Применлив модел за електронско гласање кај клиентски уреди со ограничени карактеристики

м-р Панче Рибарски

Факултет за информатички науки и компјутерско инженерство

Универзитет „Св. Кирил и Методиј“ Скопје

2017

Ментор:	Вонр. проф. д-р Љупчо Антовски Факултет за информатички науки и компјутерско инженерство, УКИМ, Скопје, Македонија
Членови на комисијата:	Проф. д-р Марјан Гушев, претседател Факултет за информатички науки и компјутерско инженерство, УКИМ, Скопје, Македонија
	Вонр. проф. д-р Љупчо Антовски, ментор Факултет за информатички науки и компјутерско инженерство, УКИМ, Скопје, Македонија
	Вонр. проф. д-р Весна Димитрова, член Факултет за информатички науки и компјутерско инженерство, УКИМ, Скопје, Македонија
	Вонр. проф. д-р Гоце Арменски, член Факултет за информатички науки и компјутерско инженерство, УКИМ, Скопје, Македонија
	Вонр. проф. д-р Александра Милева, надворешен член Факултет за информатика, Универзитет „Гоце Делчев“, Штип, Македонија
Датум на одбрана:	15.05.2017

Благодарност

Изработката на овој докторски труд беше интересно патување, со големи предизвици од професионален, но и од личен карактер. Му должам голема благодарност на мојот ментор Љупчо Антовски со кој заедно ја продолживме работата во полето на електронското гласање започната со магистерската теза. Голем придонес во градењето на оваа теза дадоа професорите Весна Димитрова, Гоце Арменски, Марјан Гушев и Александра Милева. Разговорите со нив на некоја од темите во докторскиот труд ми помогнаа при решавањето на многуте проблеми. Патот на научното усовршување го одевме заедно со помладите колеги и долгогодишни пријатели од ФИНКИ. Со Христина Михајлоска, Катарина Тројачанец и Магдалена Костоска заеднички работевме, се мотивиравме и си дававме поттик за истрајност, за целото време на изработка на тезата, што посебно ми значи.

Неизмерна благодарност им должам на моите родители Ѓорѓи и Стојка Рибарски, за сите одрекувања и жртви, неуморна поддршка и помош во секој момент од животот. Стојанче, Весна и малата Андреа, беа убаво друштво во долгите денови исполнети со работа. Најголема поддршка ми беше сопругата и колешка Билјана Тојтовска, од која добив професионална помош за математичките основи на докторатот. Таа беше упорен сопатник на ова патување, потпора во сите тешки моменти кои не беа ретки во минатите 5 години. Без нејзиниот поттик, докторскиот труд ќе беше недостижна цел во дадениот временски рок.

Благодарност на Фондацијата проф. д-р Ванчо Чифлиганец за финансиската поддршка и на сите кои придонесоа кон остварување на еден голем сон. Секако, патувањето не застанува тука, продолжуваме понатаму со следни предизвици.

Апстракт

Оваа докторска дисертација ја истражува можноста за електронското гласање кај уреди со ограничени карактеристики. Како пример за такви уреди се избрани мобилните телефони со Android оперативниот систем. Поради комплексната природа на електронското гласање и карактеристиките кои сакаме да ги исполни тој систем, дополнително се истражуваат концептите за делење на тајна и слепи потписи. Користењето на уреди со ограничени карактеристики значи прилагодување на процесорските, мемориските и комуникациските потреби на алгоритмите во системот. Поради тоа, се истражуваат елиптичните криви и криптографијата базирана на идентитети. Како резултат на истражувањето за аритметички операции над елиптични криви направивме софтверска библиотека која вклучува и методи кои се извршуваат во константно време како механизам против временски напади. При истражувањето на криптографијата базирана на идентитети наидовме на проблем со центарот за креирање на приватни клучеви. Поради тоа изработивме протокол за дистрибуирано креирање на приватниот клуч на центарот и дистрибуирано креирање на клучеви на сите останати учесници во системот. Предложивме модел за електронско гласање на уреди со ограничени карактеристики со користење на слепи потписи. Моделот користи четири ролји: DKG играч, потпишувач, гласачка кутија и бројач. Предложениот модел го имплементиравме на Android оперативниот систем при што потребното време за давање на глас е 4 секунди, а потребното време за пребројување на 100 гласа е околу 3 секунди. Со ова се покажува практичноста и употребливоста на системот на хардвер кој веќе го имаме постојано со нас.

Содржина

Глава 1

Преглед

1.1 Мотивација

Гласањето претставува начин на искажување на мнозинското мислење. Гласањето како демократски процес се користи во различни изведби - од десетина гласачи во ad-hoc сценарио до милиони гласачи во изборни процеси на ниво на држава. Со додавањето на можноста за анонимност при гласањето се наметнува потребата од високо ниво на внимателност при одржувањето на процесот. Нивото на внимателност треба да влее доверба кон процесот, односно доверба кон системот кој го одржува процесот. Во случај на електронско гласање внесуваме многу параметри на несигурност кон системот. Во случај на сложени гласачки процеси потребни се опрема и оператори за оптимална функционалност на системот. Довербата кон овој систем се сведува на чесноста на имплементаторите и на операторите - чесност која не секогаш може да се гарантира во гласачки процеси од голема вредност. Токму тоа е мотивацијата на овој докторски труд - градење на систем за електронско гласање кој значително ќе го намали влијанието на поединечни оператори на системот. Втората работа која ја земаме во предвид е цената на поседување на системот (анг. TCO - total cost of ownership). Сигурен систем за електронско гласање бара инвестирање во софтвер за системот и во хардвер на кој би се извршувало гласањето. Затоа, акцентот на овој докторски труд е свртен кон уреди со ограничени карактеристики - мобилни телефони и таблети. Покажуваме дека е можно да се имплементира сигурен систем за електронско гласање за мали гласачки процеси со секојдневен ефтин хардвер кој веќе го

поседуваме.

1.2 Основа

Основа на истражувачката работа во овој докторат од математиката и криптографијата е литературата објавена во последните 50 години, како и истражувањата од електронско гласање од последните 20 години. Во литературата се наоѓаат модели на системи за електронско гласање кои се веќе потврдени и кои се користат во индустријата за реални гласачки процеси.

Сплотувајќи модерна криптографија со парови над елиптични криви (научна област која активно се истражува во последните 15 години) и концепти за делење на тајна (непрекинато истражување во последните 30 години) добиваме начин на имплементирање на систем за електронско гласање кој е брз, лесен за користење и сигурен и доверлив за гласачите.

Како основа дополнително се користи истражувањето на магистерската теза [94] во која се покриени системите за масовно електронско гласање базирани на хомоморфни енкрипции и слепи потписи.

За крајната имплементација се користи Android оперативниот систем поради високите 87.6% учество на пазарот на мобилни платформи (според ИДЦ [64]) во вториот квартал во 2016та година). Можноста за наоѓање на ефтин хардвер во вид на мобилни телефони и таблет уреди со Android оперативниот систем е погодна за намалување на цената на поседување на системот.

1.3 Тема која се истражува

Електронското гласање е поле кое е хетерогено според областите кои ги вклучува во истражувањето. Темите кои се истражуваат во овој труд се базираат на елиптични криви, брзо пресметување на парови над елиптични криви, делење на тајна без чесен делител, криптографија со идентитети, слепи потписи и сигурни системи за електронско гласање.

Истражувањата врзани со елиптични криви и парови над елиптични криви вклучуваат креирање на софтверска библиотека за брзи пресметки на операции врз елиптични криви и парови над елиптични криви. Оваа библиотека ги користи најновите методи за забрзување на аритметичките пресметки.

Слика 1.1: Илустрација на пазарот на мобилни платформи (според ИДЦ [64]).

Како пример за ефикасноста претставува имплементацијата на BLS потписи кои се извршуваат за трипати побрзо време од останатите библиотеки (види подглава 3.4.1).

Истражувањето за делење на тајна без чесен делител и криптографија базирана на идентитети претставува начин да се добие PKI (инфраструктура за приватни клучеви, анг. Private Key Infrastructure) систем без централна точка на слабост. PKI системот користи криптографија базирана на идентитети, што во основа ја елиминира потребата од преземање на јавни клучеви (и сертификати) од PKI системот. Ова го искористивме за намалување на комплицираниот протокол при делење клучеви во системот за електронско гласање и за земање на моќта за генерирање на клучеви од едно лице и префрлување на одговорноста на креирање на клучеви на група од учесници во протоколот.

Во истражувањето на слепи потписи во криптографија базирана на идентитети анализиравме протоколи за слепи потписи и нивната потреба од пресметки и количина на податоци кои се пренесуваат при извршувањето на про-

токолот. По донесувањето на протоколите за слепи потписи "на заеднички именител"многу го олеснува бирањето на слеп потпис за специфично сценарио. Тоа понатаму го искористивме за бирање на соодветен протокол за слепи потписи во системот за електронско гласање.

Крајната тема која се истражува е систем за електронско гласање на уреди со ограничени карактеристики - мобилни уреди и таблети. Истражуваме имплементација на систем за електронско гласање без ниту една клучна точка на мобилни уреди со Android оперативниот систем. Преку користење на криптографија базирана на идентитети со дистрибуирано генерирање на клуч и слепи потписи, покажуваме реален систем за сигурно електронско гласање со уреди со ограничени карактеристики.

1.4 Цели на истражувањето и хипотези

Главна цел на овој труд ќе биде истражувањето на мобилните уреди како можна платформа за електронско гласање и дефинирање на соодветен модел за мобилно електронско гласање. Знаејќи ги слабостите на мобилните платформи соодветно можеме да избереме алгоритам кој ќе биде добро поддржан и кој ќе ги покрие сите сигурносни аспекти кои ги бара електронското гласање. Поради тоа истражувањето е во насока на нови криптографски решенија користејќи елиптични криви кои се полесни за користење земајќи го во предвид ограниченият хардвер на мобилните уреди.

Повеќето гласачки процеси се анонимни – во нив не треба да постои врска помеѓу дадениот глас и гласачот. Затоа е потребен дополнителен напор за воведување анонимност на целосната комуникација помеѓу мобилниот уред и сервисот за електронско гласање. За ова постојат криптографски начини кои овозможуваат добивање анонимен канал за пренос на податоци. Од друга страна, негативна работа кај овие криптографски протоколи е нивната комплексност и побарувањето за голем број на податоци кои можеби мобилната мрежа нема да може да ги издржи. Цел на овој труд е истражување на методи за добивање на анонимни канали преку мобилна мрежа.

Според погоре кажаното, ги дефинираме следните главни цели на истражувањето:

- истражување и оптимизирање на постојните модели за електронско гла-

сање за да бидат прифатливи во мобилна околина и дефинирање на соодветен модел,

- имплементација на моделот за електронско гласање кој ќе биде соодветен да работи на мобилни уреди со ограничени карактеристики,
- имплементирање на соодветни напредни алгоритми за криптографија кои ќе овозможат прифатливо ниво на перформанси кај мобилните клиентски уреди за електронско гласање,
- дефинирање на методи и постапки за имплементирање на мобилно електронско гласање.

Врз основа на поставените цели на истражување се истражуваат следните хипотезите во овој докторски труд:

- Дали е возможно да се направи софтверска библиотека која е лесна за користење при имплементирање на криптографски протоколи со користење на елиптични криви?
- Дали може да се имплементира PKI за користење во криптографија базирана на идентитети со дистрибуирано креирање на клучеви на уреди со ограничени карактеристики?
- Дали може лесно да се споредат постоечките протоколи за слепи потписи во криптографија со идентитети?
- Дали може да се имплементира систем за електронско гласање на уреди со ограничени карактеристики?

1.5 Општ придонес

Во овој труд ги идентификуваме следните придонеси:

- имплементирање на библиотека за брзи пресметки над елиптични криви и парови над елиптични криви,
- имплементирање на дистрибуирано креирање на клучеви во криптографија со идентитети,

- анализа и споредба на слепи потписи во криптографија со идентитети: потребни пресметки и количина на податоци кои се потребни за извршување на протоколот,
- имплементирање на систем за електронско гласање на уреди со ограничени карактеристики - мобилни уреди и таблети со Android оперативниот систем.

1.6 Методологија

Според темата на истражување на овој докторат (подглава 1.3) соодветно дефинираме методологија на истражување на секоја целина која продуцира резултати.

Истражувањето започнува со креирање на библиотека за брзи пресметки на аритметички операции врз елиптични криви и врз основните конечни полиња на тие криви. Имплементација ги следи eBACS [16] принципите на API изгледот. За демонстрирање на леснотијата на користење и ефикасноста на аритметичките операции го имплементираме BLS протоколот и го споредуваме со други имплементации.

Понатаму, во полето на ID-базирана криптографија работиме на Android имплементација на PKI систем без централна точка на моќ. Со Android имплементацијата се демонстрира успешно користење на ID-базирана криптографија и PKI систем без сертификати.

За потребите на електронското гласање, кое претставува главна цел на истражувањата, ги истражуваме ID-базираните слепи потписи. Во ова истражување се концентрираме на собирање и споредба на алгоритмите за слепи потписи во литературата. Сведувајќи ги на иста номенклатура потребите на секој од алгоритмите даваме лесен начин на бирање на протокол за слепи потписи според карактеристичните сценарија каде би се користеле овие потписи.

Методологијата за главниот дел од ова истражување претставува изработка на протокол за електронско гласање кое користи ID-базирана криптографија. Понатаму, овој протокол се имплементира на Android мобилната платформа како демонстрација дека уреди со ограничени карактеристики може да се користат во процесите за електронско гласање.

1.7 Поделба на докторскиот труд

Овој труд е поделен на девет целини. Првата глава ја објаснува мотивацијата, проблемите, истражувањата и методологијата на работа. Оваа глава дава вовед во проблематиката која се истражува во докторскиот труд и ги наведува хипотезите кои се покриваат во истражувањето.

Втората глава дава дефиниции и математички основи на материјалот кој се користи во овој труд. Овие општи математички дефиниции се користат во следните глави и врз нив се гради теоријата на овој докторски труд. Овде е даден вовед во асиметрична криптографија и во криптографија со елиптични криви. Исто така, овде е изнесена теоријата позади процесот за споделување на тајна и шемите за споделување на тајна кои се користат во пракса. Се презентира концептот и најчесто користените алгоритми за споделување од Shamir [109], верифицираното споделување од Feldman [41] и споделувањето без чесен делител од Pedersen [91]. Придонесот на оваа глава е давањето вовед во споделување на тајна и прикажување на теоријата и комплексноста на процесот преку трите најкористени методи за споделување на тајна.

Третата глава ја објаснува Панда која претставува софтверска библиотека за имплементирање на криптографски протоколи над елиптични криви и парови над елиптични криви. Во оваа глава е дадена теоријата врз која се основа Панда, најновите методи кои се користат за брзи пресметки и имплементацијата во Assembler и програмскиот јазик C. Придонесот на Панда претставува C библиотека која може да ја користи секој за пресметки со елементи од елиптични криви и брзи пресметки на парови над елиптични криви. Сите пресметки се извршуваат во константно време при што имплементираните алгоритми стануваат отпорни на временски напади (анг. timing attacks).

Во петтата глава ја објаснуваме криптографијата базирана на идентитети, проблемите со генерирање на клучеви и имплементираниот систем за PKI со дистрибуирано креирање на клучеви. Во оваа глава е направена споредба меѓу криптографијата базирана на идентитети со криптографијата базирана на сертификати. Прикажаниот проблем со посредништво на клуч е решен преку имплементирање на дистрибуирано генерирање на клуч за практична имплементација на PKI без сертификати. Нашиот придонес во оваа глава

претставува практичен систем имплементиран во Android за дистрибуирано генерирање на клуч во било кој протокол.

Шестата глава дава вовед за слепи потписи и нивно користење во криптографија базирана на идентитети. Во оваа глава даваме споредбата на најдобрите седум протоколи за слепи потписи. Придонесот во оваа глава претставуваат табелите со споредба на број на математички операции и потребен пропусен опсег за извршување на секој од овие протоколи. На овој начин може лесно да се избере протокол знаејќи ја околината во која ќе се извршува протоколот.

Седмата глава ги користи сите истражувани теми во овој труд за да имплементира сигурен систем за електронско гласање на мобилни уреди. На почеток се дава опис на поимот електронско гласање, видовите на електронско гласање кои се појавуваат во реалниот свет и карактеристиките на имплементирани системи за електронско гласање. Придонесот на оваа глава ја покрива темата на докторскиот труд. Предложениот систем за електронско гласање е имплементиран на Android мобилната платформа. Со ова се покажува практичната примена на елиптични криви и криптографијата базирана на идентитети во еден комплексен систем каков што е системот за електронско гласање.

Во осмата глава се сумирани резултати од истражувањата покриени во овој докторски труд. Оваа глава користи методологија за мерење на резултати од предложениот и имплементираниот систем за електронско гласање.

Деветтата глава ја заклучува целата работа од докторскиот труд. Давајќи преглед на завршеното истражување заклучуваме кој е придонесот од покриените теми и имплементираните системи. Наведуваме отворени прашања и идна работа која произлегува од истражуваната тема, а која не можеше да се вклучи во докторскиот труд.

Глава 2

Основи

Во оваа глава се зададени основните математички и криптографски терминологии кои се користат во овој докторски труд. Ги дефинираме потребните алгоебарски структури на кои е изградена криптографската теорија, според материјалот изложен во [68] и [120]. Воведуваме и основни криптографски поими потребни за понатамошните излагања, за што се повикуваме на [118], [32], [88] и [112]. Во литературата се сретнува различна нотација за терминологијата која ќе ја користиме овде. Во оваа глава ќе ја поставиме нотацијата која ќе се користи во остатокот од овој докторски труд. Исто така, овде е презентирана основната теорија за делење на тајна со чесен делител и без чесен делител.

2.1 Алгебарска основа

Дефиниција 2.1. Група $\langle G, \circ \rangle$ претставува конечно или бесконечно множество заедно со бинарна операција $\circ$ дефинирана на G, која ги задоволува следните својства:

- (Затвореност) Доколку x и y се два елемента од G, тогаш и $x \circ y$ исто така припаѓа во G, $\forall x, y \in G, x \circ y \in G$,
- (Асоцијативност) За сите x, y, z од G важи $(x \circ y) \circ z = x \circ (y \circ z)$,
- (Неутрален елемент) Постои елемент e во G т.ш. за секој x од G важи $e \circ x = x \circ e = x$. Елементот e се нарекува неутрален елемент за операцијата $\circ$,

- (Инверзен елемент) За секој x во G постои инверзен елемент означен со x^{-1} т.ш. $x \circ x^{-1} = x^{-1} \circ x = e$.

Забелешка. *Неутралните елементи во однос на адитивно и мултипликативно означени операции ќе ги означуваме со* 0 *и* 1.

Забелешка. *Групата G се нарекува Абелова (или комутативна) доколку за сите x, y во G важи $x \circ y = y \circ x$.*

Пример. *Пример за група претставува множеството $Z_m = \{0, 1, ..., m-1\}$ над кое е дефинирана операцијата собирање по модул m, $m \in \mathbb{N}$. Неутрален елемент во оваа група е елементот* 0. *Секој елемент x од оваа група има инверзен елемент $-x$ т.ш. $x + (-x) = (-x) + x = 0$.*

Дефиниција 2.2. Ред на групата G претставува кардиналноста на групата, означена со $|G|$. Групата G е конечна доколку $|G|$ е позитивен природен број, инаку е бесконечна група.

Дефиниција 2.3. Ред на елементот a од групата G е најмалиот позитивен природен број n за кој $a^n = a \times ... \times a = e$, доколку таков број постои. Доколку $a^n \neq e$ за секој позитивен природен број n, тогаш велиме дека елементот a е од бесконечен ред.

Забелешка. *За прост број p групата G е примарна во однос на p (се нарекува p-примарна или p-група) доколку редот на секој елемент е степен на бројот p.*

Забелешка. *Во конечни групи сите елементи се од конечен ред. Во бесконечни групи може да постојат елементи и со конечен и со бесконечен ред.*

Доколку на групата е дефинирана операцијата собирање, тогаш групата се нарекува адитивна група и инверзната операција е одземање. Доколку на групата е дефинирана операцијата множење, тогаш групата се нарекува мултипликативна група и инверзната операција е делење.

Дефиниција 2.4. Прстен $\langle R, +, \times \rangle$ претставува множество на кое се дефинирани две операции $+$ и $\times$ со следните својства:

- R е комутативна група во однос на операцијата $+$ со неутрален елемент 0,

- Операцијата $\times$ е асоцијативна и има неутрален елемент 1 кој е различен од 0,
- Операцијата $\times$ е дистрибутивна во однос на $+$, т.е. за секој x, y, z од R важи $x \times (y + z) = x \times y + x \times z$.

Забелешка. *Прстеноm R се нарекува комуmаmивен доколку оnерацијаmа $\times$ е комуmаmивна во однос на R.*

Пример. *Множесmвоmо од цели броеви Z заедно со оnерацииmе собирање и множење на цели броеви nреmсmавува nрсmен. Исmо mака, множесmвоmо од сиmе nолиноми над Z со оnерацииmе собирање и множење на nолиноми исmо mака nреmсmавува nрсmен.*

Дефиниција 2.5. Поле F претставува комутативен прстен во кој секој ненулти елемент е инверзибилен.

Пример. *Пример за nоле nреmсmавува множесmвоmо реални броеви R со оnерации $+$ и $\times$, во кое неуmрален елеменm во однос на оnерацијаmа собирање е 0, а неуmрален елеменm во однос на оnерацијаmа множење е 1. Секој елеменm x во R има инверзен елеменm $-x$ во однос на оnерацијаmа $+$. Секој елеменm y кој е различен од 0 има инверзен елеменm $\frac{1}{y}$ во однос на оnерацијаmа $\times$.*

Во криптографијата главно се разгледуваат полиња со конечен број на елементи. Овие полиња се нарекуваат конечни полиња или полиња на Галоа. Бројот на елементи во конечното поле е редот или кардиналност на полето.

Теорема 2.6. *Поле од ред m nосmои само доколку m е од облик $m = p^n$, каде $n \in \mathbb{N}$ и p е nросm број. Во овој случај p се нарекува каракmерисmика на конечноmо nоле.*

Забелешка. *Полињаmа во кои $n = 1$ се нарекувааm nолиња од nросm ред, односно nросmи nолиња. Двеmе оnерации во овие nолиња се собирање и множење nо модул p.*

Теорема 2.7. *Нека p е nросm број. $Z_p = 0, 1, 2, ..., p - 1$ со оnерации $+$ и $\times modulo p$ е nоле кое се означува со $GF(o)$. Се нарекува nросmо nоле, односно nоле на Галоа со nросm број на елеменmи. Сиmе ненулmи елеменmи на $GF(p)$*

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

(a)

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

(b)

Слика 2.1: Илустрација на собирање и множење во $GF(3)$

имаат инверзен елемент и целата аритметика во $GF(p)$ се прави по модул p.

Забелешка. *Прстенот Z_p кој го дефинира $GF(p)$ исто така претставува конечно поле.*

Пример. *Нека за пример земеме конечно поле $GF(3) = \{0, 1, 2\}$. Алгебарските операции собирање и множење над ова поле се дадени во сликата 2.1.*

Забелешка. *Најмалото конечно поле претставува $GF(2)$.*

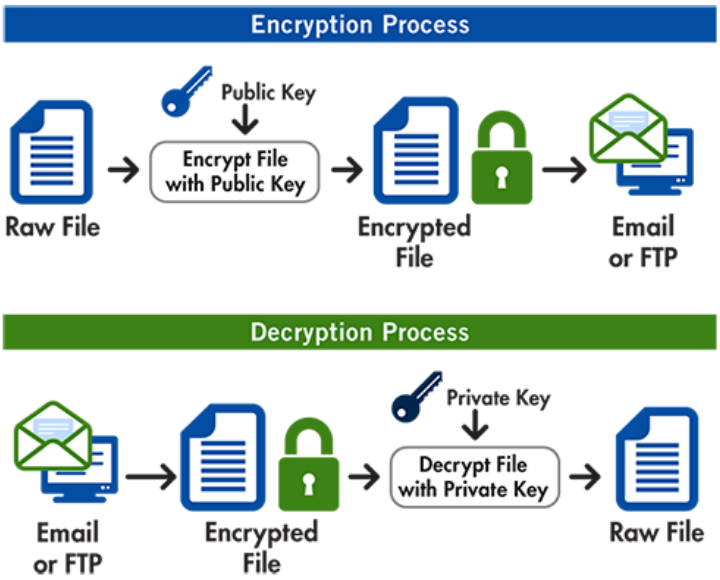
Дефиниција 2.8. Нека $\langle P, +, \circ \rangle$ е поле и нека $S \subseteq P$. $\langle S, +, \circ \rangle$ е потполе на P доколку $0, 1 \in S$ е затворено во однос на $+, \circ$ и барањето инверзен елемент. Во овој случај P се нарекува проширување на полето S и се означува со P/S.

Пример. *Полето на комплексни броеви C со операциите $+$ и $\times$ претставува проширување на полето на реални броеви R, а полето на реални броеви $\langle R, +, \times \rangle$ претставува проширување на полето на рационални броеви $\langle Q, +, \times \rangle$.*

Пример. *Доколку p е прост број и $n \in \mathbb{N}$, тогаш конечното поле $GF(p^n)$ има p^n елементи. Ова поле претставува проширување на полето $GF(p)$ кое има p елементи.*

2.2 Асиметрична криптографија

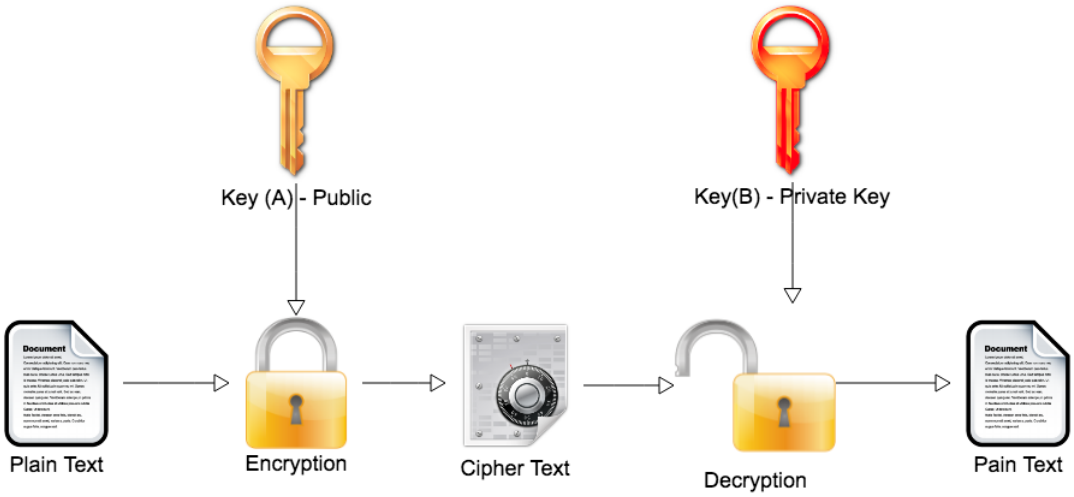
Асиметричната криптографија (исто така позната како криптографија со јавен клуч) претставува релативно нов тип на криптографија кој масовно се употребува за различни криптографски потреби. За разлика од симетричната криптографија, каде се употребува еден ист клуч за криптирање и декриптирање, кај асиметричната криптографија постојат пар клучеви за



Слика 2.2: Илустрација на криптирање и декриптирање со асиметрична криптографија.

оваа цел (види слика 2.2). Едниот клуч, наречен јавен клуч, се користи за криптирање, додека другиот клуч, наречен приватен клуч, се користи за декриптирање. Според именувањето веднаш може да се заклучи дека јавниот клуч е познат и се објавува јавно, а приватниот клуч се чува тајно кај сопственикот и не смее никој да го дознае.

Асиметричната криптографија го надминува големиот проблем со размена на клучеви кој го има симетричната криптографија. Во асиметричната криптографија секој играч го генерира парот јавен/приватен клуч. Јавниот клуч го дава на заинтересираните играчи, а приватниот го чува тајно. Асиметричната криптографија е прикажана визуелно на сликата 2.3. Ова значи дека се скока чекорот на размена на приватни клучеви кој е вклучен во симетричната криптографија. Сепак, слабата точка на асиметричната криптографија е брзината. Симетричната криптографија е илјадници пати побрза од асиметричната. Затоа во реалните системи се користат и двата типа - асиметричниот се користи за договарање и размена на симетричен клуч, а потоа комуникацијата продолжува преку брза симетрична криптографија. Пример за ова е SSL/TLS протоколот [42] кој се користи масовно во веб прелистувачите и веб серверите.



Слика 2.3: Илустрација на асиметрична криптографија со јавен и приватен клуч.

Првото откритие на асиметричен тип на криптографија е во 1970 година од James H. Ellis во Владиниот центар за комуникации на Велика Британија (анг. Government Communication Headquarters - GCHQ), кое немало практичен начин на имплементација. Во 1973 година Clifford Cocks, исто така во GCHQ, имплементира асиметричен тип на криптографија кој денес е познат како RSA [113]. Во 1974 година, Malcolm J. Williamson го истражувал и го направил тоа што денес е познато како Diffie-Hellman протокол за размена на клучеви [121]. Овие истражувања биле класифицирани како тајна бидејќи биле дел од воени истражувања во криптографијата. Иако биле извонредни откритија во областа на криптографијата, не се искористиле во времето на откривање поради малата процесирачка моќ на тогашните компјутери. Работата на овие научници останале тајна се до 1997 година кога се декласифицирани од GCHQ.

Прв асиметричен систем предложен во јавноста е од Whitfield Diffie и Martin Hellman кои го пронашле познатиот Diffie-Hellman протокол за размена на клучеви [37] во 1976 година, независно од истражувањата на Williamson vo 1974 година. Овој протокол како математички проблем го користи степенувањето во конечни полиња. Во 1977 година е измислен познатиот RSA асиметричен криптосистем од Ron Rivest, Adi Shamir и Leonard Adleman [98]. Овој алгоритам како математички проблем го користи факторирањето на го-

леми броеви.

Графички прикажано, асиметричната криптографија се користи како на сликата 2.3. Некои од местата каде се користи асиметричната криптографија се:

- се користи за енкрипција без потреба од разменување на сензитивни и тајни клучеви,
- може да се користи асиметрична енкрипција за размена на таен симетричен клуч или за договарање на заеднички таен клуч,
- дигиталните потписи се користат за интегритет, автентикација и неодрекливост.

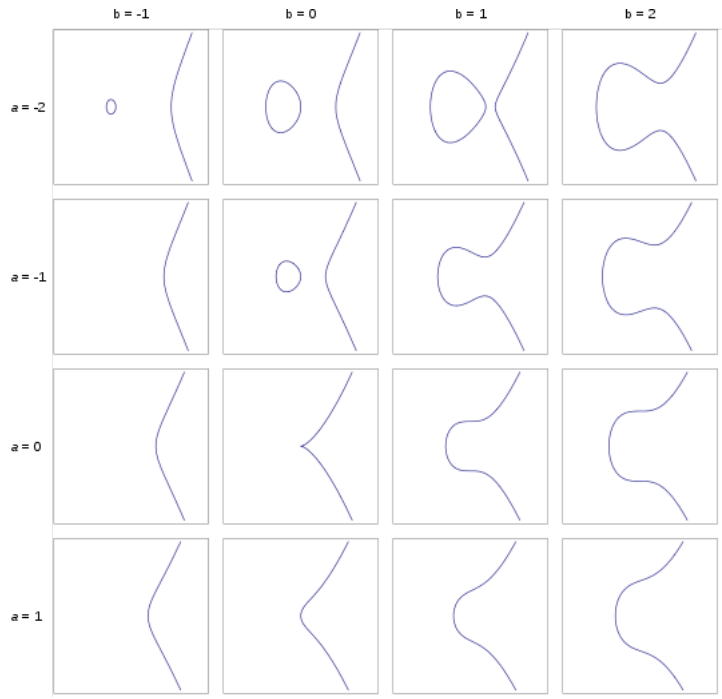
2.3 Криптографија со елиптични криви

Интересот за елиптичните криви датира уште од пред нашата ера, при што резултатите може да се сретнат во познатата Arithmetica од Диофант. И денес тие се активно поле на истражување во математиката. Во поновите времиња елиптичните криви се истражуваат од осумдесетите години на минатиот век. На пример, важен резултат во теорија на броеви е доказот на последната теорема на Ферма во кој се искористени елиптичните криви. Во криптографијата се користат за факторизација на броеви и во асиметрични криптографски системи базирани на елиптични криви над конечни полиња [88, Глава 9].

Елиптичните криви се алгебарски криви зададени со равенка од облик $y^2 = x^3 + Ax + B$. Тие претставуваат кубични криви изоморфни на торус [118, Глава 2.1]. Кога зборуваме за крива секогаш мислиме на парот (x, y) кој е решение на равенката со која е претставена кривата. Вредностите кои може да ги добијат координатите на кривата може да дојдат од различни полиња, меѓутоа за криптографски потреби најинтересни се конечните полиња.

Дефиниција 2.9. Елиптична крива E над полето Z_p, $p > 3$, е множеството од сите парови (x, y) од Z_p кои ја задоволуваат следната равенка:

$$y^2 = x^3 + Ax + B \ mod \ p \tag{2.1}$$



Слика 2.4: Илустрација на визуелизирање на елиптична крива над полето на реални броеви.

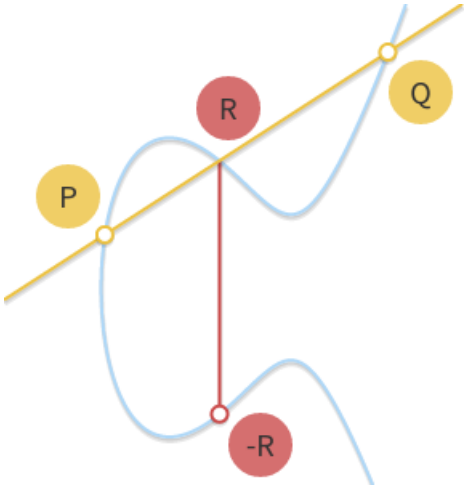
Коефициентите a и b исто така припаѓаат на Z_p со условот дискриминантата на равенката 2.1 да е различна од нула:

$$4a^3 + 27b^2 \neq 0 \ mod \ p \tag{2.2}$$

Во множеството на парови (x, y) се додава и имагинарна точка, наречена точка на бесконечност $\mathcal{O}$.

Доколку ја претставиме графички оваа равенка над некое конечно поле нема да добиеме слика на крива како што би очекувале. Но, доколку претставувањето е над полето на реални броеви добиваме интересни визуелизации на кривата (види слика 2.4).

Од важност е да дефинираме операција на елиптични криви над Z_p во однос на која таа ќе биде група. Таа операција треба да е затворена над сите елементи (x, y) кои ја задоволуваат равенката на елиптична крива. За таа цел дефинираме операција "собирање"на произволни точки $P = (x_1, y_1)$ и $Q = (x_2, y_2)$ од кривата чиј збир е $P + Q = R$, каде R е повторно точка на кривата.



Слика 2.5: Илустрација на операцијата собирање на елиптична крива преку полето на реални броеви.

Дефиниција 2.10. Нека E е елиптична крива дефинирана од равенката $y^2 = x^3 + Ax + B$. Нека $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ се точки на кривата E и притоа $P_1, P_2 \neq \mathcal{O}$. Дефинираме $P_1 + P_2 = P_3 = (x_3, y_3)$ според следното:

1. Доколку $x_1 \neq x_2$ тогаш $x_3 = m^2 - x_1 - x_2$, $y_3 = m(x_1 - x_3) - y_1$, каде $m = \frac{y_2 - y_1}{x_2 - x_1}$,
2. Доколку $x_1 = x_2$ но $y_1 \neq y_2$, тогаш $P_1 + P_2 = \mathcal{O}$,
3. Доколку $P_1 = P_2$ и $y_1 \neq 0$ тогаш $x_3 = m^2 - 2x_1$, $y_3 = m(x_1 - x_3) - y_1$, каде $m = \frac{3x_1^2 + A}{2y_1}$,
4. Доколку $P_1 = P_2$ и $y_1 = 0$ тогаш $P_1 + P_2 = \mathcal{O}$.

Уште повеќе, дефинираме $P + \mathcal{O} = P$ за сите точки $P \in E$.

Забелешка. *Елиптична крива E е затворена над дефинираната операција собирање на точки. Се докажува дека точките на елиптичната крива формираат адитивна абелова група заедно со единечната точка $\mathcal{O}$.*

Забелешка. *Геометриска илустрација на операцијата "собирање"на две различни точки на елиптична крива над полето на реални броеви може да се види на сликата 2.5.*

Скаларно множење nP, $n \in \mathbb{N}$ претставува собирање на точката P сама со себе точно n пати. По вообичаениот пристап, ова претставива n пати извршување на операцијата собирање. Но, од практична гледна точка оваа операција има $\mathcal{O}(2^k)$ сложеност, каде k е бројот на битови во записот на n. Затоа ќе користиме побрзи алгоритми за скаларното множење.

Дефиниција 2.11. "Дуплирај и додади"алгоритмот претставува ефикасен алгоритам за скаларно множење nP на точката P од елиптичната крива и природниот број n. Алгоритмот оди по битовата репрезентација на n од десно кон лево. Доколку наиде на 1, го собира досегашниот резултат со привремена променлива. Доколку наиде на 0, ја дуплира привремената променлива.

Забелешка. *Овој алгоритам има сложеност $\mathcal{O}(k)$ во однос на бројот на битови k на природниот број n.*

Во оваа докторска десертација ќе се користат многу поими и резултати кои се стандардни во теоријата на елиптични криви. Но, детално воведување на овие поими излегува од доменот на темата на овој докторат. Такви се на пример, точка во бесконечност (дефинирана преку проективни простори), Miller-лупата, билиниеарни функции, типови на парови, степен на вклучување, афини и проективни координати и координати на Јакоби како нивна модификација потребна за забрзување на некои процедури.

Елиптичните криви се користат во библиотеката Панда објаснета во Глава 3.1. Таму се имплементирани математичките операции над елиптичните криви кои ги воведовме во оваа глава. Уште повеќе, Панда вклучува имплементација на математичките операции на основните групи над кои се дефинираат елиптичните криви. Затоа Панда претставува комплетно решение при имплементирање на протоколи со елиптични криви.

2.4 Делење на тајна

2.4.1 Концепт на делење на тајна

Концептот на делење на тајна (анг. Secret Sharing) подразбира поделба на некоја тајна информација на повеќе делови кои се доделуваат на учесници во

протоколот. На тој начин секој учесник би имал свој дел од тајната информација без да има било какви сознанија за целосната информација. Кога ќе се соберат доволен број на учесници со делови од тајната, тогаш тајната информација може да се реконструира. Во спротивност тоа не може да се направи. Доволниот број на учесници кои треба да се соберат за реконструкција на тајната се нарекува прагова вредност (анг. threshold value).

Можните улоги во протокол на делење на тајна се делител и играчи. Делителот ја има тајната информација, ја дели на делови и ги дистрибуира тие делови на играчите. При делењето на информацијата делителот решава која е праговата вредност, односно колку играчи е потребно да се соберат за да се реконструира тајната информација. Бидејќи делителот на почетокот ја има тајната, се смета дека тој е доверлив и дека нема да го прекрши протоколот. Дополнително, се смета дека после делењето делителот ја брише тајната информација. Кога ќе се соберат број на играчи еднаков или поголем од праговата вредност, тогаш нивните делови може да ја реконструираат тајната информација.

Дефиниција 2.12. Шема на делење на тајна (t, n) се нарекува делењето на тајна S на делови $S_1, S_2, ..., S_n$ на следниот начин:

- било кои t или повеќе собрани делови може го реконструираат S при што $t \leq n$;
- било кои $t-1$ или помалку собрани делови не даваат никаква информација за S.

Шемата се состои од две фази:

- Фаза на делење: делителот ја дели тајната S и дава дел S_i на играчот P_i;
- Фаза на реконструкција: сите играчи ги даваат доделените тајни делови S_i и се реконструира тајната S.

Оваа шема се нарекува шема на делење со прагова вредност t.

Концептот на делење на тајна е дефиниран во 1979 година од Adi Shamir [109] и George Blakley [19] независно еден од друг. Овој концепт се користи во сценарија кога имаме сензитивни податоци (тајна информација) кои

не треба да бидат "видливи"цело време. Пример за тоа се криптографски клучеви, тајни документи, ПИН броеви и други слични типови на податоци. Преку шемата на делење на тајна (види Дефиниција 2.12) може да се подели информацијата на повеќе учесници. На овој начин ниту еден учесник во оваа шема нема пристап до поделената информација. Само доколку се соберат доволен број на учесници (прагова вредност) може да се реконструира поделената информација.

2.4.2 Шема за делење на тајна на Shamir

Шемата за делење на тајна од Adi Shamir [109] е начин за делење на тајна со прагова вредност t. Шемата се темели на идејата дека точно две различни точки се доволни за дефинирање на права, точно три различни точки се доволни за дефинирање на парабола, и така натаму, што доведува до фактот дека точно t различни точки се доволни за дефинирање на полином од степен $t-1$. Со ова знаење може да се дефинира шемата за делење на тајна од Adi Shamir.

Нека n претставува бројот на учесници во протоколот за делење на тајна. Нека s претставува тајната која треба да се подели на играчите во протоколот. Нека t претставува бројот на играчи кои се доволни за реконструкција на тајната s. Вака дефинирата шема ја нарекуваме (t, n) шема на делење. Делителот на тајната го дефинира полиномот $f(x)$ на следниот начин:

$$f(x) = s + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_{t-1} x^{t-1} \tag{2.3}$$

Коефициентите $a_1, a_2, a_3, ..., a_{t-1}$ се случајно избрани позитивни броеви, додека нултиот коефициент s претставува тајната која треба да се дели. Извршителот на протоколот за делење на тајна пресметува делови од тајната за секој играч P_i во протоколот на следниот начин:

$$(i, f(i)), i = 1, ..., n \tag{2.4}$$

Извршителот на протоколот го дава парот (2.4) од индекс и евалуација на

полиномот за тој индекс на соодветниот играч P_i. Преносот на овие податоци треба да биде сигурен, односно никој не смее да ги дознае податоците на i-тиот играч освен P_i.

Според фактот дека точно t различни точки се доволни за дефинирање на полином од степен $t-1$ заклучуваме дека се доволни t точки (парови дадени на играчите P_i) за реконструкција на целиот полином $f(x)$. Реконструкцијата може да се изврши со процес познат под името интерполација. Интерполација претставува процес на конструкција на математичка функција која најдобро одговара на дадени точки. Бидејќи точките во нашиот протокол беа изведени преку евалуирање на полиномот $f(x)$, тогаш може да се користи Лагранжовата интерполација на полиноми.

Лагранжовата интерполација се дефинира над множество од точки (x_0, y_0), (x_1, y_1),...(x_k, y_k) кои се добиени од полиномот $f(x)$ од степен $k-1$ како $(i, f(i))$, $i = 1, 2, ..., k$. Интерполираниот полином над наведените точки е даден со следната линеарна комбинација:

$$L(x) = \sum_{j=0}^{t} y_j l_j(x) \tag{2.5}$$

каде $l_j(x)$ претставува Лагранжов базен полином:

$$l_j(x) = \prod_{\substack{0 \le m \le t \\ m \neq j}} \frac{x - x_m}{x_j - x_m} \quad , \; j = 0, 1, 2, ..., t \tag{2.6}$$

При решавањето на $L(x)$ всушност се добива интерполиран $f(x)$. Меѓутоа, интересот во интерполацијата треба да биде насочен само кон нултиот коефициент - тајната s. Нема потреба да се пресметуваат останатите коефициенти на $f(x)$ бидејќи тие се помошни случајно избрани броеви. Во тој случај Лагранжовиот базен полином изгледа вака:

$$l_j(x) = \prod_{\substack{0 \le m \le t \\ m \neq j}} \frac{x_m}{x_j - x_m} \quad , \; j = 0, 1, 2, ..., t \tag{2.7}$$

2.4.3 Верифицирана шема за делење на тајна

Шемата за делење на тајна овозможува делење на тајната на делови кои се даваат на играчите во протоколот. Во овој едноставен случај (како што е шемата на Shamir), играчите во протоколот веруваат во претпоставката дека доделените делови се точни, односно дека делителот го извршувал коректно протоколот. Доколку делителот е малициозен и не сака да дозволи реконструирање на тајната, тој може да направи намерни грешки во фазата на делење. Во такви случаи се користи верифицирана шема за делење на тајна (анг. verified secret sharing). Овој начин на делење на тајна прво е откриен во 1985 од Baruch Awerbuch, Silvio Micali и Shafi Goldwasser [30].

За остварување на верифицирана шема за делење на тајна е потребно проширување во фазата на делење на тајната. Во оваа фаза, играчите во протоколот може да остварат повеќе рунди на комуникација со цел да ги верифицираат применените делови од тајната. Доколку делителот е малициозен и дава неточни делови од тајната, во оваа фаза тоа може да се утврди и да се прекине протоколот. Доколку сите поделени делови од тајната ја поминуваат верификацијата, тогаш може недвосмислено да се реконструира тајната доколку се соберат доволен број на учесници кои ќе учествуваат во фазата на реконструкција.

Често користена верифицирана шема за делење на тајна е шемата на Paul Feldman [41] објаснета следното поглавје.

2.4.4 Шема за делење на тајна на Feldman

Верифицирана шема за делење на тајна на Feldman [41] претставува надградба на шемата за делење на Shamir [109]. Во шемата на Feldman е искористена шема за хомоморфна енкрипција за добивање на верифицирана шема за делење на тајна.

Исто како во шемата на Shamir, ги дефинираме n, s и t да бидат соодветно бројот на играчи во протоколот за делење на тајна, тајната која треба да се подели и праговата вредност за бројот на играчи кои треба да ја реконструираат тајната. Дополнително, делителот избира генератор g од групата во која работи G. Потоа, делителот на тајната го дефинира полиномот $f(x)$ на следниот начин:

$$f(x) = s + a_1x + a_2x^2 + a_3x^3 + ... + a_{t-1}x^{t-1} \tag{2.8}$$

Повторно исто како во шемата на Shamir, коефициентите $a_1, a_2, a_3, ..., a_{t-1}$ се случајно избрани позитивни броеви, нултиот коефициент s претставува тајната која треба да се дели. Извршителот на протоколот за делење на тајна пресметува делови од тајната за секој играч P_i во протоколот на следниот начин:

$$(i, f(i)), i = 1, ..., n \tag{2.9}$$

Дополнително, делителот ги пресметува и заложувањата за коефициентите на полиномот $f(x)$ и неговата евалуација во i:

$$\begin{aligned} c_0 &= g^s, \\ c_1 &= g^{a_1} \\ c_2 &= g^{a_2} \\ &... \\ c_{t-1} &= g^{a_{t-1}} \\ d_i &= g^{f(i)} \end{aligned}$$

По овие пресметки, делителот му ги праќа точката $(i, f(i))$ и заложувањата $c_0, c_1, ..., c_{t-1}$ и d_i на играчот P_i. Играчот P_i сега е во можност да верифицира дека точката $(i, f(i))$ е вистинска евалуација на $f(x)$ во i преку пресметување на следната равенка:

$$\begin{aligned} c_0 c_1^i ... c_{t-1}^{i^{t-1}} &= \\ g^s (g^{a_1})^i ... (g^{a_{t-1}})^{i^{t-1}} &= \\ g^s g^{a_1 i} ... g^{a_{t-1} i^{t-1}} &= \\ g^{s+a_1 i+...+a_{t-1} i^{t-1}} &= \\ g^{f(i)} &= d_i \end{aligned}$$

Со хомоморфното својство на експонентите во групата G се добива шема за верификација. Бројот на елементи кои ги дава делителот во шемата на Feldman на секој играч е зголемен за дополнителни t елементи од групата G во однос на шемата на Shamir. Исто така, бројот на аритметичките операции во шемата на Feldman е зголемен за дополнителни t степенувања во однос на шемата на Shamir. Зголемувањето на аритметички операции се зголемува и кај играчот P_i, кај него се зголемува за дополнителни $t-1$ степенувања и $t-1$ множења на елементи од групата G во однос на шемата на Shamir.

2.4.5 Делење на тајна без чесен делител

Шемите на Shamir и Feldman побаруваат делител на тајна кој мора да биде чесен. Во протоколи каде не сме сигурни дека играчот - делител е чесен треба да се користи шема за делење на тајна без чесен делител. Таква шема е шемата за Дистрибуирано генерирање на клуч (анг. Distributed Key Generation - DKG). Оваа шема прв пат ја конструира Pedersen [91] во 1992 година.

Дефиниција 2.13. Шема на делење на тајна без чесен делител (t, n) се нарекува делењето на тајна S на делови $S_1, S_2, ..., S_n$ на следниот начин:

- било кои t собрани делови или повеќе овозможуваат реконструкција на S,
- било кои $t-1$ собрани делови или помалку не даваат никаква информација за S.

Шемата се состои од две фази:

- фаза на делење - секој играч P_i дава делови од неговата тајна S_i на останатите играчи,
- фаза на реконструкција - сите играчи ги добиваат деловите од тајните на останатите играчи, се реконструираат сите тајни S_i на играчите P_i и на крај се добива поделената тајна S.

Оваа шема се нарекува шема на делење на тајна без чесен делител со прагова вредност t.

Шемата за дистрибуирано делење на тајна се користи за исти цели како шемата за делење со чесен делител 2.4.1. Новитет е елиминирање на потребата од чесен делител. Оваа шема може да се користи во протоколи каде не може да се верува на еден единствен играч кој ќе ја знае тајната S која треба да се дели. Во оваа шема тајната S не ја знае никој до крај на фазата за реконструкција. Сите играчи P_i учествуваат во тајната S преку делење на нивните тајни S_i.

Шемата на Pedersen [91] претставува пример за шема за делење на тајна без чесен делител. Оваа шема се користи во протокол за дистрибуирано делење на клуч. За оваа цел, всушност се извршуваат n паралелни инстанци на шема за делење на тајна. Секој играч P_i извршува шема за делење на тајна S_i добивајќи n делови кои ги дистрибуира на останатите играчи. На овој начин ниту еден играч нема информации за тајната S која е всушност комбинација од сите S_i. Доколку е потребно некаква операција со S (како на пример екстракција на клуч во ID-базирана криптографија) тогаш сите играчи извршуваат операција над сопственото S_i и на крај се добива резултат без да се открие S. Доколку е потребно да се открие S тогаш се извршува фазата на реконструкција при што сите играчи ги откриваат поделените делови на секое S_i за да се реконструира финалната поделена тајна S.

Глава 3

Панда

3.1 Вовед во елиптични криви

Од касните 90ти години на минатиот век и раните 2000ти, кога Ohgishi, Sakai, Kasahara [86, 102, 103] и Joux [65, 66] ги презентираа првите конструктивни имплементации на криптографски парови, се предложија многу криптографски протоколи базирани на парови. Почетоците на ID-Базираната криптографија на Boneh и Franklin [20] и кратките потписи на Boneh, Lynn и Shacham [22], беа следени од мноштво трудови кои презентираат сè повеќе и повеќе шеми базирани на парови со нови интересни криптографски функционалности. Примерите вклучуваат шеми за хиерархиска ID-Базирана енкрипција [61, 50], енкрипција базирана на атрибути [100], системи за неинтерактивни докази со нула знаење [56, 55] и случајни докази и анонимни акредитив [10].

Од друга страна, перформансите на пресметувањата на парови исто така се драстично подобрени. Клучни точки во овие истражувања се конструкциите на различни фамилии на криви пријателски кон паровите (за осврт види [44]), многу оптимизации во алгоритмот на паровите вклучувајќи елиминација на делителот во лупата на Miller, BKLS02, брзи алгоритми за пресметување на финалното степенување [107] и воведувањето на техники за намалување на лупата [58], кои доведоа до нотацијата *оптимални парови* [116]. Неколку труда презентираат брз софтвер кој пресметува 128-битни парови за различни Intel и AMD процесори [81, 18, 6, 77] и за ARM процесори со NEON поддршка [104]. Овие трудови го намалија времето потребно за пресметка на

пар на 128-битно ниво под 0.5 милисекунди [104].

Меѓутоа, овие истражувања за пресметката на парови не влијаат реално во забрзувањето на имплементирани протоколи кои користат парови. Причина за тоа е фактот што протоколите користат многу повеќе операции покрај пресметката на парови. На овие протоколи им треба брза аритметика во сите вклучени групи, брзо хеширање во групите на елиптичната крива, брзо мулти-скаларно множење (и мулти-степенување) или специфични оптимизации за пресметка на производ на парови. Затоа овие истражувања се сметаат за некомплетни од страна на дизајнерите на криптографски протоколи. При имплементирање на некој криптографски протокол би требало да се користат различни библиотеки од наведените истражувачи бидејќи во различни фази на протоколот би биле потребни математички операции од различни библиотеки. Проблемот лежи во тоа што откако дизајнерите ќе се решат за некоја библиотека за парови, обично е потребно многу труд и време за префрлување на друг софтвер или библиотека. Уште потешка е работата при мешање на повеќе библиотеки кои не се компатибилни меѓу себе.

Како што наведува авторот во [105], потребните оптимизации на пресметката на паровите или некоја друга аритметичка операција многу зависат од протоколот кој се имплементира. Тоа значи дека постојат сценарија каде пресметката на пар претставува доминантната цена во целиот протокол, додека во други сценарија многу останати аритметички операции се тесното грло (види [90]). Доколку протоколот содржи повеќе степенувања во група отколку пресметки на пар, во некои случаеви се доаѓа до заклучок дека е подобро да се избере различна фамилија на криви пријателски за парови кои дозволуваат брзи операции во групата за сметка на малку поскапото пресметување на пар (односот на степенување во група и пресметка на пар може да се види во [23]). Во имплементација која е скроена само за брзи парови обично е тешко да се манипулира со изборот на елиптични криви бидејќи таа брза имплементација е дизајнирана врз одредена елиптични крива која не може да се смени.

Оваа глава ја воведува Панда, софтверска библиотека која ги адресира погорните размислувања преку подобрувањето на пресметките на парови (и општо целата аритметика во групите) и лесно користење за дизајнери на протоколи. Овој проект е инспириран од eBACS проектот за тестирање [16], кој

дефинира различни API-ја за различни типични криптографски примитиви и протоколи (ова вклучува хеш функции, поточни шифрувачи, енкрипција со јавен клуч и криптографски потписи). Панда може да се гледа како генерализација на eBACS за функции на ниско ниво за елиптични криви и парови.

3.1.1 Тип-1, Тип-2 и Тип-3 парови

Нашата имплементација на Панда API-то моментално поддржува само одредено множество на параметри за Тип-3 парови, меѓутоа API-то е дизајнирано да поддржува различни елиптични криви кои се пријателски кон паровите. Подглавата 3.2 објаснува како API-то поддрржува и Тип-1 парови. Во досегашните пристапи кон имплементирање на сигурни (на пример 128-битни) Тип-1 парови се користеле суперсингуларни криви преку бинарни или тернарни полиња. Меѓутоа, истражувањата во решавање на DLP проблемот во мултипликативни групи од полиња со мала карактеристика во [67, 52, 7, 1] подигнаа сериозни загрижувања за сигурноста на овие конструкции. Според истражувањата на Joux се покажува дека паровите на криви преку полиња со мали карактеристики не се сигурни [46]. Затоа постои размислување во научните кругови дека овие парови не се препорачливи повеќе. Поради овој факт ние ќе вклучиме имплементација од Тип-1 парови кои користат пристап сличен на [114] и [124].

Според Chatterjee и Menezes во [28] Тип-2 паровите претставуваат чисто неефикасни имплементации на Тип-3 паровите. Поради тоа се чини дека Тип-2 паровите не придонесуваат кон протоколите кои побаруваат асиметрични парови од гледна точка на функционалност, сигурност и перформанси. Затоа експлицитно не поддржуваме Тип-2 парови, меѓутоа е многу лесно да се вклучат овие парови во Панда. Единствената разлика од аспект на API-то е тоа што недостига хеширање во втората група на аргументите на паровите.

Тип-3 паровите се сетирани на следниот начин. Парот е недегенеритативна, билинеарна функција $e : G_1 \times G_2 \to G_3$, каде G_1 и G_2 се групи од прост ред r кои содржат рационални точки на обична елиптична крива E пријателска за парови преку конечно поле $\mathbb{F}_p$ од карактеристика p, каде p е прост број. Елиптичната крива E има мал степен на вклучување k, што значи дека групата G_3 е групата на r-тите корени на взаемност во мултипликативната

група $\mathbb{F}^*_{p^k}$, т.е. сите три групи се од ред r.

3.1.2 Аритметика во групите за непарови

Панда исто така има API за аритметика во групите кои не поддржуваат ефикасно пресметување на парови (како што се елиптични криви кои не се пријателски кон парови). Доколку протоколите немаат потреба од ефикасно пресметување на парови, тогаш може да се избере група од поголемо множество на групи во кои DLP проблемот е тежок. Кога бираме група од ова множество на групи, обично бираме групи со побрза аритметика во групата.

API-то за групите ги поддржува сите функции кои се исто така поддржани во трите групи во случај на пресметка на парови. Нашата имплементација на ова API користи групи од Edwards фамилијата која се користи за Ed25519 потписите [13, 14]. Меѓутоа, оваа глава повеќе се фокусира на описот на пресметка и користење на парови во Панда.

3.1.3 Важноста за алгоритми со константно време

Покрај постојаните напади на тешките проблеми на кои се базира модерната криптографија, големи закани претставуваат и нападите од странични канали (анг. side-channel). Поспецифично, временските напади (кои често може да се извршуваат и од далечина) се покажуваат како многу моќни алатки за напад. Трудовите [87, 115, 25, 40, 119] даваат повеќе примери за временски напади на криптографски софтвер.

Некој ќе каже дека библиотека за дизајнирање на протоколи не би требало претерано да комплицира со внимавањето на овие напади, да го чува API-то чисто и да додаде заштита против овие напади само во финална имплементација која реално би се користела. Се спротивставуваме на ова од две причини. Првата причина е фактот што кога веќе еднаш ќе се објават незаштитени верзии на библиотеки (во однос на временските напади, или било кои други напади) невозможно е да се осигураме дека тие верзии нема да завршат во некој софтвер за реална употреба. Втората причина и тоа што заштитата против временски напади не додава константно зголемување. Цената за имплементирање на заштита многу зависи од дизајнот на протоколот, на алгоритмот и на параметрите кои се користат. На пример, комплетноста

на законот на група на Edwards кривите [15, 12] дозволува лесно заштитување на операцијата собирање во група против временски напади. Возможно е да се заштити собирање на две точки од Weierstrass крива против временски напади (види Подглава 3.3.2), меѓутоа вклучува значајно зголемување на комплексноста на алгоритмот и времето на извршување.

Оптимизирање на перформансите на незаштитени имплементации на криптографски протоколи може да води кон погрешни одлуки кои е многу тешко да се поправат подоцна. Панда го решава ова со нудење на сите аритметички операции во верзија која е заштитена од временски напади (се извршуваат во константно време). За операции кои не вклучуваат сензитивни податоци постојат побрзи верзии кои не се извршуваат во константно време. Овие незаштитени верзии на функциите мора да се изберат експлицитно - предефинираните функции се во константно време.

3.1.4 Слична работа

Постојат различни криптографски библиотеки кои нудат функционалност на ниско ниво - како што се аритметика во група и парови преку нивното API. Меѓутоа, API-то кое нуди пристап до овие функционалности обично е скроено да одговара на специфичните потреби на примитиви од високо ниво кои ги имплементира библиотеката. Обично библиотеките не се дизајнирани за ефикасна имплементација на различни нови протоколи. Некои библиотеки кои користат аритметика во групите дури одлучуваат да не ги откријат овие функционалности на ниско ниво преку API-то бидејќи оваа функционалност не била планирана да се користи надвор од границите на примитивите од високо ниво на библиотеката. Како на пример, посочуваме на високото ниво на API во NaCL [17]. Два примери за криптографски библиотеки со згодно API за парови и аритметика во групи се RELIC [5] и Miracl [27].

Библиотеката која е експлицитно дизајнирана за користење на различни протоколи со парови е PBC библиотеката [74]. Внимателно изработениот дизајн е причината што оваа библиотека сеуште е преферираната библиотека за имплементирање на различни протоколи и покрај фактот што не нуди најдобри перформанси и криви со висока сигурност. Панда API-то е дизајнирано со истата цел како PBC. Меѓутоа, имплементација на Панда нуди најдобри перформанси кои се досега познати и со избор на криви кои нудат

128-битна сигурност (види секција 3.3.3 и секција 3.4.3). Уште повеќе, Панда е дизајнирана како библиотека која нуди (и охрабрува) измени од различни дизајнери за рефлектирање на најдобри перформанси во аритметика во групи и пресметување на парови.

Друга библиотека за лесно имплементирање на криптографски протоколи е Charm [3]. Charm нуди Python API и користи повеќе криптографски библиотеки за да постигне добри перформанси. За пресметка на парови ја користи PBC библиотеката. Charm претставува библиотека која се наоѓа на повисоко имплементаторско ниво од Панда. Но, не ја гледаме Панда како противник на Charm, туку изразуваме надеж дека Charm би ја вклучил Панда за пресметка на аритметика во групи и пресметка на парови за забрзување на протоколите имплементирани во неговото високо-нивовско API.

3.1.5 Организација на оваа глава

Секцијата 3.2 го објаснува API-то на Панда. Секцијата 3.3 дава детали за имплементацијата на ова API и дава мерења за брзината на сите аритметички операции. Секцијата 3.4 дава пример за лесно имплементирање на протокол базиран на парови кој достигнува најдобри перформанси користејќи ја Панда.

3.2 Панда API и функционалност

Интерфејсот на процедурите во Панда се инспирирани од APIто на eBACS што значи дека интерфејсот е специфичен за C програмскиот јазик. Има многу погодности при користење на C јазикот. Тоа е јазик кој најчесто се користи при конструирање на криптографски софтвер кој треба да е многу брз во извршување (најчесто се комбинира со Асемблер). Ова значи дека користењето на C компатибилен интерфејс дозволува лесно интегрирање на брз криптографски софтвер со Панда. Уште повеќе, протоколите кои побаруваат аритметика во групи, парови, или на пример хеширање или stream cipher, лесно може да ја комбинираат Панда со софтвер тестиран и мерен во eBACS.

Во eBACS APIто сите функции ја имаат `crypto` претставката, т.е. сите

функциите започнуваат со `crypto_`. На ист начин, сите функции и типови на податоци кои се поврзани со аритметика во групи кои поддржуваат ефикасно пресметување на билинеарен пар се со `bgroup` претставката (ознака за "bilinear group"), а функциите кои пресметуваат аритметика во група кои не поддржуваат парови се со претставката `group`.

3.2.1 Податочни типови во Панда

Функционалноста која е тестирана и мерена во Панда е претставена на ниво на аритметички операции. Дизајнирањето на Панда API-то е пониско од дизајнирање на криптографски протоколи во однос на архитектура во софтверот. Во eBACS протоколот се мерат криптографски примитиви и протоколи, додека Панда се користи за мерење на аритметички операции кои треба да се користат за дизајнирање на криптографски протоколи. Ова има последици при избирањето на податочните типови во Панда во влезните и излезните податоци на функциите. Во eBACS сите функции примаат на влез низа од бајти (C податочен тип `unsigned char`) и должина на овие низи специфицирани како тип `unsigned long long`. Излезите на функциите се исто така низи од бајти. Типична имплементација на криптографски протокол во eBACS прво ги конвертира низите од бајти во некоја интерна репрезентација (што дозволува брзо извршување на операциите), потоа ги извршува сите операции користејќи ја интерната репрезентација и на крај го запишува резултатот во низа од бајти. Овие трансформации обично додаваат многу малку во цената на извршување на протоколот. Протоколите имплементирани во Панда обично користат секвенци на операции од Панда API-то и затоа сакаме да го скокнеме непотребното конвертирање на податоците од низи од бајти во интерна репрезентација на почетокот на секоја функција и обратно, на крајот на секоја функција. Имплементацијата на Панда API-то дефинира четири податочни типови во датотеката `api.h`, кои ги претставуваат елементите на трите групи G_1, G_2 и G_3 и скаларите (по модул на големината на групата). Овие податочни типови (`struct` во C) се наречени `bgroup_g1e`, `bgroup_g2e`, `bgroup_g3e` и `bgroup_scalar`. API-то нуди две функции за конвертирање, едната конвертира елемент од G_1, G_2, G_3 или скалар во уникатна низа од бајти со фиксна големина (`pack`), другата конвертира ваква низа од бајти назад во елемент од некоја од групите или скалар (`unpack`). Имплемен-

тацијата на Панда API-то ги специфицира големините на овие спакувани елементи во датотеката `api.h`:

```
#define BGROUP_G1E_PACKEDBYTES 32
#define BGROUP_G2E_PACKEDBYTES 64
#define BGROUP_G3E_PACKEDBYTES 384
#define BGROUP_SCALAR_PACKEDBYTES 32
```

Оваа датотека специфицира дека спакуваните елементи од G_1 се 32 бајти, спакуваните елементи од G_2 се 64 бајти итн. Панда автоматски генерира хедер датотека `panda_bgroup.h` од датотеката `api.h` која ги дефинира сите функции на Панда API-то. За групата G_1, функциите `unpack` и `pack` се:

```
int bgroup_g1e_unpack(bgroup_g1e *r,
                      const unsigned char b[BGROUP_G1E_PACKEDBYTES]);

void bgroup_g1e_pack(unsigned char r[BGROUP_G1E_PACKEDBYTES],
                     const bgroup_g1e *b);
```

Следејќи ја eBACS конвенцијата, `unpack` функцијата враќа integer вредност, које е нула секогаш кога се прима валидна низа од бајти која може да биде отпакувана во елемент од групите или скалар. Во случај на невалидна низа од бајти која не може да се отпакува, функцијата `unpack` враќа integer вредност која не е 0.

Наредно ќе ги објасниме API функциите за аритметика од G_1, а еквивалентни функции постојат и во G_2 и во G_3.

3.2.2 Константи во Панда

За секоја од трите групи Панда имплементацијата треба да дефинира две константи: генератор на групата и неутрален елемент на групата. За групата G_1 овие се наречени `bgroup_g1e_base` и `bgroup_g1e_neutral`. Секоја имплементација треба да се осигура дека парот евалуиран во `bgroup_g1e_base` и `bgroup_g2e_base` дава `bgroup_g3e_base` како резултат. Уште повеќе, секоја Панда имплементација дефинира две константи од типот `bgroup_scalar` за елементот нула и елементот еден во прстенот на цели броеви по модул r за

трите групи G_1, G_2, и G_3. Овие константи се наречени `bgroup_scalar_zero` и `bgroup_scalar_one`.

3.2.3 Споредување на елементи во група

Еден начин за тестирање на еднаквост на два елемента од група е користење на `bgroup_g1e_pack` функцијата на двата елемента и споредба на двете низи за еднаквост. Ова не е многу ефикасен начин, освен кога секако правиме `bgroup_g1e_pack` во функцијата која се извршува. На пример, да земеме две точки од елиптична крива во проективни координати. Конверзијата во *уникатна* низа од бајти бара трансформација во афини координати што побарува неколку инверзии и неколку множења. Споредбата на овие две точки за нивна еднаквост без конверзија во низа од бајти побарува само неколку множења. Во Панда API-то се специфицира функција за споредба

```
int bgroup_g1e_equals(const bgroup_g1e *a, const bgroup_g1e *b);
```

која враќа 1 доколку двата елемента се еднакви, или враќа 0 доколку двата елемента не се еднакви.

Како што беше наведено во воведот на Панда, оваа функција мора да гарантира дека нема да испуштаат сензитивни информации за времето на извршување на различни елементи. За случај кога двата елемента не се сензитивни податоци постои функцијата

```
int bgroup_g1e_equals_publicinputs(const bgroup_g1e *a,
                                   const bgroup_g1e *b);
```

која се однесува на истиот начин како и претходната функција, меѓутоа може да има различно временско однесување за различни операнди. Од друга страна, оваа функција може да е побрза во извршувањето од претходната функција.

3.2.4 Собирање и дуплирање

Во конкретни имплементации на парови, групите G_1 и G_2 се адитивни групи, додека групата G_3 е мултипликативна група. Од тука, основните операции за аритметика во групи се собирање и дуплирање во G_1 и G_2, и множе-

ње и квадрирање во G_3. Трите групи ги третираме како апстрактни Абелови групи и користиме заедничка нотација за операциите во групите. Многу трудови го третираат парот како црна кутија која користи мултипликативна нотација за трите групи. Наместо тоа, Панда API-то користи адитивна нотација следејќи го `crypto_scalarmult` API-то од SUPERCOP околината за тестирање во eBACS.

Собирање на два елемента, дуплирање и негација (пресметка на инверзен елемент) се прават преку следните функции:

```
void bgroup_g1e_add(bgroup_g1e *r, const bgroup_g1e *a,
                    const bgroup_g1e *b);
void bgroup_g1e_double(bgroup_g1e *r, const bgroup_g1e *a);
void bgroup_g1e_negate(bgroup_g1e *r, const bgroup_g1e *a);
```

Забележуваме дека повратната вредност секогаш се запишува во првиот аргумент на функцијата (како во eBACS API-то и во GMP API-то [51]). Исто така, забележуваме дека имплементацијата се осигурува дека функциите за собирање и дуплирање работат за сите елементи на групата и дека не се издаваат никакви временски параметри при различни влезови на функциите. Како што напоменавме и претходно, постојат потенцијално временски неконстантни верзии на овие функции кои може да се побрзи:

```
void bgroup_g1e_add_publicinputs(bgroup_g1e *r, const bgroup_g1e *a,
                                 const bgroup_g1e *b);
void bgroup_g1e_double_publicinputs(bgroup_g1e *r,
                                    const bgroup_g1e *a);
void bgroup_g1e_negate_publicinputs(bgroup_g1e *r,
                                    const bgroup_g1e *a);
```

3.2.5 Скаларно множење

Базичната функција за скаларно множење е

```
void bgroup_g1e_scalarmult(bgroup_g1e *r, const bgroup_g1e *a,
                           const bgroup_scalar *s);
```

Оваа функција може да се изврши многу побрзо кога множиме со фиксирана базна точка која е позната во времето на компајлирање. Овие потенцијално побрзи верзии се поддржани за генераторот на групата преку функцијата

```
void bgroup_g1e_scalarmult_base(bgroup_g1e *r,
                                const bgroup_scalar *s);
```

Имплементирани се дополнителни подобрувања во случај на мулти-скаларно множење, т.е. кога е потребно да се пресмета сумата $\sum_{i=0}^{m-1} s_i P_i$ од неколку скаларни множења за m скалари $s_0, \ldots, s_{m-1}$ и m елементи од групата $P_0, \ldots, P_{m-1}$. Овие пресметки се поддржани преку следната функција која во последниот аргумент (`unsigned long long`) го специфицира бројот m од скаларни множења кои треба да се извршат во сумата.

```
void bgroup_g1e_multiscalarmult(
bgroup_g1e *r, const bgroup_g1e *a,
const bgroup_scalar *s, unsigned long long alen);
```

Повторно, оваа функција треба да ги прифаќа сите можни влезови и да се извршува во константно време без да издава било какви временски информации за различни влезови. Панда API-то исто така поддржува функција во неконстантно време (`publicinputs` верзија) која е потенцијално побрза за несензитивни влезови.

3.2.6 Хеширање во G_1 и G_2

Постојат многу протоколи кои побаруваат хеширање на бит-низи во елементи од групите G_1 или G_2. Затоа во Панда APIто е имплементирано хеширање во овие две групи. Хеширањето во G_1 е:

```
void bgroup_g1e_hashfromstr(bgroup_g1e *r, const unsigned char *a,
                            unsigned long long alen);
```

За оваа функција исто така постои неконстантна верзија од (`publicinputs`) API-то. Мора да се напомене дека двете верзии на оваа функција се однесуваат различно, односно двете точки добиени со користењето на двете верзии ќе

бидат различни за ист влез. Поради конзистентност во протоколот, потребно е извршување на истата функција (или функцијата во константно време, или функцијата во неконстантно време) врз истата бит-низа во сите фази на протоколот.

3.2.7 Аритметика на скалари

Имплементирани се различни функции за аритметика на скалари по модул на редот на групата. Овие функции се потребни при градењето на различни протоколи кои можат да ја користат Панда (не само за елиптични криви и за парови над елиптични криви). Имплементираните функции во Панда API-то се следните функции:

```
void bgroup_scalar_setrandom(bgroup_scalar *r);
void bgroup_scalar_add(bgroup_scalar *r, const bgroup_scalar *s,
                       const bgroup_scalar *t);
void bgroup_scalar_sub(bgroup_scalar *r, const bgroup_scalar *s,
                       const bgroup_scalar *t);
void bgroup_scalar_negate(bgroup_scalar *r, const bgroup_scalar *s);
void bgroup_scalar_mul(bgroup_scalar *r, const bgroup_scalar *s,
                       const bgroup_scalar *t);
void bgroup_scalar_square(bgroup_scalar *r, const bgroup_scalar *s);
void bgroup_scalar_invert(bgroup_scalar *r, const bgroup_scalar *s);
int  bgroup_scalar_equals(const bgroup_scalar *s,
                          const bgroup_scalar *t);
```

Обично аритметиката на скаларите не е тесното грло во протоколи кои користат парови над елиптични криви. Уште повеќе, не очекуваме значителни забрзувања за неконстантни верзии на овие функции. Затоа API-то не вклучува `publicinputs` верзии на функциите за аритметика на скалари.

3.2.8 Парови и производ на парови

На крај, API функцијата за пресметување на пар над елиптична крива е

```
void bgroup_pairing(bgroup_g3e *r, const bgroup_g1e *a,
                    const bgroup_g2e *b);
```

Некои протоколи побаруваат или може да искористат производ на неколку парови (на пример BLS потписите во Секција 3.4). Пресметката на производ на два пара може да биде значително побрзо од пресметката на два пара независно еден од друг и потоа нивното множење. Првото објаснување е дека финалното степенување при пресметувањето на парови мора да се изврши само еднаш, а второто објаснување е дека квадрирањето во Miller лупата може да се дели помеѓу два парови. За поддршка на овие случаи, Панда API-то ја има следната функција

```
void bgroup_pairing_product(bgroup_g3e *r, const bgroup_g1e *a,
                    const bgroup_g2e *b, unsigned long long alen);
```

3.3 Имплементација на Панда референтен код

Оваа секција ја опишува референтната имплементацијата на API функциите од претходната секција. Оваа имплементација нуди платформа со 128-битна сигурност на Тип-3 парови.

3.3.1 Избор на параметри

Најдобар избор за криви пријателски кон парови на 128-битно сигурносно ниво се кривите од фамилијата Barreto-Naehrig [8] преку поле F_p со проста карактеристика со големина околу 256 бита. Ние користиме 254-битна крива $E = E_{2,254}$ која е предложена во [92] и исто така се користи во [6]. Параметарот на кривата:

$$u = -(2^{62} + 2^{55} + 1) \tag{3.1}$$

дава 254-битни прости броеви:

$$p(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1 \tag{3.2}$$

$$r(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1 \tag{3.3}$$

и равенката на кривата $E : y^2 = x^3 + 2$ над $\mathbb{F}_p$. Бидејќи степенот на вклучување е $k = 12$, нашата имплементација треба да има проширување на полето $\mathbb{F}_{p^{12}}$. Оваа екстензија е имплементирана на стандардниот начин како кула од:

$$\mathbb{F}_p \subset \mathbb{F}_{p^2} \subset \mathbb{F}_{p^6} \subset \mathbb{F}_{p^{12}} \tag{3.4}$$

Групите на елиптичната крива се $G_1 = E(\mathbb{F}_p)$ и G_2 е просторот од p-сопствени вредности на ендоморфизмот на Фробениус $F(x) = x^p$ во r-торзионата група

$$E(\mathbb{F}_{p^{12}})[r] \tag{3.5}$$

која е претставена како изоморфна група

$$G_2' = E'(\mathbb{F}_{p^2})[r] \tag{3.6}$$

на кривата од шести степен E' од E преку $\mathbb{F}_{p^{12}}$. Секогаш кога работиме со елементи од G_2 ја користиме нивната репрезентација како елементи од G_2, односно тоа се точки на крива со коефициенти во $\mathbb{F}_{p^2}$ и аритметиката со овие точки е всушност аритметика на E' преку $\mathbb{F}_{p^2}$.

3.3.2 Алгоритми

Пакување и отпакување. За да се спакуваат елементи од групите G_1 и G_2 се водиме по најчесто користениот начин на компресија на точки од елиптични криви, кој е претставен овде. За аритметика на елиптична крива точките се во Јакоби координати. За да се спакува точка, таа прво се конвертира во афини координати. Спакуваната репрезентација е 32-бајтна низа која ја има 254-битната афина x-координата заедно со LSB (least significant bit) на y-координатата во еден од двата слободни бита. Останатиот слободен бит се користи за да се репрезентира точката во бесконечност на кривата.

При вака дадена низа од бајти, алгоритмот за отпакување веднаш ја

враќа назад y-координатата и ја решава равенката на кривата за да ја добие y-координатата. Го користи LSB битот за да го одбере точното решение од можните две решенија за y-координатата. Основата на оваа операција е пресметка на квадратен корен, при што користиме два различни алгоритми за G_1 и G_2. Бидејќи $p \equiv 3 \mod 4$, во G_1 користиме $a^{(p+1)/4}$ за пресметка на квадратен корен на $a \in \mathbb{F}_p$. Во G_2 алгоритмот за отпакување го користи [2, Algorithm 9] за пресметка на квадратен корен. По добивањето на точката на кривата се проверува дали го има редот r, односно дали добиената точка е во точната подгрупа.

Елементите од G_3 се чуваат како елементи на $\mathbb{F}^*_{p^{12}}$. Алгоритмот за пакување конструира уникатна низа од бајти на дванаесетте $\mathbb{F}_p$-коефициенти на уникатниот $\mathbb{F}_{p^{12}}$-елемент од G_3. Алгоритмот за отпакување враќа назад низа од бајти во $\mathbb{F}_{p^{12}}$-елемент и проверува дали редот на добиениот елемент е r. Во овој момент, имплементацијата на Панда не компресира G_3 елементи, меѓутоа во следна итерација може големината на спакуваниот G_3-елемент да се намали на третина од $\mathbb{F}_{p^{12}}$-елементот со користење на техники опишани во [106, 53, 80, 6].

Споредба. За да се споредат елементи од групите G_1 и G_2 треба да се споредат точките кои се репрезентирани во (проективни) Јакоби координати. Стандардниот начин на споредба на овие редундантни репрезентации е да се измножат соодветните степени на Z-координатата. Ова не побарува инверзии, во споредба со конверзија во афини координати. Споредбата во групата G_3 може директно да споредува $\mathbb{F}_{p^{12}}$-елементи на соодветните компресирани репрезентации.

Хеширање во G_1 и G_2. Стандардниот алгоритам за хеширање во неконстантно време на случаен стринг во точка на елиптична крива е "пробај-и-инкрементирај" методот покажан во [22]. Пораката се конкатенира со бројач и се хешира со криптографска хеш функција во елемент на користеното конечно поле. Доколку овој добиен елемент е валидна x-координата тогаш се пресметува соодветна y-координата. Доколку добиениот елемент не е валидна x координата, се зголемува бројачот и повторно се прави процедурата. Го користиме овој метод за хеширање во неконстантно време за `publicinputs` верзијата на функцијата.

За хеширање во константно време во групите G_1 и G_2 го користиме ал-

горитмот објаснет во [43] кој е базиран на алгоритмот од Shallue и van der Woestijne објаснет во [108]. Сите условни гранки во алгоритмот (особено при бирањето на едно од трите можни решенија) се имплементирани преку операции во константно време со условно копирање.

Собирање во група. Интерната репрезентација на елементите од G_1 и G_2 се во Јакоби координати. За собирање во неконстантно време се користат формулите за собирање од Bernstein и Lange кои побаруваат 11 множења и 5 дуплирања [34]. Доколку на влез има некој од специјалните случаи кои не се покриени со формулите (собирање на две исти точки или собирање со точка во бесконечност) се користат условни гранки за префрлање на дуплирање или враќање на точка во бесконечност. Дуплирањето ги користи формулите од Lange кои побаруваат 5 дуплирања и 2 множења [35].

Не постојат комплетни формули за операцијата собирање на точки на Weierstrass крива [24, Теорема 1]. Унифицираните формули од Hussein [59, 5.5.2] можат да се справат со дуплирања, меѓутоа тие го постигнуваат тоа преку префрлање на специјалните случаи на други точки (специфично собирањето на точки од облик (x_1, y_1) и $(x_2, -y_1)$ со $x_1 \neq x_2$). Овде евалуираме две множества на формули и користиме условно копирање во константно време за да избереме меѓу двата излеза. Го правиме ова со формулите за собирање и дуплирање наведени погоре. Забележуваме дека протоколите обично не се оптоварени со операцијата собирање, туку со операцијата скаларно множење. Скаларното множење може да биде многу побрзо користејќи дедицирано собирање, се додека можеме да се осигураме дека скаларите се помали од редот на групата. Ова е исто така компатибилно со GLV/GLS декомпозицијата објаснета во следниот параграф.

Скаларен производ. За скаларно множење во Панда API-то разликуваме функции во константно време и функции во неконстантно време во трите групи на елиптичната крива. За секој наведен случај имаме имплементирано по три алгоритми: општо скаларно множење, скаларно множење со фиксирана базна точка и мулти-скаларно множење.

Скаларното множење со фиксна базна точка која е позната во времето на компајлирање е направена преку пред-пресметување на 512 мултипли на таа точка во табела и потоа користење на овие мултипли за пресметка на скаларен производ. Методот кој се користи е опишан детално во [13, 14, Section

4]. Бидејќи не очекуваме значајно забрзување со вклучување на алгоритам во неконстантно време, алгоритмот од константно време се користи во функцијата за неконстантно време.

Стандардниот случај за скаларен производ користи ефикасни ендоморфизми од BN кривата со делење на скаларите преку 2-димензионално GLV и 4-димензионално GLS декомпонирање во G_2 и G_3. Повеќе детали за GLV/GLS декомпонирање има од Bos, Costello и Naehrig во [23]. Нашиот метод во G_1 малку се разликува од методот во [23]. По скаларната декомпозиција во функцијата за константно време, заштедуваме неколку собирања преку користење на fixed-signed-window со големин 5 и две собирања по преземање, наместо користење на табела со големина на window 2 и едно собирање. Функцијата во неконстантно време користи sliding-window со големина 5. Алгоритмите во константно време во G_2 и G_3 се објаснети во [23], додека алгоритмите во неконстантно време во G_2 и G_3 користат sliding-window со големина 4.

Алгоритмите за мулти-скаларниот производ во неконстантно време прво го аплицира GLV/GLS скаларно декомпонирање. За мали мулти-скаларни производи се користи joint-signed-sliding-window скаларен производ, додека за поголеми мулти-скаларни производи (> 16 за G_1 и > 8 за G_2 и G_3) се користи Bos-Coster скаларниот производ објаснет во [36, Section 4]. Поради бавните комплетни рутини за собирање, за функциите во константно време сега се користи посебен скаларен производ и собирање на крај. За групата G_3 се чини исплатливо имплементирањето на степенување на компресирани вредности според методот на Stam и Lenstra [111]. Планираме да ја искористиме оваа оптимизација во наредни верзии на Панда.

Пресметување на парови. Алгоритмот за пресметување на оптимални ате парови на истите BN криви е како во [6]. За разлика од [6], ние не ги користиме стандардните проективни координати, туку Јакоби координатите како во [81]. Исто така, користиме "мрзлива редукција"за аритметика во проширените полиња, објаснета во [6]. Крајното степенување е имплементирано на ист начин како во [18], при што ние користиме циклотомични квадрирања како во [54, Section 3.1], меѓутоа не користиме компресирани квадрирања објаснети во [6, Секција 5.2].

Планираме да продолжиме со оптимизирање на пресметка на парови преку експериментирање на стандардни проективни координати и крајно степе-

нување со компресирани квадрирања како во [6] и преку користење на побрза аритметика на пониско ниво.

Аритметика на пониско ниво. Аритметиката на пониско ниво во $\mathbb{F}_p$ и аритметика на скалари е имплементирано во асемблер со `AMD64` инструкциско множество. Користиме Montgomery репрезентација за елементите во $\mathbb{F}_p$. Скаларите се репрезентирани во "стандардна" форма бидејќи во скаларниот производ ни треба пристап до бинарната репрезентација. Модуларната редукција на скалари ја користи редукцијата на Barrett од [9].

Сеуште немаме имплементирано асемблерска аритметика во $\mathbb{F}_{p^2}$. Планираме да ја додадеме оваа оптимизација и очекуваме големо забрзување на пресметката на парови и во аритметиката во G_2 и G_3.

Исто така, имаме компатибилна имплементација на аритметиката во поле целосно напишана во C за поддршка на останатите платформи. Ќе продолжиме со оптимизација на софтверот со асемблерски рутини за останати платформи, имено за ARM процесорите со NEON поддршка.

3.3.3 Перформанси

Го тестиравме нашиот софтвер (со $\mathbb{F}_p$ аритметика имплементирана во асемблер) на едно јадро на Intel Core i5-3210M процесор со исклучени Turbo Boost и Hyperthreading. За секоја функција се извршени 100 пресметки врз случајни податоци. Медијаната и квартилите на секој циклус се дадени во табелите 3.1, 3.2, 3.3, и 3.4.

3.4 Имплементирање протоколи со Панда

Во оваа секција ги разгледуваме BLS потписите [22] како мал пример за протокол со парови имплементиран во Панда. Го одбираме овој пример бидејќи го илустрира користењето на повеќе API функции на Панда и бидејќи криптографските потписи (за разлика од некои други покомплексни протоколи) се поддржани во eBACS проектот за мерење на перформанси [16]. Софтверот презентиран во оваа секција го имплементира eBACS API-то за криптографски потписи, а се планира поднесување на овој алгоритам за тестирање во eBACS проектот.

3.4.1 Bonneh-Lynn-Shacham шемата за потписи

Овде накратко ги опишуваме трите алгоритми на Bonneh-Lynn-Shacham (накратко BLS) [22] потписите - генерирање на клуч, потпишување и верификација. Овие алгоритми користат асиметричен пар од Тип-3. Нека $Q \in G_2$ е дефинирана базна точка за G_2.

Генерирање на клуч. Избери случаен скалар $s \in \mathbb{Z}_r^*$. Пресметај го скаларниот производ $R \leftarrow [s]Q$. Врати го R како јавен клуч и s како приватен клуч.

Потпишување. Хеширај ја пораката m во елемент M во G_1. Искористи го приватниот клуч s за да пресметаш $S = [s]M$. Врати ја x-координатата од резултатот S како потпис σ.

Верификација. По добивање на потпис σ, порака m и јавниот клуч R, пронајди елемент $S \in G_1$ т.ш. неговата x-координата одговара на σ и е од ред r. Доколку не постои таква точка одбиј го потписот. Потоа, пресметај го $t_1 \leftarrow e(S, Q)$. Пресметај го хешот $M \in G_1$ на пораката m и пресметај го $t_2 \leftarrow e(M, R)$. Потписот се прифаќа доколку $t_1 = t_2$ или $t_1 = -t_2$. Забележуваме дека ја користиме адитивната нотација во G_3. Оваа шема побарува еден скаларен производ за генерирање на клуч, еден скаларен производ за генерирање на потпис и споредба на два пара за верификација на потпис. Во нашиот случај потписот е спакувана вредност на точка од елиптичната крива која вклучува информација за знакот на точната y-координата. Затоа ја пресметуваме уникатната точка S која кореспондира со потписот σ, за на крај само да верифицираме дали важи $e(-S, Q) \cdot e(M, R) = 1$.

3.4.2 Имплементација во Панда

Нашата имплементација го следи eBACS API-то кое се состои од три функции: `crypto_sign_keypair`, `crypto_sign` и `crypto_sign_open`. Деталите за овие функции следат во следните параграфи.

Функцијата `crypto_sign_keypair` го генерира парот на јавен/приватен клуч. Тој побарува скаларен производ со фиксирана базна точка од G_2. Комплетниот код за генерирање на овој пар е даден во шемата 3.4.1. Макрото `CRYPTO_BYTES` кое е потребно за eBACS APIто се сетира на вредноста на

BGROUP_G1E_PACKEDBYTES во датотеката api.h.

```
int crypto_sign_keypair(
  unsigned char *pk,
  unsigned char *sk
)
{
  // private key //
  bgroup_scalar x;
  bgroup_scalar_setrandom(&x);
  bgroup_scalar_pack(sk, &x);

  // public key //
  bgroup_g2e r;
  bgroup_g2_scalarmult_base(&r, &x);
  bgroup_g2_pack(pk, &r);

  return 0;
}
```

Шема 3.4.1: Генерирање на јавен и приватен клуч

```
int crypto_sign(
unsigned char *sm,
unsigned long long *smlen,
const unsigned char *m,
unsigned long long mlen,
const unsigned char *sk
)
{
bgroup_g1e p, p1;
bgroup_scalar x;
int i,r;

bgroup_g1e_hashfromstr_publicinputs(&p, m, mlen);
r = bgroup_scalar_unpack(&x, sk);
bgroup_g1e_scalarmult(&p1, &p, &x);
bgroup_g1e_pack(sm, &p1);

for (i = 0; i < mlen; i++)
{
sm[i + CRYPTO_BYTES] = m[i];
*smlen = mlen + CRYPTO_BYTES;
}

return -r;
}
```

Шема 3.4.2: Генерирање на потпис

Функцијата `crypto_sign` го пресметува потписот по добивањето на пораката. Оваа функција побарува хеширање во G_1 и еден скаларен производ во G_1. Претпоставуваме дека пораката не претставува сензитивен податок, па затоа ја користиме `publicinputs` верзијата на функциите. Комплетниот код за потпишување е даден во шемата 3.4.2.

```
int crypto_sign_open(
unsigned char *m,
unsigned long long *mlen,
const unsigned char *sm,
unsigned long long smlen,
const unsigned char *pk)
{
bgroup_g1e p[2];
bgroup_g2e q[2];
bgroup_g3e r;
unsigned long long i;
int ok;
ok  = !bgroup_g1e_unpack(p, sm);
bgroup_g1e_negate_publicinputs(p, p);
q[0] = bgroup_g2e_base;
bgroup_g1e_hashfromstr_publicinputs(p+1, sm + CRYPTO_BYTES, smlen - CRYPTO_BYTES);
ok &= !bgroup_g2e_unpack(q+1, pk);
bgroup_pairing_product(&r, p, q, 2);
ok &= bgroup_g3e_equals(&r, &bgroup_g3e_neutral);
if (ok)
{
for (i = 0; i < smlen - CRYPTO_BYTES; i++)
m[i] = sm[i + CRYPTO_BYTES];
*mlen = smlen - CRYPTO_BYTES;
return 0;
} else
{
for (i = 0; i < smlen - CRYPTO_BYTES; i++)
m[i] = 0;
*mlen = (unsigned long long) (-1);
return -1;
}
}
```

Шема 3.4.3: Верификување на потпис

Функцијата `crypto_sign_open` верифицира дали потписот припаѓа на пораката. Како што објаснивме во претходната подсекција, наивниот метод за споредба на еднаквост на два пара е да се пресметаат тие два пара и да се споредат резултатите. Меѓутоа, очигледно е дека може да се избегне пресметката на двата пара. Наместо тоа, може да се пресмета производ на два

пара и проверка дали тој производ е еднаков на еден. Во тој случај, верификацијата побарува хеширање во G_1 и еден производ на парови. Кодот за верификација на потписот е даден во шемата 3.4.3.

3.4.3 Перформанси

Ги измеривме перформансите на BLS имплементацијата на Intel Core i5-3210M кој работи на 2.5 GHz кој исто така го користевме за деталните анализи на нашата reference имплементација на Панда API-то. Генерирањето на клуч трае 378848 циклуси. Потпишувањето (на 59-битна порака) зема 434640 циклуси (ова е медијана на 10000 мерења, првиот и третиот квартил се 428616 и 511764). Верификацијата на потписот на оваа порака трае 5832584 циклуси (повторно, ова е медијана, првиот и третиот квартил се 5797640 и 5874292). Според литературата која ја најдовме на оваа тема, ова е најбрзата имплементација на BLS потписите на 128-битно сигурносно ниво. Како идна работа останува споредбата со перформансите на BLS имплементацијата од Scott вклучено во SUPERCOP. За жал не успеавме да ја искомпајлираме оваа имплементација на AMD64 платформата. Исто така, во eBACS во време на пишување не постоеше измерен резултат за "BLS" за оваа платформа.

Го извршивме тестирањето во RELIC платформата (верзија 0.3.5) на истата машина што ја користевме за тестирање на нашиот софтвер. Времињата од RELIC тестирањето покажаа 609966 наносекунди за генерирање на BLS клуч, 510775 наносекунди за потпишување и 6910615 наносекунди за верификација. На процесор со 2.5 GHz ова одговара на 1524915 циклуси за генерирање на клуч, 1276937 циклуси за потпишување и 17276537 циклуси за верификација. Ова претставува отприлика три пати побавно време на извршување од нашиот софтвер (вкупно 6646072 циклуси за Панда и вкупно 20078389 циклуси за RELIC). Темата и резултатите од оваа глава се објавени во трудот [31].

API функција	**25% кв.**	**мед.**	**75% кв.**
`bgroup_g1e_unpack`	39140	39184	39212
`bgroup_g1e_pack`	39512	39548	39568
`bgroup_g1e_hashfromstr (59 bytes)`	198780	198908	198964
`bgroup_g1e_add`	6052	6080	6100
`bgroup_g1e_double`	1204	1216	1224
`bgroup_g1e_negate`	36	36	40
`bgroup_g1e_scalarmult`	346852	347024	347180
`bgroup_g1e_scalarmult_base`	128468	128596	128696
`bgroup_g1e_multiscalarmult`			
$(n = 2)$	705564	705820	706056
$(n = 3)$	1058308	1058644	1059128
$(n = 4)$	1411188	1411644	1411944
$(n = 8)$	2822252	2823148	2826864
$(n = 32)$	11294736	11296364	11298420
$(n = 128)$	45181816	45186732	45193356
`bgroup_g1e_equals`	1124	1132	1140
`bgroup_g1e_hashfromstr_publicinputs (59 bytes)`	41752	83168	83696
`bgroup_g1e_add_publicinputs`	2456	2468	2476
`bgroup_g1e_double_publicinputs`	1180	1192	1200
`bgroup_g1e_negate_publicinputs`	36	36	40
`bgroup_g1e_scalarmult_publicinputs`	284228	288240	290788
`bgroup_g1e_scalarmult_base_publicinputs`	102184	104024	105772
`bgroup_g1e_multiscalarmult_publicinputs`			
$(n = 2)$	415076	419860	423440
$(n = 3)$	551124	556792	560712
$(n = 4)$	710416	715396	722000
$(n = 8)$	1229100	1238660	1246568
$(n = 32)$	4727808	4741472	4752772
$(n = 128)$	14590168	14605364	14635184
`bgroup_g1e_equals_publicinputs`	576	580	588

Табела 3.1: Број на циклуси за аритметички операции во G_1 на Intel Core i5-3210M.

API функција	25% кв.	мед.	75% кв.
bgroup_g2e_unpack	1864580	1864884	1865396
bgroup_g2e_pack	42080	42124	42160
bgroup_g2e_hashfromstr (59 bytes)	2435116	2435564	2439536
bgroup_g2e_add	16048	16072	16096
bgroup_g2e_double	2924	2940	2948
bgroup_g2e_negate	60	60	64
bgroup_g2e_scalarmult	764628	764808	765088
bgroup_g2e_scalarmult_base	336788	336916	337060
bgroup_g2e_multiscalarmult			
($n = 2$)	1563312	1563668	1564040
($n = 3$)	2344964	2345496	2346704
($n = 4$)	3126720	3127116	3131192
($n = 8$)	6253984	6257528	6258700
($n = 32$)	25024136	25027200	25031036
($n = 128$)	100103176	100117420	100157284
bgroup_g2e_equals	3100	3112	3124
bgroup_g2e_hashfromstr_publicinputs (59 bytes)	298524	299884	894696
bgroup_g2e_add_publicinputs	6572	6596	6608
bgroup_g2e_double_publicinputs	2960	2972	2992
bgroup_g2e_negate_publicinputs	60	60	64
bgroup_g2e_scalarmult_publicinputs	612012	625636	635656
bgroup_g2e_scalarmult_base_publicinputs	273468	278372	283056
bgroup_g2e_multiscalarmult_publicinputs			
($n = 2$)	1031736	1043332	1060796
($n = 3$)	1477392	1492796	1510148
($n = 4$)	1889684	1912744	1928124
($n = 8$)	3443640	3467764	3489032
($n = 32$)	10293932	10329088	10366420
($n = 128$)	32941972	32991824	33061804
bgroup_g2e_equals_publicinputs	3104	3116	3120

Табела 3.2: Број на циклуси за аритметички операции во G_2 на Intel Core i5-3210M.

API функција	**25% кв.**	**мед.**	**75% кв.**
`bgroup_g3e_unpack`	1832068	1832404	1833044
`bgroup_g3e_pack`	424	424	428
`bgroup_g3e_add`	8020	8032	8048
`bgroup_g3e_double`	5548	5560	5572
`bgroup_g3e_negate`	172	176	180
`bgroup_g3e_scalarmult`	1120300	1120552	1120936
`bgroup_g3e_scalarmult_base`	608964	609148	609320
`bgroup_g3e_multiscalarmult`			
$(n = 2)$	2255028	2255624	2258896
$(n = 3)$	3382628	3383284	3387392
$(n = 4)$	4510336	4511420	4515516
$(n = 8)$	9024924	9025736	9026820
$(n = 32)$	36100180	36103240	36109596
$(n = 128)$	144408660	144446076	144467856
`bgroup_g3e_equals`	8304	8324	8336
`bgroup_g3e_add_publicinputs`	8024	8044	8056
`bgroup_g3e_double_publicinputs`	5548	5556	5568
`bgroup_g3e_negate_publicinputs`	176	176	180
`bgroup_g3e_scalarmult_publicinputs`	852272	864136	877804
`bgroup_g3e_scalarmult_base_publicinputs`	609004	609188	609352
`bgroup_g3e_multiscalarmult_publicinputs`			
$(n = 2)$	2255104	2255424	2258836
$(n = 3)$	3382688	3383368	3387800
$(n = 4)$	4510680	4512684	4515652
$(n = 8)$	4272080	4297668	4330036
$(n = 32)$	12768868	12803832	12843124
$(n = 128)$	40764052	40825956	40876608
`bgroup_g3e_equals_publicinputs`	8304	8320	8332

Табела 3.3: Број на циклуси за аритметички операции во G_3 на Intel Core i5-3210M.

API функција	25% квартил	медијана	75% квартил
`bgroup_pairing`	2566580	2567116	2572096
`bgroup_pairing_product`			
$(n = 2)$	3831724	3832644	3837688
$(n = 3)$	5089192	5093724	5094728
$(n = 4)$	6347328	6351260	6352588
$(n = 8)$	11380604	11381384	11383420
$(n = 32)$	41565448	41569424	41588976
$(n = 128)$	162321836	162364916	162387468

Табела 3.4: Број на циклуси за пресметка на пар на Intel Core i5-3210M.

Глава 4

ID-базирана криптографија

4.1 Криптографија базирана на сертификати

Во јавната криптографија (анг. pPublic crypto) секој играч поседува два типа на клучеви: приватен и јавен. Приватниот клуч треба да се чува сигурно а јавниот клуч се остава на јавно место каде може да пристапи секој. Кога играчот A сака да криптира порака за играчот B, тогаш пораката се криптира со јавниот клуч на B. Кога играчот B ќе ја прими пораката тогаш ја декриптира со својот приватен клуч. Доколку играчот A сака да му докаже на B дека пораката е оригинално од A тогаш A ја потпишува пораката со својот приватен клуч. Во овој случај играчот B ја проверува потпишаната порака со јавниот клуч на A. Повеќе детали за асиметричната криптографија може да се видат во поглавјето 2.2.

Можеме да забележиме дека пред да се изврши операцијата криптирање, играчот A мора на некој начин да го земе јавниот клуч на B. Исто така, кога B сака да го провери потписот од A, мора да го земе јавниот клуч на A. Доколку играчите се знаат и често комуницираат меѓу себе, тогаш цената на наоѓање и земање на јавниот клуч е релативно мала во однос на цената на целата комуникација. Но, доколку некои специфични протоколи подразбираат кратка комуникација помеѓу играчи кои цело време се менуваат, тогаш цената на наоѓање и земање на јавниот клуч претставува значајно голем чекор во аспект на цена на комуникација. Во овој случај прикачувањето на јавниот клуч во секоја порака претставува зголемување на обемот на пораката, што директно води кон забавување на комуникацијата.

Во сценарија каде играчите се дел од некоја организација, тогаш организацијата може да креира тело кое ќе помага при креирањето на парот клучеви и ќе ги чува и дистрибуира јавните клучеви. Ова тело се нарекува инфраструктура за јавни клучеви (анг. public key infrastructure - PKI). Во овој случај, доколку играчот A сака да криптира порака за играчот B, ја контактира PKI инфраструктурата за да го добие јавниот клуч на B. Кога ќе ја прими пораката играчот B, ја контактира PKI инфраструктурата за да го добие јавниот клуч на A за проверка на потписот. Во PKI инфраструктурата постои Авторитет за сертификати (анг. Certificate Authority) кој ги потпишува јавните клучеви кои се доделени на членовите на PKI инфраструктурата.

Дефиниција 4.1. Инфраструктура за јавни клучеви (PKI инфраструктура) претставува тело кое менаџира со сертификати кои се издадени од Авторитет за сертификати. Инфраструктурата овозможува лесно креирање, пребарување и бришење на сертификатите од соодветниот Авторитет за сертификати.

Постоењето на PKI инфраструктура го олеснува пребарувањето на јавните клучеви на играчите за кои сакаме да криптираме пораки. Од друга страна, играчите кои сакаат да проверат потпис на порака исто така брзо и лесно доаѓаат до јавните клучеви на испраќачите. Доколку дојде до пробивање на јавниот клуч на играчот E, тогаш PKI инфраструктурата може да го повлече соодветниот сертификат и на негово место да го стави новиот сертификат со јавниот клуч соодветен на новиот приватен клуч на играчот E. Криптографијата базирана на PKI инфраструктура се нарекува криптографија базирана на сертификати.

4.2 ID-базирана криптографија

Во 1984 година Adi Shamir воведува нов тип на јавна криптографија [110]. Во овој тип јавниот клуч претставува уникатен идентификатор кој ја елиминира потребата од пребарување и земање на јавниот клуч при криптографските операции. Овој тип на криптографија се нарекува ID-Based криптографија.

Уникатниот идентификатор во ID-Based криптографијата претставува

стринг кој уникатно може да идентификува играч. Овој стринг претставува јавен клуч на кој соодветно се пресметува приватен клуч. Како пример во комуникација со ID-Based криптографија се користат email адреси како стрингови за јавни клучеви. По овој пример, доколку играчот A сака да криптира порака за играчот B, тогаш ќе ја криптира со email-от на B кој претпоставуваме дека веќе го знае. Играчот B ја декриптира пораката со својот приватен клуч, а може да го провери потписот со email-от на играчот A од кого ја добил пораката. Забележуваме дека се елиминираат чекорите на барање и земање на јавните клучеви - тие се веќе познати идентификатори на играчите. Овој тип на криптографија се нарекува и криптографија без сертификати.

Првото појавување на ID-Based криптографија е во 1984 година од Adi Shamir [110]. Тој имплементира ID-Based потписи и прикажува PKI инфраструктура во која се верифицираат потписите само од јавно-познати идентификатори - како што е email-от на корисникот. Во овој систем, корисникот при иницијализирање мора да контактира PKI инфраструктура автентицирајќи се со јавниот идентификатор - за да добие соодветен приватен клуч. Постоењето на ваква PKI Инфраструктура која ги знае сите приватни клучеви претставува најслабата точка на ID-Based криптографијата.

Поради непотполноста на криптографскиот систем (Adi Shamir измислил само ID-Based потписи) овој тип на криптографија не бил од корист се до 2000 година. Тогаш, со трудовите на Sakai [101] и Dan Boneh [21] се презентира практичен ID-Based криптосистем. Овие два труда објаснуваат комплетен ID-Based криптосистем кој може да се користи во пракса. Затоа, почнувајќи од 2001 година, почнува масовното користење на ID-Based криптографијата како полесен тип на криптографија бидејќи не побарува сертификати.

4.3 Проблеми со ID-Базирана криптографијата

4.3.1 Посредништво на клуч

Најголемата предност во ID-Based криптографијата е отсуството на сертификати. Меѓутоа, дизајнот на PKI Инфраструктурата во ID-Based криптографијата воведува проблем со посредништво на клучот. Во PKI Инфрас-

труктурата постои авторитет кој генерира приватни клучеви врз основа на идентификатор - Генератор на приватни клучеви (анг. Private Key Generator - PKG). Ова значи дека PKG има формула за пресметување на било кој приватен клуч врз основа на идентификатор. Овој факт кажува дека PKG има многу големо знаење и моќ, за разлика од Авторитетот за сертификати во класичната PKI Инфраструктура каде само се потпишува (валидира) јавниот клуч.

Решение на овој проблем дава Dan Boneh [21] преку концептот за делење на тајна (види поглавје 2.4.1). На овој начин тајната на PKG може да се подели на повеќе единки и со користење на прагова криптографија може да се пресмета приватен клуч за некој идентитет само кога потребното мнозинство ќе се согласи за тоа.

4.3.2 Повлекување на клуч

Во класичната PKI Инфраструктура лесно може да се повлече сертификат за некој идентитет. Тоа значи дека доколку некој приватен клуч се пробие, соодветниот сертификат може да се поништи во PKI Инфраструктурата. Овој процес се нарекува повлекување на клуч (анг. key revocation). Потоа, играчот може да генерира нов пар приватен/јавен клуч и да го даде во PKI Инфраструктурата за менаџирање и добивање нов сертификат. Повлекувањето на клуч во ID-Based криптографијата, споредено со истиот процес кај класичната PKI Инфраструктура, не е возможно. Кај ID-Based криптографијата постои функција која секогаш за даден идентитет (стринг) дава ист приватен клуч. Доколку овој приватен клуч е пробиен, на играчот не му останува ништо друго освен да го смени идентитетот. Поради ова велиме дека не е возможно повлекување на клуч за одреден идентитет во ID-Based PKI Инфраструктурата.

Решение на овој проблем исто така дава Dan Boneh [21] преку користење на подидентитети. На пример, доколку имаме идентитет кој е email адреса bob@company.com, може да креираме подидентитет со конкатенирање на email адресата и тековниот датум. На овој начин, секој следен ден би имале нов јавен клуч за кој би имале посебен приватен клуч. Овој јавен клуч секој може да го дознае преку спојување на email адресата и денот во кој го користи јавниот клуч. Ова значи дека нема потреба од контактирање на PKI

инфраструктурата за превземање на јавниот клуч. Добрата работа е што повлекувањето на клуч веќе не постои како проблем. Лошата работа е што Bob ќе мора секој ден да бара приватен клуч за подидентитетот на денот.

4.4 Решение со дистрибуирано генерирање на клуч

4.4.1 Вовед

Во ова поглавје ќе понудиме решение за PKI Инфраструктура базирана на ID-базирана криптографија. Проблемите со ID-Based посредништвото на клучеви објаснети во поглавјето 4.3 покажуваат дека генераторот на приватни клучеви има преголема моќ. Тоа значи дека генераторот може на своја рака да генерира приватни клучеви за било кој јавен идентитет. Затоа, базирајќи се на принципот за делење на тајна без чесен делител 2.4.5 дефинираме систем за генератор на приватни клучеви кој е дистрибуиран. На овој начин, успешно креираме PKI инфраструктура базирана на ID-базирана криптографија. Резултатите од ова поглавје се објавени во трудот [96].

Дефиниција 4.2. Сигурна PKI инфраструктура базирана на ID-базирана криптографија претставува систем за Генератор на приватни клучеви со следните карактеристики:

- системот е составен од n играчи со прагова вредност t,
- ниту еден член на системот нема моќ да генерира приватни клучеви за јавни идентитети,
- само t или повеќе собрани членови може да генерираат приватен клуч за даден јавен идентитет.

Аритметиката на предложениот систем е со елиптични криви, а програмирањето се врши во Java врз Android платформата.

Протоколот на системот е претставен во поглавјата 4.4.2, 4.4.3 и 4.4.4.

4.4.2 Протокол

Предложениот систем претставува синхрон систем на комуникација. Праќањето на пораки користи ексклузивни канали помеѓу праќачот и примачот кои се безбедни и недостапни за останатите играчи во протоколот.

Играчите во предложениот протокол се DKG Играч и Клиент. DKG Играчот претставува дел од Генераторот на клучеви на PKI Инфраструктурата, а Клиентот претставува клиент кој ќе се поврзе на PKI Инфраструктурата со својот идентитет за екстракција на приватниот клуч.

Протоколот е поделен на два главни дела:

- Поставување на системот
- Екстракција на клучеви

4.4.3 Поставување на системот

На почетокот сите DKG Играчи се поврзуваат меѓу себе, секој со секого, со сигурни и ексклузивни канали. Поставувањето на системот го иницира еден член од множеството на DKG Играчи. DKG Играчот кој го иницира поставувањето на системот го нарекуваме Лидер. Првата задача на Лидерот е да ги постави сите параметри на PKI Инфраструктурата. Параметрите кои ги поставува Лидерот се:

- елиптичната крива и нејзините параметри,
- бирање на типот на пар над елиптичната крива и бирање на трите групи G_1, G_2 и G_3 за паровите на елиптични криви,
- бирање на генератор $g\_public \in G$,
- поставување на бројот на DKG Играчи n,
- поставување на прагова вредност t и пресметување на $k = t + 1$.

Лидерот ги објавува овие параметри на огласна табла како јавни параметри. Потоа, Лидерот ја поминува листата од сите поврзани DKG играчи и им праќа порака *commandInit* за сите играчи да се иницијализираат. По

праќањето на пораката Лидерот исто така влегува во процедурата за иницијализација 4.4.1.

Играчот P_i по примањето на пораката *commandInit* ги чита јавните параметри и ја започнува процедурата за иницијализација 4.4.1. По завршувањето на иницијализацијата, играчот P_i праќа порака *commandInitFinished* до Лидерот.

1. Генерирање на $f_i(x)$:
- Случајно се избираат коефициенти $a_{i,j}, j = 0, ..., t-1$ на полиномот $f_i(x)$ при што првиот коефициент е тајната на DKG играчот P_i

2. Пресметување на заложувања:
- Се пресметуваат заложувања за коефициентите на полиномот $f_i(x)$: $c_{i,j} = g\_public^{s_i}, j = 0, ..., t-1$

3. Евалуација на полиномот:
- Се евалуира полиномот за сите останати играчи $eval_i = f_i(j), j = 1, ..., n, j \neq i$

Шема 4.4.1: Иницијализација на DKG Играч P_i

Откако Лидерот ќе прими $n-1$ пораки *commandInitFinished* ја менува состојбата на протоколот во *P_STATE_DKG_1* што означува завршување на фазата за иницијализација на играчите. Потоа, Лидерот ја започнува втората фаза во поставувањето на системот. Лидерот испраќа порака *commandSharePieces* на сите DKG играчи, а потоа влегува во процедурата за делење на заложувањата и евалуациите 4.4.2.

Играчот P_i по примањето на пораката *commandSharePieces* исто така влегува во процедурата за делење на заложувањата и евалуациите 4.4.2 која резултира со праќање на пораката *commandReceivePieces* на сите останати играчи (покажано во шема 4.4.2).

1. Делење на заложувањата:
- Играчот P_i ги спрема заложувањата $c_{i,j}, j = 0, ..., t-1$ за играчот $P_j, j = 1, ..., n, j \neq i$

2. Делење на евалуациите:
- Играчот P_i ја спрема евалуацијата на полиномот $f_i(j)$ за играчот P_j, за $j = 1, ..., n$

3. Испраќање на порака:
- Играчот P_i ја испраќа пораката *commandReceivePieces* на секој играч $P_j, j = 1, ..., n, j \neq i$

Шема 4.4.2: Делење на заложувањата и евалуациите од страна на играчот P_i

Откако Играчот P_i ќе прими порака *commandReceivePieces* влегува во процедурата на преземање на заложувањата и евалуациите на полиномот $f(x)$ од страна на Играчот $P_j, j = 1, ..., n, j \neq i$ (покажано во шема 4.4.3).

1. Преземање на заложувањата:
- Играчот P_i ги зачувува заложувањата на коефициентите на полиномот на играчот $P_j, j = 1, ..., n, j \neq i$

2. Преземање на евалуациите:
- Играчот P_i ги зачувува евалуациите на полиномот $f_j(i)$ на играчот $P_j, j = 1, ..., n, j \neq i$

3. Испраќање на порака:
- Откако Играчот P_i прими $n-1$ пораки *commandReceivePieces* ја испраќа пораката *commandSharePiecesFinished* на лидерот

Шема 4.4.3: Преземање на заложувањата и евалуациите на играчот P_i

Откако Лидерот ќе прими $n-1$ пораки *commandSharePiecesFinished* ја менува состојбата на протоколот во *P_STATE_DKG_2* што означува завршување на фазата за делење на заложувањата и евалуациите на полиномите. Потоа, Лидерот ја започнува третата фаза во поставувањето на системот. Тој испраќа порака *commandVerifyCommitmentsAndEvals* на сите DKG играчи, а потоа влегува во процедурата за проверка на заложувањата и евалуациите (покажано во шема 4.4.4).

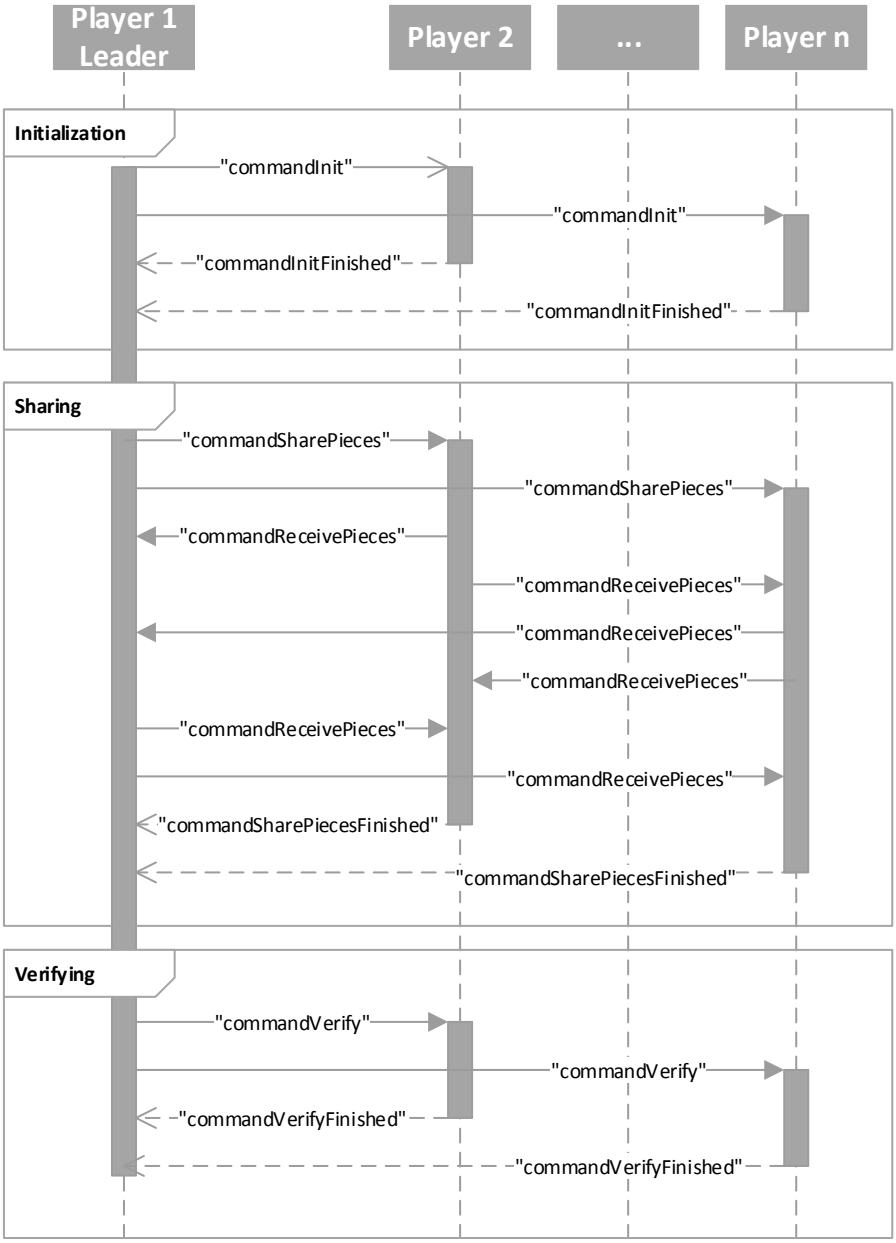
Откако Играчот P_i ќе прими порака *commandVerifyCommitmentsAndEvals* влегува во процедурата на проверка на заложувањата и евалуациите на полиномот $f(x)$ од страна на Играчот $P_j, j = 1, ..., n, j \neq i$ 4.4.4.

1. **Пресметување на заложување на групната тајна:**
   - Играчот P_i го пресметува $g\_pow\_s\_public = \prod_{j=1}^{n} c_{j,0}$
2. **Проверка на заложувањата и евалуацијата на полиномот на играчите** $P_j, j = 1, ..., n, j \neq i$**:**
   - Играчот P_i проверува дали важи равенството
   $c_{j,0}c_{j,1}...c_{j,t-1} \stackrel{?}{=} g^{f_j(i)}, j = 1, ..., n, j \neq i$
3. **Испраќање на порака:**
   - Доколку проверката е во ред играчот P_i ја испраќа пораката *commandVerifyCommitmentsAndEvalsFinished* на лидерот

Шема 4.4.4: Проверка на заложувањата и евалуациите на играчот P_i

Откако Лидерот ќе прими $n - 1$ пораки *commandVerifyCommitmentsAndEvalsFinished* ја менува состојбата на протоколот во *P_STATE_DKG_3* што означува завршување на фазата за проверка на заложувањата и евалуациите на полиномите. Оваа фаза е последна во делот од протоколот за поставување на системот. По завршувањето на оваа фаза сите DKG играчи ги поседуваат следните информации:

- елиптичната крива и нејзините параметри,
- типот на пар над елиптичната крива и трите групи G_1, G_2 и G_3 за паровите на елиптични криви,
- генераторот $g\_public \in G$,
- бројот на DKG Играчи n,
- прагова вредност t,
- заложување на тајната на целата група $g\_pow\_s\_public$,
- дел од главната тајна $s_i = a_{i,0}, i = 1, ..., n$.



Слика 4.1: Секвенцен дијаграм на фазата на поставување на системот

Секвенцниот дијаграм на поставувањето на системот е претставен на сликата 4.1.

Цената на одржување на овој протокол може да се разгледува по две основи: време на извршување на аритметички операции и број на пораки и нивната големина. Во следните табели се дадени информациите околу чинењето на извршување на овој протокол во своите три фази. Во шемите 4.4.5, 4.4.6, 4.4.7, 4.4.8, 4.4.9 и 4.4.10 е дадена цената претставена преку број и големина на пораките и цената претставена преку потребните аритметички операции.

1. **Лидерот испраќа** $(n-1)$ **порака кон останатите играчи:**
   - byte: 1
   - Integer: 2
   - G1: 1
2. **Останатите** $(n-1)$ **играчи испраќаат по една порака кон лидерот:**
   - byte: 1

Шема 4.4.5: Број на пораки во прва фаза: иницијализирање на играчи

1. **Сите играчи пресметуваат:**
   - scalar doubling: $2n+1$
   - G1 scalar multiplication: $2n(k-1)+k+1$
   - scalar addition: $n(k-1)$
2. **Лидерот пресметува**
   - G1 serialization: 1
2. **Останатите** $(n-1)$ **играчи пресметуваат**
   - G1 deserialization: 1

Шема 4.4.6: Број на операции во прва фаза: иницијализирање на играчи

1. **Лидерот испраќа** $(n-1)$ **порака кон останатите играчи:**
   - byte: 1
2. **Сите играчи испраќаат по една порака кон останатите играчи (вкупно** $n(n-1)$ **порака):**
   - byte: 1
   - G1: $k+1$
   - Scalar: 1
3. **Сите играчи враќаат по една порака кон лидерот (вкупно** $n-1$ **порака):**
   - byte: 1

Шема 4.4.7: Број на пораки во втора фаза: делење на податоци

1. **Сите играчи пресметуваат:**
   - G1 серијализаиција: $k+1$
   - скаларна серијализаиција: k

Шема 4.4.8: Број на операции во втора фаза: делење на податоци

1. **Лидерот испраќа** $(n-1)$ **порака кон останатите играчи:**
   - byte: 1
2. **Сите играчи враќаат по една порака конЛЛидерот (вкупно** $n-1$ **порака):**
   - byte: 1

Шема 4.4.9: Број на пораки во трета фаза: верификација

1. Сите играчи пресметуваат:
- G1 exponentiation: 1
- Scalar doubling: 1
- G1 doubling: 1
- Scalar exponentiation: k
- G1 scalar multiplication: $n + k$
- G1 multiplication: k
- Scalar addition: k
- G1 equality: 1

Шема 4.4.10: Број на операции во трета фаза: верификација

4.4.4 Екстракција на клучеви

Корисниците кои ќе го користат дистрибуираниот систем за PKI инфраструктура користејќи на ID-базирана криптографија треба да ги контактираат играчите кои биле дел од поставувањето на системот. За оваа цел информациите за играчите кои биле дел од поставувањето на системот се објавуваат на огласна табла. Откако системот ќе добие статус *P_STATE_DKG_3* тогаш корисниците може во било кое време да влезат во фаза на екстракција на клучеви. Фазата на екстракција очекува претходно договорен начин на идентификација на корисникот пред PKI инфраструктурата, за да може играчите во PKI инфраструктурата да генерираат делови од приватниот клуч на корисникот базиран на неговиот идентификатор.

Фазата на екстракција почнува кога корисникот ќе побара информации за PKI Инфраструктурата од било кој DKG играч кој претходно учествувал во поставувањето на системот (поглавје 4.4.3). DKG играчот ги испраќа јавните податоци за PKI инфраструктурата:

- елиптичната крива и нејзините параметри,
- типот на пар над елиптичната крива и трите групи G_1, G_2 и G_3 за паровите на елиптични криви,
- генераторот $g\_public \in G$,
- бројот на DKG играчи n,

- прагова вредност t,
- заложување на тајната на целата група $g\_pow\_s\_public$.

По добивањето на сите информации за PKI Инфраструктурата, корисникот креира ексклузивни двонасочни канали за комуникација со секој играч од PKI Инфраструктурата. По успешното поврзување со сите играчи, корисникот продолжува со извршување на протоколот за екстракција на клучот.

Првиот чекор во протоколот за екстракција на клучеви почнува кога корисникот ќе испрати порака *commandClientGetSumOfEvals* на сите играчи. Во пораката го праќа својот идентификатор id за кој сака да добие приватен клуч.

Кога играчите на PKI Инфраструктурата ќе добијат порака *commandClientGetSumOfEvals* од некој корисник, проверуваат дали идентификаторот id одговара со корисникот според претходно утврдените правила на работа на PKI Инфраструктурата. Доколку идентификацијата е успешна, секој играч P_i го пресметува делот од приватниот клуч на корисникот $priv_i = id^{s_{i,0}}$. По пресметката, испраќа порака до корисникот *commandClientReceiveSumOfEvals* заедно со $priv_i$ и $g\_public^{s_{i,0}}$.

Кога корисникот ќе прими порака *commandClientReceiveSumOfEvals* тогаш проверува дали податоците кои ги примил се конзистентни со податоците од PKI Инфраструктурата. На почеток се пресметуваат вредностите $e1$ и $e2$ и се проверува нивната еднаквост:

$$e_1 = Pairing(g\_public, priv_i) \tag{4.1}$$

$$e_2 = Pairing(g\_public^{s_{i,0}}, id) \tag{4.2}$$

$$e_1 \stackrel{?}{=} e_2 \tag{4.3}$$

Доколку $e1$ и $e2$ се еднакви (важи равенството 4.3) тогаш се прифаќа делот $priv_i$ од играчот P_i како валиден дел од приватниот клуч на корисникот.

Откако корисникот ќе собере n валидни делови од приватниот клуч, може да почне со фазата на пресметка на неговиот приватен клуч. Прво се

пресметува производ од сите n примени делови кој е пресметаниот приватен клуч на корисникот:

$$d_{id} = \prod_{i=1}^{n} priv_i \tag{4.4}$$

Потоа се пресметуваат e_1 и e_2:

$$e_1 = Pairing(g\_public, d_{id}) \tag{4.5}$$

$$e_2 = Pairing(g\_pow\_s\_public, id) \tag{4.6}$$

$$e_1 \stackrel{?}{=} e_2 \tag{4.7}$$

Со проверка на еднаквоста на овие две вредности (4.7) се верификува дека пресметаниот приватен клуч d_{id} (4.4) е валиден екстрахиран приватен клуч од сите примени делови на приватниот клуч.

Алгоритмот на протоколот за екстракција на клучеви е даден во шемата 4.4.11. Секвенцниот дијаграм за фазата на реконструкција на приватниот клуч е прикажан во графикот 4.2.

1. Барање на јавни информации за PKI Инфраструктурата:
- Корисникот испраќа порака *commandSendPlayersList* до случајно избран DKG играч од огласната табла
- Играчот добива порака *commandSendPlayersList* и испраќа порака *commandReceivePlayersList* заедно со сите информации за PKI инфраструктурата

2. Барање на дел од приватниот клуч
- Корисникот праќа порака *commandClientGetSumOfEvals* до сите играчи заедно со својот идентификатор *id*
- Секој играч P_i пресметува и испраќа $priv_i = id^{s_{i,0}}$ и $g\_public^{s_{i,0}}$ во пораката *commandClientReceiveSumOfEvals*

3. Верификација на применият дел:
- Кога корисникот ќе прими *commandClientReceiveSumOfEvals* го верифицира $priv_i = id^{s_{i,0}}$ со пресметка на $e_1 = Pairing(g\_public, priv_i)$ и $e_2 = Pairing(g\_public^{s_{i,0}}, id)$ и проверка $e_1 \stackrel{?}{=} e_2$. Доколку проверката е успешна, делот $priv_i$ од играчот P_i се прифаќа

3. Верификација и пресметка на приватниот клуч:
- Кога корисникот ќе верифицира n делови го пресметува приватниот клуч $d_{id} = \prod_{i=1}^{n} priv_i$
- Пресметаниот приватен клуч го верификува преку пресметка на $e_1 = Pairing(g\_public, d_{id})$ и $e_2 = Pairing(g\_pow\_s\_public, id)$ и проверката $e_1 \stackrel{?}{=} e_2$. Доколку проверката е успешна, приватниот клуч d_{id} се прифаќа како приватен клуч на корисникот *id* издаден од PKI инфраструктурата

Шема 4.4.11: Проверка на заложувањата и евалуациите на играчот P_i

И за екстракцијата на клучот претставуваме табели за цената претставена преку бројот на пратени пораки и нивната содржина и преку операциите што ги извршува секој играч на протоколот.

Слика 4.2: Секвенцен дијаграм на фазата на екстракција на клучот

1. **Клиентот испраќа n пораки до сите DKG играчи:**
   - byte: 1
   - String: 1
2. **Сите играчи враќаат по една порака кон клиентот (вкупно n пораки):**
   - byte: 1
   - G2: 1

Шема 4.4.12: Број на пораки при екстракција на приватниот клуч

1. **Сите DKG играчи пресметуваат:**
   - SHA1 хеширање $h_1(byte[]) -> G2$: 1
   - G2 скаларно множење: 1
   - G2 серијализација: 1
2. **Клиентот пресметува:**
   - G2 десеријализација: 1
   - пресметување на пар: $2(n+1)$
   - G3 еднаквост: $n+1$
   - G2 множење: n

Шема 4.4.13: Број на операции во трета фаза: верификација

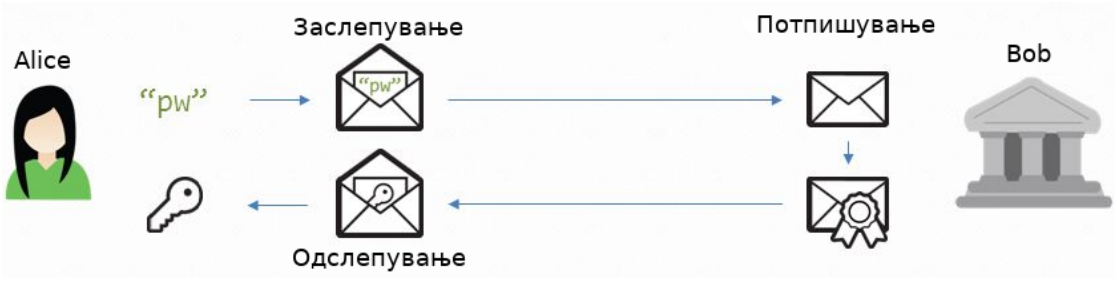
Глава 5

ID-базирани слепи потписи

5.1 Вовед

Слепите потписи претставуваат интерактивна шема за потпишување помеѓу корисник и потпишувач. Оваа интерактивна шема обично се користи за постигнување на анонимност при користењето на дигиталните потписи во протоколите. Прв пат е претставена во 1982 година од David Chaum [29] при што слепите потписи се користат во сценарио за анонимизирање при користење на електронски пари, а подоцна и за анонимизирање на гласачот во електронско гласање.

Слепите потписи можат многу лесно да се објаснат со користење на обични коверти со индиго хартија (како на слика 5.1). Во ова сценарио, Alice ја става пораката на која сака да добие слеп потпис во коверт и го праќа на потпишувачот Bob. Кога Bob ќе провери дека Alice има потреба од слеп потпис, удира печат на ковертот без да го отвори и го враќа на Alice. Сега



Слика 5.1: Илустрација на добивање на слеп потпис.

Alice го отвара ковертот и ја вади оригиналната порака врз која со помош на индиго хартијата останал печатот на Bob. Сега Alice може да на било кој да покаже дека има потпис од Bob, без Bob да има било какви информации за пораката која ја потпишал.

Формално кажано, доколку сакаме да добиеме слеп потпис на пораката M, ја претвараме порака во слепа порака M_b и од потпишувачот добиваме слеп потпис S_b. Потпишувачот не може да открие никакво знаење за M гледајќи ја M_b и секој може да провери дека S_b е потпис од потпишувачот за пораката M.

Постојат повеќе типови на слепи потписи, како фер потписи (овозможуваат повлекување на слепоста во случај на спор) или парцијално слепи потписи (корисникот и потпишувачот се договараат на некоја взаемно позната информација од пораката M). Според Horster [60] слепите потписи се поделени на четири категории: скриени, слаби, интерактивно слепи и јако слепи. Оваа категоризација ги дели пораките како (а) документ и (б) параметри на потписот, и четирите категории на слепи потписи го одредуваат знаењето на потпишувачот за релацијата на (а) со (б).

Слепите потписи се искористуваат во многу криптографски протоколи. Електронското гласање ги користи за да се добијат авторизирани гласачки ливчиња без да се наруши анонимноста на гласот. Во овој протокол гласачот бара слеп потпис на пополнетото и криптирано гласачко ливче. Потпишувачот слепо го потпишува криптираното гласачко ливче без да ја дознае содржината. Потоа, гласачот го екстрахира потписот на потпишувачот и го праќа потпишаното гласачко ливче анонимно во гласачката кутија. Слепите потписи се искористени во многу системи за електронско гласање [45, 79, 70, 71].

ID-базирани слепи потписи се слепи потписи кои се основани на ID-базирана криптографија (види глава 4). Во овој случај, потписот се проверува врз основа на идентитетот на потпишувачот, а не врз основа на некоја друга информација која претставува јавен клуч на потпишувачот. На овој начин корисникот кој сака слеп потпис (и останатите кои ќе прават проверки) не мора да пристапува до PKI инфраструктура за да го добие јавниот клуч на потпишувачот. За проверка на потписот доволно е да се знае идентитетот на потпишувачот - нешто што се смета за познато и пред почетокот на фазата на

барање слеп потпис. Бидејќи ID-базираната криптографија ги има овие погодности, особено погодноста на поефтино постоење на PKI инфраструктура, постојат повеќе системи за електронско гласање кои користат ID-базирани слепи потписи [47, 123, 72].

Бидејќи слепите потписи во протокол за електронско гласање се користат при креирање на секое гласачко ливче, од голема важност е оваа операција да не стане тесно грло во протоколот. Оваа глава разгледува ID-базирани слепи потписи и ги споредува аритметичките операции за нивно извршување. Разгледуваме седум трудови кои предлагаат ID-базиран протокол за слепи потписи [122, 39, 62, 48, 69, 93, 49] објавени во последните десет години. Оваа глава ќе ги анализира предложените протоколи и ќе ги стави во заедничка нотација полесна за споредба. Потоа, протоколите се анализираат со специфична околина за имплементација. На тој начин се добива слика за цената на протоколите во однос на потребното време и потребното количество податоци кои се пренесуваат за извршување на протоколите.

5.2 Дефиниција

Постојат повеќе дефиниции на протоколите за слепи потписи. Овде ќе ја користиме дефиницијата 5.1 за формално дефинирање на протоколите кои ги анализираме во оваа глава.

Дефиниција 5.1. Шема за ID-базиран слеп потпис се состои од четири алгоритми: *поставување*, *екстракција*, *издавање* и *верификација*. Алгоритмот *поставување* се извршува од тело наречено PKG (public key generator) кое генерира општи параметри и приватен клуч. Општите параметри се објавуваат јавно, а приватниот клуч се чува тајно во рамки на PKG. Алгоритмот *екстракција* има како влез општи параметри и приватен клуч и генерира приватен клуч D_{id} за идентитет ID. Алгоритмот *издавање* слепо потпишува порака и издава слеп потпис. Алгоритмот *верификација* има како влез потпис, порака, ID и општите параметри, а како излез дава 1 доколку потписот е верификуван и 0 доколку потписот не е верификуван.

Наведените протоколи користат неколку сигурносни претпоставки во своите алгоритми на кои ја базираат сигурноста на шемата:

- **Decisional Diffie-Hellman (DDH):** за дадени P, aP, bP, cP испечати yes доколку $c = ab$, или испечати no инаку.
- **Computational Diffie-Hellman (CDH):** за дадени P, aP, bP испечати abP.
- **Gap Diffie-Hellman group (GDH):** Групата G е GDH група доколку постои ефикасен алгоритам со полиномно време кој го решава DDH проблемот, и не постои алгоритам со полиномно време кој го решава CDH проблемот.
- **Bilinear Diffie-Hellman (BDH):** за дадени P, aP, bP, cP испечати $e(P, P)^{abc}$.
- **Decisional Bilinear Diffie-Hellman (DBDH):** за дадени P, aP, bP, cP, r, a, b, c испечати yes доколку $r = e(P, P)^{abc}$, или испечати no инаку.
- **ROS problem:** За дадена оракл случајна функција $F : Z_q^l \to Z_q$, најди ги коефициентите $a_{k,i} \in Z_q$ и решлив систем од $l + 1$ различни равенки по непознатите $c1, c2, ..., c_l$ над Z_q: $a_{k,1}c_1 + ... + a_{k,l}c_l = F(a_{k,1}, ..., a_{k,l})$, за $k = 1, 2, ..., t$.
- **Bilinear Diffie-Hellman Inversion (BDHI):** за дадени aP, bP, cP пресметај S, T така што $e(S, T) = e(P, P)^{abc}$.

5.3 Шеми за слепи потписи

Во оваа подглава ги разгледуваме предложените шеми за слепи потписи [122, 39, 62, 48, 69, 93, 49]. Секоја од овие шеми е анализирана и објаснета во сопствената секција.

5.3.1 Шема ZK1

Оваа шема [122] е првата ID-Базирана шема за слепи потписи од билинеарни парови. Шемата е претставена преку алгоритмите за поставување, екстракција, издавање и верификација:

[Поставување]: Нека P е генератор на GDH групата G_1. PKG избира случајно $s \in Z_q^*$ и го поставува $P_{pub} = sP$. Се дефинираат две хеш функции

преку функциите $H_1 : \{0,1\}^* \to Z_q$ и $H_2 : \{0,1\}^* \to G_1$. Општите параметри кои се објавуваат се $\{G_1, G_2, q, P, P_{pub}, H_1, H_2\}$, додека s е приватниот клуч на PKG.

[Екстракција]: За даден идентитет ID, PKG пресметува јавен клуч $Q_{ID} = H_2(ID)$ и го враќа приватниот клуч $S_{ID} = sQ_{ID}$.

[Издавање]: Овој алгоритам е претставен како интерактивен протокол помеѓу потпишувачот и корисникот:

- **Заложување:** Потпишувачот случајно избира $r \in Z_q^*$, пресметува $R = rP$ и го праќа R на корисникот.
- **Заслепување:** Корисникот случајно избира $a, b \in Z_q^*$, пресметува $c = H_1(m, e(bQ_{ID} + R + aP, P_{pub})) + b$ и го испраќа c на потпишувачот.
- **Потпишување:** Потпишувачот пресметува $S = cS_{ID} + rP_{pub}$ и го испраќа S на корисникот.
- **Одслепување:** Корисникот пресметува $S^{'} = S + aP_{pub}$ и $c^{'} = c - b$ и печати $\{m, S^{'}, c^{'}\}$.

[Верификација]: Се верифицира дека $c^{'} \stackrel{?}{=} H_1(m, e(S^{'}, P)e(Q_{ID}, P_{pub})^{-c^{'}})$.

Авторите на трудот [122] имаат доказ дека шемата е слепа и дека сигурноста на генеричкиот паралелен напад зависи од ROS проблемот. Но, Wagner во [117] тврди дека ROS проблемот може да биде пробиен во субекспоненцијално време, па за да се биде резистентен на нападот, q мора да биде барем 1600 бита. Овој факт ја прави шемата неефикасна во пракса.

5.3.2 Шема ZK2

Авторите во [122] предлагаат нова шема за слепи потписи која не се потпира на ROS проблемот [39]. Нивна шема е дадена преку следните алгоритми:

[Поставување]: Нека P е генератор на GDH групата G_1. PKG избира случајно $s \in Z_q^*$ и го поставува $P_{pub} = sP$. Се дефинираат две хеш функции преку функциите $H_1 : \{0,1\}^* \to Z_q$ и $H_2 : \{0,1\}^* \to G_1$. Општите параметри кои се објавуваат се $\{G_1, G_2, q, P, P_{pub}, H_1, H_2\}$, додека s е приватниот клуч на PKG.

[Екстракција]: За даден идентите ID, PKG пресметува јавен клуч $Q_{ID} = H_2(ID)$ и го враќа приватниот клуч $S_{ID} = sQ_{ID}$.

[Издавање]: Овој алгоритам е претставен како интерактивен протокол помеѓу потпишувачот и корисникот:

- **Заложување:** Потпишувачот случајно избира $r \in Z_q^*$, пресметува $U = rQ_{ID}$ и го праќа U на корисникот.
- **Заслепување:** Корисникот случајно избира $\alpha, \beta \in Z_q^*$, ги пресметува $U'\alpha U + \alpha\beta Q_{ID}$ и $h = \alpha^{-1}H_1(m, U') + \beta$, и го испраќа h на потпишувачот.
- **Потпишување:** Потпишувачот го пресметува $V = (r + h)S_{ID}$ и го праќа V на корисникот.
- **Одслепување:** Корисникот го пресметува $V' = \alpha V$ и печати $\{m, U', V'\}$.

[Верификација]: Се верифицира дека $e(V', P) \stackrel{?}{=} e(U' + H_1(m, U')Q_{ID}, P_{pub})$.

Оваа шема има доказ дека шемата е слепа, но нема формален доказ против "one-more signature"нападот.

5.3.3 Шема HCW

Авторите во [62] го напаѓаат тврдењето за неможност на фалсификување под "generic parallel"напади на шемата ZK2 [39] и даваат нова ефикасна шема за слепи потписи која се потпира на ROM моделот:

[Поставување]: Нека P е генератор на GDH група G_1. PKG случајно избира $r \in Z_q^*$ и го поставува $P_{pub} = sP$. Потоа, се дефинираат две хеш функции $H_1 : \{0,1\}^* \times G_2 \to Z_q^*$ и $H_2 : \{0,1\}^* \to G_1$. Општите параметри кои се објавуваат се $\{G_1, G_2, e, q, P, P_{pub}, H_1, H_2\}$, додека s е приватниот клуч на PKG.

[Екстракција]: За даден идентите ID, PKG го пресметува $P_{ID} = H_2(ID)$ и го враќа приватниот клуч $S_{ID} = sP_{ID}$.

[Издавање]: Овој алгоритам е претставен како интерактивен протокол помеѓу потпишувачот и корисникот:

- **Заложување:** Потпишувачот случајно избириа $r \in Z_q^*$, го пресметува $R' = e(P_{ID}, P_{pub})^r$ и го праќа R' на корисникот.

- **Заслепување:** Корисникот случајно избира $t_1, t_2 \in Z_q^*$, ги пресметува $R = R'^{t_1} e(P_1, P)^{t_2}$, $h = H_1(m, R)$, $h' = ht_1$ и го испраќа h' на потпишувачот.

- **Потпишување:** Потпишувачот го пресметува $V' = (rh' + 1)S_{ID}$ и го испраќа V' на корисникот.

- **Одслепување:** Корисникот проверува дали важи еднаквоста $e(V', P) \stackrel{?}{=} R^{th'} e(P_{ID}, P_{pub})$, го пресметува $V = V' + ht_2 P_1$ и ги печати $\{m, R, V\}$.

[Верификација]: Се верифицира дека $e(V, P) \stackrel{?}{=} R^{H_1(m,R)} e(P_{ID}, P_{pub})$.

Авторите на оваа шема ја докажуваат неможноста за фалсификување според ROM моделот, но не даваат формален доказа за "one-more signature" нападот, исто како и [39].

5.3.4 Шема GWWF1

Оваа шема е првата шема за слепи потписи која предлага оптимален број на рунди во интерактивниот протокол за издавање на потписот [48]. GWWF1 има протокол од една рунда комуникација помеѓу корисникот и потпишувачот при издавањето на потписот. Шемата е прикажана преку следните алгоритми:

[Поставување]: Нека P е генератор на GDH група G_1. PKG го избира случајно $r \in Z_q^*$ и го поставува $P_{pub} = sP$. Потоа се дефинираат две хеш функции преку $H_1, H_2 : \{0,1\}^* \to G_1$. Општите параметри се кои се објавуваат од поставувањето се $\{G_1, G_2, e, q, P, P_{pub}, H_1, H_2\}$, додека s е приватниот клуч на PKG.

[Екстракција]: За даден идентитет ID, PKG го пресметува $Q_{ID} = H_1(ID)$ и го враќа приватниот клуч $D_{ID} = sQ_{ID}$.

[Издавање]: Овој алгоритам е претставен како интерактивен протокол помеѓу потпишувачот и корисникот:

- **Заслепување:** Корисникот случајно избира $r_1 \in Z_q^*$; го пресметува $P'_m = r_1 H_2(m)$ и го испраќа P'_m на потпишувачот.

- **Потпишување:** Потпишувачот случајно избира $x_{ID} \in Z_q^*$, ги пресметува $A' = x_{ID} P'_m$, $B' = x_{ID}^{-1} D_{ID}$, $C' = x_{ID} P$, и ги испраќа A', B' и C' на корисникот.

- **Одслепување:** Корисникот верифицира дека $e(A', P) \stackrel{?}{=} e(P'_m, C')$ и дека $e(Q_{ID}, P_{pub}) \stackrel{?}{=} e(B', C')$. Потоа случајно избира $r_2 \in Z_q^*$, ги пресметува $A = r_2 r_1^{-1} A'$, $B = r_2^{-1} B'$ и $C = r_2 C'$ и ги печати $\{m, A, B, C\}$.

[Верификација]: Се пресметува $P_m = H_2(m)$ и се верифицира дека важи $e(A, P) \stackrel{?}{=} e(P_m, C)$ и $e(Q_{ID}, Ppub) \stackrel{?}{=} e(B, C)$.

Авторите на оваа шема имаат доказ дека алгоритмот има својство на слепи потписи и имаат доказ за сигурност од "one-more signature"напад од паралелни избрани пораки и избрани ID напади. Овие докази се базирани на "one-more BDHI"претпоставката која е предложена од авторите (слична на "one-mora RSA-inversion"во [11] која ја докажува шемата на Chaum [29]).

5.3.5 Шема KKS

Шемата KKS [69] е всушност генерализација на модифицирани ElGamal потписи кои се проширени да бидат слепи потписи преку шемата [26]. Horster покажува дека користејќи ја генералната равенка за потписи $A \equiv \alpha B + kC \pmod{q}$ може да се добијат многу варијанти на шема за слепи потписи. KKS шемата има 18 варијанти кои се добиени со пермутација на генералната равенка за потписи. Овде претставуваме една варијанта претставена преку следните алгоритми:

[Поставување]: Нека P е генератор на GDH група G_1. PKG случајно избира $r \in Z_q^*$ и го поставува $P_{pub} = sP$. Потоа, се дефинираат две хеш функции $H_1 : \{0,1\}^* \times G_2 \to Z_q^*$ и $H_2 : \{0,1\}^* \to G_1$. Општите параметри кои се објавуваат се $\{G_1, G_2, e, q, P, P_{pub}, H_1, H_2\}$, додека s е приватниот клуч на PKG.

[Екстракција]: За даден идентитет ID, PKG го пресметува $P_{ID} = H_2(ID)$ и го враќа приватниот клуч $S_{ID} = sP_{ID}$.

[Издавање]: Овој алгоритам е претставен како интерактивен протокол помеѓу потпишувачот и корисникот:

- **Заложување:** Потпишувачот случајно избира $k \in Z_q^*$, го пресметува $t = e(P, P)^k$ и го испраќа t на корисникот.

- **Заслепување:** Корисникот случајно избира $a, b \in Z_q^*$, ги пресметува $t = t^a e(P, P)^b$, $r = H_1(m, t)$, $r = a^{-1} r$ и го испраќа r на корисникот.

- **Потпишување:** Потпишувачот го пресметува $U = rS_{ID} + kP$ и го испраќа U на корисникот.

- **Одслепување:** Корисникот верифица дека $e(U,P)e(Q_{ID},P_{pub})^{-r} \stackrel{?}{=} r$, го пресметува $U = aU + bP$ и ги печати $\{m, r, U\}$.

[Верификација]: Верифицирај дека $e(U,P)e(Q_{ID},P_{pub})^{-r} \stackrel{?}{=} r^m$.

Авторите даваат доказ за својство на слепи потписи, но не даваат формален доказ против "one-more signature"нападот.

5.3.6 Шема RARG

Шемата RARG [93] во конструкцијата е многу слична на шемите [62] и [69]. Всушност сите овие шеми се базирани на Hess ID-Based шемата за потпис [57].

[Поставување]: Нека P е генератор на GDH група G_1. PKG случајно избира $r \in Z_q^*$ и го поставува $P_{pub} = sP$. Потоа, се дефинираат две хеш функции $H_1 : \{0,1\}^* \times G_2 \to Z_q^*$ и $H_2 : \{0,1\}^* \to G_1$. Општите параметри кои се објавуваат се $\{G_1, G_2, e, q, P, P_{pub}, H_1, H_2\}$, додека s е приватниот клуч на PKG.

[Екстракција]: За даден идентите ID, PKG го пресметува $P_{ID} = H_2(ID)$ и го враќа приватниот клуч $S_{ID} = sP_{ID}$.

[Издавање]: Овој алгоритам е претставен како интерактивен протокол помеѓу потпишувачот и корисникот:

- **Заложување:** Потпишувачот случајно избира $k \in Z_q^*$, го пресметува $R = e(P,P)^k$ и го испраќа R на корисникот.

- **Заслепување:** Корисникот случајно ги избира $a, b \in Z_q^*$, ги пресметува $R' = e(bQ_{IDs} + aP, P_{pub})R$, $V = H_1(m, R') + b$ и го испраќа V на потпишувачот.

- **Потпишување:** Потпишувачот го пресметува $S = Vd_{IDs} + kP$ и го испраќа S на корисникот.

- **Одслепување:** Корисникот ги пресметува $S' = S + aP_{pub}$, $V' = V - b$ и ги печати $\{m, S', V'\}$.

[Верификација]: Верифицирај дека

$$V' \stackrel{?}{=} H_1(m, e(S', P)e(Q_{IDs}, P_{pub})^{-V'}).$$

Авторите го докажуваат својството за слепи потписи и за неможност на фалсификување според Hess ID-Based потписите [57].

5.3.7 Шема GWWF2

Оваа шема [49] е подобрена верзија на GWWF1 шемата [48]. Во шемата GWWF2 авторите даваат две верзии на шемата GWWF1 - IBBS-1 и IBBS-2. Двете шеми се многу слични со GWWF1 шемата. Нема да одиме во детали со овие шеми, меѓутоа ќе ги вклучиме во резултатите од оваа глава.

5.4 Квантитативна споредба на шемите за слепи потписи

Во оваа глава ќе покажеме квантитативни споредби на 7те шеми за слепи потписи кои ги разгледуваме [122, 39, 62, 48, 69, 93, 49]. Квантитативните споредби ќе бидат на ниво на број на аритметички операции и на ниво на потребното количество на информација која се разменува меѓу играчите. Од првата споредба, знаејќи ја елиптичната крива што се користи, се добива директно време на извршување кај секој играч во протоколот. Од втората споредба се добива информација за бројот и големината на пораките. Повторно со избирање на конкретна елиптична крива се добиваат точно бројот на пренесени бајти помеѓу играчите за извршување на слепиот потпис.

5.4.1 Споредба на бројот на аритметички операции

Првата споредба е на бројот на аритметички операции кои треба да ги направи секој играч во протоколот за слепи потписи. Издвоени се следните аритметички операции во основното поле (операции над скалари) и во полињата на елиптичните криви (операции над елементи на G_1, G_2 и G_3):

- P - пар над елиптична крива;

Табела 5.1: Пресметковна цена на шемите: P - пар; M - множење на скалар со елемент од G_1; A - собирање на два елементи од G_1; H - хеш функција $H : \{0,1\}^* \to G_1$; M_s - множење на два скалари; A_s - собирање на два скалари; I_s - инверзија на скалар; C_s - споредба на два скалари; H_s - хеш функција $H_s : \{0,1\}^* \times G_2 \to Z_q^*$; E_p - степенување на пар-елемент; M_p - множење на два пар-елементи; C_p - споредба на два пар-елементи.

шема		P	M	A	H	M_s	A_s	I_s	C_s	H_s	E_p	M_p	C_p
ZK1	S		3	1									
	U	1	3	3			1			1			
	V	2						1		1	1	1	
ZK2	S		2				1						
	U		3	1		2	1	1		1			
	fV	2	1	1						1			1
HCW	S	1	1				1				1		
	U	3	1	1	1	2					2	2	
	V	2			1						1	1	1
GWWF1	S		3					1					
	U	4	4		1	1		2					2
	V	4			1								2
KKS	S	1	2	1							1		
	U					1		2		1	3	2	1
	V	2						1		1	1		
RARG	S	1	2	1							1		
	U	1	4	2			2			1			
	V	2						1	1	1	1	1	
IBBS-1	S		3					1					
	U	4	4		1	1		2					2
	V	4			1								2
IBBS-2	S		3	1									
	U	4	5	4	1			1					2
	V	4		1	1								2

- M - множење на скалар и елемент од G_1;
- A - собирање на два елементи од G_1;
- H - хеш функција $H : \{0,1\}^* \rightarrow G_1$;
- M_s - множење на два скалари;
- A_s - собирање на два скалари;
- I_s - инверзија на скалар;
- C_s - споредба на два скалари;
- H_s - хеш функција $H_s : \{0,1\}^* \times G_2 \rightarrow Z_q^*$;
- E_p - степенување на пар-елемент;
- M_p - множење на два пар-елементи;
- C_p - споредба на два пар-елементи.

Споредбата на шемите за слепи потписи на ниво на број на аритметички операции (подглава 5.1) е добар параметар кога одлучуваме која шема да ја користиме. Кога имплементираме некој протокол во кој треба да се вклучат слепи потписи, обично веќе имаме одлучено кои елиптични криви ќе ги користиме. Кога веќе знаеме која елиптична крива се користи, тогаш го знаеме и времето на извршување на секоја аритметичка операција. Поради природата на аритметичките операции над различните елиптични криви, се случува некаде операциите над скаларите да се бавни за сметка на побрзите пресметувања на пар над елиптичната крива. На други елиптични криви пак, операциите над скаларите или множењето на скалар со елемент од групата на елиптичната крива е побрзо за сметка на многу бавното пресметување на пар над елиптичната крива. Затоа, споредбата на шемите во која се знае точниот број на аритметички операции дава добар споредбен поглед при бирањето на шема за имплементирање. Нашата споредба е поделена за трите играчи во протоколот за слепи потписи (U-корисник, S-потпишувач, V-верификатор). Тоа значи дека кога ќе го знаеме сценариото за кое треба да се избере шема за слеп потпис, ќе знаеме кој играч е тесно грло и соодветно ќе видиме во

табелата тој играч да има што е можно помалку тешки аритметички операции. Од друга страна, доколку некој играч многу ретко ќе се вклучува во сценариото, може да избереме шема во која тој играч ќе го носи главниот товар на тешките аритметички операции.

5.4.2 Споредба на пропусност

Втората споредба е за потребите за пропусност на разгледаните шеми за слепи потписи. Во оваа споредба се земаат во предвид податоците кои се разменуваат помеѓу играчите на протоколот во различните фази. Фазите се:

- C: S до U - Фаза на заложување, се праќаат податоци од потпишувач до корисник;
- B: U до S - Фаза на заслепување, се праќаат податоци од корисник до потпишувач;
- S: S до U - Фаза на потпшување, се праќаат податоци од потпишувач до корисник;
- P: U до BB - Фаза на објавување на потписот, се праќаат податоци од корисник до огласна табла.

Во табелата 5.2 е дадена споредба на информациите што се праќаат помеѓу играчите за разгледаните шеми. Во шемите се праќаат елементи од основното поле (скалар), елементи од G_1 и од G_3. Во табелата 5.3 е избрана конкретна крива според Barreto-Naehrig фамилијата на криви и точно е пресметан бројот на бајти што се праќа во секоја фаза на протоколот при извршувањето на шемата за слепи потписи. Двете табели даваат многу добри показатели кога треба да се бира шема за имплементација на слепи потписи. Во некои протоколи информацијата за пропусност е многу важна. Ова се случува кога за извршување на протоколот се користи канал за комуникација кој е со мал опсег и праќањето на информации преку тој канал може да е многу бавен или многу скап. Со директна споредба на шемите можеме точно да видиме која побарува најмала пропусност за извршување на протоколот. Со издвојување на фазите може точно да видиме колку пропусност е потребно во секоја насока на комуникација. Ова е потребно во специфични

Табела 5.2: Цена за пропусност на шемите: G_1 - елемент од G_1; G_p - пар-елемент; s - скалар; C - фаза на заложување; B - фаза на заслепување; S - фаза на потпишување; P - фаза на објавување на потпис.

шема	насока	G_1	G_p	s
ZK1	C: S до U	1		
	B: U до S			1
	S: S до U	1		
	P: U до BB	1		1
ZK2	C: S до U	1		
	B: U до S			1
	S: S до U			1
	P: U до BB	2		
HCW	C: S до U		1	
	B: U до S			1
	S: S до U	1		
	P: U до BB	1	1	
GWWF1	B: U до S	1		
	S: S до U	3		
	P: U до BB	3		
KKS	C: S до U		1	
	B: U до S			1
	S: S до U	1		
	P: U до BB	1		1
RARG	C: S до U		1	
	B: U до S			1
	S: S до U	1		
	P: U до BB	1		1
GWWF1,IBBS-1	B: U до S	1		
	S: S до U	3		
	P: U до BB	3		
GWWF1,IBBS-2	B: U до S	1		
	S: S до U	3		
	P: U до BB	3		

Табела 5.3: Цена за пропусност на шемите: дадени потреби од пропусност во секоја фаза и вкупна пропусност за потпишувачкиот протокол. Резултатите се претставени во бајти.

шема	насока	цена:
ZK1	C: S до U	32
	B: U до S	32
	S: S до U	32
	P: U до BB	64
	вкупно:	160
ZK2	C: S до U	32
	B: U до S	32
	S: S до U	32
	P: U до BB	64
	вкупно:	160
HCW	C: S до U	384
	B: U до S	32
	S: S до U	32
	P: U до BB	416
	вкупно:	864
GWWF1	B: U до S	32
	S: S до U	96
	P: U до BB	96
	вкупно:	224
KKS	C: S до U	384
	B: U до S	32
	S: S до U	32
	P: U до BB	64
	вкупно:	512
RARG	C: S до U	384
	B: U до S	32
	S: S до U	32
	P: U до BB	64
	вкупно:	512
GWWF1,IBBS-1	B: U до S	32
	S: S до U	96
	P: U до BB	96
	вкупно:	224
GWWF1,IBBS-2	B: U до S	32
	S: S до U	96
	P: U до BB	96
	вкупно:	224

сценарија кога можеби брзината и можноста на комуникациските канали е различна во различни насоки на размена на податоци.

5.4.3 Заклучок

Шемите за слепи потписи претставуваат многу важен криптографски елемент во протоколите каде е потребна анонимноста. Во ова истражување целиме на користење на слепите потписи за креирање на систем за електронско гласање. Користењето на слепите потписи претставува еден од двата начини за постигнување на анонимност на гласачите при предавањето на гласот во гласачки кутии (вториот начин е користење на хомоморфни енкрипции). Од друга страна, користењето на ID-Based криптографија во предложениот систем за електронско гласање подразбира користење на слепи потписи во ID-Based околина. Истражувањето на трудовите за ID-Based слепи потписи во последните десет години покажа дека бирањето на шема за слепи потписи многу зависи од сценариото каде ќе се користат слепите потписи. Предложените шеми варираат во бројот на чекори во интерактивниот дел на извршувањето на протоколот, во типот на аритметички операции кои се извршуваат и во количеството информации кои се пренесуваат помеѓу учесниците во протоколот. За таа цел предложивме унифицирана нотација на предложените шеми за слепи потписи и направивме рамка за споредба на слепите потписи.

Унифицираната нотација му дозволува на истражувачот (или имплементаторот на протоколи) да ги дознае суштинските разлики помеѓу протоколите. Оваа нотација е претставена во секоја од подглавите за петте разгледувани шеми за слепи потписи: 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.5, 5.3.6 и 5.3.7.

Рамката за споредба се однесува на квантитативна анализа на шемите во однос на аритметичките операции кои се извршуваат и количеството на информации кои се пренесуваат. Резултатите од предложената рамка за споредба се презентирани во три табели. Табелата 5.1 ги сумира аритметичките операции кои ги прави секој учесник во извршувањето на протоколот. Доколку некој имплементатор ја знае фамилијата на елиптични криви и процесирачката моќ на учесниците во протоколот, лесно може да се одлучи за соодветна шема бидејќи ќе знае колку време е потребно за извршување на секоја од наведените аритметички операции. Табелата 5.2 ги дава бројот на

елементи кои се пренесуваат помеѓу учесниците. Овие елементи може да бидат од основното поле на кое се базираат елиптичните криви, или од некоја од групите на елиптичните криви. Повторно, доколку некој имплементатор има избрано фамилија на елиптични криви, веднаш може да добие претстава колку бајти ќе се пренесат во секоја фаза од извршувањето на протоколот. Пример за ова се дава во табелата 5.3 доколку е избрана Barreto-Naehrig фамилијата на елиптични криви во 128-битно сигурносно ниво. На овој начин едноставно може да направи споредба во однос на потребите за пропусност на истражуваните шеми за слепи потписи. Резултите од истражувањето на ова поглавје се објавени во трудот [95].

Глава 6

Електронско гласање

6.1 Опис

Гласањето како процес претставува корен на сите демократски процеси во светот. Од најстари времиња, па сè до денес, гласањето претставува начин на дознавање на прецизното мислење на група од луѓе кон множество од понудени опции. Овој процес се користи од обични гласања на состаноци, до гласања каде народот искажува мислење кон избрани луѓе како претставници на највисоки државни функции. Овие гласачки процеси може да бидат јавни, но поради специфичноста на сценаријата, понекогаш гласачките процеси мора да бидат анонимни. Ова значи дека секој има право да гласа, меѓутоа никој не треба да има начин да ја дознае врската гласач-глас.

Постојат уште многу други специфичности во гласачките процеси, од кои во оваа дисертација ќе се задржиме на сигурноста. Сигурноста во гласачкиот процес претставува особина која кажува дека процесот не може да се фалсификува, или на кој било друг начин злоупотреби за менување на резултатите во било која насока од вистинската. Токму оваа особина претставува слаба точка на многу системи за гласање - класични или електронски.

Електронско гласање претставува верзија на гласачки процес кој се одвива со помош на компјутерски системи. Оваа категоризација е многу широка и може да опфати: механички кутии за гласање, оптичко отчитување на гласачки ливчиња, гласање на компјутер во гласачка просторија, гласање на компјутер преку интернет и многу други варијации на користење на компјутер во гласачкиот процес. Историски гледано, електронското гласање се

користи од 1959 година, кога United Aircrafts користеле скенери за броење гласачки ливчиња, при што морало да се користи специјално мастило за пополнување на ливчето. Од тоа оптичко броење до денешните системи за гласање преку интернет се истражувале многу електронско гласање кои во секоја нова верзија носат подобрувања во полето на електронско гласање.

6.2 Видови на системи за електронско гласање

Како што наведовме во описот (подглава 6.1), постојат повеќе видови на системи за електронско гласање. Овде ќе ги наведеме најпопуларните системи кои сеуште имаат примена во светот.

6.2.1 Дупчени картички

Системот за електронско гласање кој користи дупчени картички се смета за најстар систем кој се користел во масовни гласачки процеси. Во овој систем, гласачот добива гласачко ливче на кое треба да направи дупка на позицијата која сака да ја избере. Дупката може да се направи со специјален уред (дупчалка) за да не се направи штета на гласачкото ливче. Потоа, постојат два начини за продолжување на процесот. Едниот начин е гласачкото ливче да се стави во гласачка кутија, а вториот начин е веднаш гласачот да го процесира гласачкото ливче на машина (табулатор) во која го става гласачкото ливче.

Првата машина од овој тип е Votomatic на компанијата IBM која датира од 1965 година [63]. Интересен е фактот што до 1996 година на претседателските избори во Америка слични уреди користеле околу 37% од гласачкото тело. Меѓутоа, овие уреди имале многу големи критики во претседателските избори во 2000 година во Флорида [76]. Истражувањата покажале дека уредите за броење на дупчени картички во Флорида направиле проблем кај 50000 гласачи бидејќи картичките биле лошо продупчени.

6.2.2 Оптичко скенирање

Овој начин на систем за електронско гласање исто така претпоставува класични гласачки ливчиња. Гласачот го пополнува гласачкото ливче со

обично пенкало, графитен молив, пенкало со специјално мастило или со електронско пенкало [85]. Потоа ливчето исто така може да се стави во гласачка кутија, или директно може да се процесира на оптичка машина која го отчитува гласачкото ливче. Овој тип на системи за електронско гласање дозволува и рачно пребројување доколку се користат гласачки кутии. Оптичките системи со електронско пенкало користат камера или сензор во пенкалото кое забележува која позиција била избрана, па затоа не е потребно дополнително користење на машина за скенирање на ливчето.

6.2.3 Директно електронско снимање

Директното електронско снимање (анг. Direct-Recording Electronic - DRE) претставува машина која дава избор на гласачот преку механички копчиња или екран на допир. Кога гласачот ја избира опцијата, машина го снима неговиот глас. Запишаниот глас може инстантно да се пренесе преку мрежа до некоја централна точка, или може да се сними во локална меморија на машина која подоцна се презема за броење. Овие машини почнуваат да се користат во Бразил во 1996 година, а потоа масовно се користат во САД и во други држави за потребите на електронското гласање. Меѓутоа, овие машини се многу критикувани поради големиот број на недостатоци најдени во нив.

6.2.4 Интернет гласање

Електронското гласање во овој случај го користи интернет како медиум за комуникација. Може да биде во контролирани услови, каде гласачот пристапува до уред кој е поврзан на интернет, меѓутоа во просторија со контролиран пристап. Може да биде и во неконтролирани услови, каде гласачот гласа преку негов уред поврзан на интернет.

Овој начин на гласање има најмногу предности во однос на корисниците на системот. Организаторите може лесно да одржат гласачки процеси користејќи многу малку ресурси, а гласачите имаат можност да гласаат комфорно од своите домови. Најголемата дилема при користење на овие системи е сигурноста. Потребно е многу повеќе убедување на гласачите да веруваат во систем кој е поставен на интернет, отколку да веруваат на класичен гласачки систем. Криптографијата која се користи во овие видови на системи понеко-

гаш е многу сложена, тешка за објаснување, па гласачите немаат верба и не можат да проверат дека системот е сигурен и дека точно ги брои гласовите без фалсификување.

6.3 Карактеристики на системи за електронско гласање

Постојат многу различни системи за електронско гласање. Сите овие системи се борат да имаат што е можно повеќе позитивни карактеристики, избегнувајќи ги негативните. Овде ќе ги наведеме и објасниме накратко карактеристиките [78] [99] [33] [82] кои сакаат да ги имаат системите за електронско гласање.

6.3.1 Право на гласање

Карактеристиката за право на гласање (анг. eligibility) кажува дека системот ќе дозволи право на гласање само на оние гласачи кои се на гласачкиот список. Никој друг нема да има право на гласање, и не може да даде валидно гласачко ливче.

6.3.2 Фер спроведување

Фер спроведувањето во еден систем за електронско гласање (анг. fairness) претставува карактеристиката дека во ниту еден момент не може да се дознае резултат кој може да влијае на гласачите кои сеуште не гласале. Ова значи дека до времето на затворање на гласачкиот процес не смее да се вршат пресметувања на резултат од ниту еден учесник во гласачкиот систем.

6.3.3 Точност

Точноста на резултатот (анг. accuracy) претставува карактеристика на системот за електронско гласање кој вели дека резултатот мора да е точно избројан од сите дадени гласачки ливчиња. Ниту едно гласачко ливче не може да биде елиминирано од броењето, или на било кој начин сменето од

оригиналната форма која ја пополнил гласачот. Исто така, ниту едно гласачко ливче не може да се брои како невалидно, доколку е валидно пополнето и поднесено од гласачот.

6.3.4 Верификација

Можноста за проверка на гласачкиот процес се нарекува верификација. Формално, во гласачките системи постојат два начини на верификација: индивидуална (анг. individual verifiability) и универзална верификација (анг. universal verifiability). Првиот начин, индивидуалната верификација, претставува начин на верификување од гласачите во гласачкиот процес. Тоа значи дека гласачот може да провери дали неговиот глас е избројан, и уште повеќе, дека е избројан во точната опција која гласачот ја избрал. Вториот начин, универзалната верификација, претставува начин на верификување на резултатите од било кој (внатрешен учесник или надворешно лице). Универзалната верификација обично се прави од страна на мониторинг организации кои ги валидираат (верификуваат) резултатите од гласачкиот процес.

6.3.5 Приватност на гласот

Приватноста претставува најважен дел во еден систем за електронско гласање. Приватноста на гласот (анг. vote privacy) претставува основно право во анонимните гласачки процеси. Врската гласач-глас во никој случај не треба да биде откриена од системот за електронско гласање. Доколку некој учесник во гласачкиот процес е во можност да ја дознае оваа врска, тоа е напаѓање на основните принципи на приватноста и анонимноста на гласачите.

6.3.6 Без пишана трага

Карактеристиката без пишана трага (анг. receipt-freeness) означува дека системот за електронско гласање не дава никаков доказ на гласачот околу информацијата како гласал. Тоа значи дека подоцна гласачот не може да докаже никому дека гласал на одреден начин. На овој начин се избегнува леснотијата во убедување на гласачи како да гласаат преку употреба на сила или закани. Доколку гласачот не може да докаже некому како гласал, тогаш

опаѓа мотивацијата за убедување на гласачи преку употреба на сила или закани.

6.3.7 Надежност

Надежноста (анг. reliability) на еден систем за електронско гласање мора да биде на многу високо ниво. Под надежност на систем се мисли на робусност и непрекинатост во извршувањето на работата. Ова значи дека во ниту еден случај системот не смее да ги изгуби веќе внесените гласови, да дозволи менување на гласовите или да почне со неконзистентно работење. Во случај на дефект на повисоко ниво (прекин не електрична струја или прекин на комуникации) системот мора да дозволи продолжување со работата по отстранување на настанатите дефекти.

6.3.8 Мобилност

Системот за електронско гласање не смее да постави ограничувања за локацијата на гласање доколку гласачкиот процес не поставува локациски рестрикции. Ова значи дека доколку процесот дозволува гласање од било кое место, тогаш системот треба да дозволи таква можност за гласачите. Доколку процесот предвидува гласање со лично присуство, тогаш мобилноста се ограничува на точно специфицирани локации за гласање.

6.3.9 Цена на користење

Цената на користење на системот за електронско гласање треба да биде разумна според светските и локалните стандарди за електронски системи. Доколку цената на користење е ниска, тогаш може се повеќе да се користат системи за електронско гласање во гласачките процеси. Во масовните гласачки процеси, цената на користење на електронски систем станува одлучувачки фактор бидејќи скоро секогаш е поефтино од класичното хартиено гласање. Во малите гласачки процеси може цената на користење да биде пречка бидејќи во тој случај класичното хартиено гласање претставува поефтината варијанта. Секако, симнувањето на цената на користење на систем за електронско гласање претставува главен мотив за користење на тие системи.

6.4 Слична работа

Постојат неколку системи за мобилно електронско гласање во истражувачката литература. Во оваа подглава ги наведуваме предложените системи за мобилно гласање и ги анализираме можните недостатоци во нив.

Системот предложен во [89] имплементира идентични гласачки кутии со повеќекратна физичка администрација (анг. Identical Ballot Boxes with Physical Multiple Administration - IBB/PMA). Овој систем е базиран на J2ME платформата и прв пат е презентиран во [4]. Овој систем користи автентикациски сервер кој има улога да автентицира гласачи и да брои ливчиња. Во системот постои сервер за дистрибуција на гласот кој го прима гласот од гласачот, го копира и го праќа во идентичните гласачки кутии. Авторите на овој систем имплементираат таканаречен сигурен сервер кој стои помеѓу гласачот и останатите сервери во системот за да обезбеди анонимност на гласот. Комуникацијата помеѓу сите играчи во протоколот користи SSL за енкриптирање на сообраќајот. Главната идеја во IBB/PMA системот е во тоа што сите учесници во гласачкиот процес (идентификувани како PMA) би имале сопствени гласачки кутии (идентификувани како IBB). Поради ова, секој учесник би требало да го извршува протоколот чесно бидејќи серверот за дистрибуција на гласот праќа идентична копија од гласот во сите гласачки кутии. Доколку некој учесник е нечесен и сака да го промени примениот глас во сопствената гласачка кутија, тој учесник веднаш ќе биде фатен поради тоа што ќе има различен резултат од останатите учесници. Проблемот со овој систем е претпоставката дека серверот за дистрибуција чесно го копира гласачкото ливче добиено од гласачот и претпоставката дека сигурниот сервер нема да ја чува информацијата за врската гласач - глас. Доколку овие сервери се пробијат, тогаш својствата точност и приватност на гласот нема да бидат исполнети. Уште повеќе, овие сервери претставуваат клучни точки во системот и доколку барем еден престане да работи, тогаш целиот систем нема да може да продолжи нормално да работи.

Системот предложен во [38] имплементира мобилно гласање со помош на Android платформата. Предложениот протокол користи слепи потписи, RSA и AES како криптографски блокови во својата имплементација. Во наведените фази гласачот се најпрво се регистрира пред почетокот на гласачкиот

процес. Потоа гласачот се автентицира со кредениците добиени при регистрирање при што добива слеп потпис. Гласачот го пополнува гласачкото ливче и заедно со слепиот потпис ги испраќа во гласачката кутија. Кога гласачкиот процес завршува, сите примени гласачки ливчиња се декриптираат, слепите потписи се проверуваат и сите валидни гласачки ливчиња се пребројуваат. При имплементацијата, овој систем ја користи библиотеката BouncyCastle. Проблемот со овој систем лежи во приватноста на гласачкото ливче и во робусноста на системот. Имено, гласачката кутија при примање на гласачкото ливче може да ја зачува адресата на гласачот и подоцна при декриптирање да го спои гласачкото ливче со гласачот. Проблемот со робусноста не е воопшто земен во предвид од страна на авторите. Во предложениот систем сите играчи во протоколот се репрезентирани од само една инстанца. Доколку барем еден од овие играчи стане недостапен од било која причина, тогаш целиот систем нема да може да продолжи со работа.

Авторите во [84] исто така презентираат систем за мобилно гласање на Android платформата (овој систем е прво презентиран во [83]). Системот е имплементира дистрибуирано генерирање на клуч според Pedersen користејќи го ElGamal криптосистемот. Ова значи дека се одржува робусноста на системот со прагова вредност и со парцијални енкрипции на гласачкото ливче. Кога ќе се соберат доволен број на парцијални енкрипции за одредено гласачко ливче (да се стигне до праговата вредност), тогаш гласачкото ливче може да се декриптира. Проблемот со робусноста во овој систем се јавува во друг играч во протоколот, односно во веб огласната табла (анг. Web Bulletin Board - WBB). Доколку се нападне WBB и тој се отстрани од системот, тогаш целиот гласачки процес ќе стане нефункционален. Уште повеќе, овој систем предлага голема моќ во играчот WBB претпоставувајќи дека секогаш ќе се однесува чесно.

Третиот предложен систем во Android платформата е [75] кој го имплементира системот за електронско гласање предложен во [73]. Авторите користат елиптични криви, парови над елиптични криви, слепи потписи и BLS потписи. Овој систем е најблизок до нашиот предложен систем, како во искористените криптографски елементи, така и во извршувањето на протоколот. Гласачот се автентицира со автентикациски сервер, добивајќи празно гласачко ливче со слеп потпис. Потоа, гласачот го пополнува гласачкото ливче и го

испраќа заедно со слепиот потпис на гласачки сервер. Гласачкиот сервер го проверува слепиот потпис и доколку е валиден го зачувува за понатамошно процесирање. Кога завршува гласачкиот процес, гласачкиот сервер ги брои сите валидни гласачки ливчиња и ги објавува резултатите. Проблемот во ова сценарио е во тоа што гласачкиот сервер може да го дознае идентитетот на гласачот зачувувајќи ја неговата адреса при примањето на гласачкото ливче. Потоа, во фазата на броење, гласачкиот сервер може лесно да го поврзе гласачкото ливче со адресата на гласачот пробивајќи го својството за приватност на гласот. Исто така, предложениот систем не го исполнува својството за робусност на системот бидејќи и автентикацискиот сервер и гласачкиот сервер се претставени само со по една инстанца која е претставува клучна точка во системот.

6.5 Предложен модел на систем за електронско гласање

Предложениот систем за електронско гласање претставува комплетен систем кој е составен од повеќе модули. Системот користи ID-Based криптографија и DKG систем за чување на приватниот клуч. Сите модули се конструирани со прагова вредност со што се дозволува одржливост на системот во случај на паѓање на некои играчи во протоколот.

6.5.1 Играчи во протоколот

Во системот за електронско гласање постојат пет типови на играчи:

- DKG играч (DKG Player)
- Потпишувач (Issuer)
- Гласачка кутија (Voting box)
- Бројач (Tallier)
- Гласач (Voter)

Сите типови на играчи имаат своја улога во системот за електронско гласање. Не смее да има играч кој ќе има повеќе типови на улоги (на пример да биде и гласачка кутија и бројач). Ова својство е потребно бидејќи со здружување на знаењето помеѓу различните типови на играчи возможно е пробивање на сигурноста на гласот и на анонимноста на гласачот.

Во следните подглави се објаснува улогата на секој тип на играчи.

6.5.1.1 DKG играч

DKG играчите го имплементираат протоколот за PKI инфраструктура базирана на ID-Based криптографија (подглава 4.4). Ова претставува иницијалниот чекор кој мора да се изврши за да се постави и подготви системот за електронско гласање. Овде се дефинираат вкупниот број на играчи n и праговата вредност t. Според дефиницијата 4.2 ниту еден член нема моќ да генерира приватни клучеви за било кој идентитет - само t собрани членови од DKG играчите имаат моќ да генерираат приватен клуч за даден идентитет. Откако ќе заврши фазата на поставување на системот (подглава 4.4.3) тогаш и самите DKG играчи ќе го искористат тој систем за да добијат приватни клучеви за своите идентитети. Потоа оваа група на играчи дефинира дека DKG модулот е спремен за понатамошна работа. Понатаму, улогата на секој DKG играч претставува примање на барање за дел од приватен клуч, автентикација на барателот и генерирање на дел од приватниот клуч на барателот (подглава 4.4.4). Кога барателот ќе ги добие сите делови од приватниот клуч ќе може да го генерира сопствениот приватен клуч.

Во некои случаи DKG играчите може да бидат вклучени само во едно гласање и да постојат само за тоа гласање. Ова е пример за гласачки процеси со мал број на гласачи при што во секој нареден гласачки процес DKG системот повторно ќе се креира. Во гласачки процеси кои имаат голем број на гласачи и процеси кои се извршуваат на чести интервали подобро е DKG системот да биде креиран еднаш и да се користи постојано. Во овој случај сите членови кои веќе генерирале приватен клуч од постоечкиот DKG систем нема потреба повторно да ја прават истата операција. Во овој случај се заштедува на време за поставување на DKG систем и за генерирање и дистрибуција на приватните клучеви - се добива класичен перзистентен/постојан PKI систем. Негативната работа за користење на постојан DKG систем е потребната

административна работа за негово одржување.

6.5.1.2 Потпишувач

Типот на играчи Потпишувач претставува дел од модулот за слепо потпишување на гласачките ливчиња (деф. 5.1). Во предложениот систем за електронско гласање Потпишувачите ја имаат листата од гласачи и може да автентицираат гласач според негова идентификација. Овој модул во системот ја знае врската гласач-глас и затоа потпишувачите не смеат да ја дознаат содржината на гласачките ливчиња. Затоа се користи шема за слепи потписи, при што конкретен потпишувач по автентикацијата слепо го потпишува гласачкото ливче на гласачот. Гласачот го користи слепиот потпис на гласачкото ливче за подоцна анонимно да го прати својот глас, а примачот на тој глас да може да провери дека гласот е валиден.

Сите потпишувачи претставуваат доделени играчи во системот за електронско гласање. Овие играчи не смеат да имаат други улоги во гласачките процеси. Потпишувачите се иницијализираат веднаш по креирањето на DKG системот. Иницијализацијата претставува побарување на приватен клуч од DKG системот и генерирање на приватниот клуч по добивањето на сите делови. Доколку потпишувачите имаат функција само за еден гласачки процес, тогаш во нивното иницијализирање веднаш го добиваат списокот на валидни гласачи. Овие играчи по завршувањето на гласачкиот процес се исклучуваат од системот. Во гласачки процеси кои се повторуваат често и кои имаат голем број на гласачи може Потпишувачите да бидат активни цело време - исто како и DKG системот. Тогаш, пред секој гласачки процес потпишувачите ја добиваат конкретната листа од гласачи и продолжуваат со работа, односно нема потреба од трошење време на повторно иницијализирање.

6.5.1.3 Гласачка кутија

Гласачка кутија претставува тип на играч кој ќе ги добива гласачките ливчиња од гласачите. Бидејќи не користиме анонимни канали, овие играчи можат да го дознаат идентитетот на гласачот. Конкретно, гласачките кутии не смеат да имаат листи од гласачи, меѓутоа поаѓајќи од фактот дека може да се дознае мрежната адреса на гласачот, можно е гласачката кутија да открие идентитет на гласач. Поради тоа, гласачките ливчиња кои ги праќаат

гласачите се енкриптирани со приватен клуч кој го знаат само гласачите. На овој начин, гласачките кутии кои ги содржат ливчињата не можат да ја видат нивната содржина и го поврзат дадениот глас со гласачот.

Овој тип на играчи исто така мора да е од доделен тип - доколку конкретен играч е гласачка кутија, не смее да биде ниту еден друг тип на играч. Доколку се дозволи гласачка кутија да биде и улога бројач, можно е да се открие врската гласач-глас бидејќи типот гласачка кутија може да го дознае идентитетот на одреден гласач и неговиот енкриптиран глас, а типот бројач може да ја открие содржината на енкриптираниот глас.

Иницијализијата на гласачките кутии се прави веднаш по кревањето на DKG системот бидејќи го користат DKG системот за генерирање на приватните клучеви. Исто како и кај потпишувачите, може за секој гласачки процес да се креираат и иницијализираат нови потпишувачи, а може и да се користат еднаш иницијализирани гласачки кутии во повеќе гласачки процеси. Во вториот случај се заштедува на време за иницијализирање и добивање на приватните клучеви.

6.5.1.4 Бројач

Типот на играч бројач претставува играч кој учествува во системот за електронско гласање за броење на дадените гласови. Овие играчи ги добиваат приватните клучеви од гласачите за секој даден глас во гласачките кутии според верифицирана шема за делење на тајна (подглава 2.4.3). Меѓутоа, секој бројач добива само дел од приватниот клуч на секој даден глас - што значи дека не може веднаш да се здружи со гласачка кутија за веднаш да ги декриптира гласовите и да ги брои гласовите пред време. Кога ќе заврши гласачкиот процес, еден бројач (или повеќе) започнува со фаза на рекреирање на приватниот клуч на секој даден глас. За да го оствари ова, бројачот ги контактира другите бројачи да му ги пратат деловите за конкретниот гласач. Кога ќе добие доволен број на делови од приватниот клуч (според претходно дефинирана прагова вредност) тогаш бројачот го бара енкриптираниот глас од гласачките кутии. Гласачката кутија што го има конкретниот енкриптиран глас го праќа на бројачот. По ова бројачот го декриптира гласот со генерираниот приватен клуч на гласачот и го брои гласот. Кога ќе заврши целиот процес (со поминување на сите добиени делови од приватни клучеви)

бројачот ги објавува резултатите. Доколку се избере повеќе бројачи да го прават овој процес се споредуваат добиените резултати и се проверува дали тие се исти помеѓу различните бројачи.

Исто како и другите типови на играчи, и играчот кој е бројач не смее да биде дел од другите модули во системот. Бројачите се иницијализираат веднаш после DKG системот бидејќи е потребно да ги добијат своите приватни клучеви од DKG играчите. Исто така, може еднаш иницијализирани бројачи да се искористуваат за повеќе гласачки процеси - штедејќи време на иницијализација и добивање на приватни клучеви.

6.5.1.5 Гласач

Играчот гласач претставува корисник на системот за електронско гласање. Неговата улога е да пополни гласачко ливче и доколку има авторизација на гласачкиот процес, неговото ливче да биде точно избројано.

Гласачите прво го преземаат својот приватен клуч од DKG системот. Потоа, кога има валиден гласачки процес може да пристапат кон гласање. Прво е потребно да добијат слеп потпис од потпишувач. По успешното земање на слеп потпис на гласачкото ливче, гласачот го енкриптира гласачкото ливче и слепиот потпис. Потоа, го праќа енкриптираното гласачко ливче на гласачка кутија. Клучот со кој го криптирал гласачкото ливче го дели на сите бројачи според верифицирана шема за делење на тајна 2.4.3.

По завршувањето на гласачкиот процес гласачот може да провери дали неговиот глас е во листата на избројани гласови и индивидуално го потврди резултатот. Доколку гласачкиот систем одржува чести гласачки процеси со истите гласачки списоци, тогаш гласачот може да го чува приватниот клуч од DKG системот и да го искористува низ повеќето гласачки процеси заштедувајќи време за генерирање на клуч.

6.5.2 Шема за гласање

Шемата на предложениот систем за електронско гласање се извршува во повеќе чекори.

6.5.2.1 Креирање на конфигурација

Првиот чекор е креирање на конфигурација за конкретен гласачки процес. Конфигурацијата се состои од следните параметри:

- Име на гласачки процес;
- Опис на гласачкиот процес;
- Гласачко прашање;
- Вид на гласачко ливче:
  - Еден избор од повеќе;
  - Слободни неколку избори од повеќе;
  - Точно K избори од N;
  - Слободен текстуален избор.
- Број на избори на гласачко ливче;
- Вредност за K на гласачко ливче;
- Вредност n за PKI инфраструктурата;
- Вредност t за PKI инфраструктурата;
- DKG Играчи (име, уред, адреса);
- Потпишувачи (име, уред, адреса);
- Гласачки кутии (име, уред, адреса);
- Бројачи (име, уред, адреса).

Бидејќи предложениот електронски систем работи на мобилни уреди, би било комплицирано сите наведени параметри да се внесуваат на мобилните уреди. Затоа е имплементиран администраторски вебсајт за конфигурирање на уредите. По внесувањето на параметрите за гласачкиот процес, вебсајтот креира QR код кој се скенира од мобилните уреди. На овој начин може брзо и лесно да се конфигурира гласачки процес и да се пренесе конфигурацијата на мобилните уреди кои ќе го водат процесот.

6.5.2.2 PKI инфраструктура

Вториот чекор е подигнување на PKI базирано на ID-Based криптографија и DKG. Овој чекор е објаснет во подглавата 4.4. Во оваа фаза се идентификуваат DKG играчите кои учествуваат во креирањето на приватниот клуч на PKI инфраструктурата. Пред започнување на фазата се дефинираат точно бројот на DKG играчи n и праговата вредност на PKI инфраструктурата t. По завршувањето на оваа фаза во секое време доволно е да има t активни DKG играчи за да можат да се конструираат приватни клучеви за останатите играчи во шемата за гласање и нивните јавни идентификатори.

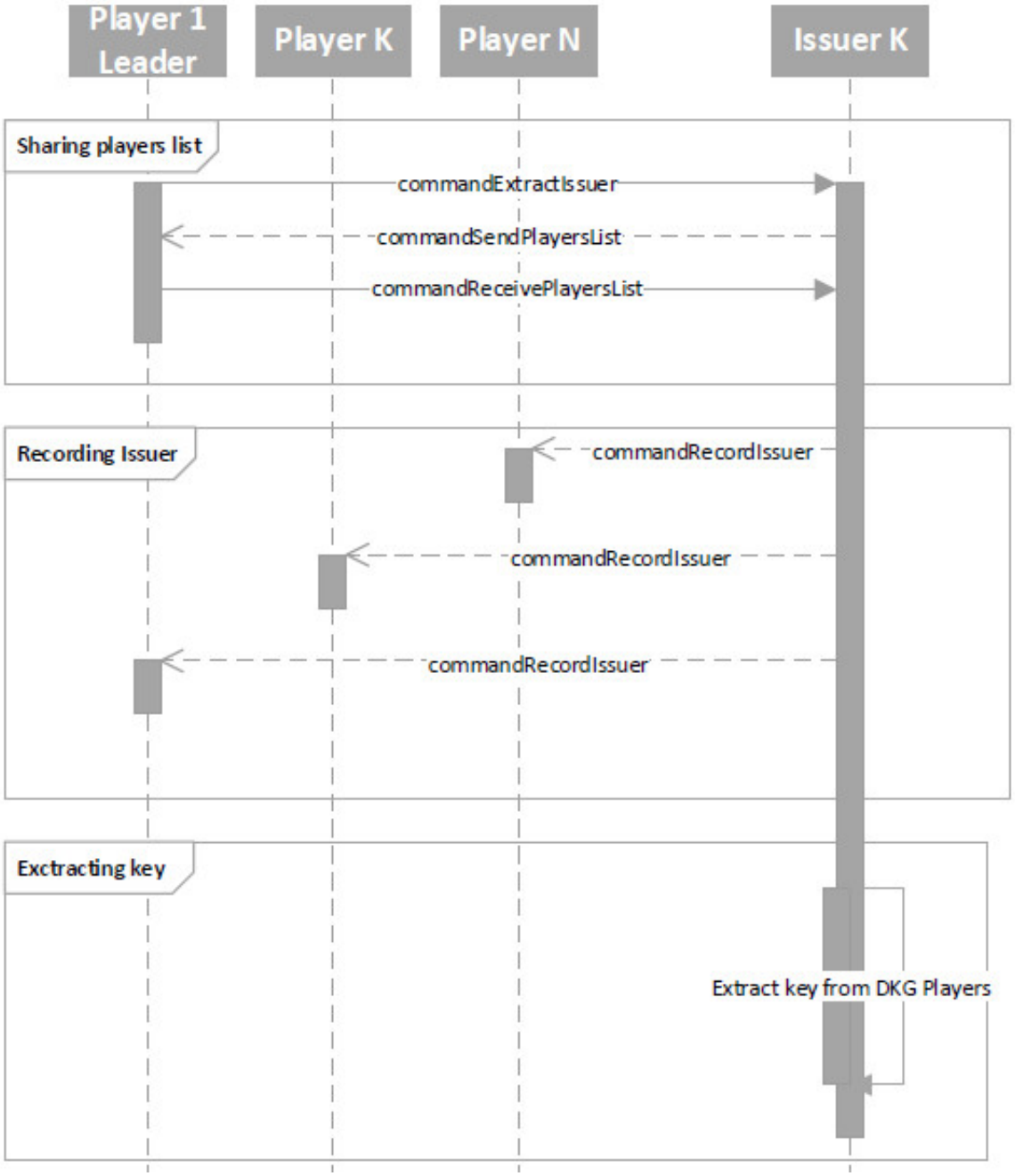
6.5.2.3 Иницирање на останатите играчи

Иницирањето на останатите играчи се состои од праќање на команда за доделување улога, преземање на листата од параметри и DKG Играчи и екстрахирање на приватниот клуч. На почетокот, на пример за потпишувачите, еден DKG играч праќа порака commandExtractIssuer до секој потпишувач од конфигурацијата на гласачкиот процес. Потоа потпишувачот му враќа порака commandSendPlayersList за да ја побара конфигурацијата на гласачкиот процес и листата на DKG играчи. По ова, DKG играчот му ги праќа бараните информации со порака commandReceivePlayersList. Сега потпишувачот може да се иницира со добиените информации и потоа праќа порака commandRecordIssuer на сите DKG играчи. Потоа потпишувачот влегува во процедура за екстракција на приватниот клуч според главата 4.4.4.

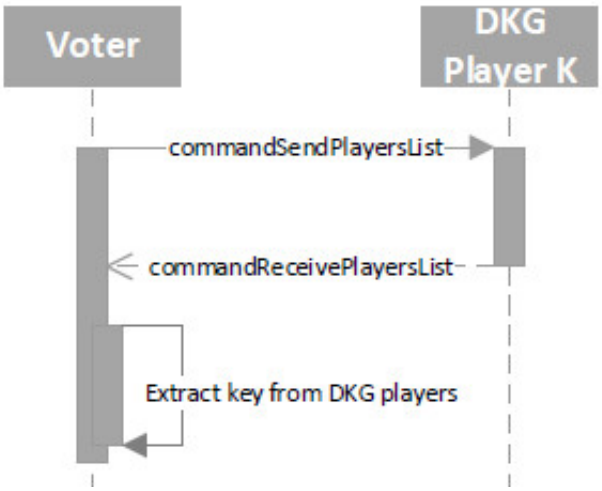
Наведениот процес е ист и за Гласачките кутии и за Бројачите. Процесот графички е прикажан преку секвенцниот дијаграм 6.1.

6.5.2.4 Гласање

Гласањето може да започне кога системот е конфигуриран, PKI инфраструктурата е наместена и сите останати играчи ги имаат екстрахирано своите приватни клучеви. На почетокот гласачот треба да знае барем еден DKG играч. Ова може да го дознае преку вебсајтот на кој ја има листата од DKG играчи. Откако ќе се поврзе барем на еден DKG играч може да започне со процедурата за иницијализација (дијаграм 6.2). Во оваа процедура гласачот праќа порака commandSendPlayersList до DKG играчот. На оваа порака DKG



Слика 6.1: Секвенцен дијаграм на фазата на иницијализација на останатите играчи (прикажано со потпишувач).



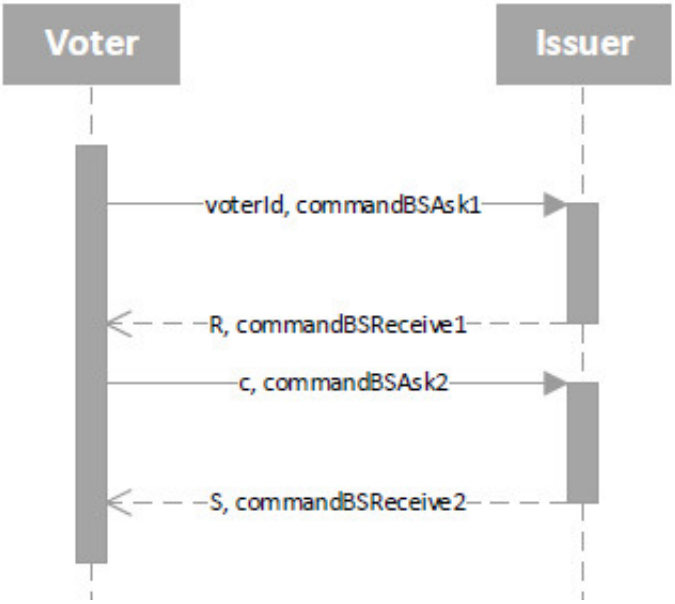
Слика 6.2: Секвенцен дијаграм на фазата на иницијализација на гласачите.

играчот му одговара со сите податоци за гласачкиот процес на гласачот со порака commandReceivePlayersList. По ова, гласачот ја започнува процедурата за екстракција на приватниот клуч според главата 4.4.4.

По иницијализацијата на гласачот тој може да започне со гласање. Гласањето се одвива во три чекори:

- земи слеп потпис од случајно избран Потпишувач;
- испрати енкриптирано гласачко ливче до случајно избрана Гласачка кутија;
- испрати делови од приватниот клуч за гласачкото ливче до Бројачите.

Земањето на слеп потпис се одвива по алгоритмот ZK1 (подглава 5.3.1). По овој протокол гласачот прво праќа наредба commandBSAsk1 до случајно избран потпишувач заедно со своја идентификација. Потпишувачот проверува дали идентификацијата постои во листата на дозволени гласачи. Доколку постои го извршува чекорот заложување од алгоритмот ZK1 и го праќа на гласачот со порака commandBSReceive1. Гласачот ја прима пораката и го извршува чекорот заслепување. Ги праќа податоците од овој чекор на потпишувачот со пораката commandBSAsk2. Потпишувачот го извршува чекорот потпишување и ја праќа пораката BSReceive2 на гласачот. Со примањето на оваа порака гласачот добива слеп потпис на гласачкото ливче, без потпишувачот да има било каква информација за содржината на гласачкото ливче.



Слика 6.3: Секвенцен дијаграм на добивање на слеп потпис.

Чекорот на земање на слеп потпис е прикажан графички во секвенцниот дијаграм 6.3.

По добивање на слеп потпис, гласачот треба да го испрати гласачкото ливче и слепиот потпис во гласачка кутија. За да се сочува анонимноста, гласачката кутија не смее да ја дознае содржината на гласачкото ливче. Затоа, гласачот генерира случаен таен клуч за AES енкрипција. Потоа, го енкриптира гласачкото ливче и слепиот потпис со тајниот клуч користејќи AES енкрипција. Добиениот енкриптиран текст го праќа на случајно избрана гласачка кутија со пораката commandReceiveEncryptedBallot. На овој начин, иако гласачката кутија знае кој го праќа ливчето (бидејќи не користиме анонимни канали), не може да го декриптира ливчето за да ја дознае содржината. Потоа, гласачот го дели приватниот клуч од гласачкото ливче според верифицираната шема за делење на тајна на Feldman (подглава 2.4.4) користејќи ги n и t од конфигурацијата на бројачите. Добиените делови од приватниот клуч ги праќа на бројачите со пораката commandReceiveVoterCsAndEval и со таен идентитет.

Цената на одвивање на протоколот за гласање е претставена преку бројот и големината на пратените пораки и преку аритметичките операции што ги извршуваат играчите во протоколот. Сумираните податоци за давањето на

глас се дадени во следните шеми.

1. Гласачот испраќа една порака кон случајно избрана гласачка кутија: - byte: 1 - byte array of encrypted vote: 1 **2. Гласачот испраќа порака кон секој бројач (вкупно n пораки):** - byte: 1 - byte array of hashed id: 1 - G1: k - S: 1

Шема 6.5.1: Број на пораки при давање на глас

1. Гласачот пресметува: - random scalar: k - serialization of BlindSignature object: 1 - AES encryption of serialized BlindSignature object: 1 - G1 scalar multiplication: k - scalar addition: $nk + n$ - scalar multiplication: $2nk$ - SHA 1 hash: 1 - G1 serialization: k - scalar serialization: 2

Шема 6.5.2: Број на операции во трета фаза: верификација

6.5.2.5 Броење на гласови

Кога ќе заврши фазата на гласање, потребно е да започне броењето на гласови. Броењето започнува кога некој DKG играч ќе прати порака commandStartTally кон случајно избран бројач. За споредба на бројачите (и зголемена веродостојност на резултатите) може да се прати порака и на сите бројачи да започнат процедура за броење. Кога одреден бројач ќе добие порака за

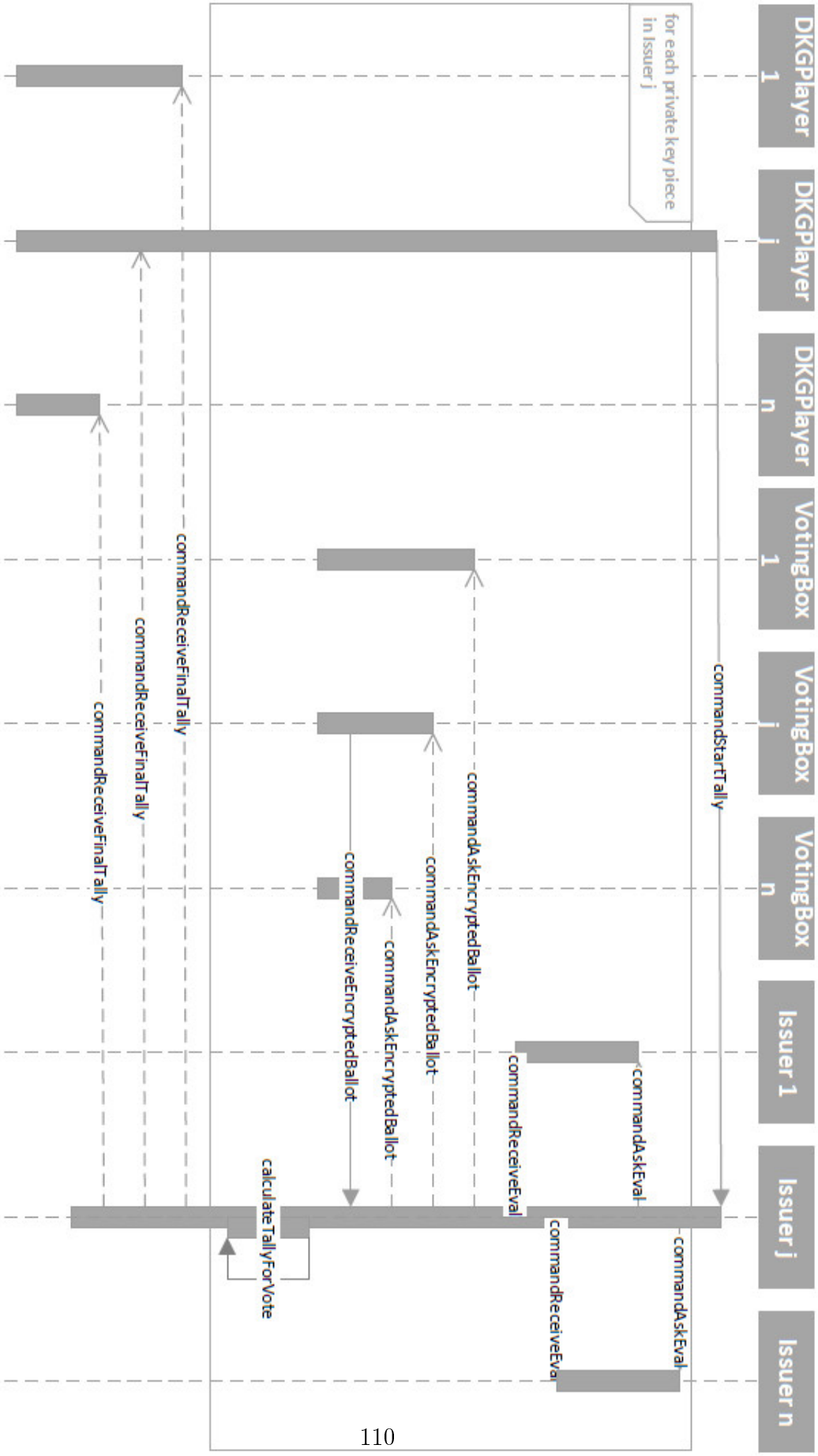
започнување на броење, почнува да итерира по сите примени делови од таен клуч на гласач. Кога ќе најде дел од таен клуч, праќа порака до сите останати бројачи commandAskEval со тајниот идентитет на гласачот. Откако ќе собере доволно делови од тајниот клуч почнува реконструкција на тајниот клуч (подглава 2.4.4). Доколку е се во ред и реконструкцијата е успешна, бројачот ги контактира сите гласачки кутии за да го добие енкриптираното гласачко ливче на гласачот со порака commandaAkEncryptedBallot. Гласачката кутија која го има енкриптираното гласачко ливче за гласачот со наведениот таен идентитет го праќа назад до бројачот со порака commandReceiveEncryptedBallot. По ова, бројачот го декриптира гласачкото ливче со реконструираниот приватен клуч. Декриптираниот глас го вбројува во резултатите од гласањето. Кога бројачот ќе ги измине сите делови од приватни клучеви броењето се смета за завршено. По завршувањето, бројачот го праќа финалниот резултат на сите DKG играчи со пораката commandReceiveFinalTally. Процедурата за броење на гласови е опишана во шемата 6.5.3 и во секвенцниот дијаграм 6.4.

Со оваа фаза завршува гласачкиот процес и резултатите се јавно објавени. По овој момент сите играчи ги даваат добиените информации и секој може да направи увид во точноста на резултатите.

1. **Отпочнување на броење:**
   - DKG играчот j праќа порака commandStartTally кон случаен бројач j
2. **Преземање на делови од приватниот клуч:**
   - За секој пронајден дел од приватен клуч праќа порака commandAskEval кон останатите бројачи заедно со идентитетот на делот од приватниот клуч
   - Кога бројач ќе добие порака commandAskEval го враќа назад делот од приватниот клуч за добиениот идентитет
   - Кога бројачот j ќе собере доволен број на делови од приватниот клуч го екстрахира приватниот клуч од добиените делови
3. **Преземање на енкриптирано гласачко ливче:**
   - За секој екстрахиран приватен клуч, бројачот j испраќа порака askEncryptedBallot до сите гласачки кутии заедно со идентитетот на приватниот клуч
   - Кога гласачка кутија ќе добие порака askEncryptedBallot за даден идентитет, проверува дали има гласачко ливче за наведениот идентитет и доколку има го враќа назад со порака receiveEncryptedBallot
   - Кога бројачот j ќе добие енкриптирано гласачко ливче, го декриптира со екстрахираниот приватен клуч
4. **Испраќање на резултати:**
   - Кога бројачот j ги измине сите делови од приватни клучеви, сите добиени резултати ги праќа на сите DKG играчи со пораката commandReceiveFinalTally

Шема 6.5.3: Процедура за броење на гласови

Цената на одвивање на протоколот за броење на гласовите е претставена преку бројот и големината на пратените пораки и преку аритметичките операции што ги извршуваат играчите во протоколот. Сумираните податоци за давањето на глас се дадени во следните шеми.



Слика 6.4: Секвенцен дијаграм на броење гласови.

1. **DKG Играч испраќа порака кон случајно избран Бројач:**
   - byte: 1
2. **Бројачот за секој најден дел од приватен клуч испраќа по $n_{talliers} - 1$ порака кон останатите Бројачи:**
   - byte: 1
   - byte array of hashed id: 1
3. **Бројач праќа дел од приватен клуч за одреден гласач (вкупно $n_{talliers} - 1$ пораки):**
   - byte: 1
   - byte array of hashed id: 1
   - G1: k
   - S: 1
4. **Бројач праќа порака кон сите Гласачки кутии за енкриптиран глас за даден гласач (вкупно n_{voting_boxes}):**
   - byte: 1
   - byte array of hashed id: 1
4. **Гласачка кутија праќа енкриптиран глас:**
   - byte: 1
   - byte array of hashed id: 1
   - byte array of encrypted vote: 1

Шема 6.5.4: Број на пораки при собирање на еден глас

1. При добивање на сите делови од приватниот клуч за даден Гласач и неговиот енкриптиран глас Бројачот за секој глас пресметува:
- scalar negate: $k(k-1)+1$
- scalar multiplication: $k(2k+1)$
- scalar substraction: $k(k-1)$
- scalar division: $k+1$
- scalar addition: k
- pairing: 2
- G3 scalar exponentiation: 1
- G3 multiplication: 1
- SHA1 hash: 1
- scalar equality test: 1

Шема 6.5.5: Број на пораки при собирање на еден глас

6.6 Имплементација и резултати

Целта на овој докторски труд е прикажување на возможен и практичен систем за електронско мобилно гласање на уреди со ограничени карактеристики. За таа цел го имплементиравме предложениот модел за електронско гласање на најзастапената мобилна платформа на пазарот, мобилната платформа Android (според ИДЦ [64]). Како алатка за програмирање во Android го користевме Android Studio и Android SDK верзија 22 (повеќе позната како Android 5.1 Lollipop). Минималната верзија на која може да работи оваа апликација е Android SDK 16 (позната како Android 4.1 Jelly Bean). Бидејќи оваа верзија е објавена во јули 2012та, тоа значи дека апликацијата покрива телефони кои се продаваат веќе 5 години.

Во апликацијата се имплементирани криптографските протоколи според кои се извршува предложениот модел, а како комуникација се користат Bluetooth или WiFi Direct технологиите. Поради некомплетната техничка имплементација на Bluetooth чиповите во мобилните уреди кои ги користевме за тестирање, не успеавме да поврземе повеќе од 5 уреди во секој-со-секого Bluetooth врска. За сценарија каде се потребни повеќе мобилни уреди мо-

же да се користи WiFi Direct технологијата при што мобилните телефони се поврзуваат во мрежа без потреба од дополнителни WiFi уреди.

Тестирањето се изврши на десет уреди од марката LG Joy со чипсет Qualcomm MSM8210 Snapdragon 200, процесор Dual-core 1.2 GHz Cortex-A7 и 512MB RAM меморија. Сите фази од процесот на гласањето се извршија по 1000 пати за да се пресмета средна вредност која е презентирана во резултатите. Покрај историодните уреди, тестирањето се изврши и на различни Android уреди, вклучувајќи телефони од марките LG, Samsung, Asus и Xiaomi кои имаат различни верзии на Android оперативниот систем, од Jelly Bean (Android 16) до Marshmallow (Android 23). Со ова покажавме дека апликацијата работи во хетерогена средина при што големата фрагментација на Android платформата не претставува закана за компатибилност на апликацијата.

При тестирањето на апликација ги меревме времињата на извршување на секоја од четирите главни фази: поставување на системот, екстрахирање на клуч, гласање и броење на гласовите. Сите мерења се направени на вистински мобилни уреди со различни перформанси користејќи ја WiFi Direct технологијата за комуникација. Во фазите поставување на системот и екстрахирање на клуч работевме со $(n, t) \in \{(2,1), (3,1), (3,2), (4,1), (4,2), (4,3)\}$, а во фазите за гласање и броење на гласови со $(n, t) \in \{(2,1), (3,2), (4,3)\}$.

Времињата на извршување на првата фаза, поставување на системот, се дадени во табелите 6.1 и 6.2. Во оваа фаза идентификуваме случајно избран DKG играч како лидер, а останатите DKG играчи ги означуваме како регуларни играчи. Во табелите се прикажани одделно времињата за иницијализација на DKG играч (step1), делење на заложувањата и евалуациите (step2) и проверка на примените заложувања и евалуации (step3). Во сценарио каде имаме вкупно четири DKG играчи со прагова вредност од три DKG играчи, вкупното време на извршување на протоколот трае 3,3 секунди. Ова е визуелно покажано во графикот 6.5.

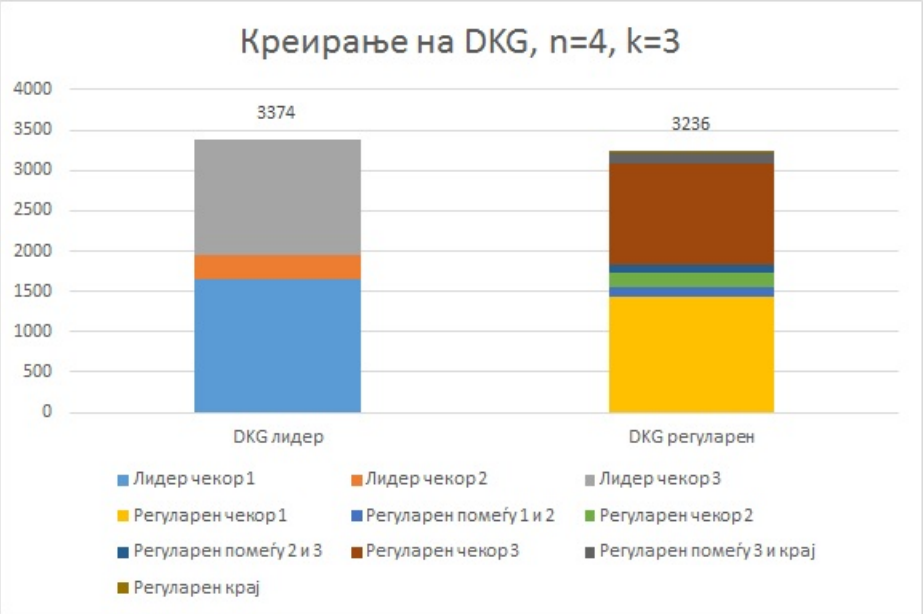
Фазата на екстрахирање на клучеви првиот пат се случува веднаш по завршувањето на DKG системот. Во овој момент, играчите потпишувач, гласачка кутија и бројач ги контактираат DKG играчите со цел да ги добијат своите приватни клучеви. Времињата на извршување на оваа фаза се дадени во табелите 6.3 и 6.4. Во табелите гледаме време кое е потребно секој DKG

Табела 6.1: Време на извршување на фазата поставување на системот за DKG Лидер (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=3, t=2	n=4, t=1	n=4, t=2	n=4, t=3
Чекор 1	1132	1126	1394	1145	1414	1654
Чекор 2	181	221	271	229	301	306
Чекор 3	748	1037	1115	1391	1419	1424
Вкупно	2060	2384	2781	2765	3133	3384

Табела 6.2: Време на извршување на фазата поставување на системот за DKG Член (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=3, t=2	n=4, t=1	n=4, t=2	n=4, t=3
Чекор 1	974	960	1243	922	1208	1443
Чекор 2	119	141	126	129	153	119
Чекор 3	89	116	161	109	197	179
Чекор 4	83	93	96	71	86	96
Чекор 5	632	919	966	1185	1253	1256
Чекор 6	105	125	141	147	177	133
Чекор 7	12	12	12	12	13	11
Бкупно	2014	2367	2745	2575	3086	3238



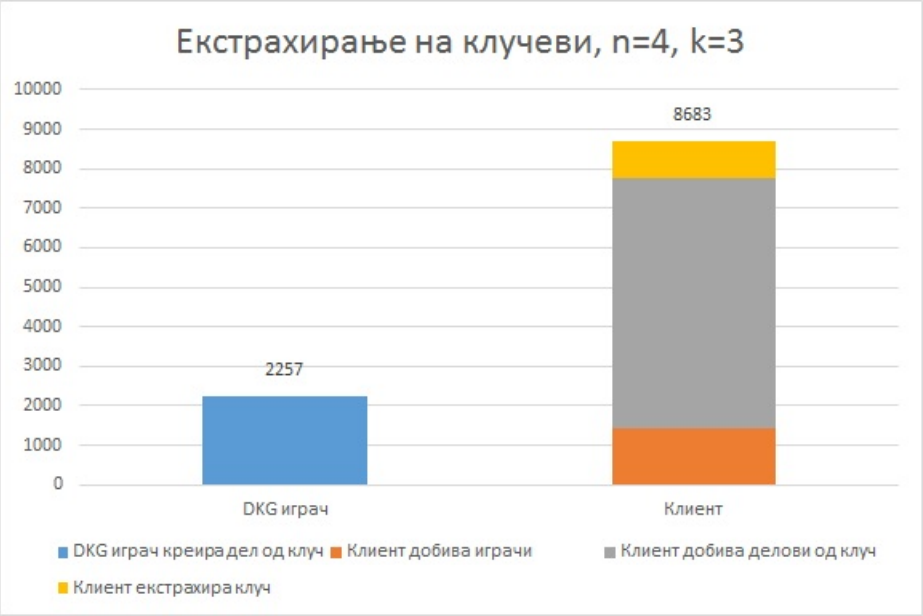
Слика 6.5: Време на извршување на фазата поставување на системот (вредностите се дадени во милисекунди).

Табела 6.3: Време на извршување на фазата екстрахирање на клуч за потпишувач (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

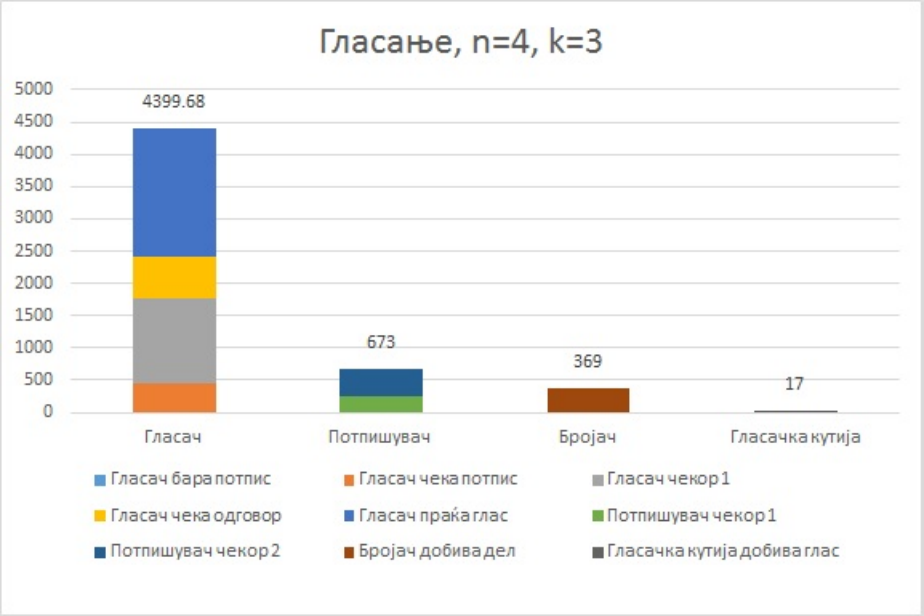
	n=2, t=1	n=3, t=2	n=3, t=2	n=4, t=1	n=4, t=2	n=4, t=3
Чекор 1	1632	1585	1456	1472	1524	1479
Чекор 2	15	16	16	19	23	22
Чекор 3	5	8	6	8	9	12
Чекор 4	4652	6020	5381	6026	6020	6389
Чекор 5	986	1015	855	851	855	939
Вкупно	7291	8643	7714	8376	8431	8840

Табела 6.4: Време на извршување на фазата екстрахирање на клуч за DKG играч (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=3, t=2	n=4, t=1	n=4, t=2	n=4, t=3
Чекор 1	2224	2196	2239	2124	2241	2257



Слика 6.6: Време на извршување на фазата екстрахирање на клучеви (вредностите се дадени во милисекунди).



Слика 6.7: Време на извршување на фазата гласање (вредностите се дадени во милисекунди).

играч да пресмета дел од клучот за бараниот идентитет и време кое е потребно за клиентот да екстрахира приватен клуч од DKG играчите. Во случај на DKG клиент, тоа време го делиме на време за добивање на листа од DKG играчи, барање на делови од клучот и конструирање на приватниот клуч од добиените делови. Секој DKG играч во просек троши по 711 милисекунди при барање за екстракција од DKG клиент. Во сценарио со четири DKG играчи со прагова вредност од три DKG играчи, секој клиент троши во просек по 8,6 секунди за екстрахирање на приватен клуч од DKG групата. Ова е прикажано во графикот 6.6.

Фазата која е најважна во секој гласачки процес е фазата на гласање. Времињата на извршување од оваа фаза се дадени во табелите 6.5, 6.6, 6.7 и 6.8. Во оваа фаза учествуваат играчите гласач, потпишувач, гласачка кутија и бројач. Во графикот овие играчи се прикажани како посебни колони со вкупното време на извршување при гласање на еден гласач. Кај гласачот времињата се поделени на барање на слеп потпис, чекање на иницијални вредности од потпишувач, прва фаза за добивање на слеп потпис, чекање на слеп потпис и гласање. Вредностите од овој график се дадени во сценарио со четири DKG играчи со прагова вредност од три DKG играчи, еден потпишу-

Табела 6.5: Време на извршување на фазата гласање за гласач (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=4, t=3
Чекор 1	3	2	2
Чекор 2	464	457	454
Чекор 3	1315	1315	1315
Чекор 4	676	640	635
Чекор 5	1619	1813	1995
Вкупно	4077	4227	4401

Табела 6.6: Време на извршување на фазата гласање потпишувач (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=4, t=3
Потпис 1	283	260	260
Потпис 2	460	440	414
Вкупно	743	700	674

Табела 6.7: Време на извршување на фазата гласач за гласачка кутија (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=4, t=3
Потпис 1	283	260	260
Потпис 2	460	440	414
Вкупно	743	700	674

Табела 6.8: Време на извршување на фазата гласање за бројач (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=4, t=3
Чекор 1	350	355	369

вач, една гласачка кутија и четири бројачи со шема $n = 1$, $t = 3$. Средното време кое е потребно за гласање кај гласачот е 4,4 секунди. За тоа време потпишувачот троши просечно 673 милисекунди во двата чекора при издавање на слеп потпис. Гласачката кутија просечно троши по 17 милисекунди за зачувување на секој пратен енкриптиран глас од гласач. На четирите бројачи во ова сценарио им е потребно по 369 милисекунди за зачувување на деловите од приватниот клуч за декриптирање на пратениот глас. Ова е прикажано во графикот 6.7.

Табела 6.9: Време на извршување на фазата броење за главен бројач (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=4, t=3
Чекор 1	2338	2976	3731
Чекор 2	10	10	18
Вкупно	**2348**	**2986**	**3749**

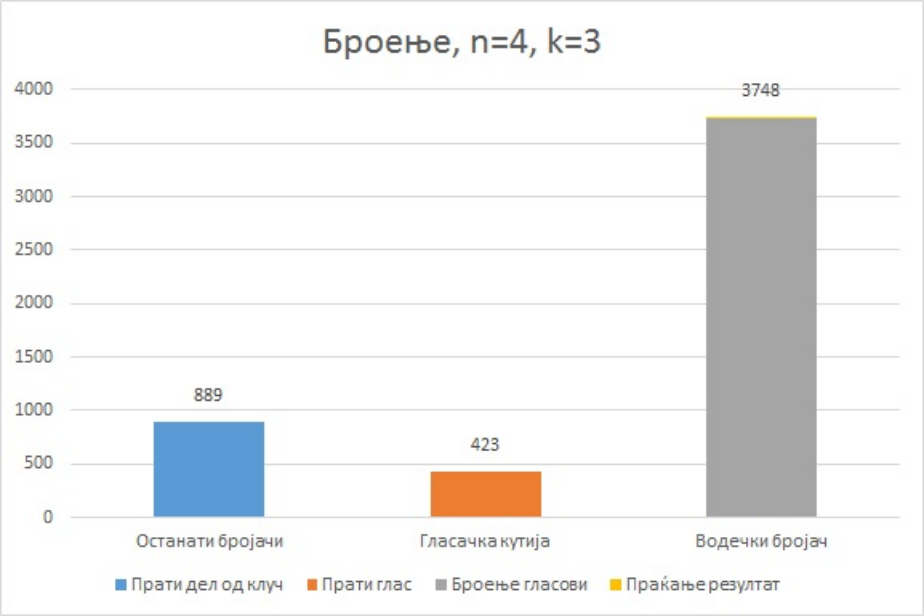
Табела 6.10: Време на извршување на фазата броење за останати бројачи (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=4, t=3
Чекор 1	7	7	9

Табела 6.11: Време на извршување на фазата броење за гласачка кутија (средно време од 1000 извршувања, вредностите се дадени во милисекунди)

	n=2, t=1	n=3, t=2	n=4, t=3
Чекор 1	3	3	4

Фазата броење се извршува на крајот од гласачкиот процес. Во овој чекор се бира случајно барем еден бројач да го изврши протоколот за броење на гласови. Времето потребно да се процесира еден глас од страна на бројачот е дадено во табелите 6.9, 6.10 и 6.11. Во табелите се идентификувани играчите бројач, гласачка кутија и водечки бројач. Во овој случај, бројач претставува играч од кој се побарува дел од приватен клуч за даден глас, а водечки бројач претставува бројачот кој го извршува протоколот за броење на гласови, односно овој бројач ги контактира останатите бројачи за да го комплетира приватниот клуч за гласот кој го побарува од гласачките кутии. Во просек, еден бројач троши по 889 милисекунди за секои 100 побарани делови од

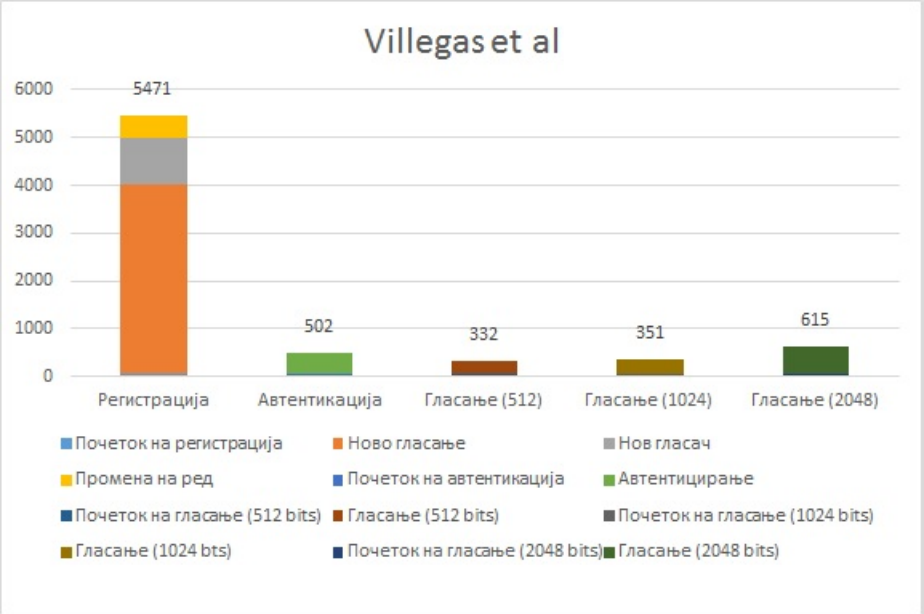


Слика 6.8: Време на извршување на фазата броење (вредностите се дадени во милисекунди).

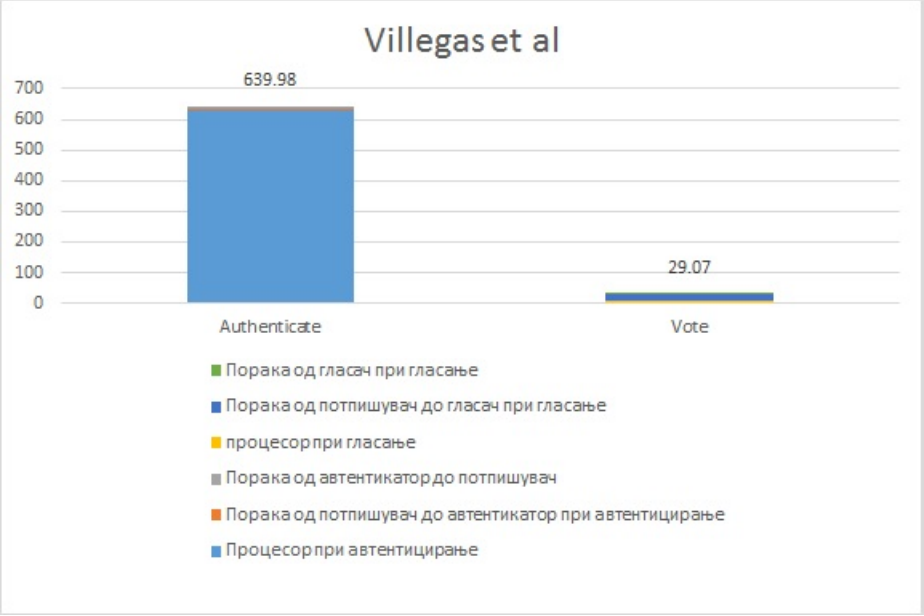
глас, една гласачка кутија троши по 423 милисекунди за секои 100 побарани енкриптирани гласови и водечкиот бројач троши 3.7 секунди за спојување на приватниот клуч, декриптирање и броење на 100 гласа. Ова е графички прикажано во графикот 6.8.

6.7 Споредба и заклучок

Како споредба, најдовме информации за времињата на извршување за два од четирите системи опишани во подглавата 6.4. Времето за извршување на системот [38] е дадено во графикот 6.9, а времето за извршување на системот [75] е дадено во графикот 6.10. Бидејќи информациите за потребното време не се совпаѓаат со нашите прикажани фази, не е возможно директна споредба на системите за гласање. Исто така, овие системи користат постари модели на Android телефони кои имаат побавни процесори од уредите на кои го тестиравме нашиот систем. За споредба, системот предложен во [38] е тестиран на Android 8 (Android 2.2 Froyo) на уредот Xperia Play, а системот предложен во [75] е тестиран на Android 10 (Android 2.3.3 Gingerbread) на уредот Xperia S LT26i. Бидејќи кодот од овие системи не е јавно достапен, не



Слика 6.9: Време на извршување на системот [38] (вредностите се дадени во милисекунди).



Слика 6.10: Време на извршување на системот [75] (вредностите се дадени во милисекунди).

сме во состојба да ги споредиме системите на идентични платформи и да ги прикажеме времињата на еден график за споредба. Предложениот систем и резултатите се презентирани во трудот [97] кој е во процес на објавување.

Од имплементацијата на предложениот систем за електронско гласање може да се види дека е возможно извршување на гласачки процеси на уреди со ограничени карактеристики. Ваков систем може да се користи во т.н. ад-хок гласачки процеси поради мобилноста на опремата која е потребна за извршување. Всушност, за извршување на гласачки процес не е потребно да се има дедицирирана опрема, доволно е да се искористат Android мобилните уреди од самите учесници во гласачкиот процес. Доколку е потребно да се организираат поголеми гласачки процеси, повторно може да се искористи овој систем при што би се поставило посебно множество од уреди во секое гласачко место. Сите овие гласачки места ќе бидат изолирани гласачки процеси, па на крај ќе биде потребно сумирање на резултатите за добивање на крајни резултати.

Глава 7

Заклучок

7.1 Преглед и придонес со резултати

Темата електронско гласање претставува широко поле на истражување. Во овој докторат покривме различни теми, почнувајќи од елиптични криви и парови на елиптични криви, до имплементација на систем за електронско гласање на Android мобилната платформа. Направивме софтверска библиотека за брзо извршување на аритметички операции со елиптични криви и со елементи на основните полиња врз кои работат елиптичните криви. За потребата на темата истражувавме за ID-Базирана криптографија и понудивме решение за PKI инфраструктура без сертификати и без ниту една клучна точка. Дополнително беа истражувани и ID-Базирани слепи потписи објавени во литературата. За таа цел направивме целосно порамнување во номенклатурата на избраните алгоритми за слепи потписи. Финалниот продукт е табела за лесна споредба на алгоритмите за слепи потписи: споредба на потребни аритметички операции и потребен пропусен опсег при размената на податоци. Ги обединивме овие истражувања за потребите на систем за електронско гласање кој ќе може да се извршува на уреди со ограничени карактеристики. Ефективно покажавме дека со користење на ID-Базирана криптографија и користејќи елиптични криви може да се направи ефикасен електронски систем користејќи ја Android мобилната платформа.

Придонесот од имплементацијата на Панда во глава 3 претставува библиотека за аритметички операции со елиптични криви и пресметување на пар над елиптични криви. Оваа библиотека нуди заштита од временски напади

бидејќи за секоја аритметичка операција нуди функција која се извршува во константнто време. Перформансите на оваа библиотека се дадени во табелите 3.1, 3.2, 3.3 и 3.4 во глава 3. Овие резултати покажуваат дека библиотеката има многу брза имплементација која може да се користи за реални проекти. Пример за ова даваме во секција 3.4.1 каде е покажана реална имплементација и перформанси на BLS потписите со помош на Панда библиотеката. Темата и резултатите од оваа глава се објавени во трудот [31].

Во главата 4 претставеното решение и имплементација на PKI инфраструктура без сертификати покажува дека ID-базирана криптографија може се користи и класичната јавна криптографија. Придонесот на PKI инфраструктура без сертификати се гледа во елиминирањето на потребата за преземање или постојано транспортирање на јавни клучеви. Слабоста на овој систем поради големата моќ на центарот за генерирање на приватни клучеви е успешно отстранета со помош на дистрибуирано креирање на приватен клуч. Резултатите од овој систем за дистрибуирано креирање на приватен клуч и екстракција на клучеви од членовите на центарот за генерирање на приватни клучеви се дадени во табелите 6.1, 6.2, 6.3 и 6.4. Резултатите од оваа тема се објавени во трудот [96].

Истражуваните слепи потписи со ID-базирана криптографија беа потребни за креирање на системот за електронско гласање. За таа цел во главата 5 беа истражувани алгоритми за слепи потписи со ID-базирана криптографија. Придонесот од оваа глава претставува унификацијата на нотацијата на алгоритмите и директната споредба по бројот и типот на аритметичките операции и по потребниот пропусен опсег за извршување на алгоритмите. Резултите од овој дел се дадени во табелите 5.1, 5.2 и 5.3 и истите се објавени во трудот [95].

Главната тема на овој докторски труд беше дизајн на систем за електронско гласање и негова имплементација на уреди со ограничени карактеристики. Системот и имплементацијата се дадени во главата 6. Придонесот на оваа глава е всушност и главниот придонес на докторскиот труд: апликација за електронско гласање која може да се користи за електронско гласање со помош на мобилни телефони со мали перформанси. Перформансите со користењето на Android уреди од марката LG Joy со процесор Dual-core 1.2 GHz Cortex-A7 и 512MB RAM меморија се дадени во графиците 6.5, 6.6, 6.7

и 6.8. Резултатите се многу добри и практично покажуваат реална употребна вредност на предложениот систем за електронско гласање. Системот и резултатите се презентирани во трудот [97] кој е во процес на објавување. Покрај наведената марка, апликацијата беше тестирана и на поголемо множество на уреди од марките LG, Samsung, Xiaomi и Asus со што ефективно се покажа дека апликацијата нема проблеми со големата фрагментираност на Android оперативниот систем. Апликацијата успешно работеше во средина од различни уреди и го извршуваше предложениот протокол за електронско гласање.

7.2 Отворени прашања

При истражувањето на темата на овој докторски труд се отвориja повеќе различни прашања поврзани со електронското гласање. Бидејќи постојат повеќе начини да се изведе електронско гласање, повеќето отворени прашања се насочени кон имплементирање на останатите протоколи за електронско гласање на уреди со ограничени карактеристики. Во оваа подглава ги наведуваме отворените прашања кои произлегоа од истражувањето на оваа тема.

1. При изработката на системот за електронско гласање со помош на уреди со мали карактеристики избравме протокол со користење на слепи потписи за зачувување на тајноста. Би било интересно да се погледне споредба на перформансите со електронско гласање кое користи друг протокол (на пример хомоморфни енкрипции);

2. При имплементирањето на криптографските протоколи користевме ID-Базирана криптографија со помош на парови над елиптични криви. Како отворено прашање се поставува ефикасноста на систем со користење на други криви каде пресметувањето на паровите е побавно, меѓутоа аритметиката на ниско ниво е побрза. Исто така би било од интерес користењето на класична PKI инфраструктура со сертификати;

3. Blockchain технологијата зема голем замав во последно време, и во праксата и во теоријата. Како отворено прашање стои корисноста на оваа технологија во електронското гласање, особено во делот кај заложувањата на информациите;

4. Како отворено прашање може да го сметаме планираното истражување за хибриден модел со комбинација на уреди со ограничени карактеристики и класични серверски компоненти (подглава 7.3). Овој хибриден модел би овозможил организирање на масовни гласачки процеси (поддржан од класичните серверски компоненти) со достапни уреди како крајни точки (користејќи уреди со ограничени карактеристики).

7.3 Идна работа

Специфичноста на темата за електронско гласање наложи истражување на повеќе различни теми од криптографијата и од информатичките технологии. Секоја глава покрива одредена тема, за на крај сите теми да се искористат во главата за електронско гласање. При истражувањето дојдовме до одредени теми кои ги оставивме како идна работа бидејќи се интересни за понатамошен развој.

Во главата за Панда (глава 3) направивме библиотека за работа со Barreto-Naehrig елиптичните криви. Како идна работа на библиотеката ги дефинираме следните работи:

- Хеширање во втората група на аргументите на паровите
- Оптимизирање на пресметката на паровите преку користење на стандардни проективни координати и пресметување на крајното степенување со помош на compressed squarings како што е објаснето во [6];
- Оптимизирање и забрзување на аритметиката на пониско ниво
- Оптимизирање на аритметиката со асемблерски рутини за ARM процесорите со NEON поддршка;
- Поддршка за работа со други криви во кои аритметиката на пониско ниво е побрза од Barreto-Naehrig кривите за протоколи кои не користат парови (или работата со парови е занемарливо помала од обемот на работа со аритметика на пониско ниво).

Во главата за електронско гласање (глава 6) понудивме решение за мобилно електронско гласање со помош на Android платформата. Како идна работа во ова поле ги дефиниравме следните работи:

- Хибриден модел на систем за електронско гласање кој користи уреди со ограничени карактеристики кај крајните корисници и компјутерски уреди во серверскиот дел. Овој систем би понудил поголем капацитет за масовни гласачки процеси;
- Истражување на други платформи со уреди со ограничени карактеристики кои се присутни на пазарот со значаен удел (на пример iOS платформата).
- Вклучување на поспецифични уреди во системите за електронско гласање: сим картички, паметни картички, микроконтролери и останати уреди кои се со многу ниска цена и кои може да се искористат за исполнување на повеќе карактеристики на системот за електронско гласање (подглава 6.3).

Планираме континуирано да продолжиме со работата на системи за електронско гласање. Овие системи покажуваат дека станува практично користењето на електронско гласање за олеснување на организирањето на гласачките процеси.

Листа на слики

Листа на табели

ЛИСТА НА ТАБЕЛИ

Листа на шеми

Референци

[1] Adj, G., Menezes, A., Oliveira, T., and Rodríguez-Henríquez, F. Weakness of $\mathbb{F}_{3^{6 \cdot 509}}$ for discrete logarithm cryptography. Cryptology ePrint Archive, Report 2013/446, 2013. http://eprint.iacr.org/2013/446/. 29

[2] Adj, G., and Rodríguez-Henríquez, F. Square root computation over even extension fields. Cryptology ePrint Archive, Report 2012/685, 2012. http://eprint.iacr.org/. 41

[3] Akinyele, J. A., Garman, C., Miers, I., Pagano, M. W., Rushanan, M., Green, M., and Rubin, A. D. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering 3*, 2 (2013), 111–128. http://eprint.iacr.org/2011/617/. 32

[4] Alefragis, P., S., Lounis, S., K., Triantafillou, V., D., and Voros, N., S. An electronic voting scheme with physical multiple administrators and identical ballot boxes. In *International Conference on WWW/Internet*. 2004, p. 99. 95

[5] Aranha, D. F., and Gouvêa, C. P. L. RELIC is an Efficient LIbrary for Cryptography. http://code.google.com/p/relic-toolkit/, 2013. 31

[6] Aranha, D. F., Karabina, K., Longa, P., Gebotys, C. H., and López, J. Faster explicit formulas for computing pairings over ordinary curves. In *Advances in Cryptology – Eurocrypt 2011* (2011), K. G. Paterson, Ed., vol. 6632 of *Lecture Notes in Computer Science*, Springer, pp. 48–68. http://eprint.iacr.org/2010/526/. 27, 39, 41, 43, 44, 126

[7] Barbulescu, R., Gaudry, P., Joux, A., and Thomé, E. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. Cryptology ePrint Archive, Report 2013/400, 2013. http://eprint.iacr.org/2013/400/. 29

[8] Barreto, P. S., and Naehrig, M. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography* (2006), B. Preneel and S. Tavares, Eds., vol. 3897 of *Lecture Notes in Computer Science*, Springer, pp. 319–331. http://cryptosith.org/papers/#bn. 39

[9] Barrett, P. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In *Advances in Cryptology – CRYPTO '86* (1987), A. M. Odlyzko, Ed., vol. 263 of *Lecture Notes in Computer Science*, Springer, pp. 311–323. 44

[10] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., and Shacham, H. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology – CRYPTO 2009* (2009), S. Halevi, Ed., vol. 5677 of *Lecture Notes in Computer Science*, Springer, pp. 108–125. http://research.microsoft.com/pubs/122759/anoncred.pdf. 27

[11] Bellare, M., Namprempre, C., Pointcheval, D., and Semanko, M. The power of rsa inversion oracles and the security of chaumian rsa-based blind signature scheme. In *In Financial Cryptography* (2001), pp. 319–338. 78

[12] Bernstein, D. J., Birkner, P., Joye, M., Lange, T., and Peters, C. Twisted Edwards curves. In *Progress in Cryptology – AFRICACRYPT 2008* (2008), S. Vaudenay, Ed., vol. 5023 of *Lecture Notes in Computer Science*, Springer, pp. 389–405. http://cr.yp.to/papers.html#twisted. 31

[13] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., and Yang, B.-Y. High-speed high-security signatures. In *Cryptographic Hardware and Embedded Systems – CHES 2011* (2011), B. Preneel and T. Takagi, Eds.,

vol. 6917 of *Lecture Notes in Computer Science*, Springer, pp. 124–142. see also full version [14]. 30, 43, 137

[14] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., and Yang, B.-Y. High-speed high-security signatures. *Journal of Cryptographic Engineering 2*, 2 (2012), 77–89. http://cryptojedi.org/papers/#ed25519, see also short version [13]. 30, 43, 137

[15] Bernstein, D. J., and Lange, T. Faster addition and doubling on elliptic curves. In *Advances in Cryptology – ASIACRYPT 2007* (2007), K. Kurosawa, Ed., vol. 4833 of *Lecture Notes in Computer Science*, Springer, pp. 29–50. http://cr.yp.to/papers.html#newelliptic. 31

[16] Bernstein, D. J., and Lange, T. eBACS: ECRYPT benchmarking of cryptographic systems, 2013. http://bench.cr.yp.to (accessed 2013-08-15). 6, 28, 44

[17] Bernstein, D. J., Lange, T., and Schwabe, P. The security impact of a new cryptographic library. In *Progress in Cryptology – LATINCRYPT 2012* (2012), A. Hevia and G. Neven, Eds., vol. 7533 of *Lecture Notes in Computer Science*, Springer, pp. 159–176. http://cryptojedi.org/papers/#coolnacl. 31

[18] Beuchat, J.-L., González Díaz, J. E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., and Teruya, T. High-speed software implementation of the optimal ate pairing over Barreto-Naehrig curves, 2010. http://eprint.iacr.org/2010/354/. 27, 43

[19] Blakley, G. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference* (Monval, NJ, USA, 1979), AFIPS Press, pp. 313–317. 19

[20] Boneh, D., and Franklin, M. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO 2001*, J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*. Springer, 2001, pp. 213–229. http://www.iacr.org/archive/crypto2001/21390212.pdf. 27

[21] Boneh, D., and Franklin, M. Identity-based encryption from the weil pairing. In *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 213–229. 55, 56

[22] Boneh, D., Lynn, B., and Shacham, H. Short signatures from the Weil pairing. *Journal of Cryptology 17*, 4 (2004), 297–319. http://crypto.stanford.edu/~dabo/pubs/papers/weilsigs.ps. 27, 41, 44, 45

[23] Bos, J. W., Costello, C., and Naehrig, M. Exponentiating in pairing groups. In *Selected Areas in Cryptography – SAC 2013* (2013), T. Lange, K. Lauter, and P. Lisonek, Eds., vol. to appear of *Lecture Notes in Computer Science*. http://cryptosith.org/papers/#exppair. 28, 43

[24] Bosma, W., and Lenstra, H. W. Complete systems of two addition laws for elliptic curves. *Journal of Number Theory 53* (1995), 229–240. http://www.math.ru.nl/~bosma/pubs/JNT1995.pdf. 42

[25] Brumley, B. B., and Tuveri, N. Remote timing attacks are still practical. In *Computer Security—ESORICS 2011* (2011), V. Atluri and C. Diaz, Eds., vol. 6879 of *LNCS*, Springer, pp. 355–371. http://eprint.iacr.org/2011/232/. 30

[26] Camenisch, J., Piveteau, J.-M., and Stadler, M. Blind signatures based on the discrete logarithm problem. In *Advances in Cryptology EUROCRYPT'94*, A. Santis, Ed., vol. 950 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1995, pp. 428–432. 78

[27] Certivox. MIRACL Cryptographic SDK. http://www.certivox.com/miracl, 2013. 31

[28] Chatterjee, S., and Menezes, A. On cryptographic protocols employing asymmetric pairings – the role of ψ revisited. *Discrete Applied Mathematics 159* (2011), 1311–1322. http://eprint.iacr.org/2009/480/. 29

[29] Chaum, D. Blind signatures for untraceable payments. In *Advances in Cryptology*, D. Chaum, R. Rivest, and A. Sherman, Eds. Springer US, 1983, pp. 199–203. 71, 78

[30] Chor, B., Goldwasser, S., Micali, S., and Awerbuch, B. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 1985), SFCS '85, IEEE Computer Society, pp. 383–395. 22

[31] Chuengsatiansup, C., Naehrig, M., Ribarski, P., and Schwabe, P. Panda: Pairings and arithmetic. In *Pairing-Based Cryptography – Pairing 2013* (2014), Z. Cao and F. Zhang, Eds., vol. 8365 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, pp. 229–250. Document ID: 775a51985db9972bde7bd2acddf1d2a2, `http://cryptojedi.org/papers/#panda`. 48, 124

[32] Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., and Vercauteren, F. *Handbook of Elliptic and Hyperelliptic Curve Cryptography, Second Edition*, 2nd ed. Chapman & Hall/CRC, 2012. 9

[33] Cytron, R. K., Cranor, L. F., and Cranor, L. F. Design and implementation of a practical security-conscious electronic polling system. Tech. rep., 1996. 92

[34] Daniel, B., and Tanja, L. Addition formulas. `http://www.hyperelliptic.org/EFD/g1p/auto-shortw-jacobian-0.html#addition-add-2007-bl`, 2016. 42

[35] Daniel, B., and Tanja, L. Addition formulas. `http://www.hyperelliptic.org/EFD/g1p/auto-shortw-jacobian-0.html#doubling-dbl-2009-l`, 2016. 42

[36] de Rooij, P. Efficient exponentiation using precomputation and vector addition chains. In *Advances in Cryptology – EUROCRYPT '94* (1995), A. D. Santis, Ed., vol. 950 of *Lecture Notes in Computer Science*, Springer, pp. 389–399. 43

[37] Diffie, W., and Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theor. 22*, 6 (Sept. 1976), 644–654. 14

[38] Eliver, Pérez, V., Gina, G.-G., Gualberto, Aguilar, T., and Héctor, Flores, G. Implementation of electronic voting system in mobile phones with android operating system. In *Journal of Emerging Trends in Computing and Information Sciences*, vol. 4/9. 2013, pp. 728–737. 95, 119, 120, 130

[39] Fangguo, Z., and Kwangjo, K. Efficient id-based blind signature and proxy signature. In *In Proceedings of ACISP 2003, LNCS 2727* (2003), Springer-Verlag, pp. 312–323. 73, 74, 75, 76, 77, 80

[40] Fardan, N. J. A., and Paterson, K. G. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *2013 IEEE Symposium on Security and Privacy* (2013), IEEE Computer Society, pp. 526–540. `www.isg.rhul.ac.uk/tls/TLStiming.pdf`. 30

[41] Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 1987), SFCS '87, IEEE Computer Society, pp. 427–438. 7, 22

[42] Foundation, A. Ssl/tls strong encryption: An introduction, 2016. 13

[43] Fouque, P.-A., and Tibouchi, M. Indifferentiable hashing to Barreto–Naehrig curves. In *Progress in Cryptology – LATINCRYPT 2012*, A. Hevia and G. Neven, Eds., vol. 7533 of *Lecture Notes in Computer Science*. Springer, 2012, pp. 1–17. `www.di.ens.fr/~fouque/pub/latincrypt12.pdf`. 42

[44] Freeman, D., Scott, M., and Teske, E. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology 23*, 2 (2010), 224–280. `http://eprint.iacr.org/2006/372/`. 27

[45] Fujioka, A., Okamoto, T., and Ohta, K. A practical secret voting scheme for large scale elections. In *Advances in Cryptology AUSCRYPT '92*, J. Seberry and Y. Zheng, Eds., vol. 718 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1993, pp. 244–251. 72

[46] Galbraith, S. Joux kills pairings in characteristic 2, 2013. http://ellipticnews.wordpress.com/2013/05/22/joux-kills-pairings-in-characteristic-2/ (accessed 2016-09-12). 29

[47] Gallegos-García, G., Gómez-Cárdenas, R., and Duchén-Sánchez, G. I. Identity based threshold cryptography and blind signatures for electronic voting. *W. Trans. on Comp. 9*, 1 (Jan. 2010), 62–71. 73

[48] Gao, W., Wang, G., Wang, X., and Li, F. One-round id-based blind signature scheme without ros assumption. In *Proceedings of the 2Nd International Conference on Pairing-Based Cryptography* (Berlin, Heidelberg, 2008), Pairing '08, Springer-Verlag, pp. 316–331. 73, 74, 77, 80

[49] Gao, W., Wang, G., Wang, X., and Li, F. Round-optimal id-based blind signature schemes without ros assumption. *Journal of Communications 7*, 12 (2012). 73, 74, 80

[50] Gentry, C., and Silverberg, A. Hierarchical id-based cryptography. In *Advances in Cryptology – ASIACRYPT 2002* (2002), Y. Zheng, Ed., vol. 2501 of *Lecture Notes in Computer Science*, Springer, pp. 548–566. http://www.cs.ucdavis.edu/~franklin/ecs228/pubs/extra_pubs/hibe.pdf. 27

[51] The GNU MP library, 2013. http://gmplib.org/ (accessed 2013-11-02). 36

[52] Göloğlu, F., Granger, R., McGuire, G., and Zumbrägel, J. Solving a 6120-bit DLP on a desktop computer. In *Selected Areas in Cryptography* (2013), T. Lange, K. Lauter, and P. Lisonek, Eds., vol. to appear of *Lecture Notes in Computer Science*, Springer. http://eprint.iacr.org/2013/306. 29

[53] Granger, R., Page, D., and Stam, M. On small characteristic algebraic tori in pairing-based cryptography. *IACR Cryptology ePrint Archive* (2004), 132. http://eprint.iacr.org/2004/132. 41

[54] Granger, R., and Scott, M. Faster squaring in the cyclotomic subgroup of sixth degree extensions. In *Public Key Cryptography – PKC 2010* (2010), P. Q. Nguyen and D. Pointcheval, Eds., vol. 6056 of *LNCS*, Springer, pp. 209–223. 43

[55] Groth, J. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology – ASIACRYPT 2010* (2010), M. Abe, Ed., vol. 6477 of *Lecture Notes in Computer Science*, Springer, pp. 321–340. `http://www.cs.ucl.ac.uk/staff/J.Groth/ShortNIZK.pdf`. 27

[56] Groth, J., and Sahai, A. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput. 41*, 5 (2012), 1193–1232. `http://www0.cs.ucl.ac.uk/staff/J.Groth/WImoduleFull.pdf`. 27

[57] Hess, F. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography*, K. Nyberg and H. Heys, Eds., vol. 2595 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 310–324. 79, 80

[58] Hess, F., Smart, N. P., and Vercauteren, F. The eta pairing revisited. *IEEE Transactions on Information Theory 52*, 10 (2006), 4595–4602. `http://eprint.iacr.org/2006/110`. 27

[59] Hışıl, H. *Elliptic Curves, Group Law, and Efficient Computation*. PhD thesis, Queensland University of Technology, 2010. `http://eprints.qut.edu.au/33233/`. 42

[60] Horster, P., Petersen, H., Horster, P., and Petersen, H. Classification of blind signature schemes and examples of hidden and weak blind signatures, 1994. 72

[61] Horwitz, J., and Lynn, B. Toward hierarchical identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2002* (2002), L. R. Knudsen, Ed., vol. 2332 of *Lecture Notes in Computer Science*, Springer, pp. 466–481. `http://theory.stanford.edu/~horwitz/pubs/hibe.pdf`. 27

[62] Huang, Z., Chen, K., and Wang, Y. Efficient identity-based signatures and blind signatures. In *Proceedings of the 4th International Conference on Cryptology and Network Security* (Berlin, Heidelberg, 2005), CANS'05, Springer-Verlag, pp. 120–133. 73, 74, 76, 79, 80

[63] IBM. Votomatic, 2016. 90

[64] IDC. Smartphone os market share, 2016 q2. `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`, 2016. 2, 3, 112, 129

[65] Joux, A. A one round protocol for tripartite Diffie-Hellman. In *Algorithmic Number Theory* (2000), W. Bosma, Ed., vol. 1838 of *Lecture Notes in Computer Science*, Springer, pp. 385–393. `cgi.di.uoa.gr/~aggelos/crypto/page4/assets/joux-tripartite.pdf`. 27

[66] Joux, A. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology 17*, 4 (2004), 263–276. 27

[67] Joux, A. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. In *Selected Areas in Cryptography* (2013), T. Lange, K. Lauter, and P. Lisonek, Eds., vol. to appear of *Lecture Notes in Computer Science*, Springer. invited paper, `http://eprint.iacr.org/2013/095/`. 29

[68] Kalajdzic, G. *Algebra*, 2 ed. Matematicki fakultet Beograd, 2000. 9

[69] Kalkan, S., Kaya, K., and Selcuk, A. Generalized id-based blind signatures from bilinear pairings. In *Computer and Information Sciences, 2008. ISCIS '08. 23rd International Symposium on* (Oct 2008), pp. 1–6. 73, 74, 78, 79, 80

[70] Kharchineh, B., and Ettelaee, M. A new electronic voting protocol using a new blind signature scheme. In *Future Networks, 2010. ICFN '10. Second International Conference on* (Jan 2010), pp. 190–194. 72

[71] Li, C.-T., Hwang, M.-S., and Lai, Y.-C. A verifiable electronic voting scheme over the internet. In *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on* (April 2009), pp. 449–454. 72

[72] Lopez-Garcia, L., Perez, L. J. D., and Rodriguez-Henriquez, F. A pairing-based blind signature e-voting scheme. *The Computer Journal* (2013). 73

[73] López-García, L., Perez, L. J. D., and Rodríguez-Henríquez, F. A pairing-based blind signature e-voting scheme. *The Computer Journal* (2013). 96

[74] Lynn, B. PBC library – the pairing-based cryptography library. `http://crypto.stanford.edu/pbc/`, 2013. 31

[75] María, de Lourdes, L. G., Asdrúbal, López, C., Javier, Silva, P., and Miguel, Ángel, L. C. An e-voting system for Android Smartphones. *Revista Facultad de IngenierÃa Universidad de Antioquia* (09 2014), 9 – 19. 96, 119, 120, 130

[76] Mebane, W. R. The wrong man is president! overvotes in the 2000 presidential election in florida. *Perspectives on Politics 2* (9 2004), 525–535. 90

[77] MITSUNARI, S. A fast implementation of the optimal ate pairing over BN curve on Intel Haswell processor. Cryptology ePrint Archive, Report 2013/362, 2013. `http://eprint.iacr.org/2013/362/`. 27

[78] Mote, Jr., C. D. Report of the national workshop on internet voting: Issues and research agenda. In *Proceedings of the 2000 Annual National Conference on Digital Government Research* (2000), dg.o '00, Digital Government Society of North America, pp. 1–59. 92

[79] Mu, Y., and Varadharajan, V. Anonymous secure e-voting over a network. In *Computer Security Applications Conference, 1998. Proceedings. 14th Annual* (Dec 1998), pp. 293–299. 72

[80] Naehrig, M., Barreto, P. S., and Schwabe, P. On compressible pairings and their computation. In *Progress in Cryptology - AFRICACRYPT 2008* (2008), S. Vaudenay, Ed., vol. 5023 of *Lecture Notes in Computer Science*, Springer, pp. 371–388. `http://eprint.iacr.org/2007/429/`. 41

[81] Naehrig, M., Niederhagen, R., and Schwabe, P. New software speed records for cryptographic pairings. In *Progress in Cryptology – LATINCRYPT 2010* (2010), M. Abdalla and P. S. Barreto, Eds., vol. 6212 of *Lecture Notes in Computer Science*, Springer, pp. 109–123. updated version: `http://cryptojedi.org/users/peter/#dclxvi`. 27, 43

[82] Neumann, P. G. Security criteria for electronic voting. In *Proc. 16th National Computer Security Conference* (Baltimore, Maryland, Sep 1993), NIST/NCSC. 92

[83] Neumann, S., Kulyk, O., Murati, L., and Volkamer, M. Towards a practical mobile application for election authorities (demo). In *4th International Conference on e-Voting and Identity (VoteID13)* (July 2013). 96

[84] Neumann, S., Kulyk, O., and Volkamer, M. A usable android application implementing distributed cryptography for election authorities. In *2014 Ninth International Conference on Availability, Reliability and Security* (Sept 2014), pp. 207–216. 96

[85] of State Legislatures, N. C. Voting equipment. `http://www.ncsl.org/research/elections-and-campaigns/voting-equipment.aspx`, 2016. 91

[86] Ohgishi, K., Sakai, R., and Kasahara, M. Notes on ID-based key sharing systems over elliptic curve (in Japanese). Tech. Rep. ISEC99-57, IEICE, 1999. 27

[87] Osvik, D. A., Shamir, A., and Tromer, E. Cache attacks and countermeasures: the case of AES. In *Topics in Cryptology – CT-RSA 2006* (2006), D. Pointcheval, Ed., vol. 3860 of *Lecture Notes in Computer Science*, Springer, pp. 1–20. `http://eprint.iacr.org/2005/271/`. 30

[88] Paar, C., and Pelzl, J. *Understanding Cryptography*, 1 ed. Springer-Verlag Berlin Heidelberg, 2010. 9, 15

[89] Panayiotis, A., Vassilis, T., and Nickolaos, V. Mobile based e-democracy: The ibb/pma e-voting system. In *11th Panhellenic Conference on Informatics (PCI 2007)*, T. Papatheodorou, Ed. 2007. 95

[90] Parno, B., Gentry, C., Howell, J., and Raykova, M. Pinocchio: Nearly practical verifiable computation. In *Proceedings of the IEEE Symposium on Security and Privacy* (2013), IEEE, Ed. `http://eprint.iacr.org/2013/279`. 28

[91] Pedersen, T. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO '91*, J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1992, pp. 129–140. 7, 24, 25

[92] Pereira, G. C., Jr, M. A. S., Naehrig, M., and Barreto, P. S. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software 84*, 8 (2011), 1319–1326. `http://cryptojedi.org/papers/#fast-bn`. 39

[93] Rao B., U., and K. A., A. An id-based blind signature scheme from bilinear pairings. In *International Journal of Computer Science and Security Volume (4) Issue (1)* (march 2010), pp. 98–106. 73, 74, 79, 80

[94] Ribarski, P. Модел на систем за масовно електронско гласање. Магистерска теза, Природно математички факултет - Скопје, 2011. 2

[95] Ribarski, P., and Antovski, L. Comparison of id-based blind signatures from pairings for e-voting protocols. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (May 2014), pp. 1394–1399. 87, 124

[96] Ribarski, P., and Antovski, L. Distributed private key generator for id-based public key infrastructure. In *ICT-ACT Proceedings 2016* (2016), G. Stojanov and A. Kulakov, Eds., Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg. 57, 124

[97] Ribarski, P., and Antovski, L. mvoting on android (to be published). 121, 125

[98] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21*, 2 (Feb. 1978), 120–126. 14

[99] Safevote. Election requirements, 2016. 92

[100] Sahai, A., and Waters, B. Fuzzy identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2005* (2005), R. Cramer, Ed., vol. 3494 of *Lecture Notes in Computer Science*, Springer, pp. 457–473. http://eprint.iacr.org/2004/086/. 27

[101] Sakai, R., O., K., and Kasahara, M. Cryptosystems based on pairings. In *Symposium on Cryptography and Information Security (SCIS)* (2000). 55

[102] Sakai, R., Ohgishi, K., and Kasahara, M. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan* (2000), pp. 135–148. 27

[103] Sakai, R., Ohgishi, K., and Kasahara, M. Cryptosystems based on pairing over elliptic curve (in Japanese). In *The 2001 Symposium on Cryptography and Information Security, Oiso, Japan* (2001), pp. 23–26. 27

[104] Sánchez, A. H., and Rodríguez-Henríquez, F. NEON implementation of an attribute-based encryption scheme. In *Applied Cryptography and Network Security* (2013), M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds., vol. 7954 of *Lecture Notes in Computer Science*, Springer, pp. 322–338. http://cacr.uwaterloo.ca/techreports/2013/cacr2013-07.pdf. 27, 28

[105] Scott, M. On the efficient implementation of pairing-based protocols. In *Cryptography and Coding* (2011), L. Chen, Ed., vol. 7089 of *Lecture Notes in Computer Science*, Springer, pp. 296–308. http://eprint.iacr.org/2011/334/. 28

[106] Scott, M., and Barreto, P. S. Compressed pairings. In *Advances in Cryptology – CRYPTO 2004* (2004), M. K. Franklin, Ed., vol. 3152 of *Lecture Notes in Computer Science*, Springer, pp. 140–156. 41

[107] Scott, M., Benger, N., Charlemagne, M., Perez, L. J. D., and Kachisa, E. J. On the final exponentiation for calculating pairings on ordinary elliptic curves. In *Pairing-Based Cryptography – Pairing 2009*

(2009), H. Shacham and B. Waters, Eds., vol. 5671 of *Lecture Notes in Computer Science*, Springer, pp. 78–88. eprint.iacr.org/2008/490/. 27

[108] Shallue, A., and van de Woestijne, C. E. Construction of rational points on elliptic curves over finite fields. In *Proceedings of the 7th international conference on Algorithmic Number Theory* (Berlin, Heidelberg, 2006), ANTS'06, Springer-Verlag, pp. 510–524. 42

[109] Shamir, A. How to share a secret. *Commun. ACM 22*, 11 (Nov. 1979), 612–613. 7, 19, 20, 22

[110] Shamir, A. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology*, G. Blakley and D. Chaum, Eds., vol. 196 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1985, pp. 47–53. 54, 55

[111] Stam, M., and Lenstra, A. K. Efficient subgroup exponentiation in quadratic and sixth degree extensions. In *Cryptographic Hardware and Embedded Systems – CHES 2002* (2003), B. S. K. Jr., Çetin Kaya Koç, and C. Paar, Eds., vol. 2523 of *LNCS*, Springer, pp. 318–332. 43

[112] Stinson, D. *Cryptography: Theory and Practice,Second Edition*, 2nd ed. CRC/C&H, 2002. 9

[113] Telegraph, T. The unsung genius who secured britain's computer defences and paved the way for safe online shopping, 2016. 14

[114] Teruya, T., Saito, K., Kanayama, N., Kawahara, Y., Kobayashi, T., and Okamoto, E. Constructing symmetric pairings over supersingular elliptic curves with embedding degree three. In *Pairing-Based Cryptography – Pairing 2013* (2013), Z. Cao and F. Zhang, Eds., Lecture Notes in Computer Science, Springer. 29

[115] Tromer, E., Osvik, D. A., and Shamir, A. Efficient cache attacks on AES, and countermeasures. *Journal of Cryptology 23*, 1 (2010), 37–71. http://people.csail.mit.edu/tromer/papers/cache-joc-official.pdf. 30

[116] Vercauteren, F. Optimal pairings. *IEEE Transactions on Information Theory 56*, 1 (2010). http://www.cosic.esat.kuleuven.be/publications/article-1039.pdf. 27

[117] Wagner, D. A generalized birthday problem. In *Proceedings of the 22Nd Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 2002), CRYPTO '02, Springer-Verlag, pp. 288–303. 75

[118] Washington, L. C. *Elliptic Curves: Number Theory and Cryptography, Second Edition*, 2 ed. Chapman & Hall/CRC, 2008. 9, 15

[119] Yarom, Y., and Falkner, K. Flush+reload: a high resolution, low noise, L3 cache side-channel attack. Cryptology ePrint Archive, Report 2013/448, 2013. http://eprint.iacr.org/2013/448/. 30

[120] Z. Grulovic, M. *Osnovi teorije grupa*, 1 ed. Institut za matematiku u Novom Sadu, 1997. 9

[121] ZDNet. Gchq pioneers on birth of public key crypto, 2016. 14

[122] Zhang, F., and Kim, K. Id-based blind signature and ring signature from pairings. In *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology* (London, UK, UK, 2002), ASIACRYPT '02, Springer-Verlag, pp. 533–547. 73, 74, 75, 80

[123] Zhang, L., Hu, Y., Tian, X., and Yang, Y. Novel identity-based blind signature for electronic voting system. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on* (March 2010), vol. 2, pp. 122–125. 73

[124] Zhang, X., and Wang, K. Fast symmetric pairing revisited. In *Pairing-Based Cryptography – Pairing 2013* (2013), Z. Cao and F. Zhang, Eds., Lecture Notes in Computer Science, Springer. 29