се могу разликовати начини паковања порука као и протоколи комуникације. Видови транспорта које подржава *CTAP* протокол су:

- *Bluetooth* (нпр. *Android Key*)
- *NFC* (нпр. *Yubikey*)
- *USB* (нпр. *Yubikey*)

2.3 *Webauthn (W3C Web Authentication)* спецификација

Webauthn спецификација дефинише АПИ помоћу кога се може издати и користити генерисани јавни кључ за одређену веб апликацију. Методе овог АПИ-ја у позадини комуницирају са аутентификатором путем *CTAP* протокола.

Јавни кључ се генерише на основу добијених података од стране веб апликације и његова сврха може бити јака аутентикација корисника, потписивање корисничких налога или давање пристанка на одређену акцију од стране корисника. Пар приватни/јавни кључ се чува на аутентификатору на ком је и креиран на захтев поуздане стране и може му се приступити само од поуздане стране за коју је и креиран уз одобрење корисника.

Постоје два типа операција које поуздане стране могу захтевати од аутентификатора:

1. Захтев за регистрацију, где се, на основу података прослеђених од поуздане стране, на аутентификатору креира пар приватног и јавног кључа за одређеног корисника. Захтев за регистрацију може бити издат за корисника који је претходно већ регистрован код поуздане стране или током саме регистрације корисника.
2. Захтев за аутентификацију, где поуздана страна као одговор добија потврду од аутентификатора да је корисник за којег је јавни кључ издат присутан и да је дао одобрење за коришћење истог.

Web Authentication API се састоји од имплементације `PublicKeyCredential` интерфејса за управљање креденцијалима и инфраструктуре помоћу које ти креденцијали могу да се користе са акцијама `navigator.credentials.create()` и `navigator.credentials.get()` које се користе при регистрацији и аутентификацији корисника, респективно.

2.3.1 Церемонија регистрације креденцијала

Као резултат ове церемоније креира се нови кориснички креденцијал који се региструје на серверу и који корисник у будућности може користити за аутентификацију.

1. Корисник посети одређени сајт (поуздану страну) на коме или поседује кориснички налог на који је улован коришћењем корисничког имена и лозинке

или коришћењем претходно додатог другог аутентификатора, или жели да креира кориснички налог.

2. Поуздана страна покрене скрипту за издавање новог креденцијала (**Пример 5**).
3. Клијентска платформа проналази аутентификатор.
4. Клијент се повезује са аутентификатором.
5. Кориснику се приказује интерфејс од стране аутентификатора преко кога корисник може да изврши потребну акцију (биометријска аутентификација, унос пина, додиривање *USB* кључа итд.).
6. Клијент добија одговор од аутентификатора који се прослеђује до поуздане стране.
7. Уколико је аутентификација била успешна поуздана страна шаље креиране креденцијале серверу који их чува у својој бази уз кориснички налог. Заједно са креденцијалима могу бити послати и подаци о пореклу аутентификатора и његовим карактеристикама.

При иницирању овог процеса, поуздана страна може захтевати кроз параметре захтева да се корисник аутентификује са тачно одређеним аутентификатором (нпр. *Windows Hello*). Тада ће клијент проверити да ли је такав тип аутентификатора доступан на платформи. Уколико јесте процес регистрације корисника ће бити настављен, у супротном ће бити враћена грешка.

2.3.2 Церемонија аутентификације корисника

Да би ова церемонија била покренута, претходно се мора извршити процес регистрације креденцијала за корисника. Ова церемонија се користи када је потребно аутентификовати корисника са претходно регистрованим креденцијалима код поуздане стране.

1. Корисник посети одређени сајт (поуздану страну) за који је претходно регистровао креденцијале. Поуздана страна покрене скрипту за аутентификацију корисника (**Пример 4**).

2. Скрипта кроз позив АПИ-ја захтева од клијента потврду аутентификације. При слању захтева кроз параметре се могу проследити информације које би клијенту помогле да сузи избор прихватљивих аутентификатора (нпр. тип аутентификатора, идентификаторе креденцијала које је корисник регистровао код поуздане стране а који могу бити добијени помоћу прослеђеног корисничког имена итд.).
3. Клијентска платформа проналази доступне аутентификаторе.
4. Клијент се повезује са аутентификатором.
5. Кориснику се приказује интерфејс од стране аутентификатора. Овај интерфејс може кориснику приказати листу прихватљивих креденцијала које корисник може користити за аутентификацију.
6. Аутентификатор захтева интеракцију од стране корисника којом ће потврдити своје присуство на пример: биометријску аутентификацију, унос ПИН кода, додоиривање *USB* кључа итд.
7. Аутентификатор одговор шаље клијенту који га прослеђује до поуздане стране. У случају да се корисник неуспешно аутентификовао или одбио аутентификацију, биће враћена грешка.
8. У случају успешне аутентификације, потврда аутентификације се прослеђује до сервера који поверава идентификатор креденцијала са којим је аутентификација извршена као и потпис потврде аутентификације. У случају да су провере успешно извршене корисник ће бити аутентификован од стране сервера.

2.3.3 Повлачење креденцијала

Регистровани кориснички креденцијали могу бити повучени из више разлога:

- Корисник је изгубио аутентификатор (лаптоп, телефон или *USB* кључ) – у овом случају корисник одлази на веб сајт на коме је креденцијал регистрован, улогује се неком од других доступних метода и на страни са листом својих креденцијала уклони изгубљени креденцијал;

- Сервер може повући кориснички креденцијал уколико је детектовао да он није активан одређени временски период;
- Корисник обрише креденцијал са аутентификатора тако што врати аутентификатор на фабричка подешавања.

2.3.4 *Web Authentication API*

Web Authentication API је апи за креирање и коришћење креденцијала јанвих кључева. Идеја овог АПИ-ја је да креденцијали припадају кориснику и да њима управља *WebAuthn* аутентификатор, са којим *WebAuthn* поуздана страна комуницира кроз клијентску платформу.

Безбедност АПИ-ја је обезбеђена кроз клијента и аутентификатор. Аутентификатор који управља креденцијалима, обезбеђује да су ти креденцијали затворени унутар оквира специфичног веб сајта. То значи да креденцијали не могу да се користе за други веб сајт за који нису регистровани. Да би аутентификатор знао ком веб сајту припада сваки од генерисаних креденцијала, при захтеву за генерисање креденцијала, кроз поље извора захтева се прослеђује базни *URL* веб сајта са ког потиче захтев. При аутентификацији се, такође, базни *URL* порекла захтева просеђује до аутентификатора ради валидације порекла.

Битан аспект церемоније генерисања креденцијала и аутентификације корисника јесте одржавање приватности корисника и спречавање злонамерних поузданих страна да испитују да ли за одређеног корисника у аутентификатору постоје креденцијали издати за друге поуздане стране. Да би се то спречило као додатан параметар позивима АПИ-ја од стране клијента се прослеђује идентификатор поуздане стране *RP ID (Relying Party Identifier)*. Аутентификатор обезбеђује да ће се креденцијали креирани од стране поуздане стране користити у операцијама које су захтевани само од поуздане стране са истим *RP ID*-јем. Такође, одвајање овог идентификатора од извора захтева омогућава коришћење АПИ-ја у случајевима где једна поуздана страна поседује више извора.

Прослеђивање идентификатора поуздане стране и извора поуздане стране је обезбеђено од стране клијента.



Слика 1 - Церемонија регистравања кренцијала

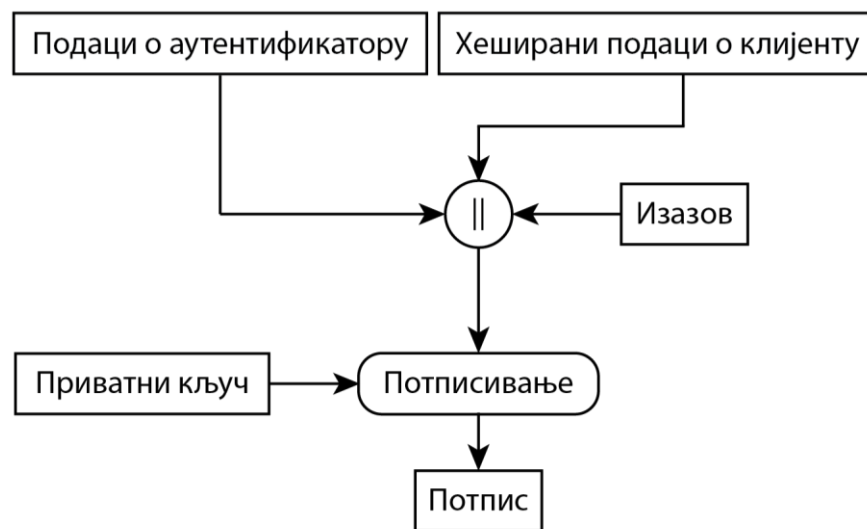


Слика 2 - Церемонија аутентификације корисника

Описи модела који се користе у церемонијама аутентификације корисника и регистравања креденцијала се могу видети у поглављу 9.

Објекти који представљају одговоре аутентификатора у себи садрже потписани садржај који је потписан приватним кључем из аутентификатора. Потписани садржај се састоји од података о аутентификатору, хешираним подацима о клијенту од којег потиче захтев и изазова сервера (поље *Challenge* захтева за регистрацију креденцијала или захтева за аутентификацију корисника). Када се овај одговор проследи до сервера, потребно је да сервер помоћу јавног кључа који одговара приватном кључу којим је садржај потписан, декодује потпис и упореди да ли се садржај поља изазова поклапа са

оним садржајем који је сервер изгенерисао за тај захтев. Такође у случају аутентификације корисника када је аутентификатор већ регистрован у систему, може се и проверити да ли се подаци о аутентификатору у потпису поклапају са оним подацима регистрованим у систему. Провера података о аутентификатору се обавезно састоји од провере колико је пута корисник нешто потписао са коришћеним аутентификатором. Сваки аутентификатор у себи чува вредност броја потписа. При регистрацији аутентификатора, потребно је да сервер сачува тренутну вредност броја потписа аутентификатора. При следећем коришћењу аутентификатора од стране корисника за аутентификацију код поуздане стране, сервер поуздане стране треба да упореди сачувану вредност броја потписа аутентификатора са оном коју је добио у потпису. Уколико је вредност добијена у потпису мања или једнака од вредности која је сачувана у систему, сматра се да је неко покушао да реплицира аутентификатор и овај захтев се одбија од стране сервера.



Слика 3 - поступак потписивања захтева у аутентификатору

2.4 Аутентификатори

Да би уређај могао да се користи као ФИДО2 аутентификатор, он мора да буде одобрен од стране ФИДО алијансе. Листа одобрених ФИДО2 аутентификатора је доступна на званичном сајту ФИДО алијансе. Тренутно најприступачнији

аутентификатори на тржишту су: *Yubikey* од компаније *Yubico*, *Windows Hello* од компаније *Microsoft* и *Android Key* од компаније *Google*.

2.4.1 *Yubikey*

Yubikey је хардверски аутентификатор развијен од компаније *Yubico* који се може користити као ФИДО2 аутентификатор. *Yubikey* се поред ФИДО2 протокола, може користити и са *U2F* и *OATH HOTP* протоколима. Ови аутентификатори подржавају корисничку аутентификацију коришћењем *USB*-а или *NFC*-а транспортних протокола и као такав се може користити са десктоп рачунарима, лаптоп рачунарима и паметним телефонима. Корисник са овим аутентификатором интерагује на следећи начин:



1. Аутентификатор прими захтев за издавање новог креденцијала или за издавање потврде о аутентификацији корисника од поуздане стране.
2. Кориснику се приказује интерфејс преко кога је потребно да корисник унесе ПИН код којим је аутентификатор заштићен.
3. Уколико је корисник унео валидан пин, биће приказана порука кориснику да додирне означено место на аутентификатору како би потврдио своје присуство.

2.4.2 *Windows Hello*

Windows Hello је хардверски аутентификатор развијен од стране компаније *Microsoft* са циљем да побољша корисничко искуство на оперативном систему *Windows* и повећа сигурност корисничких налога додавањем биометријске аутентификације. Касније је овај аутентификатор додатно усаглашен са стандардима ФИДО алијансе чиме је добио ФИДО2 сертификацију.

Windows Hello подржава три начина аутентификације корисника: аутентификацију ПИН кодом, биометријску аутентификацију читавањем отиска прста и биометријску



аутентификацију препознавањем лица. Да би било који од ова три вида аутентификације могао да се користи са поузданом страном за издавање креденцијала или аутентификацију корисника, претходно мора бити конфигурисан у оквиру самог оперативног система *Windows*. Корисник са овим аутентификатором интерагује на следећи начин:

1. Аутентификатор прими захтев за издавање новог креденцијала или за издавање потврде о аутентификацији корисника од поуздане стране.
2. Кориснику се приказује интерфејс на коме може да одабере једну од конфигурисаних метода аутентификације помоћу *Windows Hello*-а.
3. Корисник одабере жељену методу аутентификације.
4. Корисник се у зависности од одабране методе аутентификује: уношењем ПИН кода, читавањем отиска прста или препознавањем лица.

У случају биометријске аутентификације читавањем отиска прста или препознавањем лица, мора се поседовати одговарајући хардвер у виду читача отиска прста или *Windows Hello* компатибилне веб камере. Листа хардвера који је компатибилан са овим аутентификатором може се пронаћи на веб страни компаније *Microsoft*.

2.4.3 Android Key

Android Key је кључ који је уграђен у Андроид паметне телефоне. Сваки Андроид паметни телефон који поседује верзију 7 Андроида или већу је ФИДО2 сертифициван аутентификатор и као такав се може користити за ФИДО2 аутентификацију. Корисник са овим аутентификатором интерагује на следећи начин:



1. Корисник укључи блутут на паметном уређају и на платформском уређају (рачунару или лаптопу)
2. Корисник на веб сајту на коме жели да се улогује изабере опцију да се улогује помоћу регистрованог паметног уређаја

3. Платформски уређај и паметни уређај комуницирају путем блутута и кориснику се на паметном уређају приказује интерфејс на коме је потребно да потврди да он покушава да се аутентификује на одређеном веб сајту

У тренутку писања овог документа, *Android* Key је могуће користити само са Гугл налогом.

3 JWS – JSON веб потписи (JSON Web Signatures)

3.1 Преглед

JWS предстаља дигитално потписан или *MAC*-ован садржај користећи *JSON* структуре података и УРЛ енковање у основи 64. *JWS* се састоји од три целине:

- *JOSE* заглавље (*header*)
- *JWS* тело (*payload*)
- *JWS* потпис (*signature*)

JWS тело садржи корисне податке у *JSON* формату које је потребно потписати. Подаци тела могу бити у оквиру једног или у оквиру више надовезаних *JSON* објеката.

JOSE заглавље представља објекат који је сачињен од уније следећих вредности:

- *JWS* заштићено заглавље
- *JWS* незаштићено заглавље

Енкован садржај:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P  
Ok6yJV_adQssw5c
```

JWS заглавље:

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

JWS тело:

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

Пример 1 - пример једноставног *JWS*-а

3.2 *JWS* типови серијализације

JWS подржава два типа серијализације: *JWS* компактна серијализација и *JWS JSON* серијализацију.

3.2.1 JWS компакта серијализација

JWS компакта серијализација представља дигитално потписан садржај као компакти УРЛ безбедан стринг. Тај стринг је у следећем формату:

BASE64URL(UTF8(JWS Protected Header)) || ' ' ||

BASE64URL((JWS Payload) || ' ' ||

BASE64URL(JWS Signature)

Овај тип серијализације не подржава вишеструке потписе.

3.2.2 JWS JSON серијализација

Овај тип серијализације такође представља дигитално потписан садржај али за разлику од компактне серијализације, овај тип серијализације није компактан нији је УРЛ безбедан.

JWS добијен коришћењем овог типа серијализације садржи следећа поља:

- *payload* – ово поље је обавезно и садржи тело JWS-а у формату *BASE64URL(JWS Payload)*.
- *signatures* – ово поље представља низ JSON објеката. Сваки објекат представља потпис или примењен MAC над телом и заштићеним заглављем. Сваки елемент овог низа се састоји од JSON објеката у који садрже следеће елементе:
 - *protected* – овај елемент је обавезан и мора да садржи JWS заштићено заглавље у следећем формату *BASE64URL(UTF8(JWS Protected Header))*.
 - *header* – овај елемент је обавезан и садржи JWS незаштићено заглавље у ситуацијама када оно постоји, у ситуацијама када ово заглавље не постоји, ово поље мора бити празно.
 - *signature* – ово поље је обавезно и мора да садржи дигитални потпис у следећем формату *BASE64URL(UTF8(JWS Signature))*.

3.3 JOSE заглавље

JOSE заглавље представља JSON објекат чији чланови ближе описују дигитални потпис или MAC који је примењен над заштићеним заглављем и телом. Називи поља у

овкиру овог објекта морају бити јединствена. Поља која се могу појавити у оквиру заглавља су:

- *"alg"* (*Algorithm*) – заглавље које представља идентификатор криптографског алгоритма који је коришћен да се осигура *JWS*. Уколико ово заглавље није присутно или његова вредност не представља валидан идентификатор алгоритма, дигитални потис неће бити валидан.
- *"jku"* (*JWK Set URL*) – представља URI до ресурса одакле се може прибавити листа јавних кључева од којих један одговара приватном кључу који је коришћен за потписивање *JSW*-а.
- *"jwk"* (*JSON Web Key*) – јавни кључ који одговара приватном кључу који је коришћен за потписивање
- *"kid"* (*Key ID*) – идентификатор кључа који је коришћен за потписивање *JWS*-а.
- *"x5u"* (*X.509 URL*) – урл до ресурса који садржи X.509 сертификат или ланац сертификата који одговара кључу који је коришћен за потписивање *JWS*-а.
- *"x5c"* (*X.509 Certificate Chain*) – сертификат или ланац сертификата који одговарају кључу који је коришћен за потписивање *JWS*-а.
- *"x5t"* (*X.509 Certificate SHA-1 Thumbprint*) – SHA-1 тамбпринт енкодован у основи 64 сертификата који одговара кључу који је коришћен за потписивање *JWS*-а.
- *"x5t#S256"* (*X.509 Certificate SHA-256 Thumbprint*) – SHA-256 тамбпринт енкодован у основи 64 сертификата који одговара кључу који је коришћен за потписивање *JWS*-а.
- *"typ"* (*Type*) – параметар који се може користити да се назначи медиа тип *JWS*-а.
- *"cty"* (*Content Type*) – параметар који се може користити да се означи медиа тип садржаја тела *JWS*-а
- *"crit"* (*Critical*) – параметар кроз који се може нагласити да се у заглављу налазе параметри који нису покривени спецификацијом. Овај параметар садржи листу имена додатих параметара.

3.4 Генерисање и коришћење *JWS*-а

Да би се креирао *JWS*, следећи кораци се морају применити:

1. Креира се садржај који ће се користити као тело *JWS*-а.
2. Израчуна се енкодovана вредност тела *BASE64URL(JWS Payload)*
3. Креира се *JSON* објекат који садржи жељене параметре заглавља.
4. Израчуна се енкодovана вредност заглавља
BASE64URL(UTF8(JWS Protected Header))
5. У зависности од алгорита који се користи за потписивање, израчуна се *JWS* потпис над садржајем који је потребно потписати. Садржај који се потписује је:
ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload))
6. Израчуна се енкодovани потпис у облику *BASE64URL(JWS Signature)*.
7. Ако се користи *JWS JSON* серијализација, кораци 3-6 се понављају за сваки дигитални потпис или *MAC*.
8. У зависности од тога који се тип серијализације користи, на крају се креира *JWS* у одговарајућем формату.

Код валидације *JWS*-а је потребно проверити да ли је потпис или *MAC* валидан. У случају да постоји више потписа или *MAC*-ова апликација која проверава потписе може да одабере да ли је довољно да један потпис буде валидан да би *JWS* био валидан, или је потребно да се сви потписи успешно валидирају како би *JWS* био валидан. Валидација *JWS*-а подразумева следеће акције:

1. Парсира се добијени *JWS* како би се добиле серијализоване вредности компонента *JWS*-а.
2. Декодују се енкодovана репрезентација заштићеног заглавља у основи 64.
3. Проверава се да ли је резултујућа секвенца валидна UTF-8 енкодovана репрезентација *JSON* објекта.
4. Уколико се користи компактна серијализација тада ће заглавље имати вредност *JWS* заштићеног заглавља. У случају *JWS JSON* серијализације заглавље ће имати вредност уније *JWS* заштићеног заглавља и *JWS* незаштићеног заглавља.

5. Проверава се да ли валидатор уме да валидира сва додатна поља заглавља наведена у *crit* параметру заглавља.
6. Декодује се енкована репрезентација *JWS* тела у основи 64.
7. Декодује се енкована репрезентација *JWS* потписа у основи 64.
8. Вредност *JWS* потписа се упоређује са следећом вредношћу:
ASCII(BASE64URL(UTF8(JWS Protected Header))) || '.' || BASE64URL(JWS Payload)).
Начин упоређивања зависи од алгорита који је коришћен при потписивању.
Тип алгорита се налази у *alg* параметру заглавља.
9. Уколико се користи *JWS JSON* серијализација поступак 4-8 се понавља за сваки дигитални потпис или *MAC* вредност која се налази у *JWS*-у.

3.5 Предности *JWS*-а у односу на *XMLDSig*

Када је у питању поређење ове две спецификације, јако је тешко дати некој од њих предност. *XMLDSig* је доста дуже присутан од *JWS*-а и неки од делова *JWS* спецификације су писани управо по узору на *XMLDSig* спецификацију. Када је у питању сама величина потписа, предност имају *JWS* потписи који, поготово када се користи *JWS* компактна серијализација, могу имати и до 10 пута мању величину од *XMLDSig* потписа истог садржаја.

4 Спецификација имплементације

У овом поглављу ће бити изложена спецификација система који развијен и који имплементира ФИДО2 спецификацију. Сама имплементација представља систем који служи као менаџер идентитета и аутентификациони сервер и који је погодно користити када више различитих апликација које имају потребу за управљањем идентитетима корисника желе да тај посао делегирају другој апликацији. Такође, у оквиру саме апликације ће бити подржана ауторизација ресурса од стране корисника коришћењем потписивања ресурса помоћу ФИДО2 аутентификатора као и коришћењем потписивања помоћу паметних картица.

4.1 Архитектура система

Имплементирани систем се састоји из три целине које представљају три засебне апликације:

- ФИДО2 менаџер идентитета и аутентификациони сервер који представља веб апликацију која се користи као сервер за управљање идентитетима корисника и аутентификацију корисника
- Тестна веб апликација која користи имплементирани менаџер идентитета за управљање идентитетима својих корисника
- Десктоп апликација која се користи за потписивање помоћу паметних картица

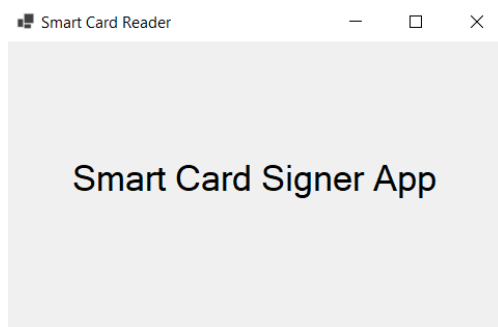
4.2 Апликација за потписивање помоћу паметних картица

Потписивање помоћу паметних картица је реализовано помоћу механизма приватних и јавних кључева. Приступ приватом кључу са паметних картица је могућ само са рачунара са којим је корисник повезао паметну картицу. Овим је обезбеђена сигурност корисничких креденцијала на паметној картици али и онемогућено једноставно потписивање ресурса путем интернета. Како веб претраживачи не

обезбеђују интерфејс који је могуће користити у сврху потписивања ресурса на интернету, било је потребно развити десктоп апликацију која би примила податке о ресурсу који је потребно потписати, те податке потписала и потписане податке вратила назад.

4.2.1 Интерфејс апликације

Интерфејс реализоване десктоп апликације је тривијалан и садржи само натпис са називом апликације.



Слика 4 - интерфејс апликације за потписивање помоћу паметне картице

4.2.2 Функционалности апликације

Апликација подржава потписивање добијеног садржаја коришћењем *JWS* стандарда и приватног кључа са паметне картице коју је корисник повезао помоћу читача са рачунаром на којем је покренута апликација за потписивање.

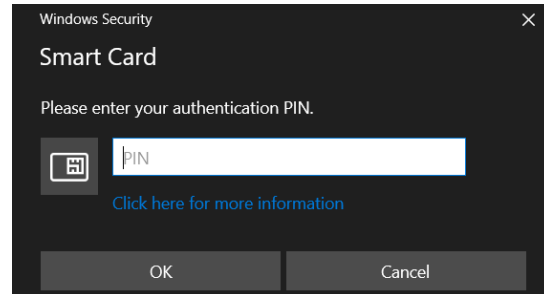
При покретању, апликација у позадини покреће мини сервер на предефинисаном порту који се користи за комуникацију са спољашњим светом. У оквиру мини сервера је имплементиран веб АПИ са методом `POST /signature`. Ова метода има само један обавезни параметар `payload` типа стринг који представља стринг репрезентацију ресурса који је потребно потписати. Када корисник жели да потпише ресурс паметном картицом, из веб претраживача се пошаље захтев ка овој методи која:

1. Прочита сертификат са паметне картице, при чему се у оквиру оперативног система кориснику прикаже интерфејс путем кога је потребно да унесе пин код

којим је заштићен приватни кључ на картици, уколико је приватни кључ заштићен;

2. Коришћењем приватног кључа са паметне картице потпише примљени садржај;

3. Потписани садржај се заједно називом алгорима помоћу кога је извршено потписивање и са сертификатом који садржи јавни кључ који је повезан са приватним кључем са којим је извршено потписивање пошаље назад издаваоцу захтева.



Слика 5 - интерфејс за унос пина

```
CspParameters cspParameters = new CspParameters(1, smartCardProviderName);
RSACryptoServiceProvider rsaProvider = new RSACryptoServiceProvider(cspParameters);
string pubKeyXml = rsaProvider.ToXmlString(false);

// Find the certificate in the CurrentUser\My store that matches the public key
X509Store x509Store = new X509Store(StoreName.My, StoreLocation.CurrentUser);
x509Store.Open(OpenFlags.ReadOnly | OpenFlags.OpenExistingOnly);
X509Certificate2 certificate = null;
foreach (X509Certificate2 cert in x509Store.Certificates)
{
    if ((cert.PublicKey.Key.ToXmlString(false) == pubKeyXml))
        if (cert.HasPrivateKey)
            certificate = cert;
}
return certificate;
```

Пример 2 - пример кода за читање сертификата са паметне картице

```
switch (certificate.PrivateKey.SignatureAlgorithm)
{
    case "RSA":
        var privateKey = certificate.GetRSAPrivateKey();
        string token = Jose.JWT.Encode(payload, privateKey, JwsAlgorithm.RS256);
        return (token, "RSA");
    case "EC":
        var ecPrivateKey = certificate.GetECDsaPrivateKey();
        string ecToken = Jose.JWT.Encode(payload, ecPrivateKey,
            JwsAlgorithm.ES256);
        return (ecToken, "EC");
}
return (null, null);
```

Пример 3 - пример кода за потписивање садржаја помоћу паметних картица

Назив параметра	Тип	Обавезан	Опис
Token	Стринг	Да	JSON веб токен у формату енкодованом у бази 64 који садржи садржај који је било потребно потписати и потписани садржај.
certificate	Стринг	Да	Сертификат енкодован у бази 64 који садржи информације о јавном кључу који је повезан са приватним кључем са којим је извршено потписивање садржаја.
algorithm	Стринг	Да	Идентификатор алгоритма којим је садржај потписан.

Табела 1 - параметри одговора методе POST /signature

4.3 ФИДО2 менаџер идентитета

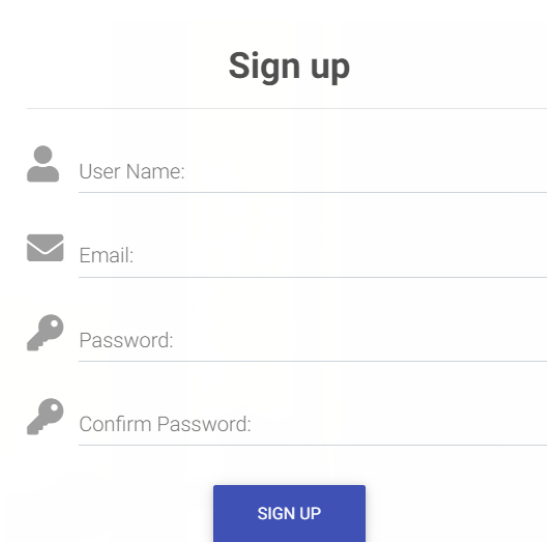
Ова апликација имплементира основне операције за управљање корисничким подацима и корисничким креденцијалима. Подржане операције су:

- Регистрација корисника;
- Логин корисника
 - Логин помоћу корисничког имена и лозинке,
 - Двофакторски логин помоћу корисничког имена, лозинке и једног од претходно регистрованих ФИДО2 аутентификатора,
 - Логин помоћу корисничког имена и једног од претходно регистрованих ФИДО2 аутентификатора;
- Управљање ФИДО2 аутентификаторима
 - Регистравање корисничких креденцијала у ФИДО2 аутентификатору и повезивање аутентификатора са корисничим налогом,
 - Брисање регистрованог ФИДО2 аутентификатора за улогованог корисника;

- Управљање паметним картицама
 - Регистровање нове паметне картице за улогованог корисника,
 - Брисање регистровано паметне картице за улогованог корисника;
- Потписивање ресурса које у овој имплементацији представљају фиктивна плаћања
 - Потписивање плаћања коришћењем претходно регистрованог ФИДО2 аутентификатора,
 - Потписивање плаћања путем приватног кључа који је повезан са претходно регистрованим паметном картицом која садржи приватни кључ.

4.3.1 Почетна страна апликације

Навигацијом на почетну страну апликације кориснику постају доступне две форме: форма за регистрацију новог корисника и форма за логин постојећег корисника.



Слика 6 - форма за регистрацију

У оквиру форме за регистрацију корисника, потребно је унети корисничко име, и-мејл адресу и корисничку лозинку. У случају да су унети валидни подаци (јединствено корисничко име и лозинка која задовољава полисе апликације) за корисника се креира нови кориснички налог. Након регистровања корисничког налога, корисник се може уловати у систем коришћењем корисничког имена и лозинке.

Када је у питању логин корисника у систем разликују се два случајева коришћења: логовање корисника са регистрованим минимум једним ФИДО2 аутентификатором и логовање корисника који нема регистровани ФИДО2 аутентификатор. Корисник који нема регистроване ФИДО2 аутентификаторе се у систем логује путем стандардне форме за логин у којој је

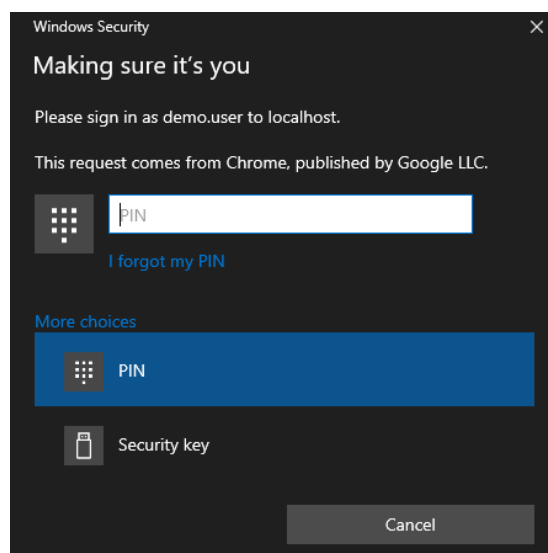
потребно да унесе своје корисничко име и лозинку. У оквиру ове форме корисник може да изабере да ли жели да његов логин буде запамћен у веб претраживачу или не. У случају када корисник има регистрован макар један ФИДО2 аутентификатор могу се разликовати два подслучаја: логин коришћењем двофакторске аутентификације и логин без лозинке.

Слика 7 - форма за логин

Уколико корисник жели да се улогује коришћењем двофакторске аутентификације, потребно је да понови идентичан унос као када је у питању логин коришћењем само лозинке. Овај унос представља први фактор аутентификације. Након успешног првог фактора аутентификације клијентска апликација контактира сервер како би од сервера добила објекат типа

`PublicKeyCredentialRequestOptions` који је потребно проследити *JavaScript* [6]

функцији `navigator.credentials.get()` како би се покренула церемонија аутентификације. Овај објекат садржи поље *Challenge* чији је садржај на серверској страни добијен коришћењем генератора случајних бројева. Садржај овог поља се кроз поменути објекат шаље аутентификатору на потписивање. Након покретања церемоније, кориснику се приказује интерфејс у оквиру кога може да изабере аутентификатор који жели да користи за додатни фактор аутентификације након чега је потребно да се изврши аутентификација са изабраним аутентификатором. Након извршене корисничке интеракције са аутентификатором одговор аутентификатора се прослеђује серверу на валидацију. Сервер валидира да ли је коришћени



Слика 8 - интерфејс за интеракцију са ФИДО2 аутентификатором

аутентификатор регистрован за корисника који жели да се улогује. Уколико јесте, након тога се коришћењем регистрованог јавног кључа декодује садржај потписан у аутентификатору и упоређује са послатим садржајем од стране сервера (параметром *Challenge*). Уколико су послати садржај и декодовани потписани садржај исти, аутентификација је успешна и корисник бива улогован.

Слика 9 - форма за логин без лозинке

Кликом на клизач *Passwordless* у оквиру логин форме, корисник може да изабере да ли жели да се улогује без лозинке, коришћењем само регистрованог ФИДО2 аутентификатора.

Акције које је потребно да изврши корисник да би се улоговао су идентичне као и у случају двофакторске аутентификације осим што сада корисник не уноси своју корисничку лозинку.

4.3.2 Управљање регистрованим ФИДО2 аутентификаторима

Кликом на дугме *Devices* у заглављу веб апликације, улоговани корисник одлази на страну за управљање регистрованим ФИДО аутентификаторима. У оквиру ове стране кориснику је доступна листа регистрованих аутентификатора. На овој страни корисник може да уклони регистровани ФИДО2 аутентификатор или да дода нови ФИДО2 аутентификатор. Како би се уклонио регистровани ФИДО2 аутентификатор, потребно је да корисник кликне на дугме *Remove* поред аутентификатора који жели да уклони. Након тога ће овај аутентификатор бити обрисан из базе и корисник неће више моћи да користи креденцијале са тог аутентификатора у апликацији.

Кликом на дугме *Register New Device* корисник покреће процес регистравања новог ФИДО2 аутентификатора. На почетку апликација са клијентске стране контактира сервер како би од сервера добила објекат типа `PublicKeyCredentialCreationOptions` који је потребно проследити *JavaScript* функцији `navigator.credentials.create()` како би се покренула церемонија издавања новог креденцијала у аутентификатору. Након тога кориснику се од стране оперативонг система рачунара прикаже интерфејс

путем кога је потребно да корисник изврши одређену акцију са аутентификатором (додиривање *USB* кључа, читавање отиска прста или препознавање лица) како би ауторизовао акцију издавања новог креденцијала на аутентификатору. У случају да је церемонија успешно завршена, серверу се шаље резултат у виду објекта *AuthenticationAttestationResponse* који је клијент добио од аутентификатора. На серверској страни се проверава и валидира садржај овог објекта. Валидација одговора подразумева проверу да ли је идентичан аутентификатор већ додат за корисника и валидацију потписа коришћењем креираних креденцијала. Ово упоређивање се ради тако што се потписани садржај декодује добијеним јавним кључем и упоређује са оригиналним садржајем. Уколико је валидација успешна, новогенерисани креденцијали ће бити повезани са корисничким налогом и сачувани у бази како би корисник могао у будућности да их користи. Да би се спречило да потенцијални нападач пресретне саобраћај и уместо оригиналног одговора од аутентификатора уметне неки други садржај (нпр. одговор који је нападач добио од аутентификатора који поседује, што би му омогућило да се са тим аутентификатором улогује као корисник који је жртва) у оквиру објекта *PublicKeyCredentialCreationOptions* се налази поље *Challenge*. Садржај овог поља је низ бајтова генерисан од стране сервера коришћењем нпр. генератора случајних бројева. Садржај овог поља се заједно са информацијама о аутентификатору и информацијама о кориснику за којег се издају креденцијали потписује новогенерисаним приватним кључем у аутентификатору. При валидацији одговора аутентификатора на серверској страни, проверава се да ли је вредност поља *Challenge* иста као вредност која је послата. Уколико није сматра се да је неко покушао да уметне другачији садржај и креденцијали неће бити додати.

Registered authenticators

Registration Date	AAGuid	Credential ID	
18-Sep-19 20:05:47	08987058-cadc-4b81-b6e1-30de50dcbe96	oosDOIZW7AUec/AtUmjilDsDqYXuOpRSXThHhQeMmNA=	REMOVE
18-Sep-19 20:06:50	6d44ba9b-f6ec-2e49-b930-0c8fe920cb73	dz8eSppbzR2Tf4ULY+AwTyL58ZiDI1yToN1yz9paZVxzglCgrjyCeqJa2jM3C2/vAxR/AUkJqa+OibI9OgsJQw==	REMOVE

REGISTER NEW DEVICE

Слика 10 - страна за управљање регистрованим ФИДО2 аутентификаторима

4.3.3 Управљање регистрованим паметним картицама

Кликом на дугме *Certificates* у заглављу веб апликације кориснику се приказује страна са регистрованим паметним картицама. Паметне картице се идентификују X509 Сертификатима које поседују у себи. У оквиру ове стране кориснику је доступна листа регистрованих паметних картица. Уколико жели да уклони регистровану паметну картицу, потребно је да корисник кликне на дугме *Remove* поред детаља о сертификату паметне картице коју жели да уклони.

Registered certificates

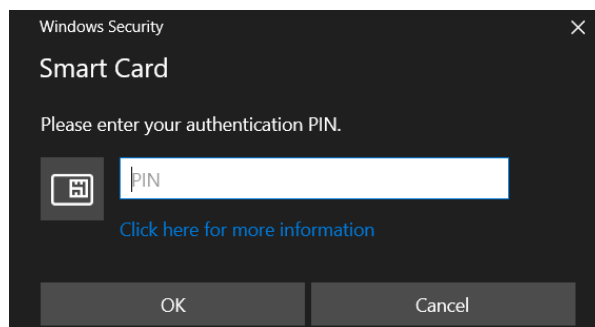
Subject		Registration Date	Thumbprint	Credential ID	
CN=МЛАЂАН МИХАЈЛОВИЋ	C=RS	18-Sep-19 20:10:20	5A928C90CB6685CEB77AFAD30C3C17DC4CAE9E36		REMOVE

Слика 11 - страна за управљање регистрованим паметним картицама

Уколико корисник жели да региструје нову паметну картицу, потребно је да кликне на дугме *Register New Certificate*.

Кликом на ово дугме шаље се захтев серверу како би се добио случајан садржај који је потребно да корисник потпише са приватним кључем са паметне картице, како би ту паметну картицу регистровао за даље коришћење. Након тога се кориснику

приказује прозор са информацијом да је потребно да укључи апликацију за потписивање помоћу паметних картица. Када корисник покрене десктоп апликацију, потребно је да кликне на дугме *Authorize* како би десктоп апликацији био послат садржај који је потребно потписати. Тада се у оквиру



Слика 12 - интерфејс за унос ПИН кода паметне картице

оперативног система кориснику приказује интерфејс путем кога је потребно да унесе ПИН код којим је паметна картица заштићена како би се извршило потписивање. Када клијент добије одговор од десктоп апликације, тај одговор се прослеђује серверу који декодује потпис јавним кључем који је добио у одговору и проверава да ли је потписани садржај идентичан послатом садржају. Уколико је валидација успешна, у систему ће се за корисника регистровати нови сертификат са паметне картице који корисник у будућности може користити да за потписивање ресурса.

4.3.4 Ауторизација ресурса

На страну за ауторизацију ресурса (у наставку плаћања) корисник долази путем тестне клијентске апликације. Више о томе ће бити речено у поглављу о тој апликацији. На овој страни корисник види детаље плаћања које потписује и може да изабере на који начин жели да потпише плаћање и самим тим изврши ауторизацију плаћања. Плаћање може бити ауторизовано коришћењем регистрованог ФИДО2 аутентификатора или коришћењем приватног кључа са регистроване паметне картице.

Уколико корисник изабере да ауторизује плаћање помоћу ФИДО2 аутентификатора, тада су кораци које је потребно одрадiti идентични као и код логина корисника без коришћења корисничке лозинке осим што се сада садржај поља *Challenge* који се потписује приватним кључем у аутентификатору не добија коришћењем генератора случајних бројева, већ ово поље садржи енкодоване информације о плаћању које се ауторизује у основи 64. У случају успешне церемоније аутентификације корисника, плаћање ће бити ауторизовано.

Уколико корисник изабере да ауторизује плаћање помоћу паметне картице, тада су кораци које је потребно одрадити идентични као и код процеса регистрације нове паметне картице. Разлика је у томе што овог пута садржај који се шаље на потписивање десктоп апликацији није садржај добијен на случајан начин већ представља *JSON* објекат који садржи информације о плаћању које се потписује. У случају успешне валидације потписа, плаћање ће бити ауторизовано.

Payment Identifier
afb60d5e-d249-4dd4-a47d-42adcb52ec20

Creditor Account
RS12308

Debtor Account
RS32453

Subject
726ea1ab-c662-4554-ace5-3c7340bfe1c7

☒ Sign using FIDO2 Authenticator
☐ Sign using smart card

AUTHORIZE PAYMENT

Слика 13 - страна за избор начина ауторизације плаћања

4.4 Тестна клијентска апликација

Идеја ове апликације јесте демонстрација како нека апликација може користити ФИДО2 менаџер идентитета за управљање корисничким налозима својих корисника. Почетна страна ове апликације не садржи никакав садржај осим заглавља са два дугмета: *Privacy* и *Home*. Почетна страна је дозвољена за неаутентификоване кориснике док страни *Privacy* могу приступити само аутентификовани корисници. Кликом на дугме *Privacy* се покреће *OAuth2* [7] процес и корисник бива редиректован на страну за логин на ФИДО2 менаџер идентитета. Када се корисник на овој страни успешно улогује, он бива редиректован назад на тестну клијентску апликацију. Сада у оквиру ове апликације може видети основне податке о свом налогу и може отићи на страну за преглед, иницирање и ауторизацију плаћања. Навигацијом на ову страну корисник може видети до тог тренутка иницирана плаћања, може да иницира ново плаћање и да покрене ауторизацију плаћања.

Кликом на дугме за иницирање плаћања кориснику се приказује једноставна форма где је потребно унети податке о плаћању. Прослеђени садржај се не валидира и у ову форму се може унети било шта. Када корисник иницира плаћање ово плаћање ће бити приказано у листи иницираних плаћања.

Кликом на дугме за ауторизацију плаћања корисник покреће процес ауторизације плаћања. У оквиру овог процеса корисник ће бити редиректован на одговарајућу страну на ФИДО2 менаџеру идентитета у оквиру које може ауторизовати плаћање. Након успешне ауторизације плаћања корисник ће бити редиректован назад на теснту клијентску апликацију.

Initiate payment						
Payment ID	Creditor Name	Creditor Account	Debtor Name	Debtor Account	Request Time	Authorization Date
0061007d-5b2a-4f7f-9187-4a7837d5fd64	Demouser	RS12308	Userdemo	Userdemo	18-Sep-19 20:26:14	Authorize

Слика 14 - интерфејс тестне клијентске апликације

5 Имплементациони детаљи

5.1 Коришћене технологије

ФИДО2 менаџер идентитета и тестна клијентска апликација су развијени у програмском језику *C#* коришћењем *.NET Core 2.2* [8] фрејмворка. Клијентска страна ових апликација је реализована помоћу *Razor View Engine*-а [9]. Део апликације који се извршава на клијентској страни а који се комуникације са аутентификатором, прослеђивања параметара од сервера до аутентификатора и назад, као и потписивања ресурса паметним картицама је реализован у *JavaScript* програмском језику. Десктоп апликација за потписивање помоћу паметних картица је такође развијена у програмском *C#* али коришћењем *.NET Core 3.0* фрејмворка. Као регистар пакета у овим пројектима је употребљен *NuGet Package Manager* који обезбеђује једноставно инсталирање и коришћење готових библиотека за *C#* програмски језик. *NuGet* пакети који су употребљени у овим апликацијама су:

- *Fido2* – библиотека која обезбеђује операције за валидацију ФИДО2 креденцијала;
- *Fido2.Models* – библиотека које обезбеђује моделе потребне да би се користила *Fido2* библиотека;
- *IdentityServer4* [10] – библиотека која садржи имплементацију *OpenID Connect* [11] и *OAuth 2.0* фрејмворка за *ASP.NET Core*;
- *jose-jwt* – библиотека која обезбеђује операције за креирање, потписивање и валидацију *JWT*-а;
- *IdentityServer4.AspNetIdentity* – библиотека која пружа неопходне операције и моделе за рад са идентитетима корисника.

5.2 Дизајн базе података

За базу података је због своје једноставности одабран *SQLite* [12]. Развијени систем користи пет табела за чување података:

- Табела за смештање корисничких налога
- Табела за смештање информација о регистрованим аутентификаторима
- Табела за смештање информација о сертификатима са регистрованих паметних картица
- Табела за смештање фиктивних плаћања који се користе као ресурси које је потребно ауторизовати
- Табела за смештање ауторизација плаћања

Табела за чување корисничких налога:

Поље	Тип	Опис
<i>Id</i>	Гуид стринг	Јединствени идентификатор корисника у систему
<i>UserName</i>	Стринг	Јединствено корисничко име
<i>Password</i>	Стринг	Корисничка лозинка
<i>Email</i>	Стринг	И-мејл адреса корисник

Табела 2 - приказ модела за чување корисничких података

Табела за чување информација о регистрованим аутентификаторима:

Поље	Тип	Опис
<i>Id</i>	Стринг	Јединствени идентификатор регистрације
<i>UserId</i>	Стринг	Идентификатор корисника за који су регистровани креденцијали
<i>PublicKeyId</i>	Стринг	Идентификатор јавног кључа са аутентификатора у енкодованом облику у бази 64
<i>PublicKeyIdBytes</i>	Низ бајтова	Идентификатор јавног кључа са аутентификатора
<i>PublicKey</i>	Низ бајтова	Јавни кључ
<i>UserHandle</i>	Низ бајтова	Идентификатор корисника за који су креирани креденцијали у аутентификатору генерисан од стране аутентификатора
<i>SignatureCounter</i>	Цели број	Бројач потписа у аутентификатору

<i>CredentialType</i>	Стринг	Тип генерисаних креденцијала
<i>RegistrationDate</i>	Датум	Датум генерисања и регистрације креденцијала
<i>AaGuid</i>	Стринг	Јединствени идентификатор аутентификатора

Табела 3 - приказ модела за чување података о регистрованим ФИДО2 аутентификаторима

Табела за чување информација о сертификатима са регистрованих паметних картица:

Поље	Тип	Опис
<i>Id</i>	Стринг	Јединствени идентификатор регистрације
<i>UserId</i>	Стринг	Идентификатор корисника за који су регистровани креденцијали
<i>Thumbprint</i>	Стринг	Идентификатор сертификата са регистроване паметне картице
<i>Certificate</i>	Стринг	Сертификат са регистроване паметне картице у енкодованом формату у бази 64
<i>RegistrationDate</i>	Датум	Датум регистрације
<i>Subject</i>	Стринг	Поље <i>Subject</i> из сертификата

Табела 4 - приказ модела за чување информација о регистрованим паметним картицама

Табела за чување фиктивних плаћања:

Поље	Тип	Опис
<i>Id</i>	Стринг	Јединствени идентификатор плаћања
<i>UserId</i>	Стринг	Идентификатор корисника за који су регистровани креденцијали
<i>CreditorAccount</i>	Стринг	Број рачуна уплатиоца
<i>DebtorAccount</i>	Стринг	Број рачуна примаоца
<i>CreditorName</i>	Датум	Име уплатиоца
<i>DebtorName</i>	Стринг	Име примаоца
<i>Amount</i>	Реалан број	Износ плаћања

<i>Status</i>	Стринг	Статус плаћања
<i>RequestDateTime</i>	Датум	Датум креирања плаћања

Табела 5 - приказ модела за чување података о фиктивним плаћањима

Табела за чување ауторизација плаћања:

Поље	Тип	Опис
<i>Id</i>	Стринг	Јединствени идентификатор ауторизације
<i>PaymentId</i>	Стринг	Идентификатор плаћања на које се односи ауторизација
<i>AuthenticatorData</i>	Стринг	Подаци о ФИДО2 аутентификатору који улазе у потпис
<i>Type</i>	Цео број	Тип потписа: 0 – ФИДО2, 1 – Паметна картица
<i>Potpis</i>	Датум	Потпис плаћања
<i>ClientData</i>	Стринг	Подаци о кориснику добијени од ФИДО2 аутентификатора који улазе у потпис
<i>PublicKeyId</i>	Стринг	Идентификатор јавног кључа који се користи за валидацију потписа
<i>AuthorizationDateTime</i>	Датум	Датум када је извршена ауторизација

Табела 6 - приказ модел за чување података о ауторизацијама плаћања

6 Закључак

У раду је приказана имплементација ФИДО2 спецификације на примеру апликације која представља менаџер идентитета и аутентификациони сервер. У том циљу изучени су *CTAP* протокол и *WebAuthn* спецификација који предстаљају градивне блокове ФИДО2 спецификације. Изнети су тренутни проблеми у вези са аутентификацијом корисника као и основне идеје и потреба за ФИДО2 спецификацијом.

Током развоја приказаног система се тежило ка томе да се без превеликог усложњавања развијени систем приближи системима који се користе у пракси. С тим у вези развијени систем приказује тренутно најзаступљеније видове аутентификације корисника и потписивања ресурса на интернету као и реализацију аутентификације и потписивања ресурса коришћењем ФИДО2 спецификације.

Из приказаних случајева коришћења се долази до закључка да ФИДО2 пружа много већу сигурност када је у питању аутентификација корисника и да на више различитих начина отежава и онемугућује нападе који би резултовали крађом идентитета или података о кориснику. На примеру развијене десктоп апликације за потписивање која ресурсе потписује коришћењем *JWS*-а и реализације потписивања ресурса помоћу ФИДО2 спецификације се примећује да ФИДО2 пружа неупоредиво лакиш и сигурнији интегрисани приступ за ауторизацију и потписивање ресурса који не подразумева коришћење додатних апликација или екстензија за веб претраживаче.

7 Коришћени термини и скраћенице

У циљу формалнијег приступа уводе се следећи појмови:

- *CTAP – Client to Authenticator protocol*, један од два саставна дела ФИДО2 протокола
- *WebAuthn - W3C Web Authentication specification*, један од два саставна дела ФИДО2 протокола
- Аутентификатор је уређај који имплементира *CTAP* протокол и који за циљ има детекцију корисничке интеракције;
- Поуздана страна (*Relying party*) – може представљати интернет претраживач, десктоп или мобилну апликацију;
- Потврда аутентификатора – потврда коју издаје аутентификатор поузданој страни, која поузданој страни гарантује да је корисник аутентификован;
- Атестација – процес генерисања креденцијала на аутентификатору;
- Клијент – веб претраживач, паметни телефон;
- Церемонија – процес у коме корисник, клијент кога корисник користи и који је притом повезан са макар једним аутентификатором, раде у спрези како би помоћу криптографских алата доказали поузданој страни да корисник контролише приватни кључ за који је код поуздане стране претходно регистрован одговарајући јавни кључ;
- *JWS – JSON Web Signature*, *JSON* веб потпис
- *JSON – JavaScript Object Notation*, објекти представљени *JavaScript* нотацијом
- *BASE64URL* – Операција енковања прослеђеног параметра у бази 64
- *MAC – Message Authentication Code*

8 Литература

- [1] „Fido Alliance,“ [На мрежи]. Available: <https://fidoalliance.org>. [Последњи приступ 16 Septembar 2019].
- [2] „World Wide Web Consortium,“ [На мрежи]. Available: <https://www.w3.org/>. [Последњи приступ 16 Septembar 2019].
- [3] „Guide to WebAuthn,“ 2019. [На мрежи]. Available: <https://webauthn.guide/>. [Последњи приступ 12 Septembar 2019].
- [4] „Client to Authenticator Protocol (CTAP),“ 2019.
- [5] C. Bormann, Universitaet Bremen TZI, P. Hoffman, VPN Consortium, „Concise Binary Object Representation (CBOR),“ October 2013. [На мрежи]. Available: <https://tools.ietf.org/html/rfc7049>. [Последњи приступ 12 Septembar 2019].
- [6] „JavaScript,“ [На мрежи]. Available: <https://www.javascript.com/>. [Последњи приступ 18 Septembar 2019].
- [7] „OAuth 2.0,“ [На мрежи]. Available: <https://oauth.net/2/>. [Последњи приступ 12 Septembar 2019].
- [8] „.NET Core,“ [На мрежи]. Available: https://en.wikipedia.org/wiki/.NET_Core. [Последњи приступ 18 Septembar 2019].
- [9] R. Anderson, L. Latham, T. Mullen и D. Vicarel, „Razor syntax reference for ASP.NET Core,“ Microsoft, 9 Maj 2019. [На мрежи]. Available: <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-2.2>. [Последњи приступ 18 Septembar 2019].
- [10] „Identity Server 4,“ Identity Server, [На мрежи]. Available: <https://identityserver4.readthedocs.io/en/latest/>. [Последњи приступ 18 Septembar 2019].
- [11] „Welcome to OpenID Connect,“ [На мрежи]. Available: <https://openid.net/connect/>. [Последњи приступ 18 Septembar 2019].
- [12] „SQLite,“ SQLite Consortium, [На мрежи]. Available: <https://www.sqlite.org/index.html>. [Последњи приступ 18 Septembar 2019].
- [13] „FIDO2 Passwordless authentication,“ 2019. [На мрежи]. Available:

<https://www.yubico.com/solutions/fido2/>. [Последњи приступ 12 Septembar 2019].

[14] „Web Authentication: An API for accessing Public Key Credentials,“ [На мрежи]. Available: <https://w3c.github.io/webauthn>. [Последњи приступ 11 Septembar 2019].

[15] Jones, Microsoft, Bradley, Sakimura и NRI, „RFC 7517 - JSON Web Signature (JWS),“ May 2015. [На мрежи]. Available: <https://tools.ietf.org/html/rfc7515>. [Последњи приступ 11 Septembar 2019].

[16] A. Liao и I. Damla, „Introducing Web Authentication in Microsoft,“ 2018 July 30. [На мрежи]. Available: <https://blogs.windows.com/msedgedev/2018/07/30/introducing-web-authentication-microsoft-edge/>. [Последњи приступ 11 Septembar 2019].

9 Додатак

9.1 Модели

9.1.1 Модел PublicKeyCredentialCreationOptions

У табели су дати параметри објекта који се прослеђује методи за креирање кренденцијала.

Назив параметра	Тип података	Обавезан	Дефиниција
clientDataHash	Низ бајтова	Да	Хеш клијентских података који зависе од контекста специфицираног од стране домаћина
rp	PublicKeyCredentialRpEntity	Да	Ова структура података описује поуздану страну за коју се издаје нови јавни кључ. Она може опционо садржати назив поуздане стране или URL до иконе за поуздану страну.
user	PublicKeyCredentialUserEntity	Да	Ова структура података описује кориснички налог за који ће нови јавни кључ бити издат код поуздане стране. Она садржи идентификатор корисника који мора бити јединствен за поуздану страну код које се корисник региструје. Ова структура може опционо садржати и корисничко име, име и презиме корисника, URL до аватара корисника итд.
publicKeyCredParams	CBOR niz	Да	Секвенца CBOR мапи које садрже парове типова јавног кључа и криптографског алгоритма. Овај параметар се користи како би клијент знао које аутентификаторе

			је корисник до сада регистровао код поуздане стране и како би онемогућио да корисник поново региструје креденцијале са истог аутентификатора.
excludeList	Секвенца објеката типа PublicKeyCredentialDescriptor	Не	Листа идентификатора јавних кључева које је потребно изузети. Уколико аутентификатор садржи неки кључ из ове листе, вратиће грешку.
extensions	CBOR мапа идентификатора екстензија	Не	Додатни параметри за аутентификатор.
options	Мапа опција аутентификатора.	Не	Додатне опције за аутентификатор кроз које се може захтевати да аутентификатор захтева интеракцију корисника (унос ПИН кода, скенирање отиска прста) или се може нагласити аутентификатору да сачува информације о кључу.
challenge	Низ бајтова	Да	Порука генерисана од стране сервера која се шаље аутентификатору на потписивање.
pinAuth	Низ бајтова	Не	
pinProtocol	Неозначени цео број	НЕ	

Табела 7 – параметри методе navigator.credentials.create

9.1.2 Модел *PublicKeyCredentialRpEntity*

Ентитет који садржи податке о поузданој страни која издаје захтеве аутентификатору.

Назив параметра	Тип података	Обавеза н	Дефиниција
Id	Стринг	Да	Идентификатор поуздане стране.
Name	Стринг	Не	Назив ентитета који шаље захтев

Табела 8 - Параметри модела *PublicKeyCredentialRpEntity*

9.1.3 Модел *PublicKeyCredentialUserEntity*

Ентитет који садржи податке о кориснику за којег се издаје захтев.

Назив параметра	Тип података	Обавеза	Дефиниција
		н	
id	Низ бајтова	Да	Идентификатор поуздане стране.
displayName	Стринг	Не	Назив корисника за којег се шаље захтев

Табела 9 - Параметри модела *PublicKeyCredentialUserEntity*

9.1.4 Модел *PublicKeyCredentialDescriptor*

Ентитет који садржи атрибуте који ближе одређују тип аутентификатора који је потребно користити током церемоније.

Назив параметра	Тип података	Обавеза	Дефиниција
		н	
id	Низ бајтова	Да	Идентификатор поуздане стране.
displayName	Стринг	Не	Назив корисника за којег се шаље захтев

Табела 10 - Параметри модела *PublicKeyCredentialDescriptor*

9.1.5 Модел *PublicKeyCredentialParameters*

Назив параметра	Тип података	Обавеза	Дефиниција
		н	
type	Стринг енумерације	Да	Тип креденцијала. Тренутно једина подржана вредност је <i>public-key</i> .
alg	Цео број	Да	Идентификатор алгоритма за потписивање са којим ће новокреидани креденцијал бити коришћен.

Табела 11 - Параметри модела *PublicKeyCredentialParameters*

9.1.6 Модел *PublicKeyCredentialRequestOptions*

У табели су дати параметри објекта који се прослеђује методи за издавање потврде о аутентификацији корисника.

Назив параметра	Тип података	Обавезан	Дефиниција
rpId	Стринг	Да	Идентификатор поуздане стране.
clientDataHash	Низ бајтова	Да	Хеш клијентских података који зависи од контекста специфицираног од стране домаћина
allowList	Секвенца објеката типа PublicKeyCredentialDecriptors		Уколико ова листа садржи један или више дескриптора, аутентификатор мора изгенерисати потврду о аутентификацији коришћењем једног од наведених дескриптора.
extensions	CBOR мапа идентификатора екстензија	Не	Додатни параметри за аутентификатор.
publicKeyCredParams	CBOR niz	Да	Секвенца CBOR мапи које садрже парове типова јавног кључа и криптографског алгорита .
options	Мапа опција аутентификатора.	Не	Додатне опције за аутентификатор кроз које се може захтевати да аутентификатор захтева интеракцију корисника (унос ПИН кода, скенирање отиска прста) или се може нагласити аутнетификатору да сачува информације о кључу.
challenge	Низ бајтова		Порука генерисана од стране сервера која се шаље аутентификатору на потписивање.
pinAuth	Низ бајтова	Не	
pinProtocol	Неозначени цео број	Не	

Табела 12 - Параметри методе navigator.credentials.get

Параметар *options* је речник или мапа које може садржати два кључа

- *up (user presence)* – када је вредност овог кључа постављена на *true* значи да се аутентификатору јавља да се захтева пристанак корисника зарад завршетка операције
- *uv (user verification)* – када је вредност овог поља постављена на *true* аутентификатору се јавља да се захтева интеракција корисника зарад завршетка операције. Примери интеракције могу бити скенирање отиска или унос ПИН кода.

9.1.7 Модел AuthenticatorAttestationResponse

Назив параметра	Тип података	Обавезан	Дефиниција
clientDataJSON	Низ бајтова	Да	JSON серијализовани клијентски подаци који су прослеђени аутентификатору приликом позива метода <i>create</i> или <i>get</i> .
attestationObject	Низ бајтова	Да	Овај елемент садржи атестациони објекат који је криптографски заштићен. У оквиру њега се налазе подаци аутентификатора који садрже јединствени идентификатор аутентификатора и јавни кључ. Овај објекат такође садржи и атестациону изјаву, чији садржај зависи од типа аутентификатора.

Табела 13 - Параметри модела *AuthenticatorAttestationResponse*

9.1.8 Модел AuthenticatorAssertionResponse

Назив параметра	Тип података	Обавезан	Дефиниција
clientDataJSON	Низ бајтова	Да	Садржај овог поља је идентичан као и садржај истог поља у моделу <i>AuthenticatorAttestationResponse</i> .
authenticatorData	Низ бајтова	Да	Ово поље садржи податке о аутентификатору који су послати од стране самог аутентификатора.
userHandle	Низ бајтова	Не	Ово поље садржи идентификатор

			корисника генерисан од стране аутентификатора при генерисању креденцијала.
--	--	--	--

Табела 14 - Параметри модела *AuthenticatorAssertionResponse*

9.2 Примери кода

```
// ask browser for credentials (browser will ask connected authenticators)
let credential;
try {
  credential = await navigator.credentials.get({ publicKey: makeAssertionOptions })
} catch (err) {
  handleUnsuccessfulLogin(err.message ? err.message : err);
  return;
}
```

Пример 4 - пример позива методе navigator.credentials.get

```

let newCredential;
try {
  newCredential = navigator.credentials.create({
    publicKey: makeCredentialOptions
  });
}
catch (err) {
  handleUnsuccessfulRegister(err.message ? err.message : err);
  return;
}

```

Пример 5 - пример позива методе navigator.credentials.create

```

var payload = document.getElementById("payload").value;
var formData = new FormData();
formData.append("payload", payload);
let res;
let jsonRes;
try {
  res = await fetch("https://localhost:5001/signature", {
    method: 'POST',
    body: formData
  }).then(response => response.json())
  .then(jsondata => {
    console.log(jsondata);
    jsonRes = jsondata;
  });
}
catch (err) {
  console.log(err);
  smartCardAuthorizationFailed();
  return;
}

```

Пример 6 - пример позива методе POST /signature