УНИВЕРЗИТЕТ У БЕОГРАДУ

ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Жарко С. Станисављевић

ВИЗУЕЛНА РЕПРЕЗЕНТАЦИЈА КРИПТОГРАФСКИХ АЛГОРИТАМА

докторска дисертација

Београд, 2014

UNIVERSITY OF BELGRADE

SCHOOL OF ELECTRICAL ENGINEERING

Žarko S. Stanisavljević

VISUAL REPRESENTATION OF CRYPTOGRAPHIC ALGORITHMS

Doctoral Dissertation

Belgrade, 2014

Ментор:
др Зоран Јовановић, редовни професор
Универзитет у Београду, Електротехнички факултет

Чланови комисије:
др Зоран Јовановић, редовни професор
Универзитет у Београду, Електротехнички факултет

др Павле Вулетић, доцент
Универзитет у Београду, Електротехнички факултет

др Душан Старчевић, редовни професор
Универзитет у Београду, Факултет организационих наука

др Бошко Николић, ванредни професор
Универзитет у Београду, Електротехнички факултет

др Славко Гајин, доцент
Универзитет у Београду, Електротехнички факултет

Датум одбране:

Изјава захвалности

Желео бих да се захвалим ментору, проф. др Зорану Јовановићу, на безрезервној подршци коју ми је пружио како приликом спровођења истраживања описаног у овом раду, тако и у читавој досадашњој научној и стручној каријери.

Захваљујем се и свим професорима са катедре за рачунарску технику и информатику Електротехничког факултета у Београду од којих сам у претходних једанаест година много тога научио и од којих сваки дан може да се научи нешто ново.

Захваљујем се и својим драгим колегама асистентима и сарадницима са катедре за рачунарску технику и информатику Електротехничког факултета у Београду са којима већ шест година делим и добре и лоше стране нашег посла.

Желео бих да се захвалим својој супрузи Јелени, која ме инспирише да свакога дана у сваком погледу будем све бољи. Поред тога што ми је пружила подршку, имала доста стрпљења за мој рад и преузела на себе већи део посла око одгајања наше кћерке, чиме ми је омогућила да овај рад напишем, помогла ми је и приликом имплементације система. Захвалност дугујем и кћерки Наталији која је у мој живот унела нову радост и оптимизам и помогла ми да напоран посао око писања рада буде много лакши.

Захвалио бих се и својим родитељима, Славку и Марини, као и брату Милошу, који су читав живот увек ту за мене да ми пруже подршку када је то потребно и да са мном поделе радост. Без њихове подршке и разумевања данас не бих био ту где сам.

На крају бих се захвалио и својој родбини и пријатељима на подршци и на разумевању за то што смо се много ређе него иначе дружили током периода израде ове дисертације.

Наслов докторске дисертације: Визуелна репрезентација криптографских алгоритама

Резиме: У овој дисертацији је описан нови систем за визуелну репрезентацију криптографских алгоритама (COALA). Главни циљ развијеног система је да се помогне студентима који су нови у области заштите података да лакше и боље савладају градиво из области криптографских алгоритама које представља основу на коју се надовезују сложеније теме. На Електротехничком факултету у Београду постоји курс под називом Заштита података на коме се најпре изучавају криптографски алгоритми на којима се заснивају сигурносни протоколи и апликације које служе да обезбеде сигурносне сервисе, а након тога се изучавају апликације које обезбеђују мрежну сигурност, док се на крају улази у проблематику сигурности рачунарских система (хакери, вируси, итд.). Праћењем резултата студената на испиту примећено је да студенти имају проблема да савладају градиво из области криптографских алгоритама, што се касније негативно одражава на њихов свеукупан успех на испиту. Анализом разлога који су довели до оваквог стања дошло се до закључка да би увођење система за визуелну репрезентацију криптографских алгоритама у оквиру лабораторијских вежби на предмету могло бити једно од решења проблема. Коришћење оваквих система је уобичајено на предметима из области рачунарске технике и информатике који изучавају софтвер. Пре развоја новог система, направљена је анализа постојећих решења како би се утврдило да ли већ постоји систем који би испунио потребе предмета и како би се установило које су то карактеристике које би тај систем требао да има да би се успешно користио у настави. Установљено је да иако постоји велики број система за визуелну репрезентацију алгоритама, релативно мали број њих омогућава визуелну репрезентацију криптографских алгоритама, док међу анализираним системима не постоји систем за визуелну репрезентацију криптографских алгоритама који у потпуности задовољава потребе предмета Заштита података на Електротехничком факултету у Београду. На основу систематизације постојећих система за визуелну репрезентацију алгоритама са посебним освртом на системе који се могу користити за визуелну репрезентацију криптографских алгоритама дефинисана је методологија развоја

оваквих система. Анализом литературе која се бави системима који се користе за помоћ у учењу дефинисан је начин коришћења развијеног система у настави. Реализовани систем омогућава праћење извршавања свих криптографских алгоритама који су објашњени на часовима предавања и вежби, приказује све детаље извршавања за сваки од подржаних алгоритама, за поједине алгоритме где је то од важности систем ради са реалним величинама параметара алгоритма које се користе у пракси и сви параметри алгоритама су конфигурабилни како би могли лако и брзо да се прикажу различити примери извршавања алгоритама.

У оквиру дисертације направљен је преглед области система за помоћ у учењу, преглед области визуелне репрезентације алгоритама, као и преглед области визуелне репрезентације криптографских алгоритама. Дат је опис криптографских алгоритама који су подржани у COALA систему. Систем има могућност визуелне репрезентације пет врста криптографских алгоритама: супституционих алгоритама (Цезар алгоритам, моноалфабетски алгоритам, *Playfair* алгоритам и *Vigenere* алгоритам), транспозиционих алгоритама (*Rail Fence* алгоритам и *Row Transposition* алгоритам), продукционих алгоритама (*Rotor Machine* алгоритам), симетричних блок алгоритама (DES алгоритам и AES алгоритам) и алгоритама са јавним кључем (*Diffie-Hellman* алгоритам и *RSA* алгоритам). Посебан акценат је стављен на објашњење детаља везаних за визуелну репрезентацију алгоритама коришћену у COALA систему. Приказан је и детаљан функционални опис COALA система за визуелну репрезентацију криптографских алгоритама. У раду су објашњени и детаљи везани за софтверску имплементацију COALA система. Дат је преглед технологија коришћених за софтверску имплементацију, објашњена је структура софтверског решења и дати су детаљи направљених проширења *Java* и *Swing* API-ја, као и начин извршавања алгоритама подржаних у COALA систему. Начин коришћења реализованог система за визуелну репрезентацију криптографских алгоритама у оквиру предмета Заштита података на Електротехничком факултету у Београду описан је кроз преглед осмишљених лабораторијских вежби у оквиру којих се користи COALA систем. У школској години 2013/2014. уведена је још једна новина на предмету, а то је коришћење система *Moodle* за испитивање и оцењивање студената у оквиру лабораторијских вежби, па је и то описано. У раду је приказана и евалуација COALA система са

аспекта ефикасности приликом примене на предмету Заштита података на Електротехничком факултету у Београду. Систем се користи у настави у претходне три школске године (уведен је у наставу 2011/2012. школске године). Резултати примене су мерени за прве две године коришћења система, с обзиром да је трећа школска година у којој се систем користи још увек у току. Евалуација је подељена на три дела. Најпре су мерени ефекти увођења COALA система у наставу на основу успешности студената на предмету у првој школској години, затим су упоређени резултати коришћења из прве и друге године, а трећи део анализе обухватио је утисак студената на основу упитника који су попуњавали. Резултати евалуације су показали да су највећу корист од увођења новог система у наставу имали студенти који иначе одлажу тренутак у коме полажу испит за касније испитне рокове у току школске године како би имали више времена да се припреме за испит и који не успевају да положе испит или га положе али са ниском оценом, као и најбољи студенти којима је коришћење система помогло да побољшају своје оцене. Анализа анкете коју су студенти попуњавали је показала да је COALA систем успешно испунио основни циљ, а то је да студенти боље разумеју градиво и детаље алгоритама који су покривени овим системом.

Кључне речи: Визуелна репрезентација алгоритама, Системи за помоћ у учењу, Криптографски алгоритми, Заштита података, Едукација у области заштите података, AES алгоритам, DES алгоритам, *Diffie-Hellman* алгоритам, RSA алгоритам.

Научна област: Електротехника и рачунарство

Ужа научна област: Рачунарска техника и информатика

УДК број: 621.3

Title: Visual representation of cryptographic algorithms

Abstract: In this dissertation a novel system for visual representation of cryptographic algorithms (CryptOgraphic Algorithms visuaL simulAtion – COALA) is presented. The main goal of the developed system is to help students who are new to the field of data security to easier and better understand the material that covers cryptographic algorithms, which represents the foundation on which the more complex topics rely on. At the School of Electrical Engineering, University of Belgrade there is a course called Data Security that covers topics from cryptographic algorithms, through security protocols and applications, to system security. By following student's results on the exam it was noticed that students tend to have problems to cope with the material covering the field of cryptographic algorithms, which later had negative impact on their overall success on the exam. After analyzing the reasons that lead to the described situation, the conclusion was drawn that the introduction of a system for visual representation of cryptographic algorithms within the laboratory exercises in the course might be a solution to the problem. Usage of similar systems is quite usual at courses that teach software related areas in the field of computer engineering and information systems. Before a new system was developed an analysis of existing solutions in the field was made in order to determine whether a system that covers the needs of the Data Security course already exists and to define which characteristics should such system have in order to be successfully used as an educational aid. It was found that even though there is a large number of systems for visual representation of algorithms (AV systems), there is only a small number of systems that can be used for visual representation of cryptographic algorithm and none of these systems fully covers the needs of the Data Security course. Based on the systematization of the existing systems for visual representation of algorithms with special attention on those that can be used for visual representation of cryptographic algorithms a methodology for the development of such system is proposed. By analysis of the literature that is concerned with the eLearning tools an efficient way of using the developed system in education process is defined. The developed system enables detailed analysis of the execution of all cryptographic algorithms that are explained at the lecture classes and the problems classes. It presents all details of the execution of supported algorithms, where it is

significant for understanding the system works with real world lengths of the algorithm parameters, and all input parameters in the algorithms are configurable in order to make it possible to easily and quickly show execution of algorithms on different examples.

As a part of the dissertation, several overviews were made: an overview of the eLearning tools field, an overview of the field covering systems for visual representation of algorithms (AV systems), and an overview of the field concerned with systems for visual representation of cryptographic algorithms. The description of cryptographic algorithms that are supported in the COALA system is presented. The system supports five different types of cryptographic algorithms: substitution algorithms (Caesar, monoalphabetic, Playfair, and Vigenere algorithms), transposition algorithms (Rail Fence and Row Transposition algorithms), production algorithms (Rotor Machine), symmetric block algorithms (DES and AES), and public-key algorithms (Diffie-Hellman and RSA). Special emphasis is placed on the explanation of details about the visual representation used in the COALA system. The detailed functional description and details about software implementation of the COALA system are presented in the dissertation. Overview of the technologies that are used, the structure of the software solution, and the modifications of the standard Java API and the Swing API, and details about the implementation of the algorithm execution are given. The way in which the developed system was used within the Data Security course is described through the presentation of the designed laboratory exercises which make use of the COALA system. In the school year 2013/2014 another novelty was introduced to the course and that is the use of the Moodle platform for examination and assessment of students at the laboratory exercises and this is also explained in the dissertation. The dissertation also shows an evaluation of the COALA system from the aspect of efficiency in the application in education. The system has been in use for the past three school years (introduced in school year 2011/2012). The results of the system use at the course have been measured for the first two years, since the third year is still in progress. The evaluation is divided in three parts. In the first part the effects of the COALA system introduction to the course is measured based on the results of students on the exam. In the second part the results from two consecutive years of the system usage have been compared. In the third part the opinion of students who used the system was analyzed through a survey conducted in the second year of usage. The

evaluation results showed that the introduction of the COALA system brought benefit to all students, especially to those students who previously could not pass the exam in the first examination period of a school year and to some of the best students to improve their grades. The analysis of the survey conducted among the students who used the system showed that the COALA system successfully fulfilled the main goal which was to help students to better understand the material and details of the algorithms that are covered by the system.

Keywords: Algorithm visualization, eLearning tools, Cryptographic algorithms, Data security, Security education, AES, DES, Diffie-Hellman, RSA.

Scientific area: Electrical and Computer Engineering

Scientific subarea: Computer Engineering

UDC number: 621.3

Садржај

1. **УВОД**

У данашње време рачунари су постали саставни део свакодневног живота. Захваљујући технолошком напретку у развоју рачунара данас се велики број разноврсних уређаја може окарактерисати као рачунар. Са друге стране, развојем Интернета дошло се до мноштва нових садржаја који су доступни коришћењем поменутих уређаја. Управо овај спој довео је до тога да се данас рачунари подједнако користе у забавне, едукативне и пословне сврхе.

Имајући у виду чињеницу да се данас путем рачунара обављају многе операције које су безбедносно осетљиве једна од битних области рачунарске технике и информатике постаје заштита података. Заштита података је одувек била битна, али је раније највише коришћена у војне сврхе. Данас ова област постаје подједнако важна сваком појединцу који користи рачунар и Интернет, без обзира на сврху коришћења.

Захваљујући доласку заштите података у центар пажње, на већини факултета у свету у оквиру студија из области рачунарске технике и информатике постоје предмети који се баве овом облашћу, па тако и на Електротехничком факултету у Београду (ЕТФ). Од 2007/2008. школске године на четвртој години основних студија одсека за софтверско инжењерство уведен је обавезни предмет под називом Заштита података. Од 2008/2009. школске године овај предмет је уведен и на четвртој години основних студија као и на мастер студијама одсека за рачунарску технику и информатику као изборни предмет. Циљ овог предмета је упознавање студената са механизмима који се примењују у области заштите података, апликацијама које се користе како би се обезбедили сигурносни сервиси и начинима примене тих апликација. Фонд часова предмета је два часа предавања, два часа вежби на табли и један час лабораторијских вежби недељно, с тим што је час лабораторијских вежби предвиђен само за студенте основних студија. Литература која се користи на предмету састоји се из књиге на енглеском језику [1] и материјала на српском језику [2]. Градиво које се изучава на предмету је веома обимно, с обзиром на то да студенти немају предзнање из ове области, а да су концепти веома комплексни. У оквиру предмета између осталог изучавају се и

криптографски алгоритми и то: симетрични и асиметрични.

Код симетричних алгоритама прво се раде класични криптографски алгоритми, који су једноставнији, али погодни за демонстрацију концепата објашњених на предавањима. Поред тога, од класичних криптографских алгоритама се полази како би студентима било јасно и због чега ови алгоритми нису могли да се користе у савременим применама. Затим се изучавају два савремена симетрична блок алгоритма *Data Encryption Standard* (DES) и *Advanced Encryption Standard* (AES). DES алгоритам је важан због тога што је то први модерни симетрични блок алгоритам који је усвојен као стандард и коришћен у комерцијалним применама у временском раздобљу од 20 година (почевши од 1977. године). Због тога је овај алгоритам погодан да се покажу принципи дизајна симетричних блок алгоритама, као и структура на којој је заснован (*Feistel* структура алгоритма), али и недостаци који су довели до тога да алгоритам у једном тренутку постане небезбедан. AES алгоритам је симетрични блок алгоритам који је данас у употреби у свим применама, па је због тога важно да студенти науче и како функционише овај алгоритам.

Код асиметричних алгоритама (односно алгоритама са јавним кључем) изучавају се *Diffie-Hellman* и RSA (*Rivest Shamir Adleman*) алгоритми. *Diffie-Hellman* алгоритам је значајан из разлога што је то први алгоритам који је икада смишљен из ове групе алгоритама и објављен заједно са концептом асиметричних алгоритама (1976. године). Овај алгоритам није практично користити за шифровање, већ само за сигурну размену заједничке тајне вредности између два корисника. RSA алгоритам је први алгоритам из ове групе алгоритама који има могућност шифровања (смишљен 1977. године). Овај алгоритам је и данас у употреби, с обзиром да до сада нису пронађени разлози који би довели до замене овог алгоритма неким новим.

Пратећи резултате студената на испиту примећено је да студенти имају проблема да савладају градиво из области криптографских алгоритама, што се касније негативно одражавало на њихов свеукупан успех на испиту. Постоји неколико разлога који су могли утицати на овакав негативан тренд. Први разлог је тај што су поједини криптографски алгоритми који су објашњени на предмету веома сложени (у питању су реални алгоритми у употреби) и због тога не постоји

могућност да се на предавањима или вежбама на табли прикаже извршавање ових алгоритама. Други разлог излази из опсега овог истраживања и захтевао би посебну социолошку анализу, а укључује чињеницу да се све већи број студената запошљава и пре него што заврши студије, па самим тим резултати на испитима постају лошији.

Оно што је уобичајено на предметима из области рачунарске технике и информатике је да се у оквиру лабораторијских вежби користе различити системи за помоћ у учењу. На предметима који су више оријентисани ка изучавању хардвера такви системи су често софтверски симулатори [3], [4], [5], [6], чији је задатак да симулирају хардвер који се изучава. На тај начин омогућено је студентима да стекну практично знање у раду са симулатором хардвера, а са друге стране није потребно направити велику инвестицију у погледу набавке потребне опреме. На предметима који изучавају софтвер често се користе системи за визуелну репрезентацију алгоритама [7], [8], [9]. Област пројектовања система за визуелну репрезентацију алгоритама има дугу традицију и велики број оваквих система је до сада реализован. Са растом популарности заштите података појављује се све више система за визуелну репрезентацију криптографских алгоритама.

Једно од потенцијалних решења проблема који је уочен на предмету заштита података могло би да буде увођење система за визуелну репрезентацију криптографских алгоритама у оквиру лабораторијских вежби на предмету. Овакав систем би требао да омогући праћење извршавања свих криптографских алгоритама који су објашњени на часовима предавања и вежби, а морао би да укључи све сложеније алгоритме за које није могуће другачије приказати како се извршавају. Систем би морао да приказује све детаље извршавања за сваки од алгоритама. За поједине алгоритме где је то од важности систем би требао да ради са реалним величинама параметара алгоритма које се користе у пракси. Сви параметри алгоритама би требали да буду конфигурабилни како би могли лако и брзо да се прикажу различити примери извршавања алгоритама.

С обзиром на чињеницу да постоји велики број система за визуелну репрезентацију алгоритама, од којих су неки и системи који омогућавају аутоматско генерисање визуелне репрезентације произвољног алгоритма, најпре

је направљен преглед постојећих система. Установљено је да иако постоји велики број система за визуелну репрезентацију алгоритама, релативно мали број њих омогућава визуелну репрезентацију криптографских алгоритама. Затим су детаљно анализирани системи који омогућавају визуелну репрезентацију криптографских алгоритама. Резултат анализе је да не постоји систем за визуелну репрезентацију криптографских алгоритама који у потпуности задовољава потребе предмета Заштита података на ЕТФ-у. На основу претходних закључака, донета је одлука да се реализује нов систем за визуелну репрезентацију криптографских алгоритама (COALA) који би у потпуности задовољио уочене потребе. На основу анализе литературе која се бави системима који се користе за помоћ у учењу осмишљен је и начин на који би се систем користио у настави у оквиру лабораторијских вежби са циљем да се студентима помогне да боље савладају градиво из ове области и тиме побољшају своје резултате на испиту.

Докторска дисертација садржи девет поглавља, скуп неопходних прилога и преглед коришћене литературе. У другом поглављу најпре је направљен преглед области система за помоћ у учењу. Потом је направљен преглед области визуелне репрезентације алгоритама, а након тога и области визуелне репрезентације криптографских алгоритама. На крају, на основу свега изложеног извучени су закључци који се могу применити приликом решавања проблема постављеног у дисертацији. У трећем поглављу је приказан опис криптографских алгоритама који су подржани у COALA систему. Систем подржава пет врста криптографских алгоритама: супституционе алгоритме, транспозиционе алгоритме, продукционе алгоритме, симетричне блок алгоритме и алгоритме са јавним кључем. У четвртом поглављу су објашњени детаљи везани за визуелну репрезентацију алгоритама коришћену у COALA систему. На основу закључака изведених у поглављу 2 произашао је дизајн коришћен у COALA систему. У петом поглављу је приказан детаљан функционални опис COALA система за визуелну репрезентацију криптографских алгоритама који је реализован. У шестом поглављу су објашњени детаљи везани за софтверску имплементацију COALA система. Прво је дат преглед технологија коришћених за софтверску имплементацију, након тога је објашњена структура софтверског решења, потом су дати детаљи неопходних проширења стандардног API-ја (*Application*

Programming Interface) која су направљена и на крају је објашњен начин извршавања алгоритама подржаних у COALA систему. У седмом поглављу је описан начин коришћења реализованог система за визуелну репрезентацију криптографских алгоритама у оквиру курса Заштита података на Електротехничком факултету у Београду. У осмом поглављу је описана евалуација COALA система са аспекта ефикасности приликом примене на курсу Заштита података на Електротехничком факултету у Београду. Систем се користи у настави у претходне три школске године (уведен је у наставу 2011/2012. школске године). У деветом поглављу даје се закључак, као критички осврт на испуњење циљева постављених на почетку овог рада, као и резиме свега урађеног.

2. ПРЕГЛЕД РЕЛЕВАНТНЕ ЛИТЕРАТУРЕ

У овом поглављу ће најпре бити направљен преглед области система за помоћ у учењу. Потом ће бити направљен преглед области визуелне репрезентације алгоритама, а након тога и области визуелне репрезентације криптографских алгоритама. На крају, на основу свега изложеног биће извучени закључци који се могу применити приликом решавања проблема постављеног у дисертацији.

2.1. ПРЕГЛЕД ОБЛАСТИ СИСТЕМА ЗА ПОМОЋ У УЧЕЊУ

Системи за помоћ у учењу (енгл. *eLearning tools*) су део савременог образовног процеса. Главни задатак ових система је да омогуће онима који их користе да лакше савладају градиво. У данашње време постоји велики број разноврсних система за помоћ у учењу. У зависности од области примене и нивоа потребног знања за коришћење система разликује се и технологија примењена за реализацију система. Био би велики изазов направити свеобухватан преглед постојећих система за помоћ у учењу. У наставку ће бити приказани неки од репрезентативних система за помоћ у учењу. Ови системи су одабрани тако да покрију што више разноврсних мултимедијалних садржаја и технологија, као и што више различитих области примене.

Табела 2.1 приказује опште информације о одабраним системима за помоћ у учењу. Ове информације обухватају: назив система, област примене система, земљу настанка система, годину настанка (за системе за које је ова информација доступна) или годину публикације система (за системе за које није доступна година настанка), линк ка систему (за системе за које постоји сајт на коме су доступни) или линк ка публикацији која описује систем (за системе који немају сајт), референца. Свим линковима је последњи пут приступано априла 2014. године.

Alice tool [10] представља 3D интерактивно анимирано окружење које је развијено како би помогло студентима који уче програмирање да разумеју како да примене технике за решавање проблема у својим програмима и како да користе уобичајене програмске конструкције. *BlueJ tool* [11] је интегрисано *Java* развојно

Табела 2.1. Преглед неких од репрезентативних система за помоћ у учењу

Назив система	Област примене	Земља	Год.	Линк	Реф.
Alice tool	програмирање	САД	2000.	http://www.alice.org/	[10]
BlueJ tool	објектно оријентисано програмирање	Аустралија, Велика Британија	2003.	http://www.bluej.org	[11]
CWRU tool set	мултидисципли--нарно инжењерство	САД	1999.	http://drushel.cwru.edu/375/index.html	[12]
Digital 3D Lego tool	сигурносни протоколи	САД	2011.	http://coitweb.uncc.edu/~wwang22/research/ papers/IEEE-TLT-Wang.pdf	[13]
Digital Violin Tutor tool	свирање виолине	Сингапур	2005.	http://www.comp.nus.edu.sg/~wangye/	[14]
Explore! tool	историја	Италија	2008.	http://ivu.di.uniba.it/people/ardito /Software/Explore/index.htm	[15]
Exspot tool	посете музејима	САД	2005.	http://www.exploratorium.edu	[16]
Family Ensemble tool	свирање клавира	Јапан	2004.	http://www.vis.uky.edu/~cheung/courses/ ee639_fall04/homeworks/p556.pdf	[17]
Haptic/Aural tool	математика	САД	2011.	http://research.vuse.vanderbilt.edu/ MEDLAB/research_files/haptics.htm	[18]
HRITEH tool set	хирургија	Швајцарска	2005.	http://www.hystsim.ethz.ch/	[19]
Humanoid Robots tool set	страни језик	Тајван	2010.	http://hci.csie.ncu.edu.tw/	[20]
Hyperscore tool	компоновање музике	САД	2004.	http://hyperscore.wordpress.com	[21]
MLVLS tool set	опште намене	Кина	2010.	http://www.carstenullrich.net/ pubs/Ullrich10Mobile.pdf	[22]
MTM tool set	опште намене	САД	2003.	http://www.csc.villanova.edu/~klassner/	[23]
PRAR tool set	опште намене	Јужна Кореја	2009.	http://cgit.nutn.edu.tw:8080/cgit/ PaperDL/CUC_110118152121.PDF	[25]
Quest Atlantis tool	опште намене	САД	2005.	http://atlantisremixed.org/	[26]
Robovie tool set	страни језик	Јапан	2005.	http://www.irc.atr.jp/~kanda/	[27]
SDLDS tool	основи рачунарске технике	Србија	2013.	http://ieeexplore.ieee.org/xpls/ abs_all.jsp?arnumber=6280603	[3]
Seeing Sound tool	свирање музичких инструмената	Аустралија	2005.	http://web.arch.usyd.edu.au/~andrew/	[28]
USIM tool set	посете музеју	САД	2002.	http://www.hpl.hp.com/research/ papers/2002/ubiquitous.html	[29]

окружење са графичким корисничким интерфејсом (енгл. *Graphical User Interface* – GUI) дизајнираним за учење основних концепата објектно оријентисаног програмирања на факултетима. *CWRU tool set* [12] подразумева да студенти који уче различите области раде заједно у тимовима како би дизајнирали, направили и испрограмирали сопствене аутономне роботе помоћу којих би се такмичили против других тимова у јавном такмичењу на крају семестра у игри сакупљања пластичних јаја. *Digital 3D Lego tool* [13] је систем за електронско учење сигурносних протокола у коме су примењене педагошке методе које се користе у играчкама конструкционим сетовима, па су сигурносне примитиве представљене као *Lego* коцкице а сигурносни протоколи као структуре које се добијају

слагањем коцкица. *Digital Violin Tutor* [14] представља софтверски мултимедијални алат дизајниран за почетнике у учењу свирања виолине како би им помогао да наштимују виолину, уоче грешке у свом свирању и препознају исправно одсвирану композицију. *Explore! tool* [15] представља мултимедијалну игру која укључује коришћење мобилних технологија за екскурзије у историјском парку за ученике у основној школи. *EXspot* [16] је алат базиран на коришћењу RFID технологије који има за циљ да побољша искуство корисника приликом посете музеју тако што му омогућава да сними детаље своје посете и касније их анализира код куће. *Family Ensemble tool* [17] представља вишекориснички музички систем за забаву и едукацију који помаже детету да научи да свира клавир тако што омогућава да родитељ и дете свирају у дуету при чему се извођење родитеља аутоматски коригује. *Haptic/Aural tool* [18] је софтверска апликација која је развијена да користи хаптички екран који реагује на додир како би помогла ученицима са оштећеним видом да уче математику у основној школи. *HRITEH tool set* [19] је генерички хируршки тренинг симулатор за хистероскопију који омогућава корисницима да увежбавају процедуру и базичне манипулације приликом спровођења процедуре. *Humanoid Robots tool set* [20] користи хуманоидне роботе за учење страног језика у основној школи. *Hyperscore tool* [21] је графички систем за компоновање мелодија уз помоћ рачунара намењен корисницима без претходног теоретског знања и обуке. *MLVLS tool set* [22] представља мобилни едукациони систем који омогућава студентима да уживо прате наставу на својим мобилним уређајима уз могућност интеракције са предавачима током трајања часа. *MTM tool set* [23] представља скуп решења за потешкоће које онемогућавају коришћење *Lego MindStorms* комплета за прављење робота у едукацији из области рачунарске технике и информатике [24]. *PRAR tool set* [25] је интерактивни систем за учење који користи препознавање образаца и измењену реалност (енгл. *pattern recognition and augmented reality*) како би помогао студентима приликом учења из литературе додавањем аудио и видео садржаја. *Quest Atlantis tool* [26] је 3D вишекорисничка виртуелна игра намењена деци узраста од 9-15 година за учење одређених области из школског плана и програма кроз низ активности које се делимично спроводе у виртуелном, а делимично у стварном свету. *Robovie tool set* [27] представља коришћење

интерактивног хуманоидног робота под називом *Robovie* као тутора за учење страног језика у основној школи. *SDLDS tool* [3] је софтверски систем развијен као подршка предметима из области основа рачунарске технике који омогућава студентима боље разумевање кроз самостално коришћење код куће и кроз лабораторијске вежбе, а наставницима помаже кроз аутоматско оцењивање и верификацију радова студената. *Seeing Sound* [28] је систем за визуелну репрезентацију звука у реалном времену који помаже музичарима приликом самосталног тренинга и замењује потребу за присуством инструктора. *USIM tool set* [29] се састоји из скупа помоћних алата који су развијени да побољшају посете модерним интерактивним музејима на сличан начин као што су аудио водичи то урадили у класичним музејима.

Пројектовање система за помоћ у учењу налази се на пресеку природних и друштвених наука. Свакако је потребно имплементирати одређени систем за шта је потребно познавање једне или више инжењерских области, али је такође потребно прво осмислити систем и то тако да буде ефикасан када се буде користио као помоћ у учењу чиме се баве друштвене науке. Постоји велики број публикација о пројектовању система за помоћ у учењу посматрано из аспекта друштвених наука [30], [31], [32], [33], [34], [35], [36], [37]. Главни закључак већине оваквих студија је да је начин коришћења система често важнији од техничких карактеристика система по питању ефикасности система као помоћи у учењу. У [37] дефинисани су принципи дизајна за коришћење мултимедијалних садржаја у системима који се користе као помоћ у учењу. Ови принципи дизајна се базирају на анализи ефеката мултимедије у едукацији. У [37] наведено је пет принципа дизајна: вишеструка репрезентација (енгл. *multiple representation*), суседност (енгл. *contiguity*), подељена пажња (енгл. *split-attention*), сажетост (енгл. *coherence*) и индивидуалне разлике (енгл. *individual differences*). Принцип вишеструке репрезентације подразумева да је боље приказати неку информацију коришћењем две или више различитих репрезентација, него само једном. Принцип суседности предлаже да када се информација презентује коришћењем више различитих репрезентација те репрезентације би требало да буду приказане истовремено. Принцип подељене пажње означава да када се више различитих репрезентација користе за презентовање исте информације различите репрезента–

Табела 2.2. Преглед испуњености принципа дизајна у одабраним системима за помоћ у учењу

Назив система	Вишеструка репрезентација	Суседност	Подељена пажња	Сажетост
Alice tool	+	+	-	+
BlueJ tool	+	-	-	+
CWRU tool set	+	-	-	+
Digital 3D Lego tool	+	+	-	+
Digital Violin Tutor tool	+	+	+	-
Explore! tool	+	+	+	+
Exspot tool	+	+	+	-
Family Ensemble tool	+	+	+	+
Haptic/Aural tool	+	+	+	+
HRITEH tool set	+	+	+	+
Humanoid Robots tool set	+	+	+	+
Hyperscore tool	+	+	+	+
MLVLS tool set	+	+	+	-
MTM tool set	+	-	-	+
PRAR tool set	+	+	+	-
Quest Atlantis tool	+	+	-	-
Robovie tool set	+	+	+	+
SDLDS tool	+	+	-	+
Seeing Sound tool	+	+	+	+
USIM tool set	+	+	+	-

ције би требало да утичу на различита чула. Принцип сажетости сугерише да би приликом мултимедијалне презентације неке информације само најважнији делови требали бити приказани. Принцип индивидуалних разлика објашњава да ефекти система за помоћ у учењу нису исти за све кориснике, па би систем требало прилагодити циљној групи корисника.

Табела 2.2 даје преглед испуњености принципа дизајна дефинисаних у [37] у системима за помоћ у учењу наведеним у табели 2.1. За сваки систем постоји ред у табели који садржи бинарну информацију о томе да ли систем испуњава одређени принцип дизајна. Принцип индивидуалних разлика се не налази у табели, јер испуњеност овог принципа није било могуће одредити на основу расположивих података о системима.

У табели 2.2 принцип вишеструке репрезентације је испуњен уколико систем

користи барем два различита типа садржаја. Из табеле се види да сви системи испуњавају принцип вишеструке репрезентације. Принцип суседности је испуњен уколико се различити типови садржаја користе симултано у систему за помоћ у учењу. Принцип подељене пажње је испуњен уколико систем за помоћ у учењу ангажује више од једног чула корисника у једном тренутку. Принцип сажетости је испуњен уколико су у систему за помоћ у учењу презентоване само неопходне информације. Анализом табеле се може видети да већина одабраних система за помоћ у учењу испуњава ове принципе дизајна у великој мери. Системи који не испуњавају ове принципе углавном имају неку специфичну намену која не захтева да сви принципи буду испуњени.

2.2. ПРЕГЛЕД ПОСТОЈЕЋИХ СИСТЕМА ЗА ВИЗУЕЛНУ РЕПРЕЗЕНТАЦИЈУ АЛГОРИТАМА

Област визуелне репрезентације алгоритама је актуелна већ дуги низ година. За то време развијено је и коришћено много разноврсних система са различитим наменама. Велики је изазов направити свеобухватан преглед постојећих система за визуелну репрезентацију алгоритама. С обзиром да област постоји већ довољно дуго написано је и неколико релевантних прегледних радова из ове области са различитим фокусом истраживања. У наставку овог поглавља најпре ће бити презентовани релевантни прегледни радови и закључци изведени на основу спроведених истраживања у тим радовима. Након тога биће направљен преглед одабраних система за визуелну репрезентацију алгоритама према таксономијама предложеним у прегледним радовима, али и на основу карактеристика система битних са аспекта визуелизације.

У раду [38] аутори се баве прегледом система за визуелну репрезентацију програма и алгоритама који су успешно коришћени у настави. Истраживање је обухватило 18 оваквих система. Фокус овог прегледа је на употребљивости система за визуелну репрезентацију алгоритама као помоћних средстава у настави. Идеја је да се на основу прегледа система за које је објављено да су били од користи у настави уочи које су то важне карактеристике оваквих система које су допринеле постизању таквог резултата. Класификација система заснована је на два критеријума: нивоу апстракције и начину имплементације. Према нивоу апстракције разликују се две врсте система (према таксономији дефинисаној у

[39]): системи за визуелну репрезентацију алгоритама (енгл. algorithm vizualizations) и системи за визуелну репрезентацију програма (енгл. program vizualizations). Прва група обухвата системе који омогућавају статичку или динамичку визуелну репрезентацију алгоритма, док друга група обухвата системе који омогућавају статичку или динамичку визуелну репрезентацију извршавања програма. Према начину имплементације разликују се три врсте система: системи који користе скрипте (енгл. script-based systems), системи који користе интерфејс (енгл. interface systems) и системи који користе преводилац (енгл. compiler-based systems). Прва група омогућава кориснику да сам бира које делове програма (алгоритма) жели да визуелизује тако што му омогућава да у коду дода позиве делу за визуелизацију, друга група нуди кориснику интерфејс за унапред дефинисану визуелизацију, док трећа група аутоматски обавља визуелизацију програма (алгоритма) који корисник пише. У обухваћеним системима испитиване су две битне карактеристике: употребљивост система и успешност коришћења система у настави. За евалуацију употребљивости система коришћене су неке од методологија дефинисане у [40]: неформална испитивања (где студенти дају своје мишљење о систему), хеуристичка испитивања (где стручњак проверава карактеристике система), технике упитника (где студенти попуњавају упитнике о систему), опсервационе студије (посматра се на који начин студенти користе систем) и контролисани експерименти (осим мишљења студената могуће је проверити и ефикасност система). За евалуацију успешности коришћења система у настави коришћен је закључак из [41] који тврди да је за ефикасност система за визуелну репрезентацију алгоритама битније на који начин га студенти користе, него шта је и како визуелно приказано. Коришћена је таксономија дефинисана у [42] која описује који су то начини интеракције приликом коришћења система за визуелну репрезентацију алгоритама: без визуелизације (енгл. no viewing), само посматрање визуелизације (енгл. viewing), одговарање на питања у вези визуелизације (енгл. responding), конфигурисање визуелизације (енгл. changing), конструисање визуелизације (енгл. constructing), приказивање визуелизације (енгл. presenting). Први ниво подразумева ситуацију у којој не постоји систем за визуелну репрезентацију алгоритама (почетно стање), други ниво подразумева постојање система за визуелну репрезентацију алгоритама у коме студенти могу

да посматрају и контролишу ток визуелизације (сви наредни нивои обухватају и овај ниво), трећи ниво подразумева да студенти одговарају на питања у вези посматране визуелизације, код четвртог нивоа студентима је омогућено да модификују визуелизацију (нпр. конфигуришу улазне параметре), пети ниво дозвољава студентима да сами конструишу визуелну репрезентацију алгоритма и шести ниво подразумева да се од студената очекује да презентују визуелизацију пред публиком. Један од закључака овог прегледа је да би употребљивост система требало проверавати комбиновањем различитих метода у процесу развоја система, али већина анализираних система није користила такав начин провере. Код већине система провера употребљивости није довела до закључака који би омогућили побољшање система. Други закључак је да системи за визуелну репрезентацију алгоритама могу бити ефикасни као помоћно средство у едукацији. Трећи закључак је да промена са првог нивоа интеракције на било који наредни ниво интеракције доноси побољшање у учењу код студената.

Мета студија ефикасности визуелне репрезентације алгоритама [41] обухвата анализу 24 експерименталних истраживања система за визуелну репрезентацију алгоритама како би се добио одговор на питање да ли су овакви системи корисни у едукацији. Једна од класификација система за визуелну репрезентацију алгоритама у оквиру ове студије је према начинима коришћења у едукацији: на предавањима (енгл. lectures – користе их наставници на часовима да лакше објасне алгоритме), за задатке (енгл. assignments – користе их студенти у оквиру домаћих задатака), за дискусију (енгл. class discussion – студенти презентују домаће задатке и дискутују о њима на часу), у лабораторији (енгл. laboratories – користе их студенти на лабораторијским вежбама), за учење (енгл. study – користе их студенти за самостално учење), за консултације (енгл. office hours – користе их наставници као помоћ на консултацијама) и за тестове (енгл. tests – користе се као део испита). Друга класификација система за визуелну репрезентацију алгоритама у оквиру ове студије је према техникама евалуације ефикасности ових система: анегдотске технике (енгл. anecdotal techniques – приказују карактеристичне сценарије извршавања алгоритма), програмске технике (енгл. programmatic techniques – користе програме који се користе да се направи визуелна репрезентација као основ за оцењивање ефикасности), аналитичке технике (енгл.

analytic evaluation techniques – код ових техника процена ефикасности се своди на употребљивост система) и емпиријске технике (енгл. empirical evaluation – на основу коришћења система изводе се закључци о ефикасности). У оквиру студије анализиране су независне и зависне променљиве у оцени ефикасности сваког од система. По питању независних променљивих уочено је да су четири различите теорије учења коришћене као основа приликом развоја и употребе анализираних система: *Epistemic Fidelity* [43] (где је претпоставка да људи памте симболичке моделе физичког света и да је некоме лакше да научи ако му се директно прикаже симболички модел), *Dual-coding* [44] (заснива се на теорији да постоје два функционално независна међусобно повезана система симболичког означавања приликом учења, један за кодирање вербалних догађаја и други за кодирање невербалних; идеја је да се визуелизација направи тако да обухвати и једно и друго), *Individual Differences* [45] (теорија која објашњава да постоје разлике у учењу код различитих појединаца; погодна је за мерење ефикасности система) и *Cognitive Constructivism* [46] (теорија која објашњава да не постоји апсолутно знање, већ да сваки појединац конструише сопствено знање на основу искустава из живота). Када говоримо о зависним променљивама, односно начинима на који је свако за себе мерио ефикасност система, постављају се два питања: шта је мерено и на који начин? Подела анализираних система према томе шта је мерено је на: системе који мере концептуално знање (разумевање алгоритма на вишем нивоу), системе који мере процедурално знање (разумевање детаља извршавања алгоритма) и системе који мере и концептуално и процедурално знање. Подела анализираних система према томе на који начин је мерено је на: тест након коришћења (енгл. post-test – где се мери резултат студената на тесту који раде након коришћења система) и побољшање након коришћења у односу на пре коришћења (енгл. pre-test to post-test improvement – где се мери побољшање након коришћења у односу на резултат пре коришћења система). Најважнији закључак ове студије је да је са аспекта ефикасности система за визуелну репрезентацију алгоритама битније на који начин студенти користе систем, него шта систем приказује.

Преглед стања области визуелне репрезентације алгоритама на основу анализе преко 500 сакупљених система за визуелну репрезентацију алгоритама је дат у

[47]. У овом раду приказан је каталог сакупљених система који је доступан путем Интернета. Каталог поред линкова ка самим системима садржи и описе и евалуације ових система. Кључне информације које су сакупљене о системима су: област примене, информације о ауторима, начин инсталације, датум настанка, у оквиру ког пројекта је настао и језик презентације. Након евалуације сваки систем се оцењује са једном од три оцене: препоручује се (енгл. Recommended), има потенцијала (енгл. Has Potential) или не препоручује се (енгл. Not Recommended). Поред категоризације системима се додељује и одредница за коју примену су погодни и то може бити једна или више од следећих седам примена: објашњавање концепта (енгл. Teaching the Concept – наставник користи систем у оквиру часа), истраживање концепта (енгл. Exploring the Concept – студенти сами истражују користећи систем), исправљање грешака (енгл. Debugging – студенти користе систем да визуелно прате извршавање програма који напишу), поређење (енгл. Comparison – приказивање сличности и разлика различитих алгоритама из истог домена), помоћ на предавању (енгл. Lecture Aid – користи се као помоћно средство приликом објашњавања неког алгоритма), лабораторијске вежбе (енгл. Lab Exercise – користи се као део лабораторијских вежби), самостално учење (енгл. Self Study – користи се уз одговарајући уџбеник за самостално учење студената). Постоји и поље ниво интеракције које описује колики је степен интеракције у сваком систему. Закључци на основу анализе сакупљених алата могу да дају увид у тренутно стање области визуелне репрезентације алгоритама. До марта 2010. године постојало је преко 500 система за визуелну репрезентацију алгоритама. Скоро сви системи настали од средине 90-их година до данас су развијени у програмском језику Java, а преко две трећине ових система може се користити путем Интернета. Већина система покрива теме из области алгоритама и структура података. Трећина система је развијена као самостални пројекат њихових аутора, мањи проценат су развијале групе аутора са неколико развијених система док је више од половине система развијано од стране група аутора који су направили више од 10 система. Сви сакупљени системи су бесплатно доступни за коришћење у едукацији, док више од половине на располагање ставља и изворни програмски код. Закључак је да иако постоји велики број расположивих система за визуелну репрезентацију алгоритама, постоји и потреба за новим.

Табела 2.3. Приказ општих информација о одабраним системима за визуелну репрезентацију алгоритама

Назив система	Област примене	Земља	Год.	Линк	Реф.
Animal	алгоритми и структуре података	Немачка	2000.	http://www.algoanim.info/AnimalAV/	[48]
Alvis	програмирање	САД	1998.	http://helplab.org/projects/alvis	[49]
PathFinder	математика	Шпанија	2009.	http://www.sciencedirect.com/science/article/pii/S1571066108005227	[50]
Dave	алгоритми и структуре података, програмирање	Грчка	2008.	http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4561697&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4561697	[51]
Jvall	програмирање	САД	2002.	http://dl.acm.org/citation.cfm?id=563362	[52]
Visage	математика, графови	Немачка	2005.	http://link.springer.com/article/10.1007/s11858-005-0027-z#	[53]
Jeliot	програмирање	Финска, Израел	2004.	http://cs.joensuu.fi/~jeliot/	[54]
Jhave	алгоритми и структуре података	САД	2005.	http://jhave.org/	[55]
MatrixPro	алгоритми и структуре података	Финска	2004.	http://www.cse.hut.fi/en/research/SVG/MatrixPro/	[56]
Raptor	програмирање	САД	2006.	http://raptor.martincarlisle.com/	[57]
Javenga	рачунарске мреже, графови	Грчка	2009.	http://onlinelibrary.wiley.com/doi/10.1002/cae.20392/abstract;jsessionid=3405C0A4BB3B667A5E8610801FA5A65A.f02t03	[58]
Jawaa	програмирање, алгоритми и структуре података	САД	2002.	https://www.cs.duke.edu/csed/jawaa2/	[59]

Табела 2.3 приказује опште информације о одабраним системима за визуелну репрезентацију алгоритама. Ове информације обухватају: назив система, област примене система, земљу настанка система, годину настанка (за системе за које је ова информација доступна) или годину публикације система (за системе за које није доступна година настанка), линк ка систему (за системе за које постоји сајт на коме су доступни) или линк ка публикацији која описује систем (за системе који немају сајт), референца. Свим линковима је последњи пут приступано априла 2014. године.

Класификација одабраних система за визуелну репрезентацију алгоритама према таксономији предложеној у [38] дата је у табели 2.4. Табела садржи: назив система, ниво апстракције, начин имплементације, технике испитивања употребљивости и начин интеракције. Поља попуњена вредношћу „-“ означавају да одређена информација о систему није доступна.

Класификација одабраних система за визуелну репрезентацију алгоритама према таксономији предложеној у [39] дата је у табели 2.5. Табела садржи: назив система, начин коришћења система, технике евалуације ефикасности система,

Табела 2.4. Класификација одабраних система за визуелну репрезентацију алгоритама према таксономији из [38]

Назив система	**Ниво апстракције**	**Начин имплементације**	**Технике испитивања употребљивости**	**Начин интеракције**
Animal	визуелна репрезентација алгоритама	скрипт	хеуристичка испитивања	само посматрање
Alvis	визуелна репрезентација програма	преводилац	контролисани експерименти	конструисање визуелизације
PathFinder	визуелна репрезентација алгоритама	интерфејс	-	-
Dave	визуелна репрезентација програма	преводилац	технике упитника, контролисани експерименти	конфигурисање визуелизације
Jvall	визуелна репрезентација програма	преводилац	-	само посматрање
Visage	визуелна репрезентација алгоритама	интерфејс	неформална испитивања	само посматрање
Jeliot	визуелна репрезентација програма	преводилац	опсервационе студије, контролисани експерименти	конфигурисање визуелизације
Jhave	визуелна репрезентација алгоритама	интерфејс	неформална испитивања, опсервационе студије	одговарање на питања
MatrixPro	визуелна репрезентација алгоритама	интерфејс	-	-
Raptor	визуелна репрезентација програма	преводилац	технике упитника	одговарање на питања
Javenga	визуелна репрезентација алгоритама	интерфејс	-	одговарање на питања
Jawaa	визуелна репрезентација алгоритама	скрипт	неформална испитивања	само посматрање

теорију учења на којој је систем заснован, тип мереног знања и начин мерења знања. Поље начин коришћења попуњен је на основу предлога аутора система на који начин би њихов систем могао да се користи или је коришћен. Поље технике евалуације ефикасности попуњено је само за системе који су вршили евалуацију ефикасности, док за остале има вредност „-“. Поље коришћена теорија учења попуњено је на основу описа примене система и начина на који би требало да буде од користи студентима и то за системе у којима је то објашњено, док је за остале вредност овог поља „-“. Поља тип мереног знања и начин мерења знања попуњавана су по истом принципу као и поље технике евалуације ефикасности.

Табела 2.6 даје приказ карактеристика одабраних система за визуелну репрезентацију алгоритама битних са аспекта визуелизације. Табела садржи:

Табела 2.5. Класификација одабраних система за визуелну репрезентацију алгоритама према таксономији из [39]

Назив система	**Начин коришћења**	**Технике евалуације ефикасности**	**Коришћена теорија учења**	**Тип мереног знања**	**Начин мерења знања**
Animal	на предавањима	-	Epistemic Fidelity	-	-
Alvis	у лабораторији, за учење	-	Cognitive Constructivism	-	-
PathFinder	на предавањима, за учење	-	-	-	-
Dave	у лабораторији	-	Cognitive Constructivism	-	-
Jvall	на предавањима, у лабораторији, на часовима	-	Epistemic Fidelity	-	-
Visage	за учење	-	Epistemic Fidelity	-	-
Jeliot	у лабораторији, за учење	-	Cognitive Constructivism	-	-
Jhave	за учење	-	Epistemic Fidelity	-	-
MatrixPro	на предавањима, за учење	-	-	-	-
Raptor	на предавањима, у лабораторији	емпиријске	Cognitive Constructivism	концептуално	тест након коришћења
Javenga	на предавањима, за учење	емпиријске	Epistemic Fidelity	концептуално и процедурално	тест након коришћења
Jawaa	на предавањима, за задатке	-	Epistemic Fidelity	-	-

назив система, динамичност система (да ли су у питању статичке или ди–намичке анимације), праћење извршавања (на ком нивоу је могуће праћење извршавања алгоритма: корак по корак, итерација, комплетан алгоритам), конфигурабилност улазних параметара (да ли и на ком нивоу је могуће конфигурисати улазне параметре алгоритма), флексибилност (да ли постоји могућност додавања нових алгоритама и на који начин) и дизајн визуелне репрезентације (колико визуелна репрезентација одговара приказу алгоритама коришћеном у литератури).

Animal [48] представља софтверски систем за визуелизацију алгоритама који омогућава генерисање визуелне репрезентације произвољног алгоритма. Корисницима је омогућено да визуелну репрезентацију алгоритма развијају користећи графички кориснички интерфејс или користећи скрипт језик. Погодан је за коришћење на предметима из области алгоритама и структура података као помоћ наставницима на предавањима.

Alvis [49] је софтверски систем за визуелизацију алгоритама који омогућава корисницима да сами напишу програмски код алгоритма за који желе да виде визуелну репрезентацију. Систем омогућава корисницима да комбинују писање програма које истог тренутка утиче на визуелни приказ алгоритма са директним

Табела 2.6. Приказ карактеристика одабраних система за визуелну репрезентацију алгоритама које су битне са аспекта визуелизације

Назив система	**Динамичност**	**Праћење извршавања**	**Конфигурабилност улазних параметара**	**Флексибилност**	**Дизајн визуелне репрезентације**
Animal	статичка анимација	корак по корак	да, у програмском коду	GUI/Script/API за додавање нових алгоритама	произвољан
Alvis	статичка анимација	корак по корак, комплетан алгоритам	да, у програмском коду	GUI/Script за додавање нових алгоритама	произвољан
PathFinder	статичка анимација	корак по корак, итерација, комплетан алгоритам	да, коришћењем визуелног едитора графа	нема могућност додавања нових алгоритама	сагласан са литературом
Dave	статичка анимација	корак по корак	да, у програмском коду	Script за додавање нових алгоритама	произвољан
Jvall	статичка анимација	корак по корак	да, у програмском коду	API за додавање нових алгоритама	сагласан са литературом
Visage	статичка анимација	комплетан алгоритам	да, коришћењем визуелног едитора графа	нема могућност додавања нових алгоритама	сагласан са литературом
Jeliot	статичка анимација	корак по корак, комплетан алгоритам	да, у програмском коду	писањем програма у Java програмском језику	сагласан са литературом
Jhave	динамичка анимација	корак по корак	да, у програмском коду	Script за додавање нових алгоритама	произвољан
MatrixPro	статичка анимација	корак по корак, комплетан алгоритам	да, коришћењем визуелног едитора	GUI за додавање нових алгоритама	сагласан са литературом
Raptor	статичка анимација	комплетан алгоритам	да, коришћењем визуелног едитора	GUI за додавање нових алгоритама	сагласан са литературом
Javenga	статичка анимација	корак по корак, комплетан алгоритам	да, коришћењем визуелног едитора	нема могућност додавања нових алгоритама	сагласан са литературом
Jawaa	статичка анимација	комплетан алгоритам	да, у програмском коду	Script за додавање нових алгоритама	произвољан

манипулацијама над објектима у визуелној репрезентацији. Намењен је за коришћење на почетним курсевима из програмирања као помоћ студентима за самостално учење или за коришћење на лабораторијским вежбама.

PathFinder [50] представља систем за визуелну репрезентацију извршавања Dijkstra алгоритма. Корисницима је омогућено да поставе улазне параметре (визуелно исцртају граф) и затим прате извршавање алгоритма за задати граф корак по корак у контролисаном окружењу. Погодан је за коришћење на

курсевима из математике за демонстрације на предавањима и као помоћ студентима за самостално учење.

Dave [51] представља софтверски систем који омогућава аутоматско генерисање визуелне репрезентације програмског кода подржаних алгоритама. Систем омогућава корисницима визуелно праћење извршавања алгоритма корак по корак кроз програмски код који описује алгоритам, а постоји и могућност додавања нових алгоритама. Погодан је за коришћење на курсевима из програмирања и алгоритама и структура података као помоћ у самосталном учењу студената.

Jvall [52] је софтверски пакет који омогућава коришћење посебне Java компоненте приликом писања програма која обезбеђује аутоматску визуелну репрезентацију уланчаних листа. Омогућава корисницима да визуелизују део програма који се односи на коришћење уланчаних листа и да након тога прате извршавање и увиде евентуалне грешке у програму. Погодан је за коришћење на курсевима из програмирања за коришћење у лабораторији, демонстрације на предавањима и самостално учење студената.

Visage [53] представља систем за визуелну репрезентацију графовских алгоритама. Омогућава корисницима да за задати проблем обиласка графа испрате примену неког од имплементираних алгоритама. Погодан је за коришћење на курсевима дискретне математике као помоћ студентима за самостално учење.

Jeliot [54] је систем за визуелну репрезентацију извршавања програма написаних у Java програмском језику, при чему је комплетна визуелизација аутоматски генерисане и не захтева никакве додатне директиве од корисника. Погодан је за учење програмирања за коришћење у оквиру лабораторијских вежби и за самостално учење студената.

Jhave [55] представља систем за визуелну репрезентацију алгоритама написаних у неком од скрипт језика које подржава. Омогућава корисницима да сами конструишу алгоритам који желе да визуелно репрезентују и да затим прате извршавање алгоритма са одабраним улазним параметрима. Погодан је за коришћење на курсевима из алгоритама и структура података као помоћ студентима у самосталном учењу.

MatrixPro [56] је систем за визуелну репрезентацију извршавања познатих алгоритама. Омогућава корисницима да направе анимацију било ког подржаног алгоритма постављајући жељене улазне параметре, а постоји и могућност да корисници сами додају нове алгоритме за визуелну репрезентацију. Погодан је за коришћење на курсевима из алгоритама и структура података као помоћ на предавањима.

Raptor [57] представља програмско окружење засновано на дијаграмима тока које омогућава корисницима да програме пишу исцртавајући дијаграме тока. Систем омогућава визуелну репрезентацију извршавања програма написаних на овај начин. Погодан је за коришћење на курсевима из програмирања као помоћ на предавањима, у лабораторији, за домаће задатке и за самостално учење студената.

Javenga [58] је систем за визуелну репрезентацију многих добро познатих графовских и мрежних алгоритама. Систем обухвата и едитор графова помоћу кога се може нацртати усмерени или неусмерени граф, омогућава корисницима да унесу улазне параметре алгоритма и затим посматрају визуелну репрезентацију у окружењу које могу да контролишу. Погодан је за коришћење на курсевима из рачунарских мрежа као помоћ на предавањима или за самостално учење студената.

Jawaa [59] представља скрипт језик за једноставно креирање статичких анимација путем Интернета. Систем обухвата примитиве, могућност креирања структура података и операција и едитор за креирање комплексних објеката. Погодан је за коришћење на курсевима из програмирања и алгоритама и структура података као помоћ на предавањима или за задавање домаћих задатака студентима.

2.3. ПРЕГЛЕД ПОСТОЈЕЋИХ СИСТЕМА ЗА ВИЗУЕЛНУ РЕПРЕЗЕНТАЦИЈУ КРИПТОГРАФСКИХ АЛГОРИТАМА

Област визуелне репрезентације криптографских алгоритама као подскуп области визуелне репрезентације алгоритама је релативно нов правац истраживања и публикације које описују резултате из ове области се појављују у последњих 7-8 година. Ова област се још увек није довољно развила да би се могао направити неки значајнији преглед области који би могао да да релевантне закључке. Самим тим у литератури нису пронађени прегледни радови из ове об–

Табела 2.7. Приказ општих информација о одабраним системима за визуелну репрезентацију криптографских алгоритама

Назив система	**Земља**	**Год.**	**Линк**	**Референца**
GRACE	Италија	2008.	http://www.di.unisa.it/research/grace/	[61]
IVATC	САД	2006.	http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1652079	[62]
MTUVisual	САД	2011.	http://www.cs.mtu.edu/~shene/NSF-4/index.html	[63], [64], [65]
DES animation	Велика Британија	2008.	http://www.cs.bham.ac.uk/research/projects/lemsys/DES/	[66]
GRASP	САД	2006.	http://prismhome.org/sites/default/files/Grasp.pdf	[67]

Табела 2.8. Класификација одабраних система за визуелну репрезентацију алгоритама према таксономији из [38]

Назив система	**Ниво апстракције**	**Начин имплементације**	**Технике испитивања употребљивости**	**Начин интеракције**
GRACE	визуелна репрезентација алгоритама	интерфејс	контролисани експерименти	конфигурисање визуелизације
IVATC	визуелна репрезентација алгоритама	интерфејс	опсервационе студије	само посматрање визуелизације
MTUVisual	визуелна репрезентација алгоритама	интерфејс	технике упитника	само посматрање визуелизације
DES animation	визуелна репрезентација алгоритама	интерфејс	неформална испитивања	само посматрање визуелизације
GRASP	визуелна репрезентација алгоритама	интерфејс	неформална испитивања	конфигурисање визуелизације

ласти. Са друге стране, значајно је у овом тренутку направити преглед области како би се видело у ком правцу се област развија. У наставку овог поглавља ће бити направљен преглед одабраних система за визуелну репрезентацију криптографских алгоритама.

Табела 2.7 приказује опште информације о одабраним системима за визуелну репрезентацију криптографских алгоритама. Ове информације обухватају: назив система, земљу настанка система, годину настанка (за системе за које је ова информација доступна) или годину публикације система (за системе за које није доступна година настанка), линк ка систему (за системе за које постоји сајт на коме су доступни) или линк ка публикацији која описује систем (за системе који немају сајт), референца. Свим линковима је последњи пут приступано априла 2014. године.

Класификација одабраних система за визуелну репрезентацију криптографских алгоритама према таксономији предложеној у [38] дата је у табели 2.8. Табела садржи: назив система, ниво апстракције, начин имплементације, технике испити–

Табела 2.9. Класификација одабраних система за визуелну репрезентацију алгоритама према таксономији из [39]

Назив система	Начин коришћења	Технике евалуације ефикасности	Коришћена теорија учења	Тип мереног знања	Начин мерења знања
GRACE	на предавањима, за учење	емпиријске технике	Cognitive Constructivism	концептуално	тест након коришћења
IVATC	на предавањима, за задатке, за учење	-	Epistemic Fidelity	-	-
MTUVisual	на предавањима, за учење	-	Epistemic Fidelity	-	-
DES animation	за учење	-	Epistemic Fidelity	-	-
GRASP	на предавањима, за учење	-	Cognitive Constructivism	-	-

вања употребљивости и начин интеракције. Поља попуњена вредношћу „-“ означавају да одређена информација о систему није доступна.

Класификација одабраних система за визуелну репрезентацију криптографских алгоритама према таксономији предложеној у [39] дата је у табели 2.9. Табела садржи: назив система, начин коришћења система, технике евалуације ефикасности система, теорију учења на којој је систем заснован, тип мереног знања и начин мерења знања. Поље начин коришћења попуњен је на основу предлога аутора система на који начин би њихов систем могао да се користи или је коришћен. Поље технике евалуације ефикасности попуњено је само за системе који су вршили евалуацију ефикасности, док за остале има вредност „-“. Поље коришћена теорија учења попуњено је на основу описа примене система и начина на који би требало да буде од користи студентима и то за системе у којима је то објашњено, док је за остале вредност овог поља „-“. Поља тип мереног знања и начин мерења знања попуњавана су по истом принципу као и поље технике евалуације ефикасности.

Табела 2.10 даје приказ карактеристика одабраних система за визуелну репрезентацију алгоритама које су битне са аспекта визуелизације. Табела садржи: назив система, динамичност система (да ли су у питању статичке или динамичке анимације), праћење извршавања (на ком нивоу је могуће праћење извршавања алгоритма: корак по корак, итерација, комплетан алгоритам), конфигурабилност улазних параметара (да ли и на ком нивоу је могуће конфигурисати улазне параметре алгоритма), флексибилност (да ли постоји могућност додавања нових алгоритама и на који начин), дизајн визуелне репрезентације (колико визуелна репрезентација одговара приказу алгоритама ко–

Табела 2.10. Приказ карактеристика одабраних система за визуелну репрезентацију алгоритама које су битне са аспекта визуелизације

Назив система	**Динамичн.**	**Праћење извршавања**	**Конфигурабилност улазних параметара**	**Флексибилн.**	**Дизајн визуелне репрезентације**	**Подржане теме**
GRACE	динамичка анимација	корак по корак	да, коришћењем визуелног едитора	писањем програма у Java програмском језику	произвољан	RSA, Diffie-Hellman, DSA, DES, X.509
IVATC	статичка анимација	корак по корак, комплетан алгоритам	да, коришћењем визуелног едитора	нема могућност додавања нових алгоритама	произвољан	Цезар, моноалфабетски, Affine, Vigenere, RC4, RSA и DES алгоритми
MTUVis.	статичка анимација	корак по корак	да, коришћењем визуелног едитора	нема могућност додавања нових алгоритама	произвољан	DES, SHA, RSA и EC алгоритми
DES animation	статичка анимација	корак по корак	да, коришћењем визуелног едитора	нема могућност додавања нових алгоритама	сагласан са литературом	DES алгоритам
GRASP	динамичка анимација	корак по корак	да, коришћењем визуелног едитора	могућност додавања нових алгоритама коришћењем предефинисан--их операција	сагласан са литературом	Diffie-Hellman алгоритам

ришћеном у литератури) и подржане теме из области заштите података.

У раду [60] објашњен је начин увођења активног учења у едукациони процес кроз коришћење система за визуелну репрезентацију алгоритама и дати су критеријуми дизајна оваквих система. Предложено је пет пожељних карактеристика приликом дизајна оваквих система: висок степен интеракције (енгл. *high interactivity*), разумљиве апстракције (енгл. *easily understood abstraction*), једноставније је боље (енгл. *simple is better*), релевантност (енгл. *relevancy*) и интуитиван кориснички интерфејс. Висок степен интеракције подразумева да корисник система има што већу улогу у визуелној репрезентацији алгоритма, што подразумева могућност конфигурисања улазних параметара, контролисање извршавања алгоритма корак по корак, лаку измену параметара и могућност повратка на претходни корак у извршавању. Разумљиве апстракције означавају да се приликом дизајна система апстракције које се користе у визуелној репрезентацији алгоритма осмисле тако да не одвуку пажњу корисника са градива које је у фокусу и да не буду сувише сложене да би корисник морао да

Табела 2.11. Примена критеријума дизајна предложених у [60] у одабраним системима за визуелну репрезентацију криптографских алгоритама

Назив система	Висок степен интеракције	Разумљиве апстракције	Једноставније је боље	Релевантност	Интуитиван кориснички интерфејс
GRACE	+	+	+	-	-
IVATC	+	+	-	-	+
MTUVisual	+	+	-	-	-
DES animation	+	+	+	-	+
GRASP	+	+	+	-	-

потроши доста времена да разуме апстракције. Једноставније је боље је карактеристика по којој је боље приказати само један концепт или један алгоритам у оквиру једне визуелне репрезентације, јер ће тако корисници лакше разумети. Релевантност подразумева да је важно исправно одабрати градиво које систем за визуелну репрезентацију алгоритама треба да прикаже. Интуитиван кориснички интерфејс представља карактеристику по којој би интерфејс система за визуелну репрезентацију алгоритама требало да буде што једноставнији и да смањи могућност грешке корисника приликом коришћења. Дефинисани критеријуми су од нарочите важности код система за визуелну репрезентацију криптографских алгоритама из аспекта ове дисертације. Због тога ће примена критеријума дизајна предложених у [60] бити анализирана у одабраним системима за визуелну репрезентацију криптографских алгоритама. Табела 2.11 садржи: назив система, висок степен интеракције, разумљиве апстракције, једноставније је боље, релевантност и робустан кориснички интерфејс. Поља попуњена вредношћу „+“ означавају да је критеријум примењен, док поља попуњена вредношћу „-“ означавају да критеријум није примењен.

GRACE [61] представља софтверски систем за визуелну репрезентацију криптографских протокола. У овом систему визуелно је репрезентован одређени број криптографских и некриптографских операција које се често користе у криптографским протоколима. Корисници могу да сами креирају сценарије који ће се извршавати у оквиру подржаних протокола. Корисници између осталог имају могућност да креирају учеснике у комуникацији, да за одређеног учесника креирају документ који може да размењује са осталим учесницима, да генеришу кључ или пар кључева у зависности од алгоритма који се користи у протоколу, да шифрују документ и да размењују поруке између креираних учесника у комуникацији. Постоји могућност да се одређени сценарио сними и касније репродукује. Подржани су протоколи који користе RSA, *Diffie-Hellman*, DSA, DES

или X.509 за комуникацију између учесника. Криптографске операције су имплементиране, али се у оквиру визуелне репрезентације могу видети само њихови резултати, а не и детаљи извршавања. Систем је реализован у програмском језику *Java* и остављена је могућност да се лако могу додати нови протоколи имплементацијом неколико нових класа које ће дефинисати понашање појединих криптографских операција у оквиру нових протокола.

IVATC [62] представља скуп интерактивних аплета за визуелну репрезентацију криптографских алгоритама. Алгоритми који су подржани у овом систему су: Цезар, моноалфабетски, *Affine*, *Vigenere*, RC4, RSA и DES. Интерфејс омогућава корисницима да унесу оригинални текст и кључ за шифровање, да добију шифровани текст за сваки од алгоритама и да интерактивно контролишу помоћне алате за анализу као што су графови са фреквенцијама појављивања слова, анализатори дужине кључа, итд. Ово омогућава да се код једноставнијих алгоритама демонстрирају и начини криптоанализе. Код RC4 алгоритма могуће је праћење формирања кључа, као и извршавања алгоритма корак по корак у итерацијама, уз приказ позиције у псеудо коду којим је алгоритам представљен. Аплет којим се врши визуелна репрезентација DES алгоритма приказује две итерације *Feistel* алгоритма у виду дијаграма са текстуалним описима операција које се извршавају у оквиру итерације уз приказ тока података кроз итерације.

MTUVisual [63], [64], [65] обухвата скуп од неколико независних система за визуелну репрезентацију алгоритама и то: DESVisual [63] који омогућава визуелну репрезентацију DES алгоритма, ECVisual [64] који омогућава визуелну репрезентацију алгоритама заснованих на елиптичкој криви и RSAVisual [65] који омогућава визуелну репрезентацију RSA алгоритма. DESVisual приказује прву итерацију DES алгоритма у виду дијаграма по принципу *Feistel* структуре са величинама података од 8 или 16 бита и величином кључа итерације од 6 или 10 бита. Корисник има могућност да види детаље иницијалне пермутације, као и саме итерације. Постоје два режима рада: праћење извршавања и вежбање. У режиму праћења извршавања све операције су извршене и корисник може да прати одређени бит кроз дијаграм кликом на тај бит. У режиму вежбања кориснику се постављају питања у току извршавања која захтевају да корисник сам израчуна поједине делове алгоритма. ECVisual омогућава кориснику да

добије визуелну репрезентацију елиптичке криве над реалним пољем или над коначним пољем простог реда, затим да изврши аритметичке операције, као и шифровање и дешифровање и да конвертује оригиналну поруку у тачку на елиптичкој криви. RSAVisual омогућава кориснику да испрати поступак рада RSA алгоритма на предефинисаним примерима, као и да увежбава познавање рада алгоритма кроз интерактивне задатке. Осим самог алгоритма чије је извршавање могуће испратити, визуелна репрезентација приказује и коришћење Еуклидовог алгоритма за проналажење инверзног броја, као и Ферматовог и Полардовог алгоритма за факторизацију и три врсте напада на RSA алгоритам.

DES animation [66] је систем за визуелну репрезентацију DES алгоритма. Систем је реализован као интернет апликација и омогућава праћење извршавања алгоритма на реалним примерима кроз предефинисану анимацију. Корисник има могућност да унесе оригинални податак у виду ASCII текста и кључ за шифровање у виду хексадецималне вредности и да започне поступак шифровања. Након тога, корисник има могућност да прати и контролише кретање кроз унапред припремљену анимацију која приказује извршавање алгоритма над подацима које је корисник унео. Корисник има могућност да прескочи преглед извршавања појединих корака у алгоритму. За сваку операцију поред демонстрације извршавања операције или делимичног извршавања операције постоји и пратеће текстуално објашњење које детаљно описује приказану операцију. Детаљи операција приказани су само за једну итерацију алгоритма.

GRASP [67] представља систем за визуелизацију криптографских протокола. Може се конфигурисати да прикаже визуелну репрезентацију било ког криптографског протокола, јер пружа посебан језик за спецификацију протокола који омогућава произвољан број учесника у комуникацији и размену порука између њих коришћењем одговарајућих команди неопходних у сигурносним протоколима. Корисници могу да модификују команде у протоколу током визуелне репрезентације извршавања протокола, чиме је омогућено да се у реалном времену види како одређена промена утиче на извршавање протокола. Корисницима је омогућено и да контролишу извршавање протокола опцијама које омогућавају померање корак унапред или корак уназад, враћање извршавања на почетак или извршавање протокола до краја.

Поред закључака изведених на основу анализе постојећих система за визуелну репрезентацију криптографских алгоритама, могу се извући и поједини закључци на основу анализе употребе система за визуелну репрезентацију криптографских алгоритама у настави. У раду [68] аутори су истраживали утицај увођења система за визуелну репрезентацију криптографских алгоритама на курс из области заштите података. Наглашено је да је коришћење техника активног учења повећало мотивацију студената и дуже држало њихову пажњу, него класично извођење наставе. Уочено је да на предметима из области заштите података постоје неке теме које су саме по себи јако занимљиве студентима (нпр. вируси, хакери, итд.), док са друге стране неке теме које се односе на коришћене концепте и теорију на којој се концепти заснивају (нпр. криптографски алгоритми) су много мање атрактивне и много теже за разумевање студентима. У [68] предложено је да би управо ове мање атрактивне теме требало да буду покривене коришћењем техника активног учења како би студентима биле занимљивије и лакше за разумевање.

2.4. ЗАКЉУЧЦИ АНАЛИЗЕ РЕЛЕВАНТНЕ ЛИТЕРАТУРЕ

Из претходне анализе може се извући неколико значајних закључака. Прво се може дефинисати како би требало да буде дизајниран систем за визуелну репрезентацију криптографских алгоритама који би требало користити. Затим се може установити како би тај систем требало користити да би био ефикасан. Може се дефинисати и начин провере ефикасности система. На крају, анализом постојећих система за визуелну репрезентацију криптографских алгоритама може се проверити да ли постоји неки систем који би испунио све претходно дефинисане захтеве.

Анализом закључака из секције 2.1 можемо констатовати да се системи за помоћ у учењу успешно користе у различитим областима и на различитим нивоима образовања. Видимо да постоји приличан број оваквих система који се користе на факултетима у настави на предметима из области рачунарске технике и информатике. Теорија која се бави применом оваквих система у образовању говори да је начин коришћења система важнији од саме реализације система када је у питању ефикасност. Принципи дизајна који су дефинисани у [37] се могу сматрати добрим препорукама приликом дизајнирања новог система за помоћ у учењу. Анализом постојећих система за визуелну репрезентацију алгоритама направљеном у секцији 2.3 могу се дефинисати ниво апстракције, начин

имплементације, теорија учења коју би требало користити, као и битне карактеристике са аспекта визуелне репрезентације које би систем за визуелну репрезентацију криптографских алгоритама требало да има. Након анализе постојећих система за визуелну репрезентацију криптографских алгоритама направљене у секцији 2.3 претходни закључци о дизајну могу се допунити запажањима карактеристичним за системе за визуелну репрезентацију криптографских алгоритама.

Препоруке за дизајн система су:

- подржани принципи: вишеструке репрезентације, суседности, подељене пажње и сажетости у што већој мери,
- ниво апстракције: визуелна репрезентација алгоритама,
- начин имплементације: интерфејс,
- коришћена теорија учења: *Epistemic Fidelity* или *Cognitive Constructivism*,
- динамичност: статичка анимација,
- праћење извршавања: корак по корак,
- конфигурабилност улазних параметара: да, коришћењем визуелног едитора,
- флексибилност: пожељна могућност додавања нових алгоритама (али није неопходна),
- дизајн визуелне репрезентације: сагласан са литературом,
- висок степен интеракције, разумљиве апстракције, једноставност визуелне репрезентације, релевантност приказаних тема и интуитиван кориснички интерфејс.

Питање начина коришћења система за визуелну репрезентацију криптографских алгоритама утиче и на дизајн самог система, али и на дизајн окружења у коме би се систем користио. У секцији 2.3 дефинисани су могући начини интеракције и начини коришћења. Закључци изведени на основу анализе постојећих система за визуелну репрезентацију алгоритама, могу се допунити и анализом из секције 2.3. Поред тога на основу анализе у секцији 2.3 могу се дефинисати и теме из области заштите података које би систем за визуелну репрезентацију криптографских алгоритама требало да подржи.

Препоруке за начин коришћења система су:

- начин интеракције: само посматрање визуелизације (минимално, а пожељно и више),
- начин коришћења: на предавањима, за учење, у лабораторији, за задатке (што више могућности то боље),
- подржане теме: класични криптографски алгоритми, симетрични криптографски блок алгоритми (DES, AES) и асиметрични криптографски алгоритми (*Diffie-Hellman*, RSA).

У секцији 2.3 објашњене су и технике које се баве испитивањем употребљивости и ефикасности система за визуелну репрезентацију алгоритама приликом коришћења у настави. С обзиром на то да предлог решења описаног проблема управо укључује коришћење једног система за визуелну репрезентацију криптографских алгоритама у настави из области заштите података битан аспект је провера успешности таквог система.

Препоруке за проверу употребљивости и ефикасности система су:

- за евалуацију употребљивости система (једна или више од предложених техника): неформална испитивања, хеуристичка испитивања, технике упитника, опсервационе студије и/или контролисани експерименти,
- за евалуацију ефикасности система (једна или више од предложених техника): анегдотске технике, програмске технике, аналитичке технике и/или емпиријске технике,
- за врсту мереног знања код евалуације ефикасности система: концептуално знање, процедурално знање или концептуално и процедурално знање,
- за начин мерења знања код евалуације ефикасности система (један или оба начина): тест након коришћења и побољшање резултата након коришћења у односу на резултат пре коришћења.

Уколико посматрамо анализу постојећих система за визуелну репрезентацију криптографских алгоритама направљену у секцији 2.3 видећемо да већина постојећих система барем делимично покрива алгоритме који се обрађују на курсу из Заштите података. Поред тога, сви системи у доста великој мери испуњавају претходно дефинисане препоруке када је у питању дизајн и начин коришћења система за визуелну репрезентацију криптографских алгоритама. За

све анализиране системе је спроведено барем неко испитивање употребљивости, док је евалуација ефикасности система спроведена само за један од анализираних система. Из ових разлога разматрана је могућност коришћења неког од постојећих система за визуелну репрезентацију криптографских алгоритама на предмету Заштита података.

GRACE [61] (подржава RSA, *Diffie-Hellman* и DES од алгоритама на Заштити података) и GRASP [67] (подржава *Diffie-Hellman* од алгоритама на Заштити података) представљају системе за визуелну репрезентацију криптографских протокола, па је тако акценат у овим системима другачији од жељеног. Наиме, код ових система приоритет је да се прикаже начин коришћења криптографског алгоритма у оквиру протокола који се користи за сигурну комуникацију између два или више учесника у комуникацији. Због тога су детаљи извршавања самог криптографског алгоритма већим делом сакривени од корисника коме су приказани само резултати извршавања. Ова два система се међусобно разликују по томе што GRACE поред визуелне репрезентације извршавања корака у протоколу даје могућност рада и са конкретним вредностима, али без улажења у детаље извршавања алгоритма, већ уз приказ резултата, док GRASP омогућава само праћење извршавања корака у протоколу на концептуалном нивоу, без конкретних вредности. IVATC [62] подржава Цезар, моноалфабетски, *Vigenere*, RSA и DES од алгоритама на предмету Заштита података. Када су у питању класични криптографски алгоритми овај систем задовољава претходно дефинисане услове, али не подржава све класичне криптографске алгоритме који се предају на предмету Заштита података. Код DES алгоритма приказане су само две итерације *Feistel* алгоритма где се детаљи извршавања појединих операција не могу испратити, већ су објашњени текстуалним објашњењем. Код RSA алгоритма изостављени су детаљи извршавања алгоритма и кориснику је само омогућено да формира пар кључева, унесе улазни текст и добије шифровани текст. DESVisual [63] (подржава DES од алгоритама на Заштити података) приказује само прву итерацију извршавања DES алгоритма и то са редукованим величинама улазних параметара. ECVisual [64] не подржава ниједан од алгоритама који се предају на предмету Заштита података. RSAVisual [65] (подржава RSA од алгоритама на Заштити података) у великој мери испуњава претходно дефинисане услове, уз

изузетак демонстрације једног од алгоритама факторизације описаних на предмету. Проблем је и кориснички интерфејс у коме су примери или аутоматски генерисани без интеракције са корисником или захтевају од корисника познавање алгоритма и ручни унос параметара. DES animation [66] (подржава DES од алгоритама на Заштити података) у великој мери испуњава претходно дефинисане услове. Проблем је што су приказани детаљи само прве итерације алгоритма, као и то што корисник нема могућност самосталног истраживања извршавања алгоритма, већ је праћење извршавања реализовано у виду анимације у којој корисник има могућност управљања темпом анимације. Пратећи описи су веома детаљни, па је овај систем погодан за самостално учење студената код куће. У литератури није пронађен ниједан систем за визуелну репрезентацију криптографских алгоритама који подржава AES алгоритам.

Сви претходно анализирани системи су развијени да задовоље неке специфичне потребе са само појединим алгоритмима које подржавају и деловима тих алгоритама који су визуелно репрезентовани. Ови системи само делимично испуњавају захтеве који су дефинисани како би се побољшала настава из предмета Заштита података. Због тога је донета одлука да се развије нови систем за визуелну репрезентацију криптографских алгоритама (COALA) који би био у стању да у потпуности задовољи претходно дефинисане захтеве и затвори јаз између расположивих система и потреба предмета Заштита података.

3. ОПИС ПОДРЖАНИХ КРИПТОГРАФСКИХ АЛГОРИТАМА У COALA СИСТЕМУ

У овом поглављу је приказан опис криптографских алгоритама који су подржани у COALA систему. Систем подржава пет врста криптографских алгоритама: супституционе алгоритме, транспозиционе алгоритме, продукционе алгоритме, симетричне блок алгоритме и алгоритме са јавним кључем.

3.1. СУПСТИТУЦИОНИ АЛГОРИТМИ

У овом поглављу су описани супституциони криптографски алгоритми који су подржани у COALA систему. Систем подржава четири таква алгоритма: Цезар алгоритам, моноалфабетски алгоритам, *Playfair* алгоритам и *Vigenere* алгоритам.

3.1.1. ЦЕЗАР АЛГОРИТАМ

Цезар алгоритам [1] је један од најстаријих симетричних супституционих криптографских алгоритама. Смислио га је Јулије Цезар по чему је и добио назив. Алгоритам подразумева да се свако слово алфабета из оригиналне поруке мења са словом које је за три места удесно у алфабету. Управо померај у односу на оригинално слово представља кључ у овом алгоритму. Уколико посматрамо енглески алфабет који има 26 слова, онда имамо 25 различитих вредности кључа.

Данас је овај алгоритам занимљив, не само због тога што је можда један од најстаријих познатих на свету, већ и због тога што је погодан за демонстрацију слабости које утичу на успешност криптоанализе. Наиме, као што је већ речено, постоји 25 различитих кључева што је јако мали број могућности. Уколико знамо да је нека порука шифрована Цезар алгоритмом, онда је лако испробавањем свих могућих вредности кључа доћи до оригиналне поруке. Три фактора су заслужна за успешност криптоанализе нападом испробавањем свих могућих вредности: да ли је сам алгоритам познат, колики је скуп кључева и колико је познат и препознатљив сам језик оригиналне поруке.

3.1.2. МОНОАЛФАБЕТСКИ АЛГОРИТАМ

Моноалфабетски алгоритам [1] представља унапређење Цезар алгоритма, тако што дозвољава да комбинација слова за замену буде произвољна. Сада можемо имати произвољну пермутацију свих слова из алфабета као кључ. Ако посматрамо енглески алфабет то нам пружа 26! различитих могућности.

Овај алгоритам је интересантан, због тога што демонстрира нови аспект криптоанализе. Наиме, иако сада имамо изузетно велики број различитих вредности кључа, чиме је напад испробавањем знатно отежан, алгоритам се и даље не може сматрати сигурним. Разлог за то је што у сваком језику постоје одређена правила, односно карактеристике тог језика, чијом се анализом може смањити скуп могућих вредности кључа, а онда нападом испробавањем свих преосталих вредности кључа доћи до оригиналне поруке. За сваки језик се може установити релативна фреквенција појављивања слова (нпр. у енглеском језику слово е има убедљиво највећу релативну фреквенцију појављивања). За поруку која се анализира може се израчунати релативна фреквенција појављивања слова и затим упоредити са фреквенцијом за тај језик и на тај начин могу се добити одређени делови кључа, чиме се смањује скуп за претрагу. Исто се може урадити и за комбинације два или три слова.

Да би се побољшала сигурност потребно је на неки начин елиминисати карактеристике коришћеног језика у шифрованој поруци. Две методе које се користе у ту сврху у супституционим алгоритмима су: шифровање више слова оригиналне поруке заједно или коришћење више различитих кључева за шифровање.

3.1.3. *PLAYFAIR* АЛГОРИТАМ

Playfair алгоритам [1] представља симетрични супституциони криптографски алгоритам који покушава да применом методе шифровања више слова оригиналне поруке заједно елиминише карактеристике коришћеног језика из шифроване поруке. Овај алгоритам дели оригиналну поруку на слогове и сваки слог се третира као јединствена целина за замену приликом формирања шифроване поруке. Алгоритам се заснива на матрици величине 5x5 која се попуњава на основу произвољне кључне речи. Матрица се попуњава ред по ред и то тако што

се најпре у њу уписује изабрана кључна реч, али без понављања слова, а затим се остатак попуни редом према алфабету словима која претходно нису уписана у матрицу. У оригиналној варијанти алгоритма која се користи за енглески језик слова I и J се попуњавају у једно поље матрице и рачунају се као једно слово. За шифровање се користе следећа правила:

- Слогови оригиналне поруке који се састоје од истих слова морају се раздвојити коришћењем неког слова које је одабрано у ту сврху (нпр. у енглеском језику често се користи слово x).
- Уколико се слова из слога оригиналне поруке налазе у истом реду у матрици, тада се свако од њих мења са словом из истог реда из колоне са десне стране.
- Уколико се слова из слога оригиналне поруке налазе у истој колони у матрици, тада се свако од њих мења са словом из исте колоне из реда испод.
- У супротном, свако слово из слога оригиналне поруке мења се са словом из матрице које се налази у истом реду као и слово које се мења и у колони у којој се налази парњак слова које се мења.

Playfair алгоритам је увођењем слогова као јединица за шифровање значајно отежао криптоанализу на основу карактеристика коришћеног језика (нпр. уместо 26 слова у енглеском језику постоји 676 слогова), али је није у потпуности елиминисао.

3.1.4. *VIGENERE* АЛГОРИТАМ

Vigenere алгоритам [1] представља симетрични супституциони криптографски алгоритам који покушава да применом методе коришћења више различитих кључева за шифровање елиминише карактеристике коришћеног језика из шифроване поруке. Овај алгоритам спада у групу полиалфабетских субституционих алгоритама. Полиалфабетски алгоритми користе скуп од више моноалфабетских алгоритама и приликом шифровања оригиналне поруке на основу кључа се одређује који од моноалфабетских алгоритама ће бити искоришћен за шифровање сваког слова оригиналне поруке. *Vigenere* алгоритам користи скуп од 26 Цезар алгоритама са померајем од 0 до 25. Сваки алгоритам

означен је словом које у одговарајућем Цезар алгоритму мења слово а. Како би се шифровала порука потребан је кључ исте дужине као и оригинална порука. Обично се кључ формира на основу кључне речи, понављањем кључне речи до дужине оригиналне поруке. За шифровање сваког слова оригиналне поруке користи се различити Цезар алгоритам одређен словом из кључа. Пошто постоји понављање у кључу због начина његовог формирања, постоји и побољшање названо аутоматско генерисање кључа (енгл. autokey). У овој варијанти за допуну кључа до дужине оригиналне поруке уместо понављања кључне речи, на кључну реч се додаје текст из оригиналне поруке. Ни овај алгоритам не успева да у потпуности елиминише карактеристике коришћеног језика, иако их у великој мери скрива.

3.2. ТРАНСПОЗИЦИОНИ АЛГОРИТМИ

У овом поглављу су описани транспозициони криптографски алгоритми који су подржани у COALA систему. Систем подржава два таква алгоритма: *Rail Fence* алгоритам и *Row Transposition* алгоритам.

3.2.1. *RAIL FENCE* АЛГОРИТАМ

Rail Fence алгоритам [1] спада у групу симетричних транспозиционих криптографских алгоритама. Ови алгоритми за разлику од супституционих алгоритама не врше замене слова из оригиналне поруке, већ мењају редослед слова у оригиналној поруци и тиме сакривају њен садржај. Овај алгоритам је један од најједноставнијих транспозиционих алгоритама. У овом алгоритму порука се исписује наизменично одозго на доле, па одоздо на горе у дијагоналама. Број редова у којима се порука исписује представља кључ. Шифрована порука се добија тако што се порука која је исписана на претходни начин очита ред по ред.

3.2.2. *ROW TRANSPOSITION* АЛГОРИТАМ

Row Transposition алгоритам [1] је нешто сложенији алгоритам од *Rail Fence* алгоритма. У овом алгоритму оригинална порука се исписује у правоугаонику ред по ред. Свака колона исписане поруке означи се редним бројем и шифрована порука добија се читањем претходно исписане поруке колону по колону у произвољном редоследу колона према означеним редним бројевима. Редослед

колона представља кључ у овом алгоритму.

Овакав алгоритам се лако препознаје због тога што шифрована порука има исту фреквенцију појављивања слова као и оригинална порука. Уколико се алгоритам понови више пута заредом добија се боља безбедност шифроване поруке.

3.3. ПРОДУКЦИОНИ АЛГОРИТМИ

У овом поглављу су описани продукциони криптографски алгоритми који су подржани у COALA систему. Систем подржава један такав алгоритам: *Rotor Machine* алгоритам.

3.3.1. *ROTOR MACHINE* АЛГОРИТАМ

Анализом субституционих и транспозиционих алгоритама установљено је да не постоји ни један такав алгоритам који би у потпуности могао да елиминише карактеристике коришћеног језика у шифрованој поруци, осим *One-time pad* алгоритма који није практичан, јер захтева коришћење великог кључа. Установљено је и да вишеструка примена субституционих или транспозиционих алгоритама на неку поруку појачава сигурност шифроване поруке. Праву прекретницу у развоју криптографских алгоритама представља комбиновање субституционих и транспозиционих алгоритама, што се назива продукционим алгоритмима [1].

Први такви алгоритми реализовани су у виду механичких машина за шифровање, које су коришћене током Другог светског рата. Ове машине састојале су се од неколико цилиндара који су могли да ротирају независно један од другог. Сваки цилиндар има 26 улазних и 26 излазних линија које су интерно повезане у цилиндру тако да је свака улазна линија везана са тачно једном излазном линијом.

Ако бисмо свакој улазној и излазној линији једног цилиндра доделили по једно слово алфабета добили бисмо моноалфабетски супституциони алгоритам. Након шифровања једног слова оригиналне поруке, цилиндар ротира за једну позицију, на тај начин се мењају интерне везе унутар цилиндра и добија се нови моноалфабетски супституциони алгоритам за следеће слово оригиналне поруке. Након 26 ротација цилиндар се враћа на почетну позицију, што значи да са једним цилиндром имамо полиалфабетску супституциону шифру са периодом 26.

Претходно описана шифра са коришћењем једног цилиндра је тривијална за криптоанализу и права снага ротор машина лежи у коришћењу већег броја цилиндара, код којих су излазне линије првог везане на улазне линије другог, излазне линије другог везане на улазне линије трећег, итд. Код овакве машине цилиндар који је најближи оператеру ротира за једну позицију након сваког шифрованог слова оригиналне поруке, након сваког комплетног периода ротације овог цилиндра, следећи у низу ротира за једну позицију, итд. Резултат је да је период сада 17576 уколико се користе три цилиндра, 456976 уколико се користе четири цилиндра, односно 11881376 уколико се користи пет цилиндара.

3.4. СИМЕТРИЧНИ БЛОК АЛГОРИТМИ

У овом поглављу су описани симетрични блок алгоритми који су подржани у COALA систему. Систем подржава два таква алгоритма: DES алгоритам и AES алгоритам.

3.4.1. DES АЛГОРИТАМ

DES алгоритам [69] је типичан представник савремених симетричних криптографских блок алгоритама. Овај алгоритам има могућност шифровања блокова величине 64 бита коришћењем кључа величине 56 бита. Заснован је на *Feistel* структури алгоритма, тако да се за шифровање и дешифровање користи идентичан алгоритам. DES има широк спектар примене и коришћен је као главни криптографски алгоритам за обезбеђивање тајности више од 20 година, од усвајања стандарда 1977. године све до 1999. године када је проглашен за недовољно безбедан и замењен најпре новијом верзијом 3DES, а затим и потпуно новим стандардом AES.

Алгоритам прихвата два податка као улаз: оригиналну поруку за шифровање и кључ за шифровање. Састоји се од иницијалне пермутације, затим шеснаест идентичних итерација, 32-битне замене и инверзне иницијалне пермутације. Од кључа се применом одговарајућих операција обезбеђују кључеви итерације који се користе у свакој од шеснаест итерација.

Иницијална пермутација служи да од оригиналног 64-битног блока направи

улазни податак у прву итерацију алгоритма. У свакој од шеснаест итерација користи се механизам који важи код Feistel структуре алгоритма. Улазни блок се дели на леву и десну половину једнаке дужине. Десна половина пролази кроз итерацију непромењена и постаје лева половина за следећу итерацију ($L_i=R_{i-1}$). Затим се на десну половину и кључ итерације примени функција итерације ($F(R_{i-1}, K_i)$) и резултат ове функције се користи да се логичком операцијом ексклузивно ИЛИ помеша са левом половином (L_{i-1}). Резултат ове операције је нова десна половина блока (R_i). Функција итерације ($F(R_{i-1}, K_i)$) подразумева следеће операције:

- пермутацију са експанзијом којом се десна половина блока (R_{i-1}) са 32 бита прошири на 48 бита, што је такође величина и кључа итерације (K_i),
- затим логичку операцију ексклузивно ИЛИ примењену на претходно проширену десну половином блока и кључ итерације,
- резултујућих 48 бита се пропушта кроз алгоритам замене који даје 32 бита као излаз и
- на крају се извршава још једна пермутација добијених 32 бита.

Алгоритам замене који се користи реализован је коришћењем 8 табела замене (S-box табела) од којих свака за улазних 6 бита даје 4 бита на излазу. Свака табела састоји се од четири различита четворобитна супституциона алгоритма. Бит највеће и најмање тежине улазног 6-битног податка одређују који од четири алгоритма ће бити примењен (ред у табели), док средња четири бита бивају замењени вредношћу из табеле (одређују колону у табели). Након шеснаест итерација и 32-битне замене и инверзне иницијалне пермутације добија се 64-битни шифровани блок.

Генерисање кључева итерације подразумева да се од оригиналног 56-битног кључа формира шеснаест различитих 48-битних кључева који ће се користити у израчунавању функције итерације у свакој итерацији. Најпре се на 64-битну унету вредност кључа примени пермутација, која осим мешања бита и одбацује сваки осми бит, тако да ефективна дужина кључа остаје 56 бита. Затим се у шеснаест итерација 56 бита дели у две половине од по 28 бита и изврши се кружно

померање улево за сваку половину посебно за једно или два места (што је дефинисано распоредом ротације кључа), а након тога се новом пермутацијом из сваке 28-битне половине одаберe по 24 бита који заједно формирају 48-битни кључ итерације. Резултат након ротације (56 бита) представља улаз у следећу итерацију за формирање кључа итерације.

3.4.2. AES АЛГОРИТАМ

AES алгоритам [70], [71] спада у групу симетричних криптографских блок алгоритама. Алгоритам постоји од 2001. године, када је издат као стандард од стране *United States National Institute of Standards and Technology* (US NIST). Стварању AES алгоритма претходило је проглашење *Data Encryption Standard* (DES) алгоритма, који се претходно користио у исте сврхе, за недовољно безбедан алгоритам 1997. године. Главни разлози за сумње у сигурност DES алгоритма били су постојање успешних аналитичких напада на алгоритам и смањивање времена потребног за нападе претраживањем свих вредности кључа. Ови недостаци били су последица наглог развоја рачунарске технологије у том периоду и значајног побољшања доступног хардвера. Пре доношења одлуке да се развија нови алгоритам, направљен је покушај да се постојећи унапреди троструком узастопном применом (3DES) [72], али иако је побољшана безбедност постојећег алгоритма, дугорочно гледано проблем није решен. Због тога US NIST исте године издаје позив за предлоге за нови стандард назван AES. Општа спецификација новог алгоритма гласила је да је потребно предложити симетрични блок алгоритам са величином блока који се шифрује од 128 бита, са подршком за величине кључа за шифровање од 128, 192 и 256 бита и сигурношћу једнаком или већом од 3DES алгоритма. У првом кругу, одржаном 1998. године, прихваћено је 15 кандидата, да би у другом кругу, одржаном 1999. године у ужи избор ушло следећих пет кандидата:

- MARS (IBM) [73],
- RC6 (USA) [74],
- *Rijndael* (*Belgium*) [75],
- *Serpent* (*Euro*) [76] и

- *Twofish* (USA) [77].

У финалном кругу избора, одржаном 2000. године, изабран је *Rijndael* алгоритам, који су осмислила два криптографа из Белгије: *Joan Daemen* и *Vincent Rijmen*.

AES алгоритам је симетрични криптографски блок алгоритам са величином блока који се шифрује од 128 бита и подршком за величине кључа за шифровање од 128, 192 и 256 бита. Податке обрађује у итерацијама, чији број зависи од величине кључа (за кључ од 128 бита има 10 итерација, за кључ од 192 бита има 12 итерација, а за кључ од 256 бита има 14 итерација). За сваки блок података који је шифрован одређеним кључем потребан је исти кључ за дешифровање.

У алгоритму се подаци, који се обрађују, третирају као структура, која се састоји од четири групе од по четири бајта. Ова структура назива се стање и може се графички представити као матрица 4x4. Сви кораци у алгоритму спроводе се над стањем. Кључ се проширује коришћењем посебно осмишљене функције тако да се у свакој итерацији добије различити 128-битни кључ итерације. Кључ итерације се, попут података, представља као четири групе од по четири бајта и тако користи у одговарајућој итерацији.

Све итерације алгоритма, осим последње, приликом шифровања, обухватају следеће кораке који се изводе над стањем:

- замена бајтова (енгл. *substitute bytes*),
- померање редова (енгл. *shift rows*),
- мешање колона (енгл. *mix columns*) и
- додавање кључа итерације (енгл. *add round key*).

Замена бајтова подразумева коришћење једне табеле замене величине 16x16 бајтова (S-box), која садржи пермутацију свих 256 осмобитних вредности. Сваки бајт стања се мења бајтом из табеле замене, који се узима из реда одређеног са лева 4 бита бајта који се мења, и колоне одређене са десна 4 бита бајта који се мења.

Померање редова представља кружно померање улево на нивоу бајта за сваки

ред стања. Померање се врши на следећи начин: први ред остаје непромењен, за други ред врши се кружно померање улево за један бајт, за трећи ред врши се кружно померање улево за два бајта и за четврти ред врши се кружно померање улево за три бајта.

Мешање колона се извршава за сваку колону стања посебно. Сваки бајт колоне мапира се у нову вредност која зависи од сва четири бајта те колоне. Математички се мешање колона може представити као множење матрице стања са матрицом чије су вредности прецизно дефинисане, при чему се индивидуална множења и сабирања приликом множења матрица врше у пољу *Galois* GF(2^8).

Додавање кључа итерације представља бајтовску операцију ексклузивно ИЛИ сваког бајта стања са одговарајућим бајтом кључа итерације.

Алгоритам је карактеристичан по томе што започиње и завршава се са кораком додавања кључа итерације, као и по томе што последња итерација нема операцију мешања колона. Дешифровање се врши на тај начин што сваки од описаних корака алгоритма има инверзан корак, па се на нивоу корака шифровани подаци пропуштају у инверзном редоследу кроз алгоритам и на њега се примењују инверзни кораци са кључевима итерације у инверзном редоследу у односу на шифровање.

3.5. АЛГОРИТМИ СА ЈАВНИМ КЉУЧЕМ

У овом поглављу су описани криптографски алгоритми са јавним кључем који су подржани у COALA систему. Систем подржава два таква алгоритма: *Diffie-Hellman* алгоритам и *RSA* алгоритам.

3.5.1. *DIFFIE-HELLMAN* АЛГОРИТАМ

Diffie-Hellman алгоритам [78] је први публиковани криптографски алгоритам са јавним кључем. Сврха овог алгоритма је да омогући пару корисника да међусобно размене заједничку тајну вредност која се може користити као кључ за будућу размену порука, али без потребе да се установљена заједничка тајна вредност заиста размењује између корисника.

Потребно је да постоје две јавно доступне вредности: прост број q и цео број α који представља прост корен броја q. Уколико корисници А и В треба да искористе овај алгоритам да размене заједничку тајну вредност, тада ће сваки од њих изабрати насумичан цео број $X_a < q$, односно $X_b < q$, респективно. Затим корисник А израчунава вредност $Y_a = \alpha^{Xa} \ mod \ q$, а корисник В израчунава вредност $Y_b = \alpha^{Xb} \ mod \ q$. Сваки корисник задржава X вредност као приватну, а објављује Y вредност као јавно доступну. Сада корисник А израчунава заједничку тајну вредност по формули $K = Y_b^{Xa} \ mod \ q$, а корисник В израчунава заједничку тајну вредност по формули $K = Y_a^{Xb} \ mod \ q$.

3.5.2. RSA АЛГОРИТАМ

Након што су *Diffie* и *Hellman* објавили рад у коме су представили концепт криптографских алгоритама са јавним кључем годину дана касније објављен је и први алгоритам базиран на новом концепту. Алгоритам су смислили *Ron Rivest*, *Adi Shamir* и *Len Adleman* и по њима алгоритам и носи назив RSA алгоритам [79]. У питању је криптографски блок алгоритам у коме су блокови оригиналне и шифроване поруке представљени као цели бројеви са вредностима између 0 и n-1 за неко n. Типична величина за n је 1024 бита.

Корисници RSA алгоритма пре него што су у могућности да алгоритам користе за шифровање и дешифровање морају да формирају пар кључева, који се састоји од приватног кључа који сваки корисник задржава само за себе и јавног кључа који корисници објављују јавно и који свако може да види. Поступак формирања пара кључева је следећи:

- прво се изаберу два велика проста броја p и q, тако да је $p \neq q$,
- затим се рачуна $n = p \times q$,
- након тога се рачуна Ојлерова фи функција за n: $\phi(n) = (p-1)(q-1)$,
- потом се бира вредност e, тако да су испуњени следећи услови: $\gcd(\phi(n), e) = 1$ и $1 < e < \phi(n)$,
- на крају се израчунава вредност d која је мултипликативно инверзна са e по модулу $\phi(n)$, односно $d \equiv e^{-1} \bmod \phi(n)$.

Формирани пар кључева састоји се од јавног кључа (e, n) и приватног кључа (d, n). Свако ко жели да пошаље шифровану поруку овом кориснику користи његов јавни кључ за шифровање. Било која порука M < n може се шифровати коришћењем израза: $C=M^e \ mod \ n$, где је C шифрована порука. Корисник који је власник пара кључева једини може дешифровати овако шифровану поруку користећи израз $M=C^d \ mod \ n$.

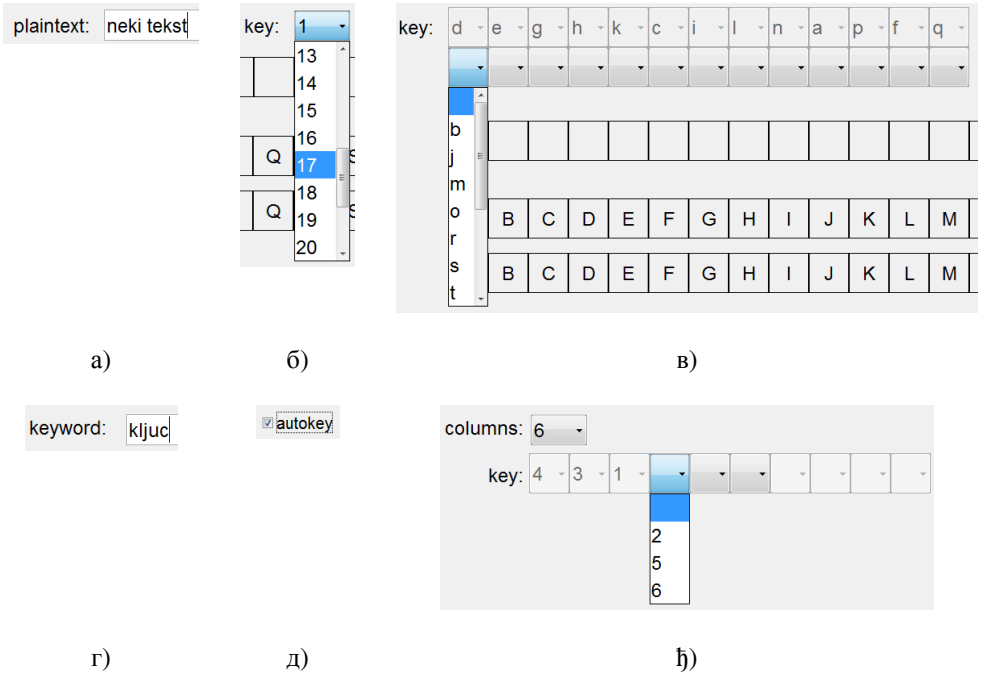
4. ВИЗУЕЛНА РЕПРЕЗЕНТАЦИЈА У COALA СИСТЕМУ

У овом поглављу биће објашњени детаљи везани за визуелну репрезентацију алгоритама коришћену у COALA систему. На основу закључака изведених у поглављу 2 произашао је дизајн коришћен у COALA систему.

4.1. ВИЗУЕЛНА РЕПРЕЗЕНТАЦИЈА КЛАСИЧНИХ КРИПТОГРАФСКИХ АЛГОРИТАМА

Код класичних криптографских алгоритама који су подржани у COALA систему (Цезар, моноалфабетски, *Playfair*, *Vigenere*, *Rail Fence*, *Row Transposition* и *Rotor Machine* алгоритми) основна идеја визуелне репрезентације је била да се постигне што боља прегледност у оквиру једног екрана на коме је приказан читав алгоритам. С обзиром да се ради о једноставним алгоритмима, уведена су одређена разумна ограничења која су омогућила бољу визуелну репрезентацију. За све алгоритме дужина поруке за коју се може приказати извршавање алгоритма ограничена је на 30 принтабилних карактера, код *Rail Fence* алгоритма максималан број редова је 10, код *Row Transposition* алгоритма максималан број колона је 10, код *Rotor Machine* алгоритма користе се три ротора чија је интерна конфигурација предефинисана и не може се мењати. Ова ограничења не умањују могућност демонстрације рада подржаних алгоритама, али омогућавају бољу организацију визуелне репрезентације ових алгоритама.

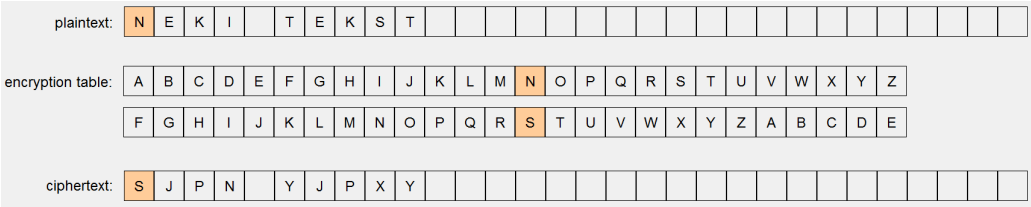
За све подржане алгоритме корисник има могућност уноса оригиналне поруке у виду текста који се састоји од принтабилних ASCII карактера (слика 4.1а). За све подржане алгоритме, осим *Rotor Machine* алгоритма, омогућен је и одабир кључа за шифровање. Ово омогућава да се рад алгоритама може лако демонстрирати на великом броју различитих примера. Унос кључа је омогућен на различите начине у зависности од потреба алгоритма. Код Цезар алгоритма користи се падајућа листа из које се бира померај (слика 4.1б). Код моноалфабетског алгоритма користи се низ од 26 падајућих листа, где се у свакој бира по једно слово кључа (слика 4.1в), при чему када се у једној листи одабере слово, оно се избацује из



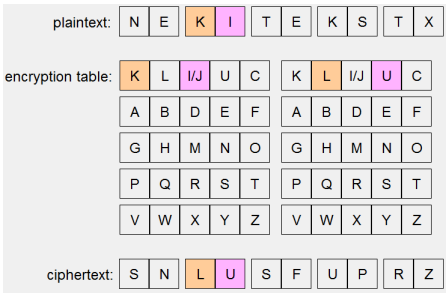
Слика 4.1. Визуелна репрезентација конфигурисања улазних параметара за класичне криптографске алгоритме подржане у COALA систему

свих осталих листа, како би на крају било одабрано 26 различитих слова. Код *Playfair* и *Vigenere* алгоритама кључна реч на основу које се формира кључ се уноси у виду текста који се састоји од принтабилних ASCII карактера (слика 4.1г), с тим што је код *Vigenere* алгоритма омогућен и одабир начина формирања кључа коришћењем оригиналне или *autokey* варијанте (слика 4.1д). Код *Rail Fence* алгоритма се из падајуће листе бира број редова који представља кључ, слично као код Цезар алгоритма (слика 4.1б), док се код *Row Transposition* алгоритма најпре одабере број колона из падајуће листе, слично као код Цезар алгоритма и *Rail Fence* алгоритма (слика 4.1б), а затим и редослед колона који представља кључ из низа падајућих листи које омогућавају да се одабере секвенца различитих бројева на сличан начин као што је то урађено код моноалфабетског алгоритма (слика 4.1ђ). Код *Rotor Machine* алгоритма број ротора, интерна конфигурација сваког од ротора и начин повезивања ротора су фиксни, тако да практично не постоји могућност одабира кључа. Оваквом реализацијом поступка уношења улазних параметара, елиминисана је могућност грешке приликом конфигурисања од стране корисника.

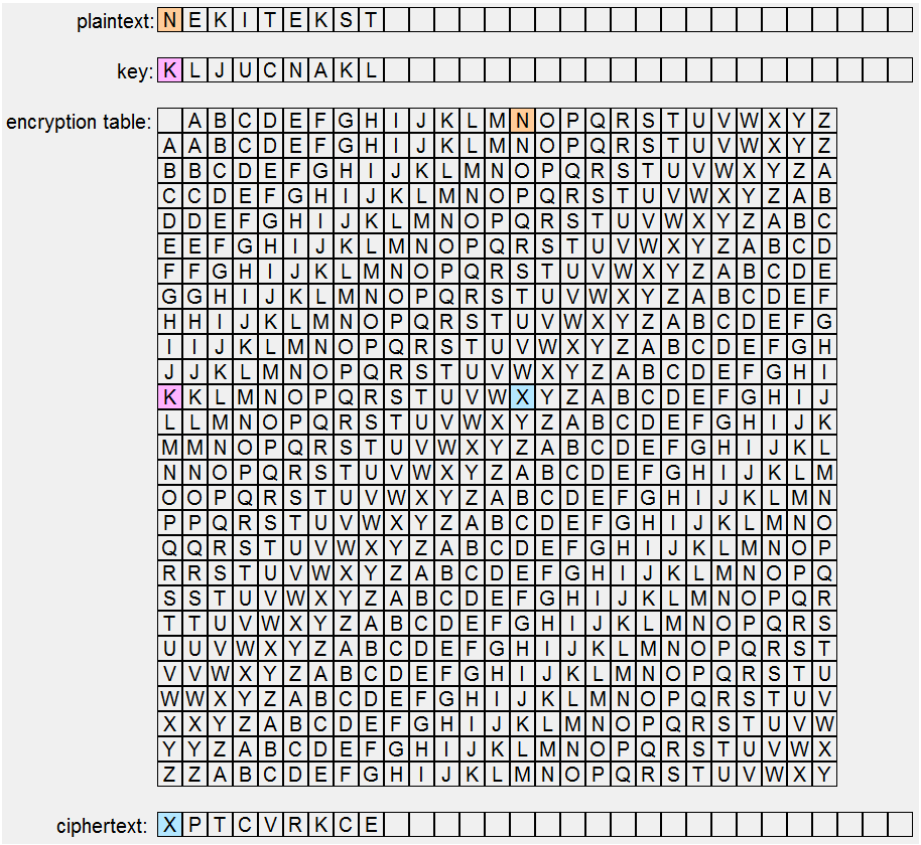
За све подржане класичне криптографске алгоритме приказана је визуелна репрезентација поступка шифровања оригиналне поруке, док је за Цезар и моно–



а)



б)



в)

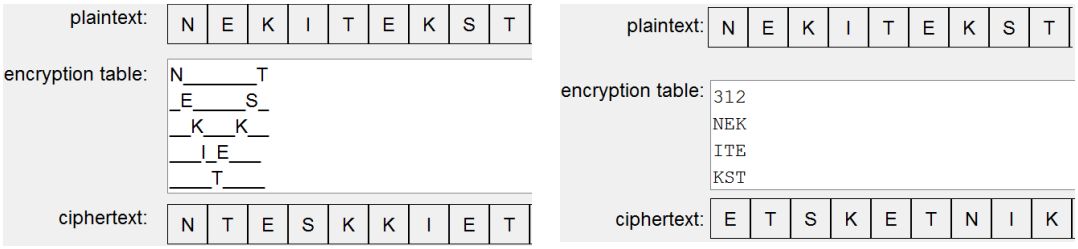
Слика 4.2. Визуелна репрезентација примењене трансформације код Цезар, моноалфабетског, *Playfair* и *Vigenere* алгоритма у COALA систему

алфабетски алгоритам на истом екрану приказано и дешифровање шифроване

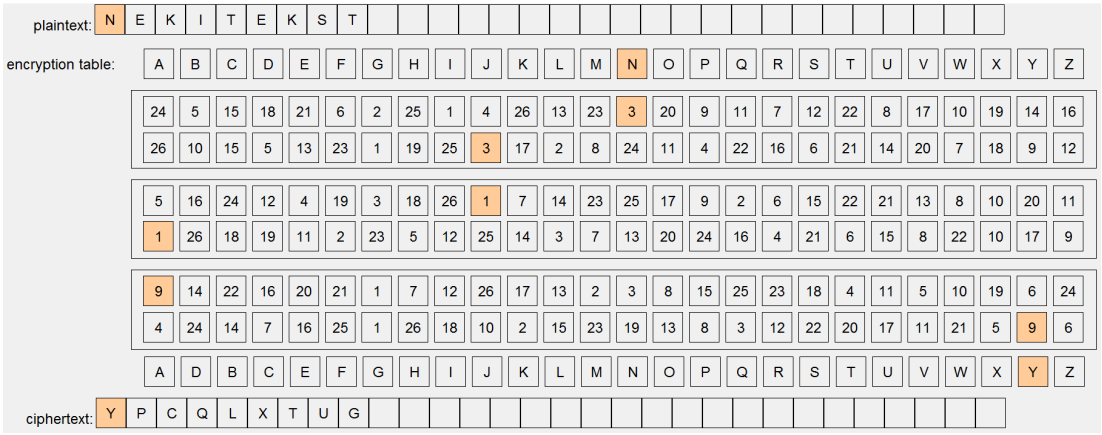
поруке. Након уношења улазних параметара визуелна репрезентација извршавања алгоритма се добија притиском на дугме *ENCRYPT*. Могуће је прекинути текућу визуелну репрезентацију алгоритма и вратити се на иницијализацију улазних параметара за исти алгоритам, коришћењем дугмета *RESET*. Повратак на почетни екран COALA система како би био одабран неки други алгоритам за визуелну репрезентацију, омогућен је притиском на дугме *BACK*. Код *Rotor Machine* алгоритма омогућено је праћење извршавања алгоритма након сваког шифрованог слова оригиналне поруке, па због тога постоје и дугмићи *NEXT* и *PREVIOUS* који приказују начин шифровања наредног, односно претходног слова, респективно.

Код Цезар, моноалфабетског, *Playfair* и *Vigenere* алгоритма користе се боје како би се јасније приказало како је шифровано свако појединачно слово оригиналне поруке. Омогућено је кориснику да позиционирањем миша на било које слово оригиналне или шифроване поруке види на који начин је дошло до одговарајуће трансформације. Код Цезар и моноалфабетског алгоритма приказане су оригинална порука, табела пресликавања и шифрована порука и у њима означена трансформација одабраног слова (слика 4.2а). Код *Playfair* алгоритма су искоришћене две матрице како би у једној могла да се види позиција слова из слога оригиналне поруке, а у другој примењена трансформација и слог који ће се појавити у шифрованој поруци (слика 4.2б). Искоришћене су две боје за означавање, по једна за свако слово из слога. Код *Vigenere* алгоритма приказана је матрица и искоришћене су три боје за означавање примењене трансформације, једна за слово из оригиналне поруке, друга за слово кључа и трећа за слово шифроване поруке (слика 4.2в).

Код *Rail Fence* и *Row Transposition* алгоритама искоришћене су текстуалне компоненте са могућношћу исписа у више редова, како би било демонстрирано на који начин је потребно исписати оригиналну поруку да би очитавањем дефинисаним у алгоритму могла да се добије шифрована порука. Код *Rail Fence* алгоритма (слика 4.3а) порука се исписује дијагонално у одабраном броју редова, наизменично одозго на доле, па одоздо на горе. Симбол „_“ је искоришћен за попуну празних места, како би био јасан начин исписа. Шифрована порука се добија очитавањем овако исписане оригиналне поруке, ред по ред. Код *Row Transposition* алгоритма (слика 4.3б) порука се исписује ред по ред у одабраном



Слика 4.3. Визуелна репрезентација примењене трансформације код *Rail Fence* и *Row Transposition* алгоритама у COALA систему



Слика 4.4. Визуелна репрезентација примењене трансформације код *Rotor Machine* алгоритма у COALA систему

броју колона, а шифрована порука се добија очитавањем овако исписане поруке по колонама и то у редоследу који је дефинисан од стране корисника.

Код *Rotor Machine* алгоритма (слика 4.4) користи се боја како би се јасније приказало како је шифровано свако појединачно слово оригиналне поруке. У овом случају означавање помоћу боје је константно и постоје контролни дугмићи помоћу којих је омогућено пребацивање на наредно, односно претходно шифровано слово, чиме се означавање бојом пребацује на то слово. Три ротора су приказана један поред другог, њихови улази и излази су означени бројевима од 1 до 26 и пермутација вредности на улазу и на излазу сваког ротора приказује интерне везе унутар ротора. Приказана су слова енглеског алфабета на улазима првог ротора и излазима трећег ротора.

4.2. ВИЗУЕЛНА РЕПРЕЗЕНТАЦИЈА СИМЕТРИЧНИХ КРИПТОГРАФСКИХ АЛГОРИТАМА

Код симетричних криптографских алгоритама који су подржани у COALA систему (DES и AES алгоритми) основна идеја визуелне репрезентације је била да

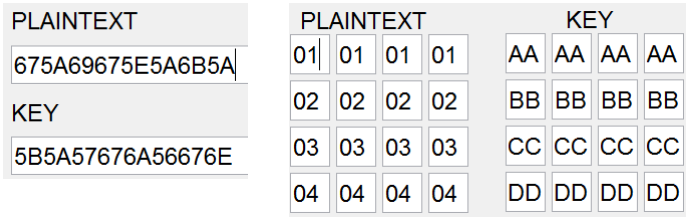
Табела 4.1. Основне разлике између DES и AES алгоритама

	DES	AES
Feistel структура	Да	Не
Величина порука	64 бита	128 бита
Величина кључа	56 бита	128/192/256 бита
Величина кључа итерације	48 бита	128 бита
Број итерација	16	10/12/14

се омогући праћење извршавања комплетног алгоритма за шифровање и за дешифровање са улазним подацима алгоритама који су истих димензија као у реалним применама. Било је потребно приказати извршавање сваке операције свих итерација алгоритма са максималним нивоом детаља у приказивању, али задржати прегледност у приказу. Улазни параметри алгоритама су лако променљиви, тако да се извршавање алгоритама може испратити на различитим примерима. Код AES алгоритма омогућена је визуелна репрезентација алгоритма са величином кључа од 128 бита.

DES и AES алгоритми спадају у групу симетричних блок алгоритама који оперишу над подацима у више итерације пре него што формирају шифровани блок података. Основне разлике између ова два алгоритма приказане су у табели 4.1. И поред разлика које постоје између ова два алгоритма, пронађен је начин да се осмисли униформни дизајн њихове визуелне репрезентације. Ово је важно због ефикасности визуелне репрезентације, као што је то објашњено у поглављу 2.

За оба алгоритма корисник има могућност уноса оригиналне поруке и кључа за шифровање, односно шифроване поруке и кључа за дешифровање. Код DES алгоритма величине поруке и кључа су нешто мање него код AES алгоритма и операције се извршавају на бинарном нивоу, па је било погодно приказ прилагодити бинарним вредностима. Ипак како би унос оригиналне поруке (или шифроване поруке) и кључа био једноставнији за корисника, омогућено је да се улазни подаци уносе у виду хексадецималних вредности (слика 4.5а). Код AES алгоритма величине поруке и кључа су нешто веће него код DES алгоритма и операције се извршавају на нивоу бајта, па је било погодно приказ прилагодити бајтовским вредностима. Унос оригиналне поруке (или шифроване поруке) и кључ–



а) б)

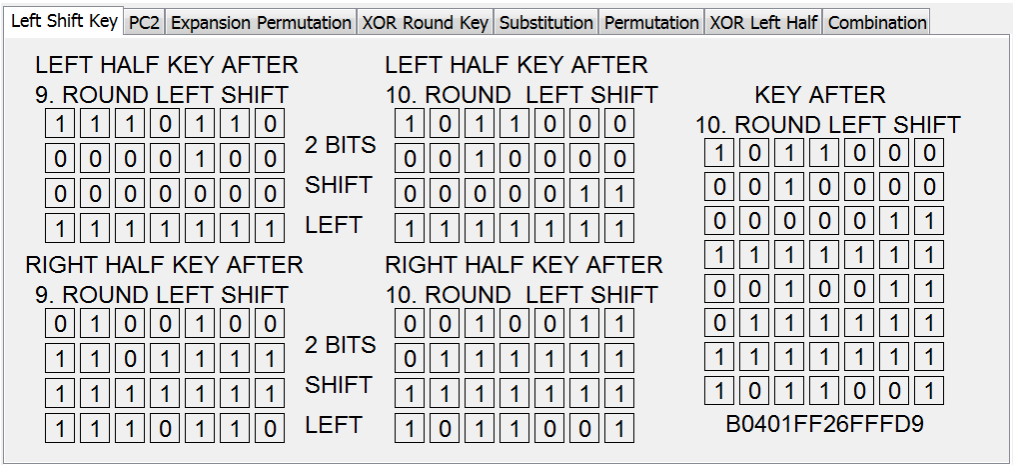
Слика 4.5. Визуелна репрезентација конфигурисања улазних параметара за DES и AES алгоритме у COALA систему

Слика 4.6. Визуелна репрезентација контролног дела за DES и AES алгоритме у COALA систему

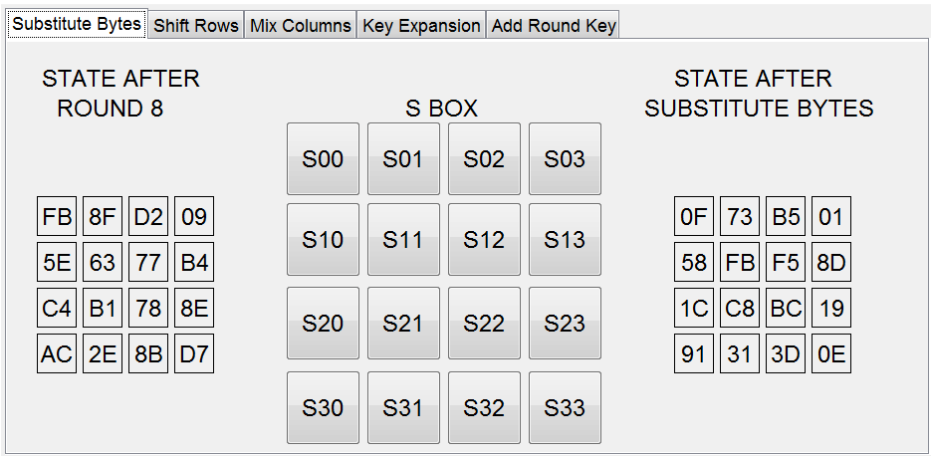
ча (слика 4.5б) омогућен је уносом хексадецималних вредности у виду две матрице 4x4 у којима свако поље представља по један бајт (састоји се од две хексадецималне цифре), као што је то дефинисано у самом алгоритму, описаном у поглављу 3. Алгоритам дефинише да се матрице попуњавају по колонама.

За оба алгоритма корисник има могућност да на почетку одабере да ли жели да прати извршавање шифровања или дешифровања одабраног алгоритма, након чега притиском на дугме *NEXT* прелази на екран за унос улазних параметара. Након уношења улазних параметара визуелна репрезентација извршавања алгоритма се добија притиском на дугме *NEXT*. Приликом праћења извршавања алгоритама корисник у доњем делу екрана има контролни део (слика 4.6) у коме има информацију о томе у којој итерацији се налази. Поред тога, у контролном делу налазе се и дугмићи *PREVIOUS ROUND*, које омогућава повратак на претходну итерацију у извршавању, *NEXT ROUND*, које омогућава прелазак на наредну итерацију у извршавању и *NEW SIMULATION*, које омогућава повратак на почетни екран и одабир неког другог алгоритма за визуелну репрезентацију.

С обзиром да је потребно приказати комплетан ток извршавања ових алгоритама, било је потребно осмислити погодан начин визуелне репрезентације. Један могући начин био би да се алгоритми визуелно прикажу у форми дијаграма, па да се на дијаграму прате међурезултати и детаљи примењених операција по итерацијама. С обзиром да оба алгоритма имају релативно велик број итерација и да се у оквиру итерација одвија више сложених операција, овакав начин визуелне



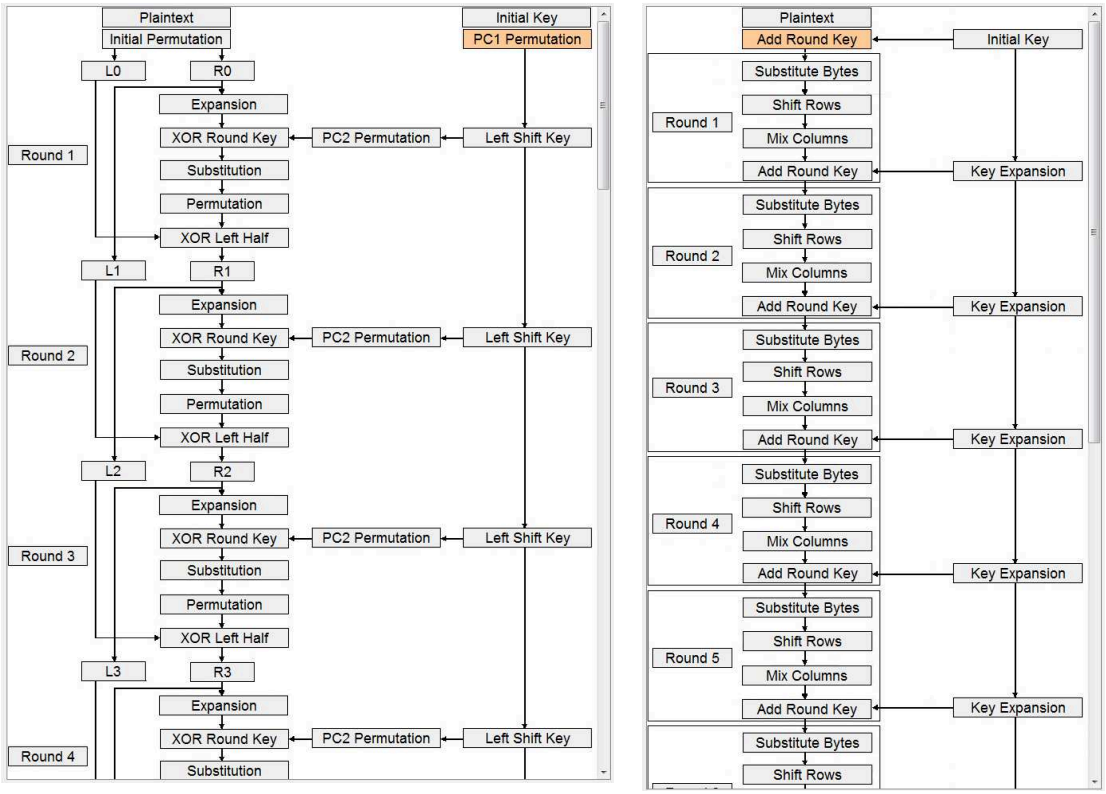
а)



б)

Слика 4.7. Визуелна репрезентација извршавања једне итерације за DES и AES алгоритме у COALA систему

репрезентације не би био довољно јасан, чиме би и ефикасност система била умањена. Други начин приказа је да се у једном тренутку прикаже једна итерација и то тако да је могуће видети детаље и међурезултате сваке операције у оквиру итерације, а да се омогући кориснику да се помера кроз итерације. Овакав начин приказа би свакако био јасан, али се овде губи на вези између система и дефиниције алгоритама, где су алгоритми представљени управо уз помоћ дијаграма. Из ових разлога у COALA систему искоришћен је комбиновани приступ, тако што су кориснику приказани детаљи извршавања једне итерације, праћени сликом дијаграма у коме је означена итерација која је приказана. Детаљи извршавања једне итерације за DES (слика 4.7а) и AES (слика 4.7б) алгоритме представљени су коришћењем засебних *tab*-ова за сваку операцију у оквиру ите–



а) б)

Слика 4.8. Визуелна репрезентација дијаграма за DES и AES алгоритме у COALA систему

A2	DD	08	77
70	16	CB	AD
F6	60	3A	AC
E8	E8	35	35

0	1	1	0	0	1	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
0	1	1	0	0	1	1	1
0	1	0	1	1	1	1	0
0	1	0	1	1	0	1	0
0	1	1	0	1	0	1	1
0	1	0	1	1	0	1	0

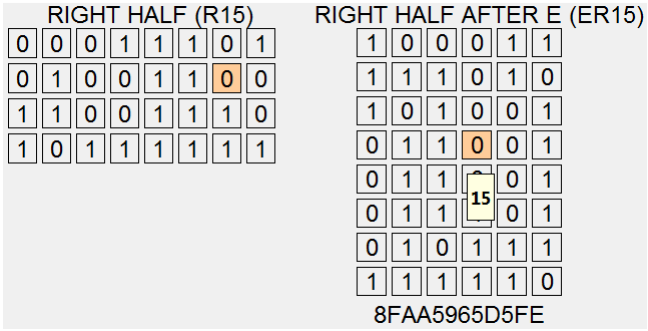
675A69675E5A6B5A

а) б)

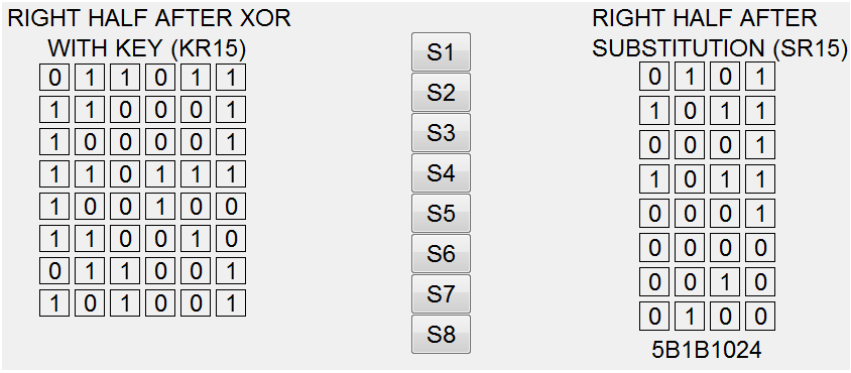
Слика 4.9. Визуелна репрезентација бинарних података за AES и DES алгоритме у COALA систему

рације. На тај начин је извршавање читаве итерације приказано у оквиру једног прозора, а корисник се лако може пребацити на праћење било које операције у оквиру итерације.

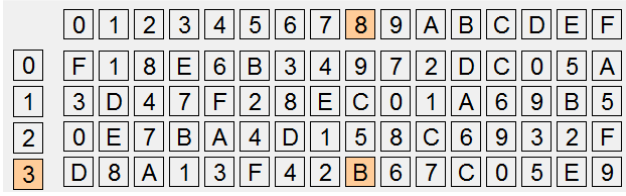
Дијаграм алгоритма чије се извршавање прати за DES (слика 4.8а) и AES (слика 4.8б) алгоритме приказује комплетну слику алгоритма, даје кориснику позицију на којој се налази у оквиру праћења извршавања алгоритма и омогућава додатну навигацију кроз праћење извршавања алгоритма.



а)



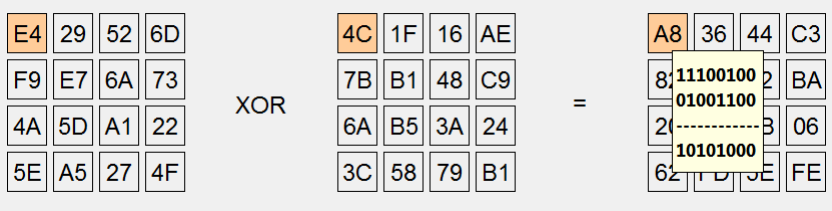
б)



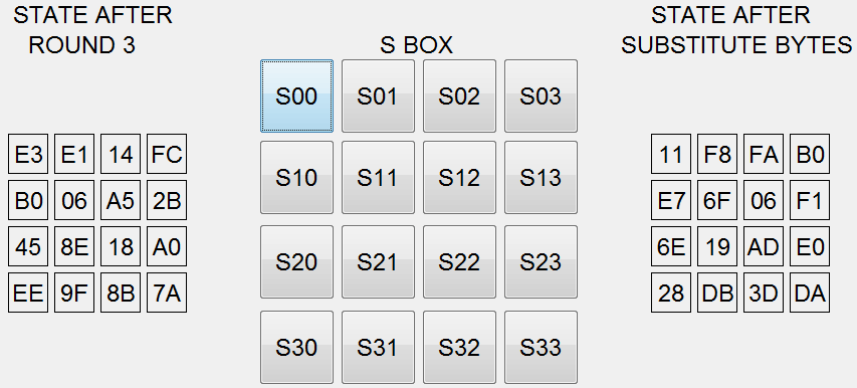
в)

Слика 4.10. Помоћна средства код визуелне репрезентације DES алгоритма у COALA систему

Једно битно питање приликом дизајна визуелне репрезентације алгоритама било је на који начин приказати бинарне податке над којима се ради. С обзиром да је код AES алгоритма дефинисано да се оригинална порука и кључ третирају као матрица 4x4 у којој свако поље представља по један бајт и вредности су приказане хексадецимално, било је логично да се преузме овакав приказ у визуелној репрезентацији (слика 4.9а). Са друге стране, код DES алгоритма оригинална порука и кључ су бинарне вредности и операције се извршавају на нивоу бита. Како би визуелна репрезентација код ових алгоритама била униформна и код DES алгоритма је примењена визуелна репрезентација података у виду матрице (слика 4.9б). У овом случају матрице су различитих димензија у зависности од њиховог значења у операцијама у којима се користе, а свако поље матрице представља по



а)



б)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

в)

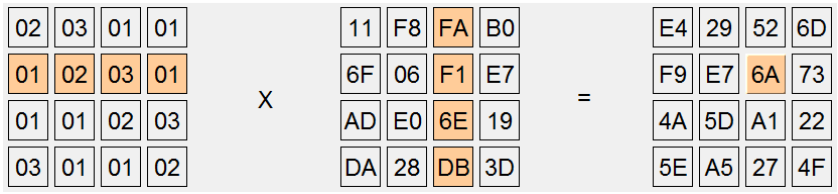
Слика 4.11. Помоћна средства код визуелне репрезентације AES алгоритма у COALA систему – први део

један бит. Како би праћење извршавања алгоритма било једноставније, поред

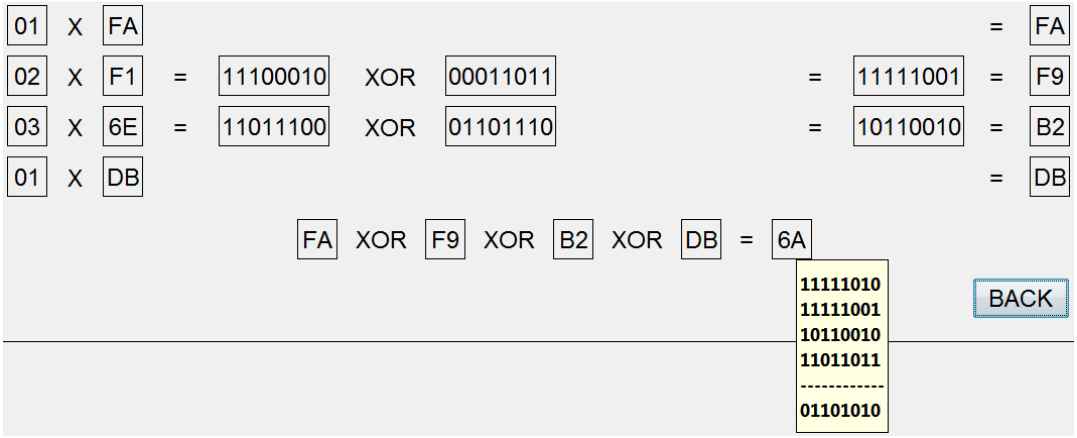
оваквог приказа код DES алгоритма постоје и хексадецималне вредности међурезултата испод матрица са бинарним вредностима.

Код DES алгоритма користи се комбинација боја и текстуалних описа како би се за сваку операцију у оквиру итерације приказало на основу које трансформације се дошло од почетне вредности до резултата операције (слика 4.10а). Омогућено је кориснику да позиционирањем миша на било које поље било које матрице у оквиру неке итерације види на који начин се дошло до одговарајућег поља у резултујућој матрици. Текстуални опис се користи како би се приказало која је била позиција одабраног бита у оригиналној матрици, чиме се индиректно приказују табеле које се користе код различитих операција DES алгоритма за трансформације. За операцију замене која постоји код DES алгоритма (слика 4.10б) омогућено је приступање свакој од осам дефинисаних табела пресликавања притиском на одговарајуће дугме и праћење добијања резултата за одабрану табелу (слика 4.10в).

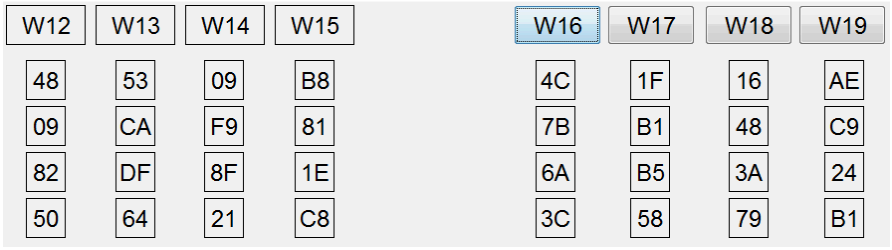
Код AES алгоритма користи се комбинација боја и текстуалних описа код операције која извршава ексклузивно ИЛИ над подацима (слика 4.11а). Омогућено је кориснику да позиционирањем миша на било које поље било које матрице у оквиру ове операције види шта су операнди, а шта резултат. Текстуални опис се користи како би се приказало на бинарном нивоу како изгледа примена ексклузивно ИЛИ операције над селектованим операндима. Код операције померања редова користи се само боја да се означи како је добијен бајт резултујуће матрице ове операције. Означавање бојом се појављује када се позиционира миш на било које поље било које матрице. За операцију замене која постоји код AES алгоритма (слика 4.11б) омогућено је приступање дефинисаној табели пресликавања за сваки бајт притиском на одговарајуће дугме и праћење добијања резултата за одабрани бајт (слика 4.11в). За операцију у којој се врши множење матрица која постоји код AES алгоритма позиционирањем миша на било које поље резултујуће матрице бојом је означен начин добијања те вредности (слика 4.12а), а омогућено је и притиском миша на било које поље видети детаљан приказ извршавања дела множења матрица којим је добијен тај бајт (слика 4.12б). За операцију добијања кључа итерације која постоји код AES алгоритма приказан је претходни кључ итерације и текући кључ итерације (слика 4.12в), а притиском



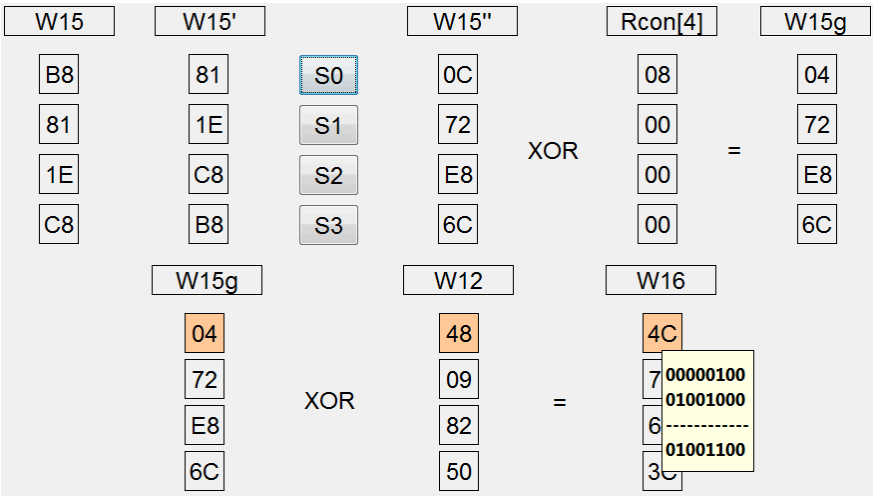
а)



б)



в)



г)

Слика 4.12. Помоћна средства код визуелне репрезентације AES алгоритма у COALA систему – други део

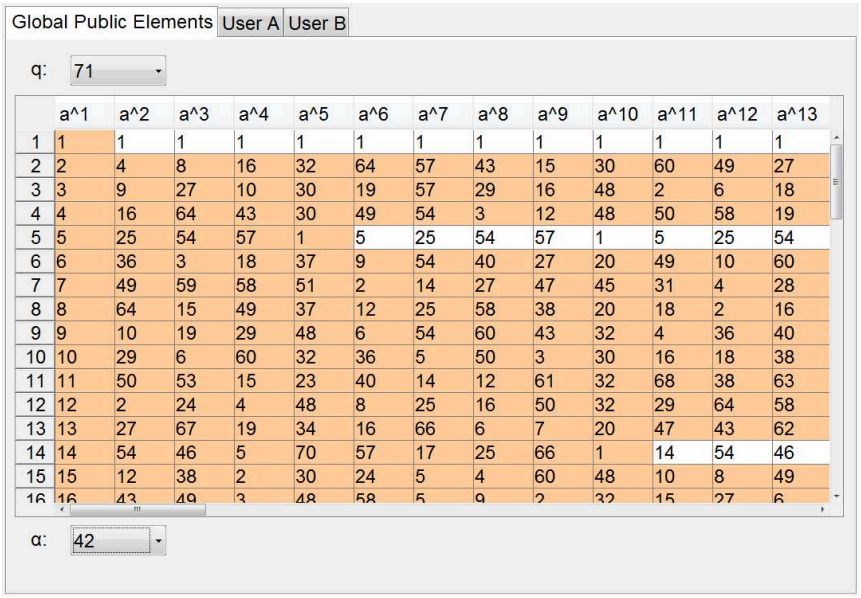
на дугме изнад сваке колоне текућег кључа итерације могуће је видети како је формирана та колона (слика 4.12г).

4.3. ВИЗУЕЛНА РЕПРЕЗЕНТАЦИЈА КРИПТОГРАФСКИХ АЛГОРИТАМА СА ЈАВНИМ КЉУЧЕМ

Код криптографских алгоритама са јавним кључем који су подржани у COALA систему (Diffie-Hellman и RSA алгоритми) основна идеја визуелне репрезентације је била да се омогући праћење извршавања комплетног алгоритма за шифровање и за дешифровање са улазним подацима алгоритама који су димензија погодних како би се приказале најважније карактеристике ових алгоритама. Разлог због чега и овде нису коришћени параметри који би били величина као у реалним применама је што би приказ таквог извршавања алгоритама био непрактичан и тежак за праћење (имајући у виду да ови параметри у реалним применама имају по 1024 бита и више). Било је потребно приказати начин формирања парова јавних и приватних кључева, као и алгоритам који се користи за ефикасну експонентизацију приликом шифровања и дешифровања. Улазни параметри алгоритама су лако променљиви, тако да се извршавање алгоритама може испратити на различитим примерима.

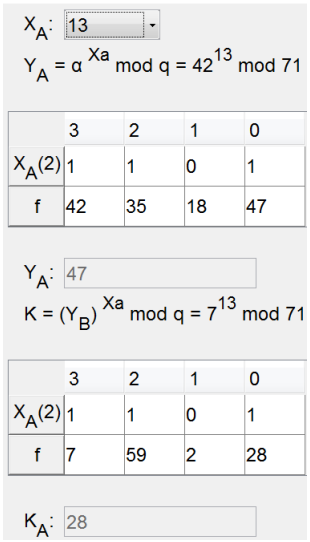
Diffie-Hellman и RSA спадају у групу алгоритама за шифровање са јавним кључем. Кључна разлика између ових алгоритама је у томе што Diffie-Hellman може да се користи само за сигурну размену вредности између два корисника, док се RSA може користити за шифровање и дешифровање података. Иако ова два алгоритма нису иста ипак је било могуће осмислити униформан начин визуелне репрезентације.

Код Diffie-Hellman алгоритма приказ визуелне репрезентације смештен је на један екран (слика 4.13), али подељен у три *tab*-а при чему један приказује одабир глобалних јавних елемената, а друга два приказују рад алгоритма за сваког од два учесника у комуникацији посебно. Приликом одабира глобалних јавних елемената (слика 4.13) кориснику је остављено да из падајуће листе одабере прост цео број за q. Листа је ограничена на просте бројеве до 1000, из раније наведених разлога. За одабрану вредност q визуелно помоћу табеле приказује се скуп свих могућих вредности за α и исте те вредности су у понуди за избор за вредност α из падајуће листе. Табела приказује по редовима све бројеве од 1 до q-1, док су по колонама



	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	4	8	16	32	64	57	43	15	30	60	49	27
3	3	9	27	10	30	19	57	29	16	48	2	6	18
4	4	16	64	43	30	49	54	3	12	48	50	58	19
5	5	25	54	57	1	5	25	54	57	1	5	25	54
6	6	36	3	18	37	9	54	40	27	20	49	10	60
7	7	49	59	58	51	2	14	27	47	45	31	4	28
8	8	64	15	49	37	12	25	58	38	20	18	2	16
9	9	10	19	29	48	6	54	60	43	32	4	36	40
10	10	29	6	60	32	36	5	50	3	30	16	18	38
11	11	50	53	15	23	40	14	12	61	32	68	38	63
12	12	2	24	4	48	8	25	16	50	32	29	64	58
13	13	27	67	19	34	16	66	6	7	20	47	43	62
14	14	54	46	5	70	57	17	25	66	1	14	54	46
15	15	12	38	2	30	24	5	4	60	48	10	8	49
16	16	43	49	3	48	58	5	9	2	32	15	27	6

Слика 4.13. Визуелна репрезентација одабира глобалних јавних елемената код Diffie-Hellman алгоритма у COALA систему



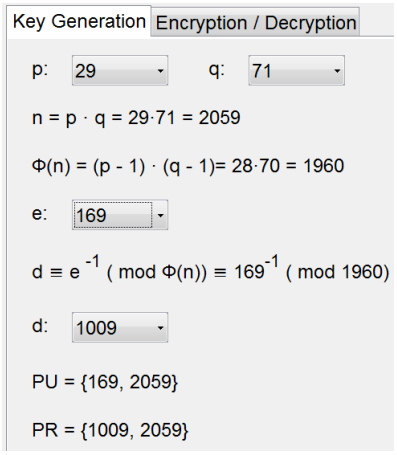
	3	2	1	0
$X_A(2)$	1	1	0	1
f	42	35	18	47

	3	2	1	0
$X_A(2)$	1	1	0	1
f	7	59	2	28

Слика 4.14. Визуелна репрезентација извршавања Diffie-Hellman алгоритма за једног од учесника у комуникацији у COALA систему

сви бројеви степеновани редом степенима од 1 до q-1, али по модулу q. У сваком реду бојом је означена најдужа секвенца без понављања. Само редови код којих је читава секвенца означена бојом, односно у којој нема понављања представљају просте корене за q и могу бити одабрани као вредност за α.

За израчунавање заједничке тајне вредности између два корисника (слика 4.14), након формирања глобалних јавно доступних вредности, потребно је да сваки корисник одабере своју тајну вредност, што је омогућено коришћењем падајућих листи у којима су понуђени бројеви од 1 до q-1 који долазе у обзир. Затим сваки

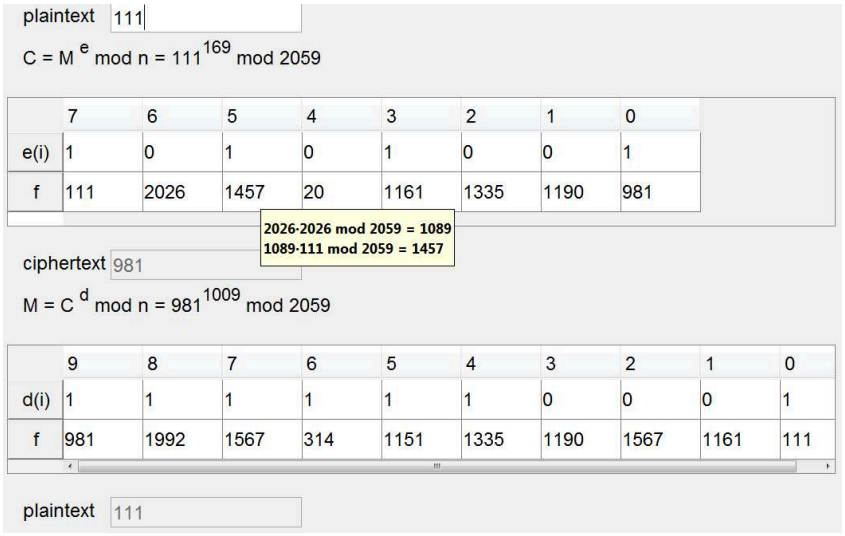


Слика 4.15. Визуелна репрезентација формирања пара кључева код RSA алгоритма у COALA систему

корисник за себе израчунава своју јавну вредност на основу претходно одабране тајне вредности. На крају сваки корисник за себе, на основу своје тајне вредности и јавне вредности другог корисника израчунава заједничку тајну вредност. Визуелно је приказана примена алгоритма за брзу експонентизацију приликом израчунавања јавне вредности и заједничке тајне вредности за сваког корисника.

Код RSA алгоритма приказ визуелне репрезентације смештен је на један екран (слика 4.15), али подељен у два *tab*-а при чему један приказује формирање пара јавни и приватни кључ, а други приказује рад алгоритма приликом шифровања помоћу јавног кључа, односно дешифровања помоћу одговарајућег приватног кључа. Приликом формирања пара кључева (слика 4.15) кориснику је остављено да из падајућих листа одабере два проста цела броја за p и q. Листе су ограничене на просте бројеве до 1000, из раније наведених разлога. За одабране вредности за p и q приказано је израчунавање n и $\Phi(n)$ и у падајућој листи за e приказане су само вредности које испуњавају услове дефинисане самим алгоритмом. Након што корисник одабере вредност за e у падајућој листи за d се појављују само вредности које испуњавају услове дефинисане алгоритмом. Након што корисник одабере и вредност за d формиран је пар приватни и јавни кључ који је и приказан.

За приказ шифровања и дешифровања са одабраним паром кључева (слика 4.16) потребно је да корисник унесе вредност оригиналне поруке. Након уноса вредности приказује се поступак шифровања, добијена шифрована вредност, као и поступак дешифровања и добијена оригинална вредност све у оквиру једног екрана. За израчунавање шифроване, односно оригиналне поруке визуелно је при–

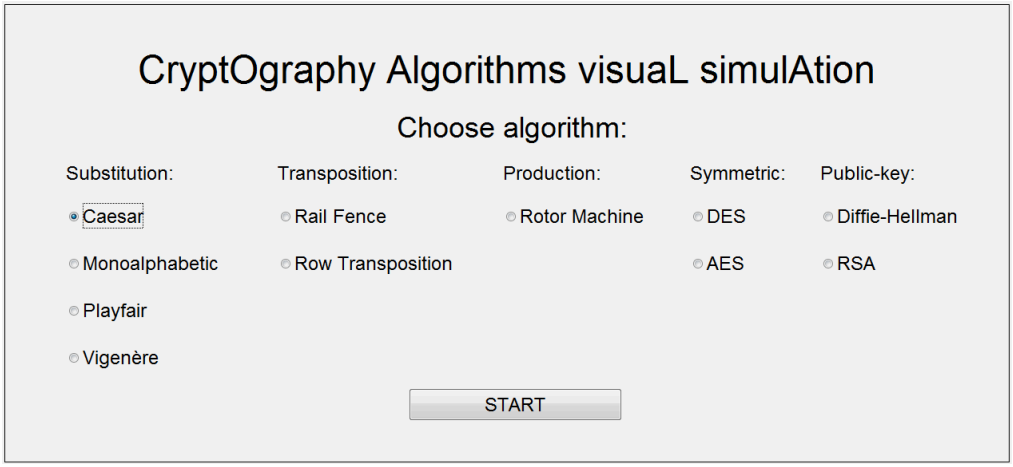


Слика 4.16. Визуелна репрезентација извршавања RSA алгоритма у COALA систему

казана примена алгоритма за брзу експонентизацију. Приказ подразумева табелу у којој свака колона означава по једну итерацију алгоритма, при чему су итерације означене у опадајућем редоследу. Први ред у табели попуњен је бинарном вредношћу експонента из израза за који се прави табела и то тако што је по један бит у свакој колони, почевши од бита највеће тежине. Други ред представља међурезултате у свакој итерацији, при чему вредност из последње (нулте) итерације представља и коначну вредност. Позиционирањем миша на било коју колону табеле, може се добити приказ израза који су коришћени у одабраној итерацији и на основу којих је добијен приказани међурезултат у виду текстуалног описа.

5. ОПИС COALA СИСТЕМА

У овом поглављу је приказан опис COALA система за визуелну репрезентацију криптографских алгоритама који је реализован. Систем има могућност визуелне репрезентације пет врста криптографских алгоритама: супституционих алгоритама, транспозиционих алгоритама, продукционих алгоритама, симетричних блок алгоритама и алгоритама са јавним кључем, који су описани у поглављу 3. Слика 5.1 приказује главни екран COALA система на коме се може одабрати жељени алгоритам за који треба приказати визуелну репрезентацију.



Слика 5.1. Почетни екран COALA система

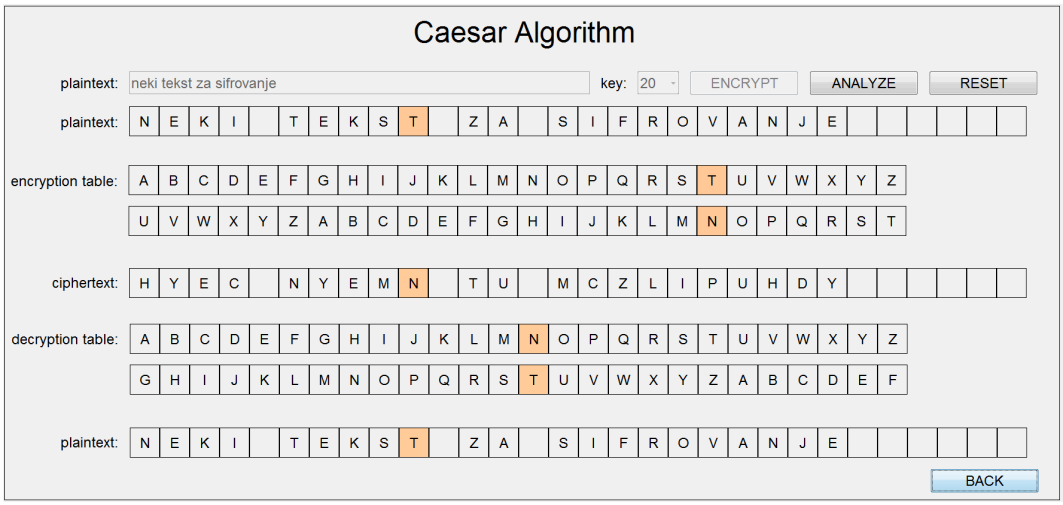
5.1. СУПСТИТУЦИОНИ АЛГОРИТМИ

У овом поглављу се разматра визуелна репрезентација супституционих алгоритама подржаних у COALA систему: Цезар алгоритам, моноалфабетски алгоритам, *Playfair* алгоритам и *Vigenere* алгоритам, као што се види на слици 5.1.

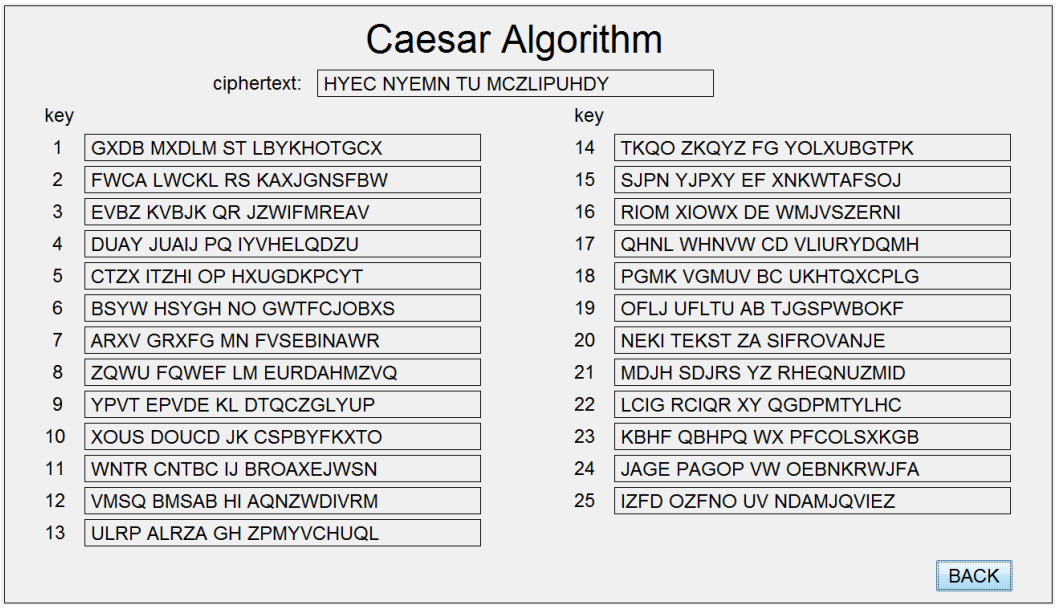
5.1.1. ЦЕЗАР АЛГОРИТАМ

На слици 5.2 приказан је екран за визуелну репрезентацију Цезар алгоритма у COALA систему, који се добија када се на почетном екрану одабере Цезар алгоритам и притисне дугме *START*.

Корисник има могућност да унесе текст који жели да шифрује и да изабере кључ који жели да користи. Притиском на дугме *ENCRYPT* добија се приказ поступка шифровања оригиналне поруке на основу табеле пресликавања, као и поступка дешифровања поруке на основу инверзне табеле пресликавања. С обзи–

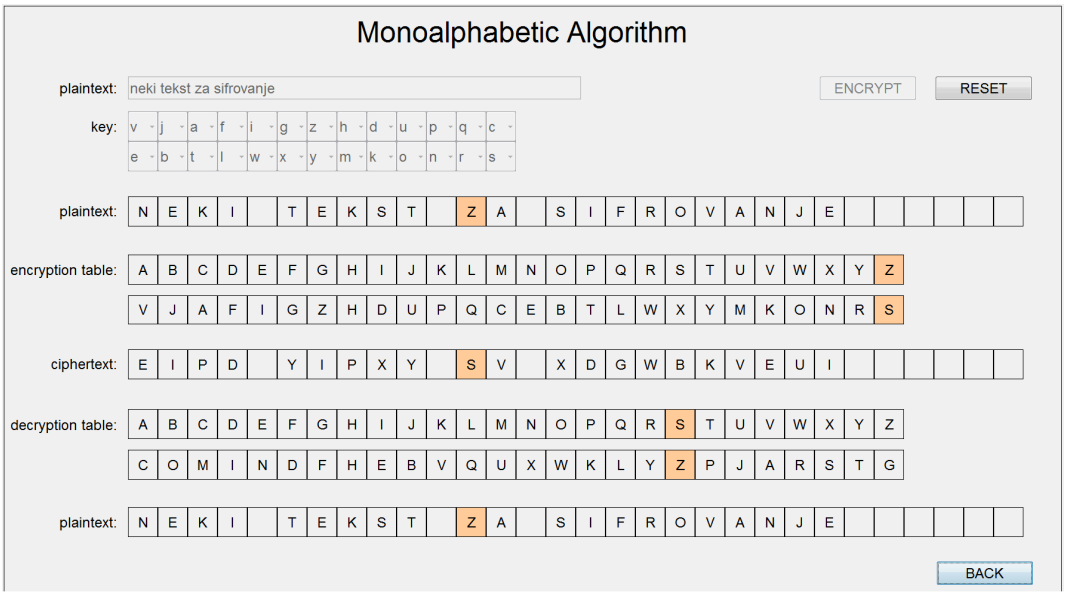


Слика 5.2. Екран за визуелну репрезентацију Цезар алгоритма у COALA систему



Слика 5.3. Екран са приказом криптоанализе шифроване поруке из примера са слике 5.2

ром да се ради о једноставном алгоритму овде је погодно приказати и поступак дешифровања, који се увек своди на инверзне операције у односу на шифровање, на истом екрану на ком је приказано и шифровање. Ради бољег разумевања, омогућено је да корисник може позиционирањем показивача миша на екрану на неко од слова оригиналне или шифроване поруке да добије додатно бојом означен поступак трансформације тог слова, као што је и приказано на слици 5.2. Цезар алгоритам је карактеристичан и по томе што постоји свега 25 различитих кључева који се могу применити за шифровање неке поруке. Због тога је ово добар тренутак да се демонстрира и како се може урадити криптоанализа уколико алгоритам није довољно безбедан. Приказ криптоанализе добија се притиском на



Слика 5.4. Екран за визуелну репрезентацију моноалфабетског алгоритма у COALA систему

дугме *ANALYZE*. За пример са слике 5.2 екран са приказом криптоанализе шифроване поруке приказан је на слици 5.3.

Код Цезар алгоритма, с обзиром да је мали скуп могућих кључева, криптоанализа се своди на испробавање свих могућих кључева, што је и приказано на слици 5.3. Као што видимо један од испробаних кључева даће оригиналну поруку (у овом случају то је кључ 20, који је претходно био изабран у примеру са слике 5.2) и само је потребно препознати која од 25 могућности би могла да буде оригинална порука.

Корисник на располагању има још и опције: *RESET* која враћа визуелну репрезентацију Цезар алгоритма у почетно стање тако да се може унети нова порука за шифровање и одабрати нова вредност кључа и *BACK* која враћа корисника на почетни екран тако да може одабрати нов алгоритам за визуелну репрезентацију.

5.1.2. МОНОАЛФАБЕТСКИ АЛГОРИТАМ

На слици 5.4 приказан је екран за визуелну репрезентацију моноалфабетског алгоритма у COALA систему, који се добија када се на почетном екрану одабере моноалфабетски алгоритам и притисне дугме *START*.

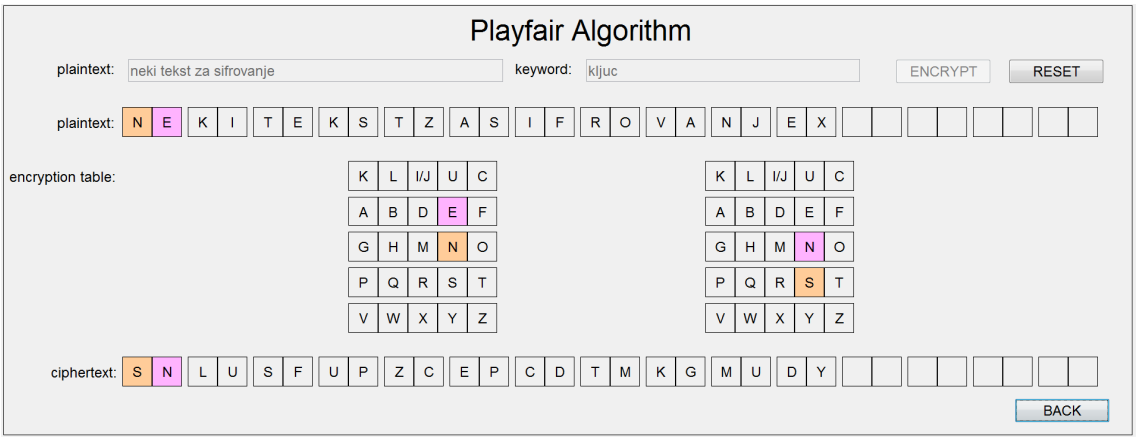
Код моноалфабетског алгоритма корисник може произвољно да изабере табелу пресликавања коју ће користити за шифровање, што је у систему омогућено одабиром 26 различитих слова кључа. Како би се смањила могућност грешке

корисника приликом одабира кључа, користе се падајуће листе за унос кључа. Падајуће листе садрже сва слова енглеског алфабета, али сваки пут када се у некој од листи одабере одређено слово, све остале листе се ажурирају тако што се из њих избацује одабрано слово и остављају само она која су и даље на располагању за одабир. Корисник затим уноси текст за шифровање и притиском на дугме *ENCRYPT* добија приказ поступка шифровања, као и поступка дешифровања, уз одговарајуће табеле пресликавања, слично као и за Цезар алгоритам. И овде је омогућено кориснику да позиционирањем показивача миша на екрану на одређено слово оригиналне или шифроване поруке добије додатно обележено како је извршена трансформација одабраног слова. Корисник на располагању има још и опције: *RESET* која враћа визуелну репрезентацију моноалфабетског алгоритма у почетно стање тако да се може унети нова порука за шифровање и одабрати нова вредност кључа и *BACK* која враћа корисника на почетни екран тако да може одабрати нов алгоритам за визуелну репрезентацију.

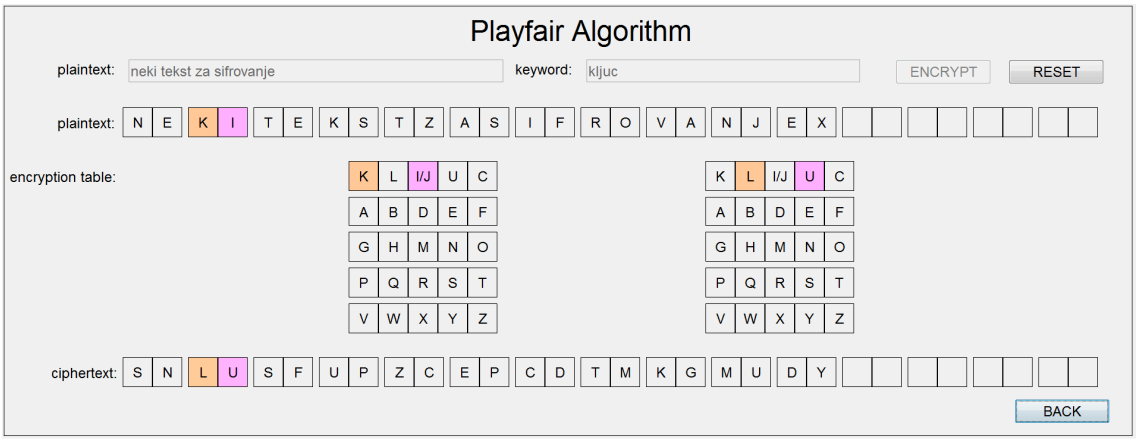
5.1.3. *PLAYFAIR* АЛГОРИТАМ

На слици 5.5 приказан је екран за визуелну репрезентацију *Playfair* алгоритма у COALA систему, који се добија када се на почетном екрану одабере *Playfair* алгоритам и притисне дугме *START*.

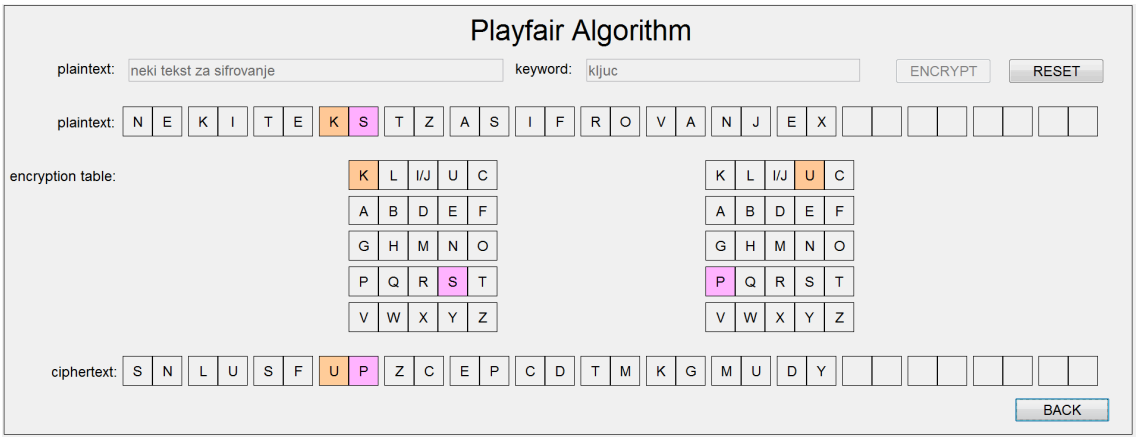
Код *Playfair* алгоритма оригинална порука се дели на слогове и затим се слогови према дефинисаним правилима шифрују помоћу матрице за шифровање која је формирана на основу одабране кључне речи, као што је приказано на слици 5.5. Кориснику је омогућено да унесе текст за шифровање, као и кључну реч која ће се користити за шифровање. Затим се притиском на дугме *ENCRYPT* добија приказ поступка шифровања. Матрица за шифровање, која је попуњена на основу кључне речи, приказана је два пута како би могло визуелно да се види примењено правило за шифровање једног слога. Кориснику је омогућено да позиционирањем показивача миша на екрану на одређени слог оригиналне или шифроване поруке, може да добије графички приказ примењеног правила за шифровање означеног слога, при чему је свако слово означено различитом бојом ради боље прегледности. На слици 5.5а, 5.5б и 5.5в видимо примену сва три правила која се користе код *Playfair* алгоритма. Корисник на располагању има још и опције: *RESET* која враћа визуелну репрезентацију *Playfair* алгоритма у почетно стање та–



а)



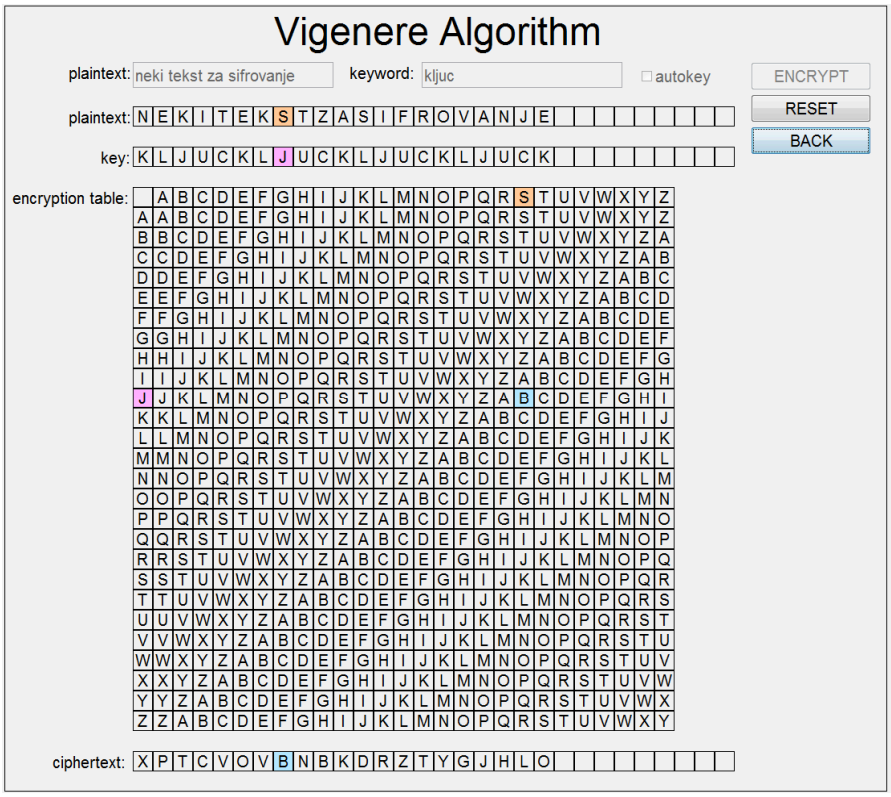
б)



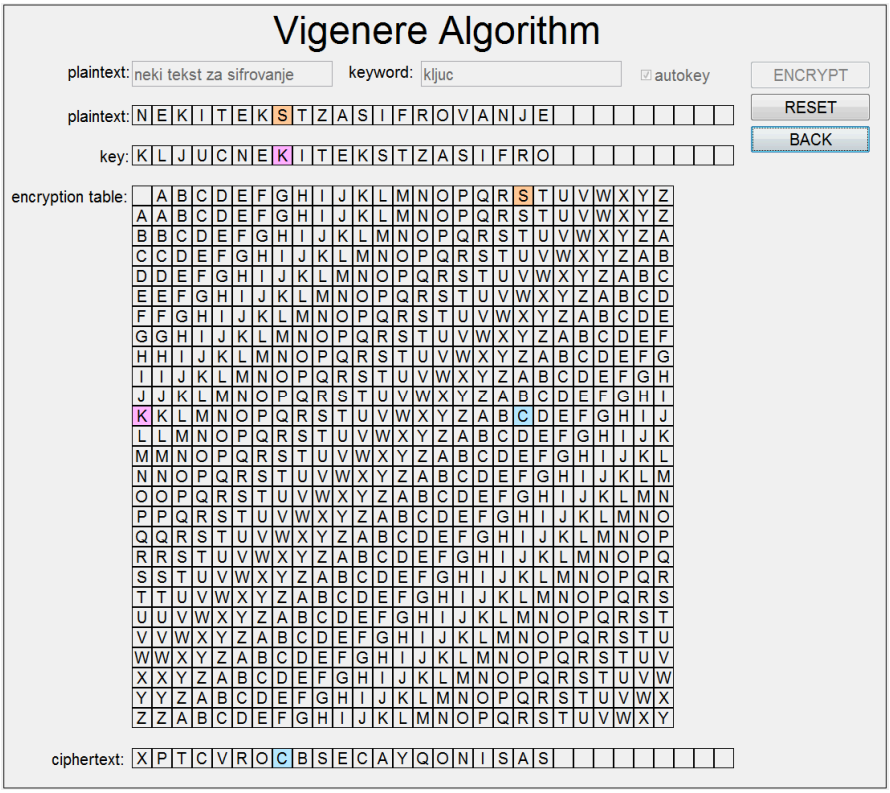
в)

Слика 5.5. Екран за визуелну репрезентацију *Playfair* алгоритма у COALA систему

ко да се може унети нова порука за шифровање и одабрати нова вредност кључа и *BACK* која враћа корисника на почетни екран тако да може одабрати нов алгоритам за визуелну репрезентацију.



а)



б)

Слика 5.6. Екран за визуелну репрезентацију *Vigenere* алгоритма у COALA систему

5.1.4. *VIGENERE* АЛГОРИТАМ

На слици 5.6 приказан је екран за визуелну репрезентацију *Vigenere* алгоритма у COALA систему, који се добија када се на почетном екрану одабере моноалфабетски алгоритам и притисне дугме *START*.

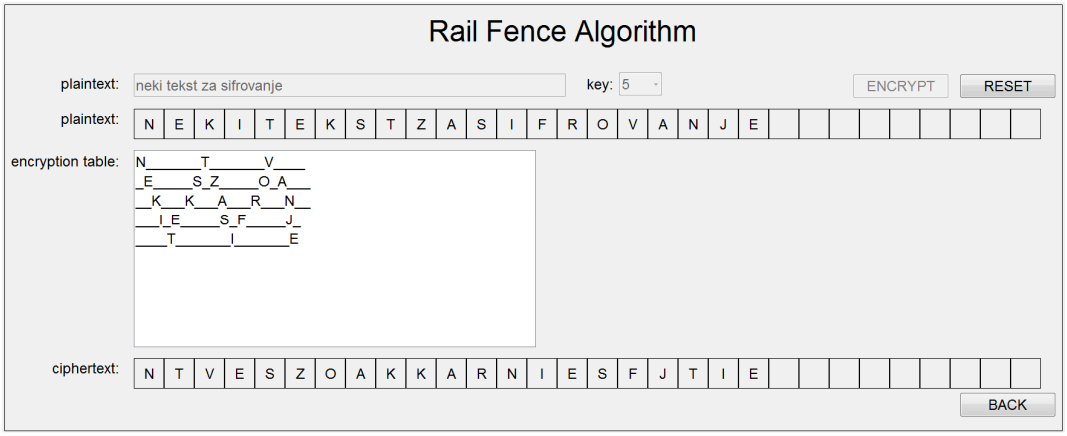
Кориснику је омогућено да унесе текст који жели да шифрује, да унесе кључ који жели да користи за шифровање и да одабере да ли жели да користи autokey побољшање или не, као што је приказано на слици 5.6. Притиском на дугме *ENCRYPT* добија се приказ поступка шифровања. Код *Vigenere* алгоритма за свако слово оригиналне поруке користи се друга табела пресликавања, па је због тога приказана матрица која приказује свих 26 могућих табела пресликавања. Слово оригиналне поруке одређује колону у матрици, а слово кључа одређује ред у матрици и из матрице се добија слово шифроване поруке. Претходно се кључ формира тако да буде исте дужине као и порука и то тако што се у оригиналној варијанти алгоритма одабрана кључна реч надовезује потребан број пута, односно у варијанти са аутокеу побољшањем се на одабрану кључну реч додаје оригинална порука до потребне дужине. Како би визуелна прегледност била боља, кориснику је омогућено да позиционирањем показивача миша на екрану на одређено слово оригиналне или шифроване поруке добије бојом означену трансформацију одабраног слова, као што је приказано на слици 5.6. Корисник на располагању има још и опције: *RESET* која враћа визуелну репрезентацију *Vigenere* алгоритма у почетно стање тако да се може унети нова порука за шифровање и одабрати нова вредност кључа и *BACK* која враћа корисника на почетни екран тако да може одабрати нов алгоритам за визуелну репрезентацију.

5.2. ТРАНСПОЗИЦИОНИ АЛГОРИТМИ

У овом поглављу се разматра визуелна репрезентација транспозиционих алгоритама подржаних у COALA систему: *Rail Fence* алгоритам и *Row Transposition* алгоритам, као што је приказано на слици 5.1.

5.2.1. *RAIL FENCE* АЛГОРИТАМ

На слици 5.7 приказан је екран за визуелну репрезентацију *Rail Fence* алгоритма у COALA систему, који се добија када се на почетном екрану одабере



Слика 5.7. Екран за визуелну репрезентацију *Rail Fence* алгоритма у COALA систему

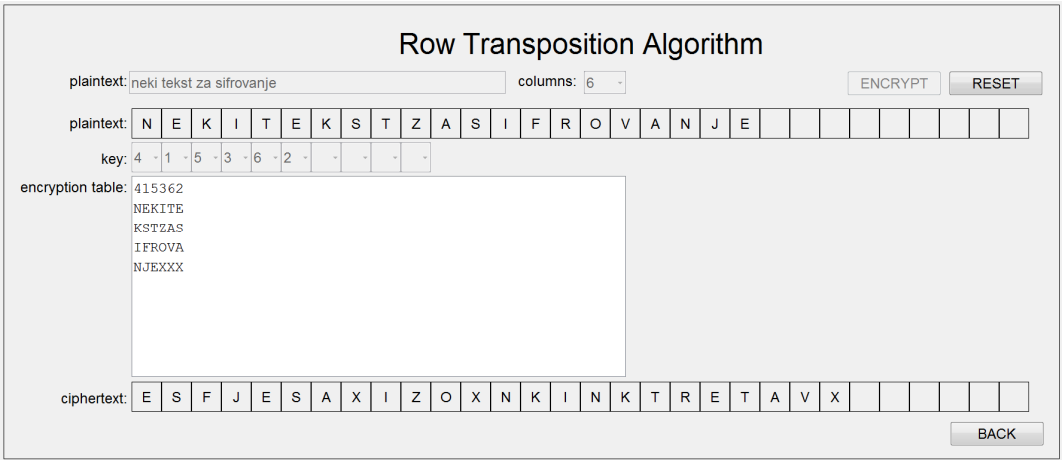
моноалфабетски алгоритам и притисне дугме *START*.

Корисник има могућност да унесе текст који жели да шифрује и да одабере кључ који жели да употреби за шифровање. Код овог алгоритма кључ је заправо број редова у којима ће бити исписана оригинална порука. На слици 5.7 приказан је пример рада овог алгоритма. Оригинална порука исписана је дијагонално у цик цак на доле, па на горе и види се облик по коме је алгоритам и добио назив. Шифрована порука добијена је ишчитавањем оригиналне поруке, која је исписана на претходно описани начин, ред по ред. Корисник на располагању има још и опције: *RESET* која враћа визуелну репрезентацију *Rail Fence* алгоритма у почетно стање тако да се може унети нова порука за шифровање и одабрати нова вредност кључа и *BACK* која враћа корисника на почетни екран тако да може одабрати нов алгоритам за визуелну репрезентацију.

5.2.2. *ROW TRANSPOSITION* АЛГОРИТАМ

На слици 5.8 приказан је екран за визуелну репрезентацију *Row Transposition* алгоритма у COALA систему, који се добија када се на почетном екрану одабере моноалфабетски алгоритам и притисне дугме *START*.

Корисник има могућност да унесе текст који жели да шифрује, да одабере број колона који жели да користи у кључу и затим да зада редослед колона што практично представља кључ у овом алгоритму, као што је приказано на слици 5.8. Код овог алгоритма оригинална порука се исписује ред по ред у онолико колона колико је корисник одабрао. Шифрована порука добија се тако што се порука која је исписана на претходно описани начин очита колону по колону и то оним редоследом који је задат кључем. Корисник на располагању има још и опције:



Слика 5.8. Екран за визуелну репрезентацију *Row Transposition* алгоритма у COALA систему

RESET која враћа визуелну репрезентацију *Row Transposition* алгоритма у почетно стање тако да се може унети нова порука за шифровање и одабрати нова вредност кључа и *BACK* која враћа корисника на почетни екран тако да може одабрати нов алгоритам за визуелну репрезентацију.

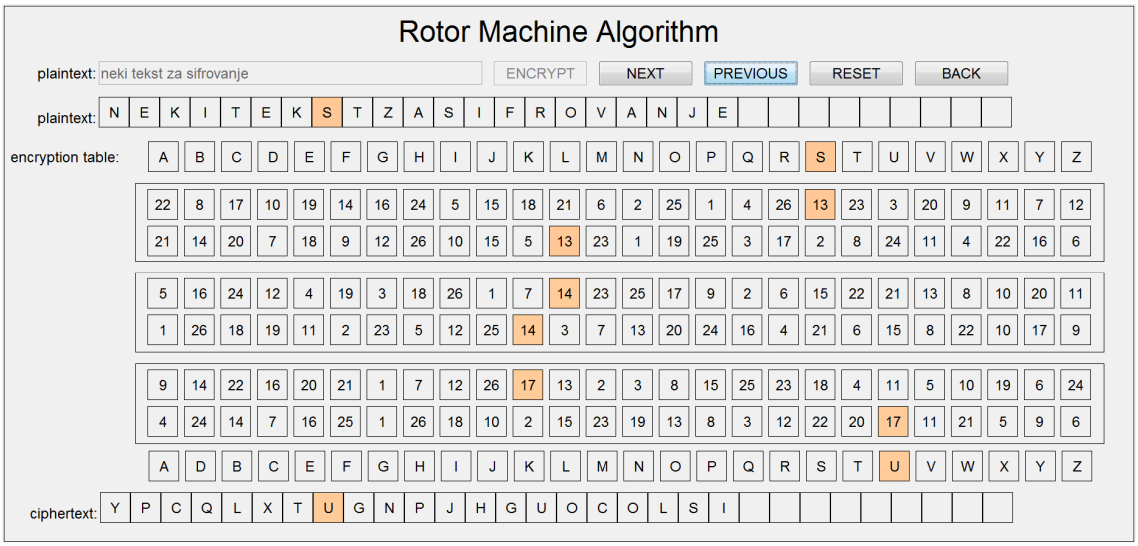
5.3. ПРОДУКЦИОНИ АЛГОРИТМИ

У овом поглављу се разматра визуелна репрезентација продукционог алгоритма подржаног у COALA систему: *Rotor Machine* алгоритма, као што је приказано на слици 5.1.

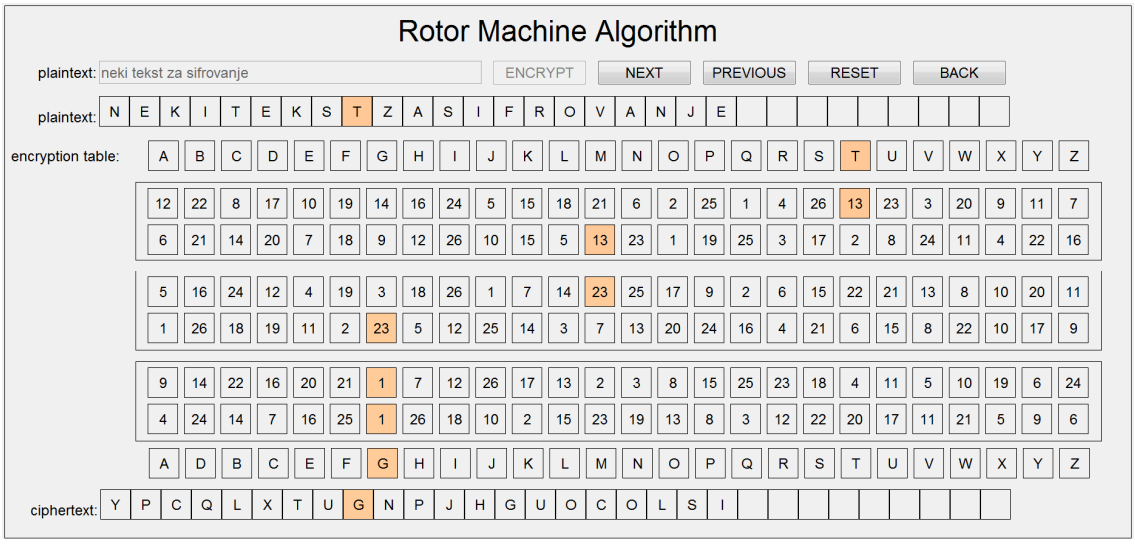
5.3.1. *ROTOR MACHINE* АЛГОРИТАМ

На слици 5.9 приказан је екран за визуелну репрезентацију *Rotor Machine* алгоритма у COALA систему, који се добија када се на почетном екрану одабере моноалфабетски алгоритам и притисне дугме *START*.

Корисник има могућност да унесе текст који жели да шифрује и притиском на дугме *ENCRYPT* добија приказ поступка шифровања, као што је приказано на слици 5.9. Приказана је ротор машина са три ротора: брзо, средње и споро ротирајући. Брзо ротирајући ротор ротира за једно место након шифровања сваког слова, средње ротирајући ротор ротира за једно место сваки пут када брзо ротирајући ротор направи цео циклус и споро ротирајући ротор ротира за једно место сваки пут када средње ротирајући ротор направи цео циклус. Овде се корисник кроз поступак шифровања креће користећи дугме *NEXT* за прелазак на



Слика 5.9. Екран за визуелну репрезентацију *Rotor Machine* алгоритма у COALA систему



Слика 5.10. Наредна итерација *Rotor Machine* алгоритма у COALA систему након итерације у примеру са слике 4.8

наредно слово оригиналне поруке и притиском на дугме *PREVIOUS* за повратак на претходно слово оригиналне поруке. Овај пут додатно означавање бојом је константно приказано за слово на коме је тренутно заустављен приказ поступка шифровања. Види се на који начин слово оригиналне поруке пролази кроз сваки од три ротора и шта се добија као слово шифроване поруке. У примеру са слике 5.9, када се притисне дугме *NEXT* добија се приказ са слике 5.10. Корисник на располагању има још и опције: *RESET* која враћа визуелну репрезентацију *Rotor Machine* алгоритма у почетно стање тако да се може унети нова порука за шифровање и одабрати нова вредност кључа и *BACK* која враћа корисника на почетни екран тако да може одабрати нов алгоритам за визуелну репрезентацију.

Слика 5.11. Екран за избор шифровања или дешифровања код DES алгоритма у COALA систему

5.4. СИМЕТРИЧНИ БЛОК АЛГОРИТМИ

У овом поглављу се разматра визуелна репрезентација симетричних блок алгоритама подржаних у COALA систему: DES алгоритам и AES алгоритам, као што је приказано на слици 5.1.

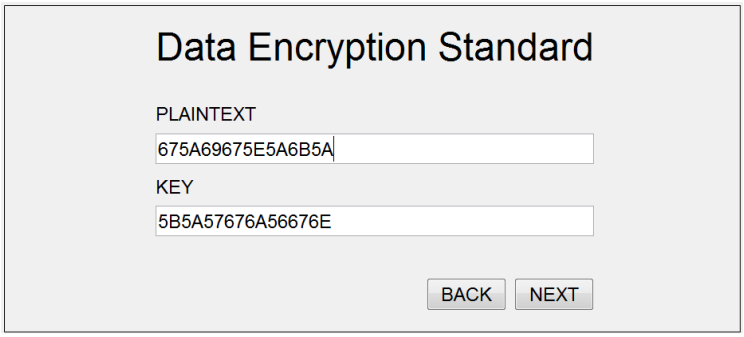
5.4.1. DES АЛГОРИТАМ

Када корисник изабере визуелну репрезентацију DES алгоритма на почетном екрану, најпре се појављује екран на коме се може изабрати да ли треба да се приказује шифровање или дешифровање (Слика 5.11).

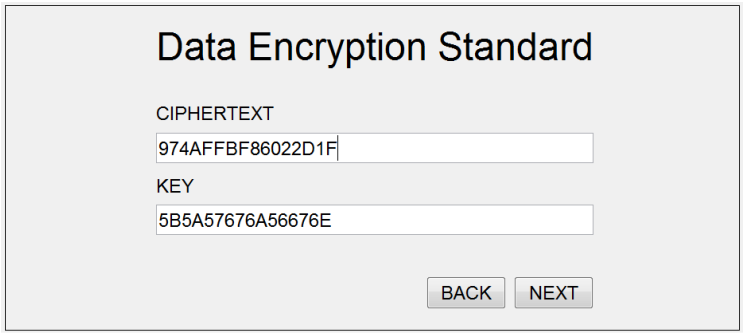
Након тога, појављује се екран за унос оригиналне поруке и кључа (Слика 5.12а) или екран за унос шифроване поруке и кључа (Слика 5.12б), у зависности од тога да ли је одабрано да се прати шифровање или дешифровање. Вредности оригиналне поруке или шифроване поруке и кључа се уносе у као бројеви у хексадецималном формату.

Након уноса улазних података приказује се екран на коме се види извршавање прве итерације алгоритма (слика 5.13). Извршавање алгоритма је презентовано тако да се у једном тренутку може видети једна итерација алгоритма. Ово омогућава корисницима да могу да прате извршавање алгоритма до најситнијих детаља. С обзиром да се итерација DES алгоритма састоји из неколико операција, за сваку операцију постоји посебан *tab* у прозору који приказује итерацију. У доњем делу екрана налази се контролни део помоћу кога корисници могу да пређу на следећу итерацију (*NEXT ROUND*), врате се на претходну итерацију (*PREVIOUS ROUND*) или да се врате на почетни екран тако да могу одабрати нов алгоритам за визуелну репрезентацију (*NEW SIMULATION*).

На слици 5.13 приказана је PC1 (*permuted choice one*) пермутација, која се на

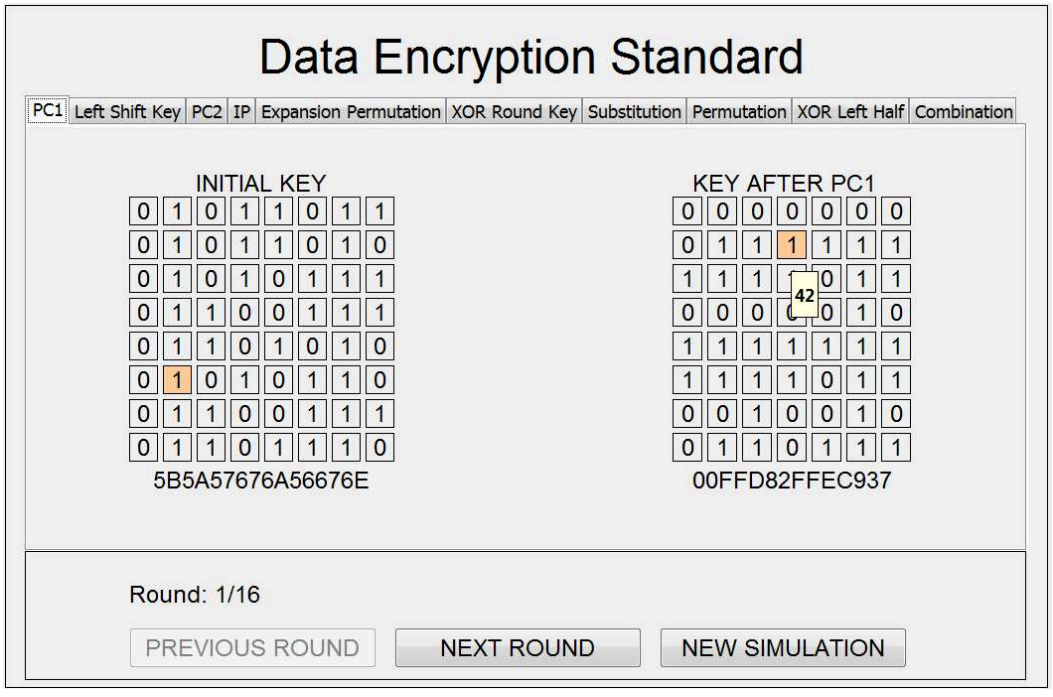


а)



б)

Слика 5.12. Екран за унос оригиналне (шифроване) поруке и кључа код DES алгоритма у COALA систему

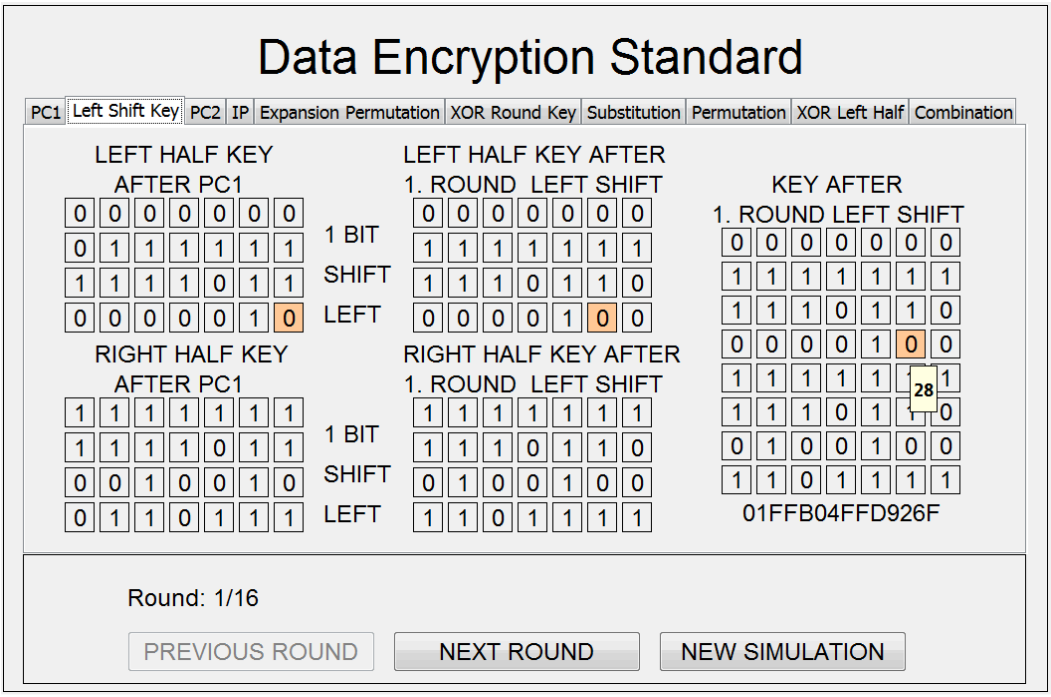


Слика 5.13. Екран за приказ прве итерације код DES алгоритма у COALA систему (PC1 пермутација)

почетку алгоритма обавља над оригиналном вредношћу унетог кључа. Са леве стране екрана приказана је бинарна вредност унетог кључа у виду матрице 8x8 у

којој свако поље представља по један бит кључа и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Испод бинарног приказа оригиналне вредности кључа види се и хексадецимална вредност унетог кључа. Са десне стране екрана приказана је бинарна вредност након извршене PC1 пермутације у виду матрице 8x7 у којој свако поље представља по један бит кључа и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Испод бинарног приказа вредности након извршене PC1 пермутације види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од две матрице на екрану бојом се означава позиција одабраног бита у обе матрице, а добија се и информација о томе која је позиција одабраног бита у оригиналној матрици у виду његовог редног броја.

На слици 5.14 приказана је операција кружног померања улево, која се обавља над резултатом PC1 пермутације. Резултат ове операције представља улаз за PC2 пермутацију у овој итерацији, али и улаз за операцију кружног померања улево за наредну итерацију. Са леве стране екрана приказана је бинарна вредност након PC1 пермутације подељена на две 28-битне половине у виду две матрице 4x7 у којима свако поље представља по један бит вредности након PC1 пермутације и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Виших 28 бита су у горњој матрици на слици, а нижих 28 бита у доњој. У средини је приказана вредност након извршене операције кружног померања улево за сваку половину појединачно, а према распреду ротације кључа, у виду две матрице 4x7 у којима свако поље представља по један бит добијене вредности. Са десне стране екрана приказана је бинарна вредност резултата након извршене операције кружног померања улево у виду матрице 8x7 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након извршене операције кружног померања улево види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од пет матрица на екрану бојом се означава позиција одабраног бита у свим матрицама у којима се налази, а добија се и информација о томе која је позиција одабраног би–

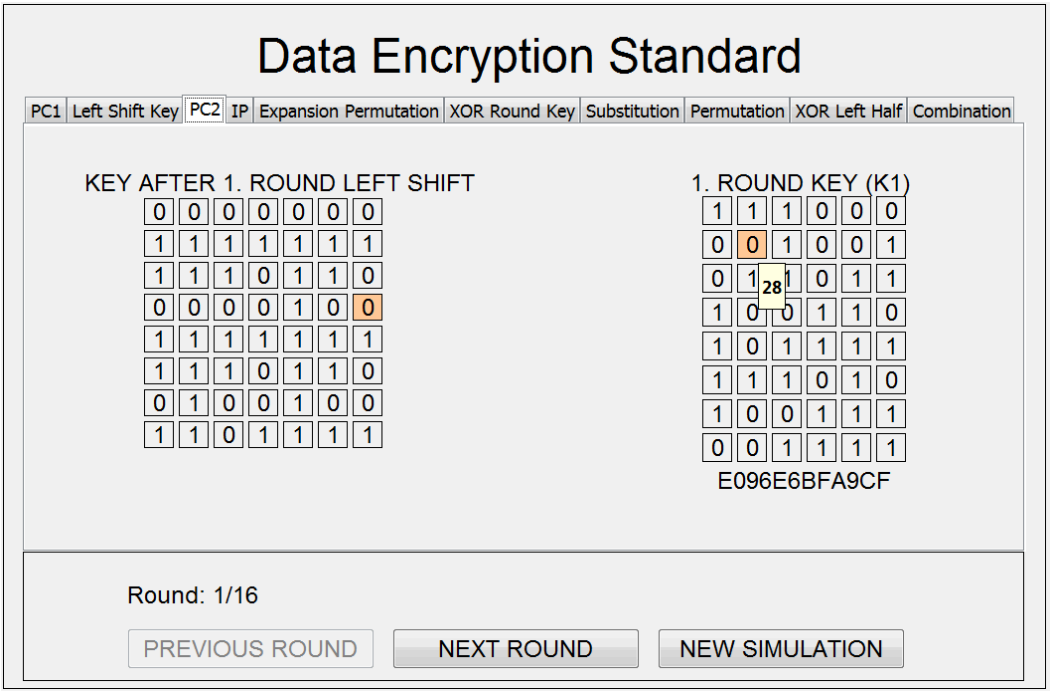


Слика 5.14. Екран за приказ операције кружног померања улево у оквиру прве итерације код DES алгоритма у COALA систему

та у матрици након PC1 пермутације у виду његовог редног броја.

На слици 5.15 приказана је PC2 (*permuted choice two*) пермутација, која се обавља над резултатом операције кружног померања улево. Са леве стране екрана приказана је бинарна вредност након операције кружног померања улево у виду матрице 8x7 у којој свако поље представља по један бит вредности након операције кружног померања улево и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Са десне стране екрана приказана је бинарна вредност након извршене PC2 пермутације у виду матрице 8x6 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након извршене PC2 пермутације види се и одговарајућа хексадецимална вредност. Ова вредност представља кључ итерације за прву итерацију. Позиционирањем миша на било које поље једне од две матрице на екрану бојом се означава позиција одабраног бита у обе матрице, а добија се и информација о томе која је позиција одабраног бита у матрици након операције кружног померања улево у виду његовог редног броја.

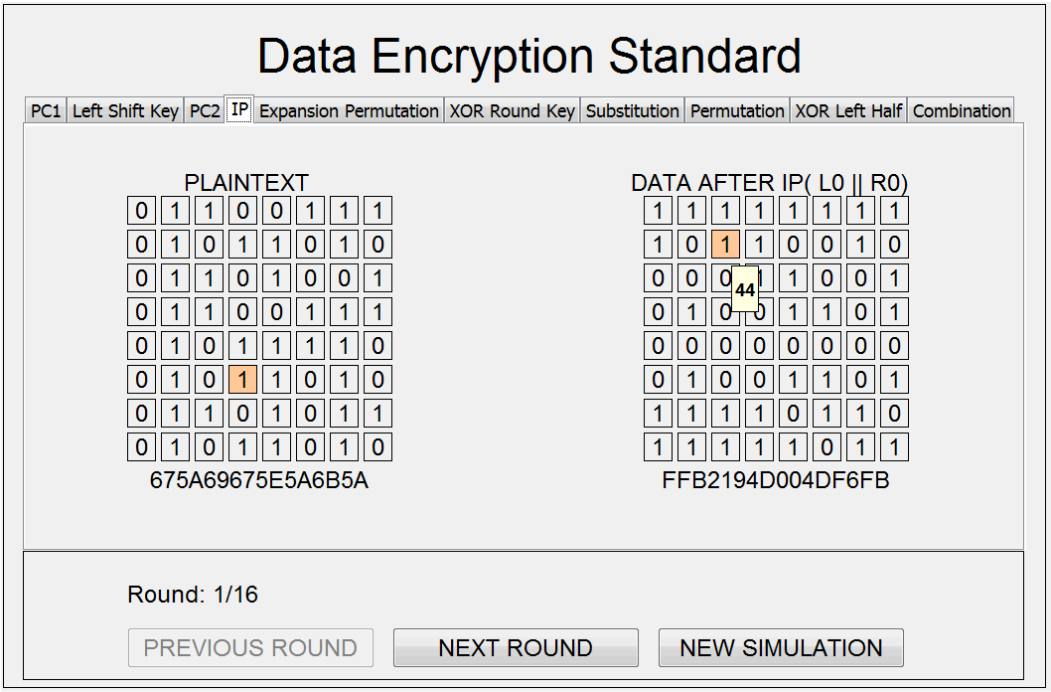
Претходне три операције су се односиле на формирање кључа итерације. PC1 пермутација се извршава само једном на почетку алгоритма и због тога је прика–



Слика 5.15. Екран за приказ PC2 пермутације у оквиру прве итерације код DES алгоритма у COALA систему

зана само у визуелној репрезентацији прве итерације, док се преостале две операције извршавају у свакој итерацији и приказане су у визуелној репрезентацији сваке итерације.

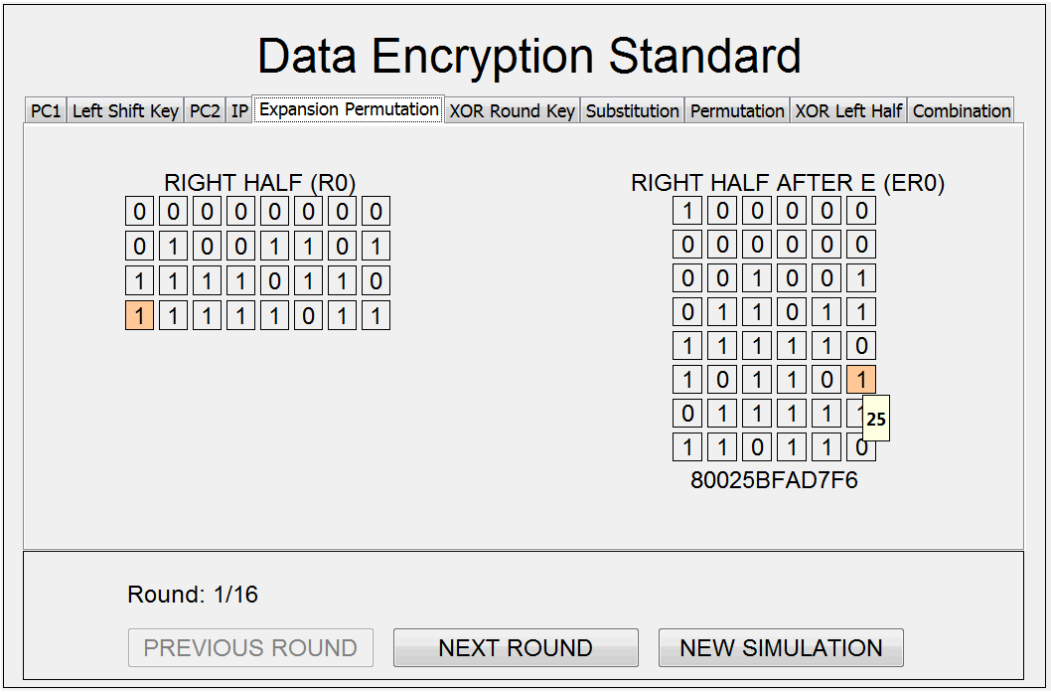
На слици 5.16 приказана је IP (*initial permutation*) пермутација, која се обавља само у првој итерацији и то над оригиналним блоком података. Са леве стране екрана приказана је бинарна вредност оригиналног блока података у виду матрице 8x8 у којој свако поље представља по један бит вредности након операције кружног померања улево и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Испод бинарног приказа вредности оригиналног блока података види се и одговарајућа хексадецимална вредност. Са десне стране екрана приказана је бинарна вредност након извршене IP пермутације у виду матрице 8x8 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након извршене IP пермутације види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од две матрице на екрану бојом се означава позиција одабраног бита у обе матрице, а добија се и информација о томе која је позиција одабраног бита у матрици оригиналног блока података у виду његовог редног броја.



Слика 5.16. Екран за приказ IP пермутације у оквиру прве итерације код DES алгоритма у COALA систему

На слици 5.17 приказана је EP (*expansion permutation*) пермутација, која се обавља као прва операција у оквиру функције сваке итерације код DES алгоритма. EP пермутација се у првој итерацији обавља над резултатом IP пермутације, а у свакој следећој итерацији над резултатом претходне итерације. Са леве стране екрана приказана је бинарна вредност десне половине улазне вредности у текућој итерацији у виду матрице 4x8 у којој свако поље представља по један бит десне половине улазне вредности и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Са десне стране екрана приказана је бинарна вредност након извршене EP пермутације у виду матрице 8x6 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након извршене EP пермутације види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од две матрице на екрану бојом се означава позиција одабраног бита у обе матрице, а добија се и информација о томе која је позиција одабраног бита у матрици десне половине улазне вредности у текућој итерацији у виду његовог редног броја.

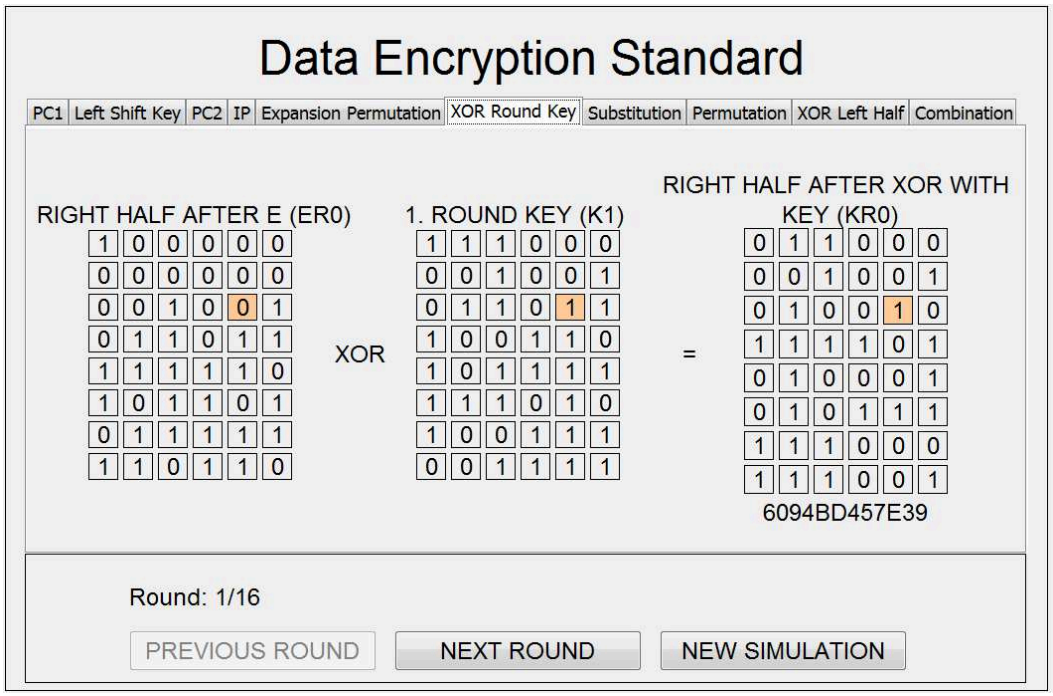
На слици 5.18 приказана је ексклузивно ИЛИ операција, која се обавља као друга операција у оквиру функције сваке итерације код DES алгоритма. Ексклу–



Слика 5.17. Екран за приказ EP пермутације у оквиру прве итерације код DES алгоритма у COALA систему

зивно ИЛИ операција се обавља над резултатом EP пермутације и одговарајућим кључем итерације добијеним након PC2 пермутације. Са леве стране екрана приказана је бинарна вредност након EP пермутације у виду матрице 8x6 у којој свако поље представља по један бит вредности након EP пермутације и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. У средини је приказана бинарна вредност након PC2 пермутације у виду матрице 8x6 у којој свако поље представља по један бит вредности након PC2 пермутације и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Са десне стране екрана приказана је бинарна вредност након извршене ексклузивно ИЛИ операције у виду матрице 8x6 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након извршене ексклузивно ИЛИ операције види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од три матрице на екрану бојом се означава начин добијања одабраног бита у резултујућој матрици.

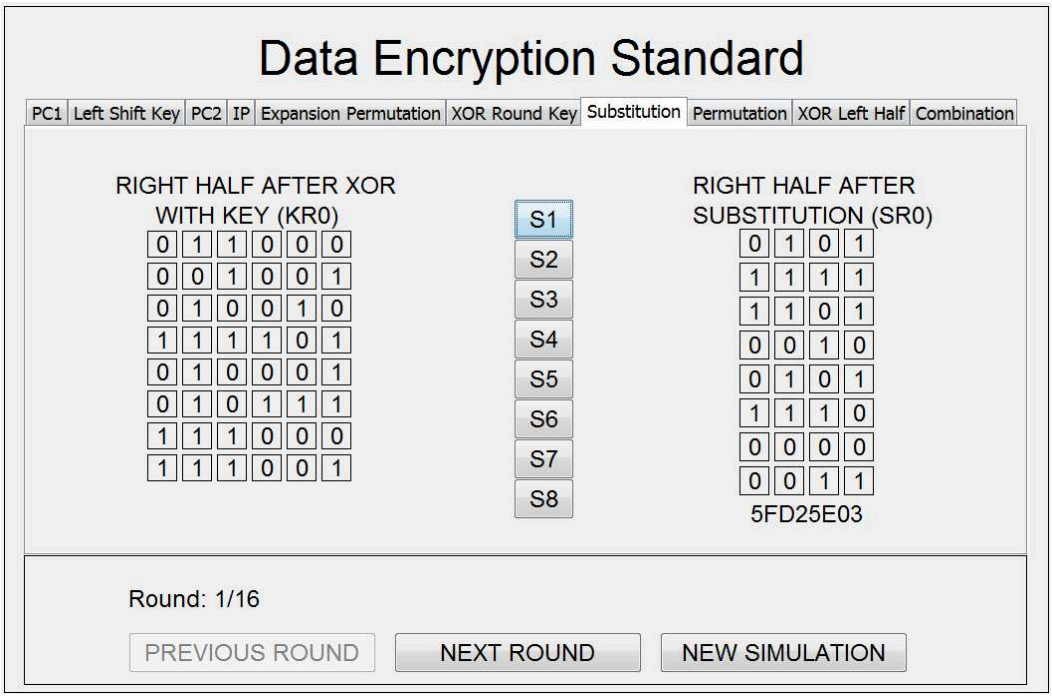
На слици 5.19а приказана операција супституције, која се обавља као трећа



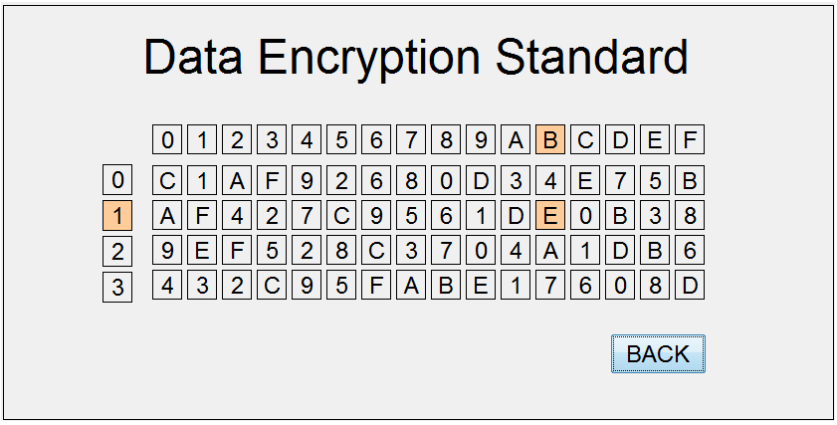
Слика 5.18. Екран за приказ ексклузивно ИЛИ операције у оквиру прве итерације код DES алгоритма у COALA систему

операција у оквиру функције сваке итерације код DES алгоритма. Операција супституције се обавља над резултатом ексклузивно ИЛИ операције. Са леве стране екрана приказана је бинарна вредност након ексклузивно ИЛИ операције у виду матрице 8x6 у којој свако поље представља по један бит вредности након ексклузивно ИЛИ операције и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. У средини је приказан низ од осам дугмића које је могуће притиснути, а који означавају осам табела замене (S-box табела) које се користе у овој операцији. Са десне стране екрана приказана је бинарна вредност након извршене операције супституције у виду матрице 8x4 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након извршене операције супституције види се и одговарајућа хексадецимална вредност. Притиском на било које од осам дугмади која представљају осам табела замене на екрану приказаном на слици 5.19а добија се детаљан приказ садржаја одговарајуће табеле замене у коме је означено како се дошло до вредности за замену. Слика 5.19б приказује табелу замене S6 из примера приказаног на слици 5.19а.

На слици 5.20 приказана је P пермутација, која се обавља као последња опе–



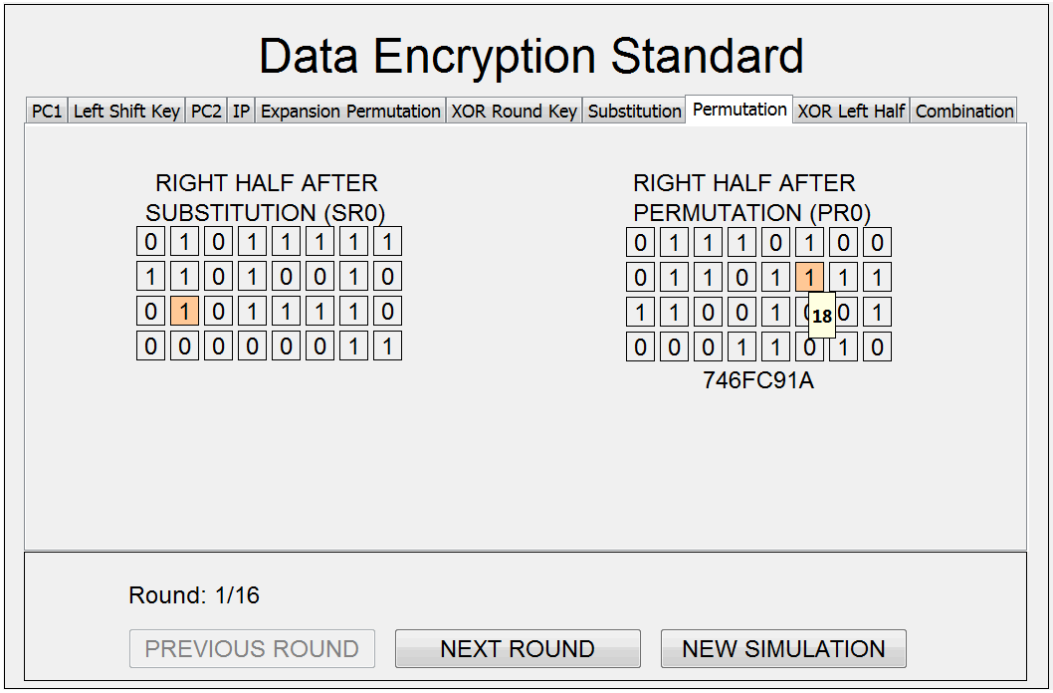
а)



б)

Слика 5.19. Екран за приказ операције супституције у оквиру прве итерације код DES алгоритма у COALA систему

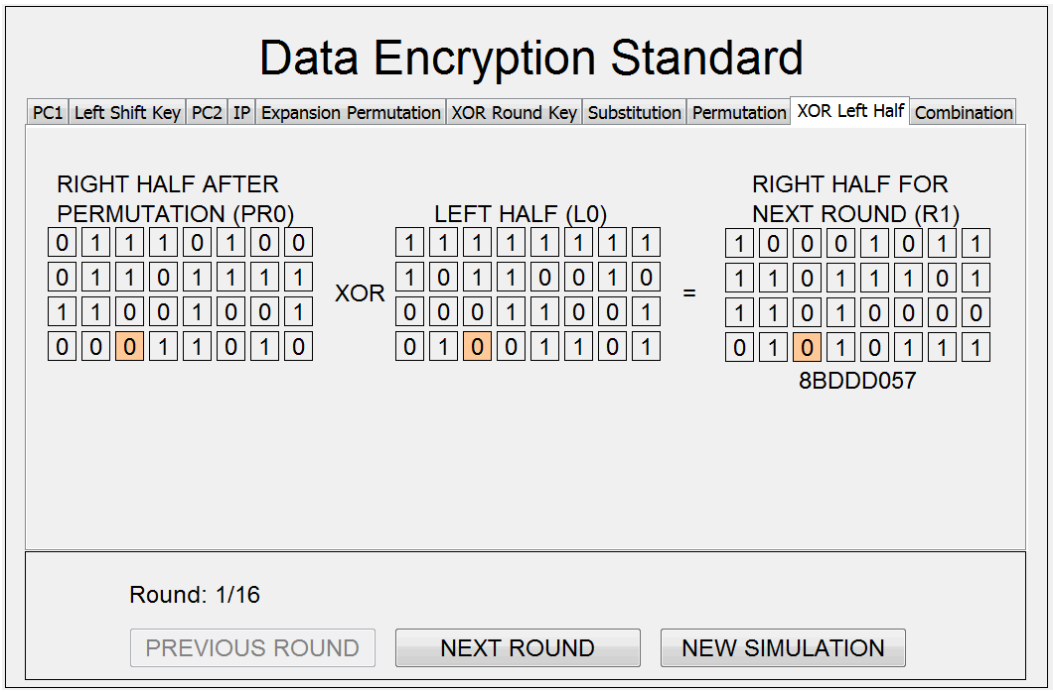
–рација у оквиру функције сваке итерације код DES алгоритма. P пермутација се обавља над резултатом операције супституције. Резултат ове операције представља резултат функције итерације ($F(R_{i-1}, K_i)$) за текућу итерацију. Са леве стране екрана приказана је бинарна вредност након операције супституције у виду матрице 4x8 у којој свако поље представља по један бит вредности након операције супституције и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Са десне стране екрана приказана је би–



Слика 5.20. Екран за приказ P пермутације у оквиру прве итерације код DES алгоритма у COALA систему

нарна вредност након извршене P пермутације у виду матрице 4x8 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након извршене P пермутације види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од две матрице на екрану бојом се означава позиција одабраног бита у обе матрице, а добија се и информација о томе која је позиција одабраног бита у матрици након операције супституције у виду његовог редног броја.

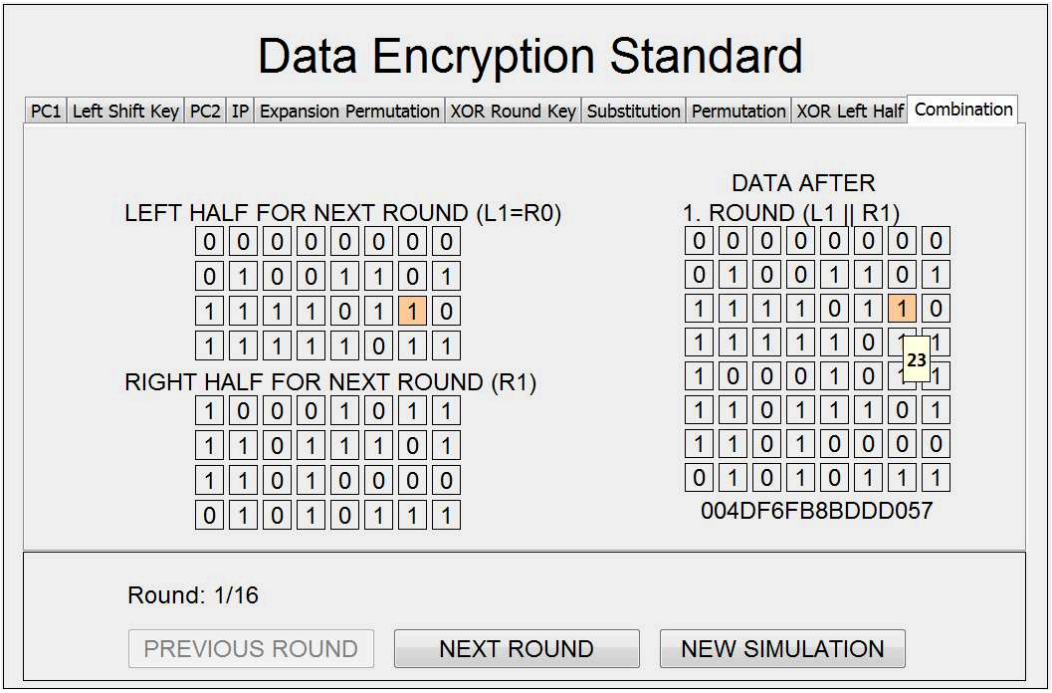
На слици 5.21 приказана је операција ексклузивно ИЛИ којом се на крају итерације добија вредност десне половине података за наредну итерацију, а која се обавља над резултатом P пермутације и левом половином података за текућу итерацију. Са леве стране екрана приказана је бинарна вредност након P пермутације у виду матрице 4x8 у којој свако поље представља по један бит вредности након P пермутације и битови су распоређени по редовима почевши од бита највеће тежине, који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. У средини је приказана бинарна вредност леве половине података за текућу итерацију у виду матрице 4x8 у којој свако поље представља по један бит вредности леве половине података за текућу итерацију и битови су распоређени по редовима почевши од бита највеће тежине,



Слика 5.21. Екран за приказ ексклузивно ИЛИ операције на крају прве итерације код DES алгоритма у COALA систему

који је први бит у првом реду, па све до бита најмање тежине, који је последњи бит у последњем реду. Са десне стране екрана приказана је бинарна вредност након извршене ексклузивно ИЛИ операције у виду матрице 4x8 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након извршене ексклузивно ИЛИ операције види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од три матрице на екрану бојом се означава начин добијања одабраног бита у резултујућој матрици.

На слици 5.22 приказано је како је добијена вредност која представља резултат прве итерације алгоритма и улаз за другу итерацију алгоритма. Са леве стране екрана приказане су једна испод друге бинарне вредности десне половине података текуће итерације (што ће бити лева половина података за наредну итерацију) у виду матрице 4x8 у којој свако поље представља по један бит и бинарне вредности леве половине података за наредну итерацију израчунате након последње ексклузивно ИЛИ операције у виду матрице 4x8 у којој свако поље представља по један бит. Са десне стране екрана приказана је бинарна вредност података за наредну итерацију у виду матрице 8x8 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности пода–

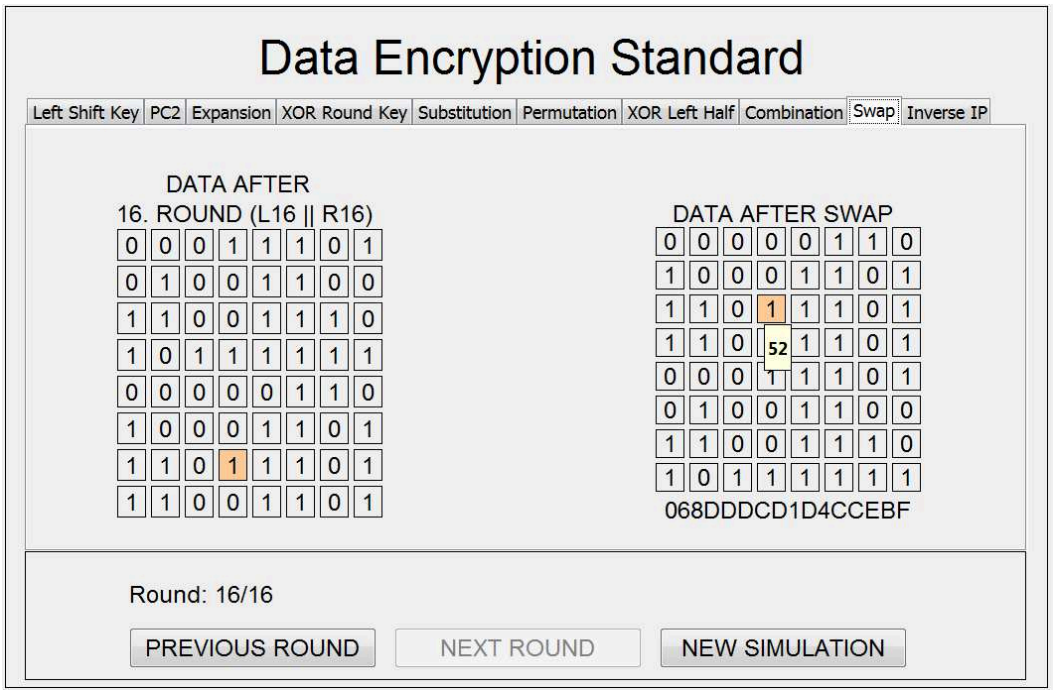


Слика 5.22. Екран за приказ вредности података за наредну итерацију на крају прве итерације код DES алгоритма у COALA систему

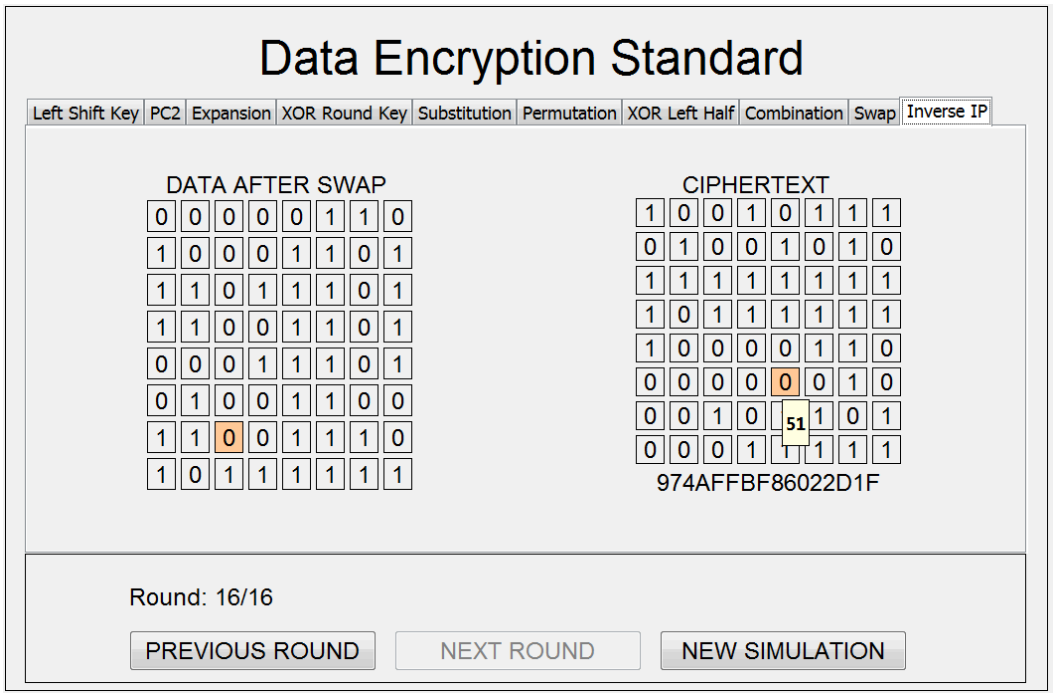
така за наредну итерацију види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од три матрице на екрану бојом се означава позиција одабраног бита у свим матрицама у којима се тај бит јавља, а добија се и информација о томе која је позиција одабраног бита у матрици вредности података за наредну итерацију у виду његовог редног броја.

На слици 5.23 приказана је операција 32-битне замене места (*32-bit swap*) која се обавља након шеснаесте итерације DES алгоритма. Са леве стране екрана приказана је бинарне вредност добијена након шеснаесте итерације алгоритма у виду матрице 8x8 у којој свако поље представља по један бит. Са десне стране екрана приказана је бинарна вредност након операције 32-битне замене места у виду матрице 8x8 у којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након операције 32-битне замене места види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од две матрице на екрану бојом се означава позиција одабраног бита у обе матрице, а добија се и информација о томе која је позиција одабраног бита у матрици након шеснаесте итерације алгоритма у виду његовог редног броја.

На слици 5.24 приказана је инверзна иницијална пермутација која се обавља након операције 32-битне замене места код DES алгоритма. Ова вредност пред–



Слика 5.23. Екран за приказ вредности након операције 32-битне замене места код DES алгоритма у COALA систему



Слика 5.24. Екран за приказ вредности након инверзне иницијалне пермутације код DES алгоритма у COALA систему

ставља шифровани блок података. Са леве стране екрана приказана је бинарна вредност добијена након операције 32-битне замене места у виду матрице 8x8 у којој свако поље представља по један бит. Са десне стране екрана приказана је бинарна вредност након инверзне иницијална пермутације у виду матрице 8x8 у

Слика 5.25. Екран за избор шифровања или дешифровања код AES алгоритма у COALA систему

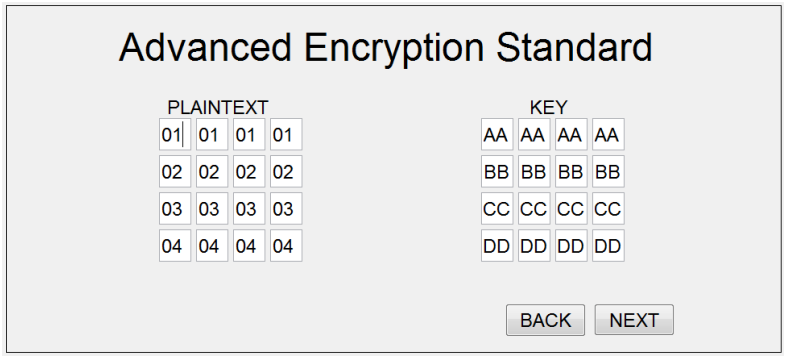
којој свако поље представља по један бит ове вредности. Испод бинарног приказа вредности након инверзне иницијалне пермутације види се и одговарајућа хексадецимална вредност. Позиционирањем миша на било које поље једне од две матрице на екрану бојом се означава позиција одабраног бита у обе матрице, а добија се и информација о томе која је позиција одабраног бита у матрици након операције 32-битне замене места у виду његовог редног броја.

5.4.2. AES АЛГОРИТАМ

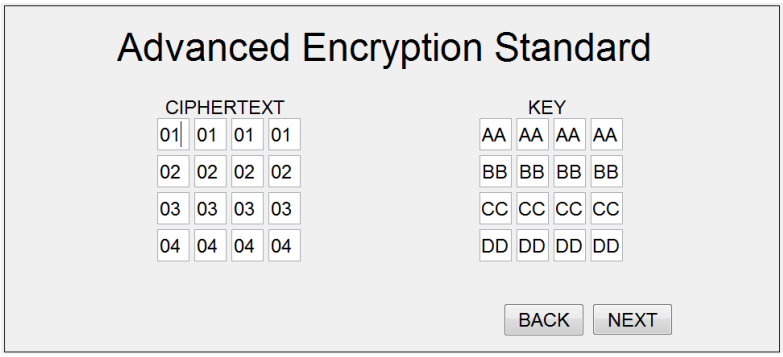
Када корисник изабере визуелну репрезентацију AES алгоритма на почетном екрану, најпре се појављује екран на коме се може изабрати да ли треба да се приказује шифровање или дешифровање (Слика 5.25).

Након тога, појављује се екран за унос оригиналне поруке и кључа (Слика 5.26а) или екран за унос шифроване поруке и кључа (Слика 5.26б), у зависности од тога да ли је одабрано да се прати шифровање или дешифровање. Поља за унос оригиналне поруке, шифроване поруке и кључа организована су као матрица 4x4 где је свако поље величине 1 бајт, а сам унос се врши уписивањем хексадецималних бројева.

Након уноса улазних података приказује се екран на коме се види извршавање прве итерације алгоритма (слика 5.27). Извршавање алгоритма је презентовано тако да се у једном тренутку може видети једна итерација алгоритма. Ово омогућава корисницима да могу да прате извршавање алгоритма до најситнијих детаља. С обзиром да се итерација AES алгоритма састоји из неколико операција, за сваку операцију постоји посебан *tab* у прозору који приказује итерацију. У доњем делу екрана налази се контролни део помоћу кога корисници могу да пређу на следећу итерацију (*NEXT ROUND*), врате се на претходну итерацију (*PREVIOUS ROUND*) или да се врате на почетни екран тако да могу одабрати нов алгоритам за визуелну репрезентацију (*NEW SIMULATION*).



а)

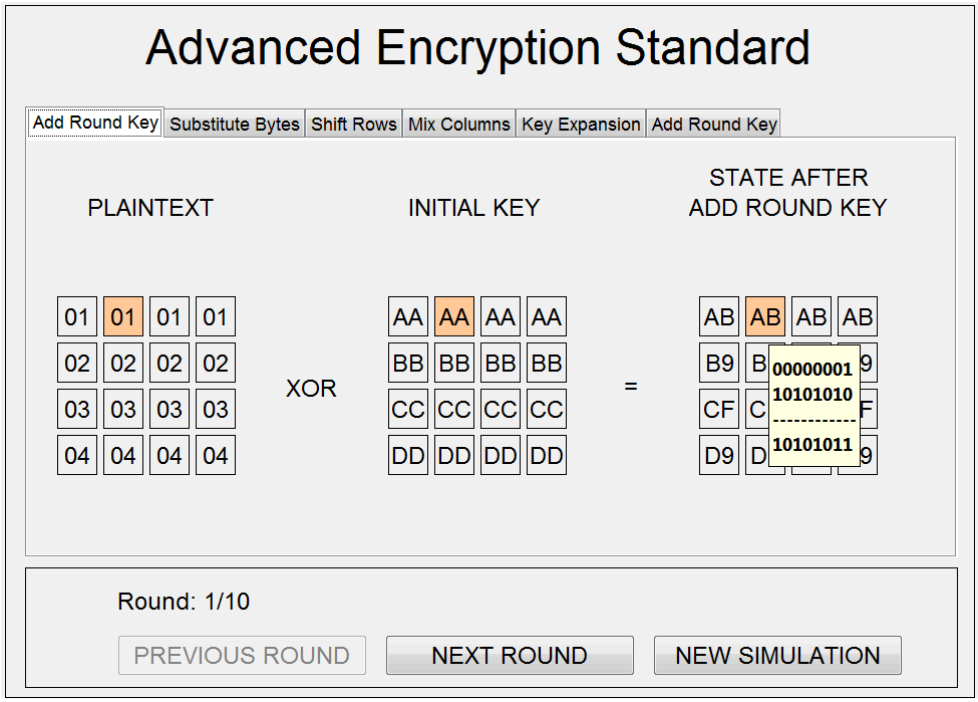


б)

Слика 5.26. Екран за унос оригиналне (шифроване) поруке и кључа код AES алгоритма у COALA систему

На слици 5.27 приказана је почетна операција додавања кључа итерације, која се на почетку алгоритма обавља над оригиналном вредношћу унетог стања и оригиналном вредношћу унетог кључа. Са леве стране екрана приказана је хексадецимална вредност унетог стања у виду матрице 4x4 у којој свако поље представља по један бајт стања. У средини екрана приказана је хексадецимална вредност унетог кључа у виду матрице 4x4 у којој свако поље представља по један бајт кључа. Са десне стране екрана приказана је хексадецимална вредност резултата операције ексклузивно ИЛИ примењене над унетим стањем и унетим кључем на нивоу бајта у виду матрице 4x4 у којој свако поље представља по један бајт стања након почетног додавања кључа итерације. Позиционирањем миша на било које поље једне од три матрице на екрану бојом се означава начин формирања резултујућег бајта на позицији одабраног бајта. У случају да се миш позиционира на неко поље резултујуће матрице добија се и текстуално појашњење извршене операције ексклузивно ИЛИ на нивоу бита.

На слици 5.28 приказана је операција замене бајтова која представља прву

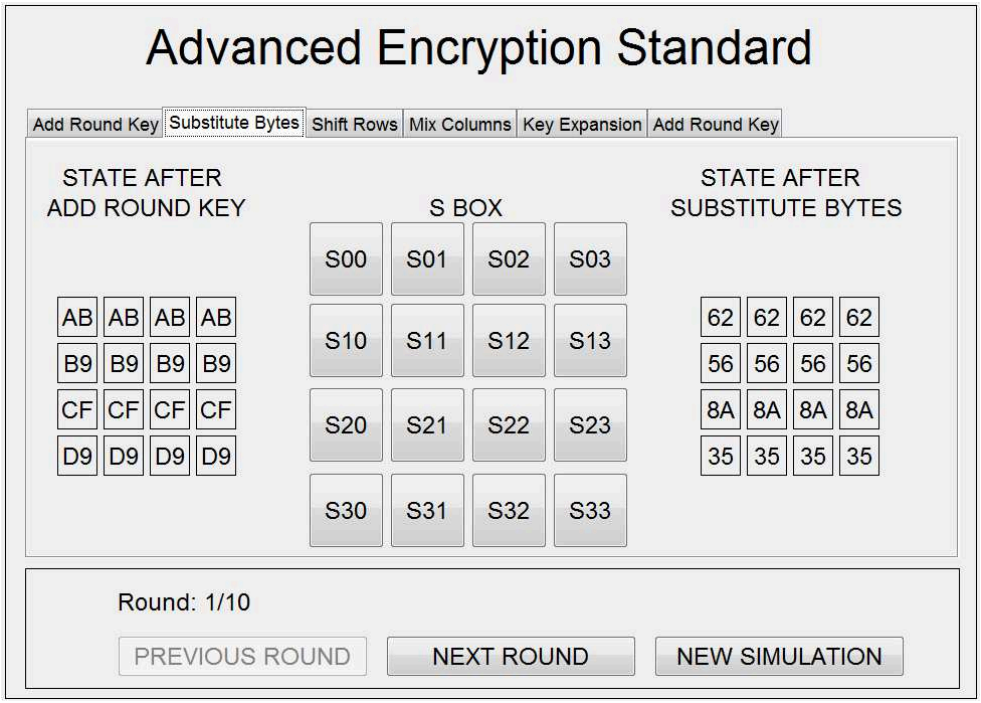


Слика 5.27. Екран за приказ прве итерације код AES алгоритма у COALA систему (операција додавање кључа итерације)

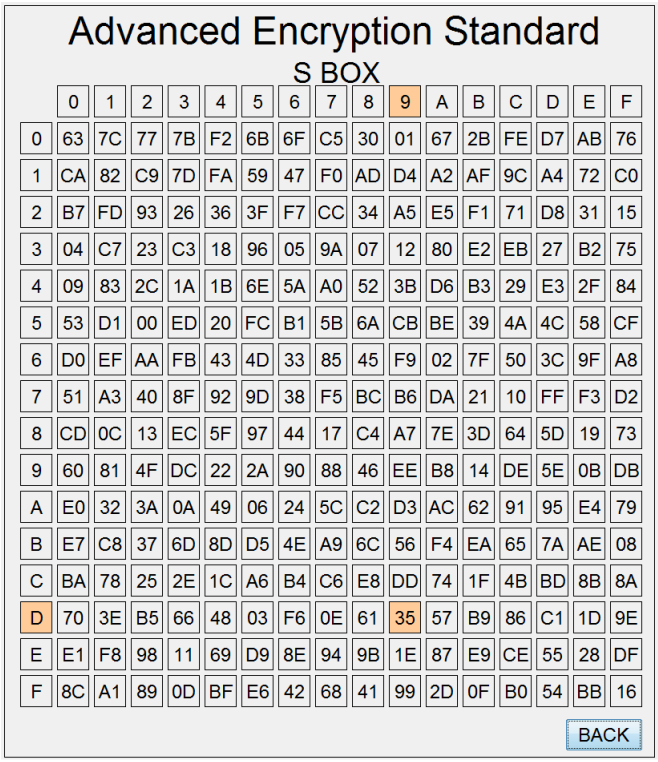
операцију сваке итерације AES алгоритма. Замена бајтова се у првој операцији обавља над стањем добијеним након почетног додавања кључа итерације, а у свим осталим итерацијама се обавља над стањем које је добијено као резултат претходне итерације. Са леве стране екрана приказана је хексадецимална вредност стања над којим се врши операција замене у виду матрице 4x4 у којој свако поље представља по један бајт стања. У средини екрана приказана је матрице 4x4 дугмића који омогућавају добијање детаљног приказа формирања одговарајућег бајта у резултујућој матрици. Са десне стране екрана приказана је хексадецимална вредност резултата операције замене бајтова примењене над стањем у виду матрице 4x4 у којој свако поље представља по један бајт стања након операције замене бајтова.

Притиском на било које дугме из матрице са дугмићима са слике 5.28 добија се детаљан приказ формирања одговарајућег бајта у резултујућој матрици. Притиском на дугме означено као S33 на слици 5.28 добија се приказ детаљног извршавања операције замене бајтова приказан на слици 5.29.

На слици 5.30 приказана је операција померања редова, која се обавља као друга операција у свакој итерацији AES алгоритма. Као улаз за операцију померања редова узима се стање након операције замене бајтова. Са леве стране

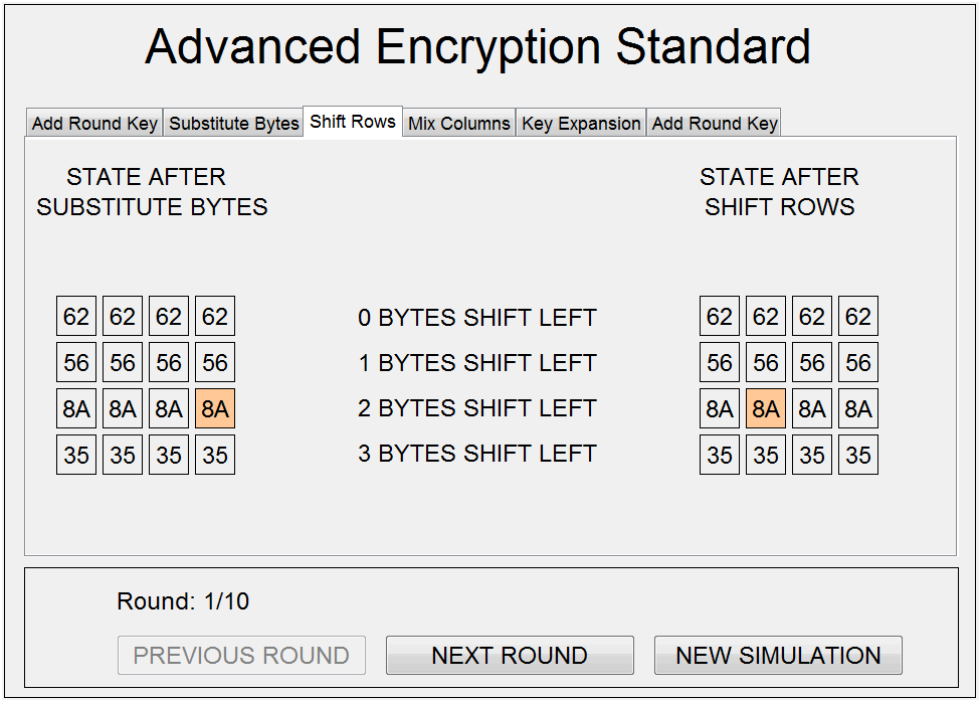


Слика 5.28. Екран за приказ операције замене бајтова код AES алгоритма у COALA систему



Слика 5.29. Екран за приказ детаља операције замене бајтова код AES алгоритма у COALA систему

екрана приказана је хексадецимална вредност улазног стања у виду матрице 4x4 у којој свако поље представља по један бајт стања. У средини екрана приказано је правило по коме се извршава операција померања редова. Са десне стране екрана

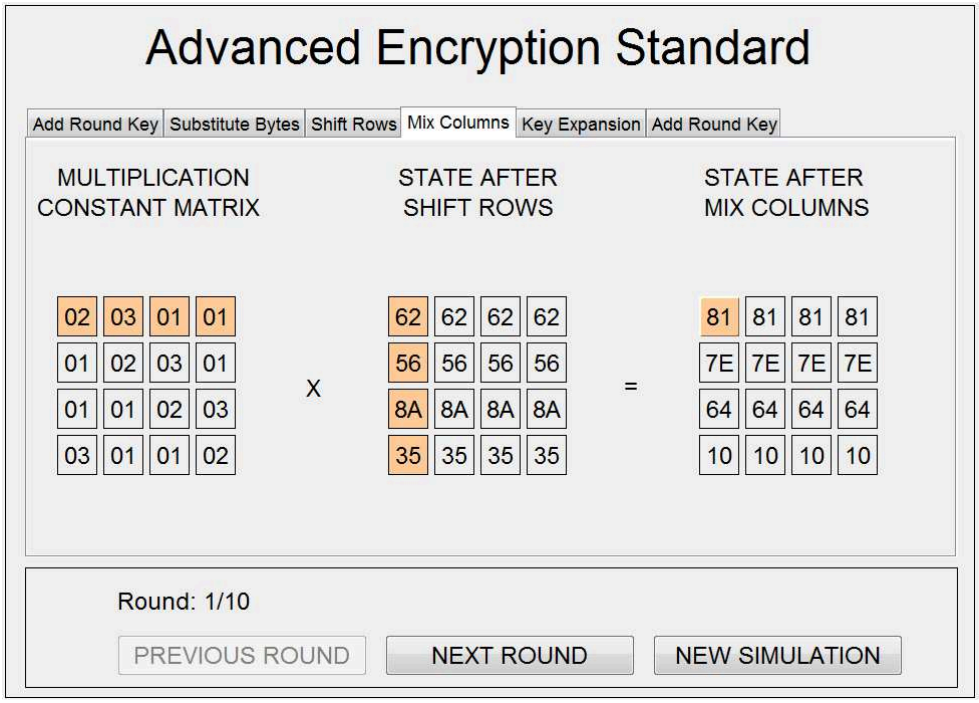


Слика 5.30. Екран за приказ операције померања редова код AES алгоритма у COALA систему

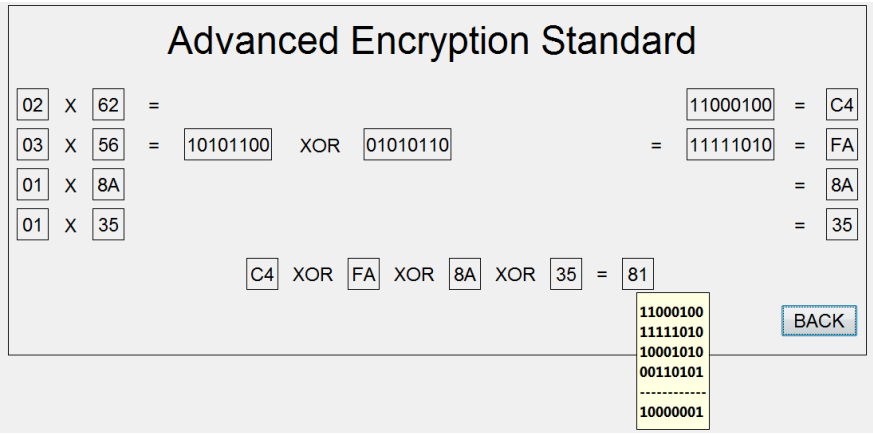
приказана је хексадецимална вредност резултата операције померања редова примењене над улазним стањем у виду матрице 4x4 у којој свако поље представља по један бајт стања након операције померања редова. Позиционирањем миша на било које поље једне од две матрице на екрану бојом се означава позиција одабраног бајта у обе матрице.

На слици 5.31 приказана је операција мешања колона, која се обавља као трећа операција у свим итерацијама AES алгоритма, осим у последњој. Као улаз за операцију мешања колона узима се стање након операције померања редова. Са леве стране екрана приказана је хексадецимална вредност константне матрице са којом се обавља множење улазног стања. У средини екрана приказана је хексадецимална вредност улазног стања у виду матрице 4x4 у којој свако поље представља по један бајт стања. Са десне стране екрана приказана је хексадецимална вредност резултата операције мешања колона примењене над улазним стањем у виду матрице 4x4 у којој свако поље представља по један бајт стања након операције мешања колона. Позиционирањем миша на било које поље резултујуће матрице на екрану бојом се означава начин формирања одабраног бајта.

За свако поље резултујуће матрице са слике 5.31 могуће је добити детаљан



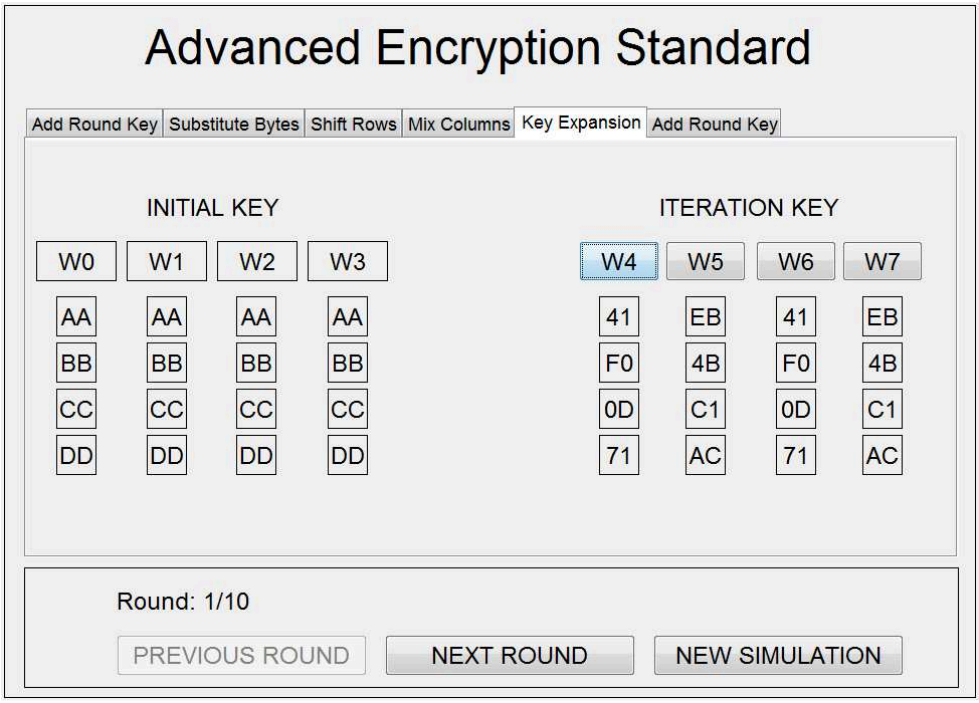
Слика 5.31. Екран за приказ операције мешања колона код AES алгоритма у COALA систему



Слика 5.32. Екран за детаљан приказ операције мешања колона код AES алгоритма у COALA систему

приказ како је одговарајући бајт формиран у оквиру операције мешања колона. Приказ се добија кликом миша на одговарајући бајт у оквиру резултујуће матрице. За пример са слике 5.31 притиском на означени бајт у резултујућој матрици добија се екран приказан на слици 5.32.

На слици 5.33 приказан је начин добијања кључа итерације за прву итерацију експанзијом кључа. Експанзијом кључа се у свакој итерацији обезбеђује нова 128-битна вредност кључа итерације која је потребна за операцију додавања кључа итерације. Због тога је у COALA систему приказ ове операције позициониран непосредно испред операције додавања кључа итерације. У свакој итерацији при–

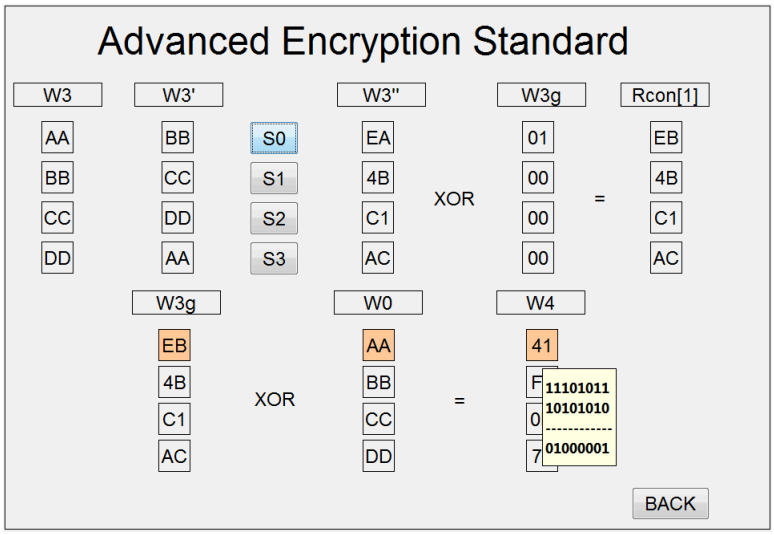


Слика 5.33. Екран за приказ добијања кључа итерације код AES алгоритма у COALA систему

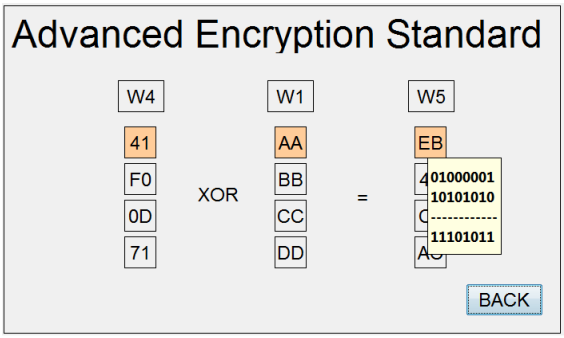
казан је само део експанзије кључа којим се формира кључ итерације за ту итерацију. Са леве стране екрана приказана је хексадецимална вредност претходног кључа итерације (у првој итерацији то је оригинални кључ) у виду матрице 4x4 у којој свако поље представља по један бајт. Колоне су означене редним бројем 32-битне речи у оквиру експандованог кључа (Wi). Са десне стране екрана приказана је хексадецимална вредност добијеног кључа итерације у виду матрице 4x4 у којој свако поље представља по један бајт кључа итерације. И овде су колоне означене редним бројем 32-битне речи у оквиру експандованог кључа (Wi), али су сада ове ознаке представљене помоћу дугмади која се могу притиснути да би се добио детаљан приказ формирања одабране речи.

За сваку колону кључа итерације са слике 5.33 могуће је добити детаљан приказ како је одговарајућа 32-битна реч формирана у оквиру експанзије кључа. Приказ се добија кликом миша на одговарајуће дугме у оквиру резултујуће матрице. За пример са слике 5.33 притиском на W4 у резултујућој матрици добија се екран приказан на слици 5.34а, а притиском на W5 у резултујућој матрици добија се екран приказан на слици 5.34б. Три од четири случаја су идентична, док је сваки четврти различит.

На слици 5.35 приказана је операција додавања кључа итерације, која се обавља



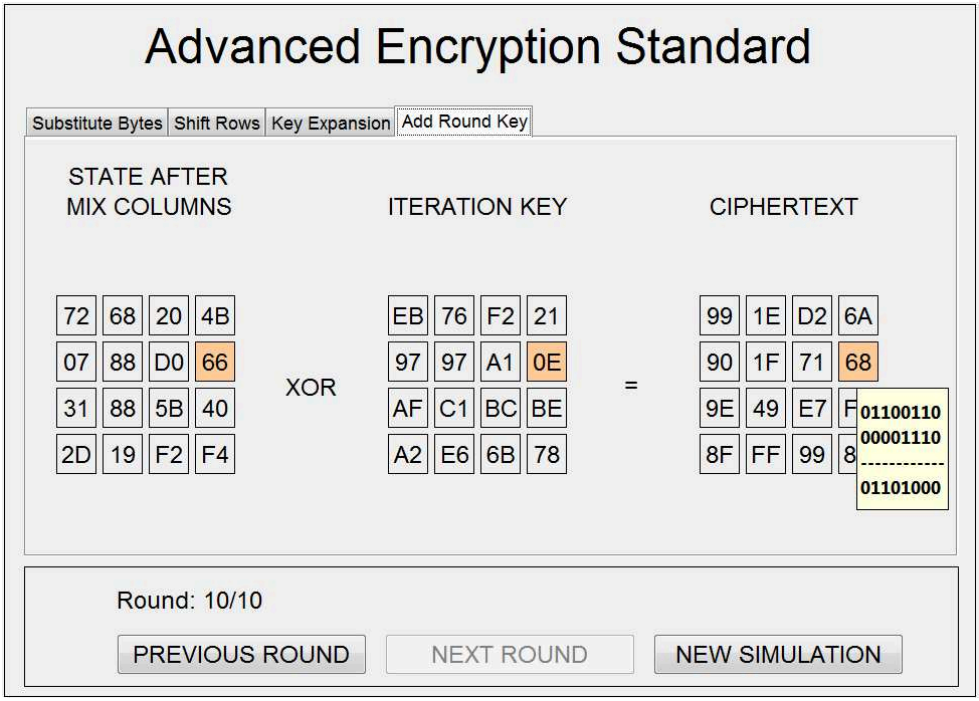
а)



б)

Слика 5.34. Екран за детаљан приказ добијања кључа итерације код AES алгоритма у COALA систему

као последња операција у свакој итерацији AES алгоритма. Операција додавања кључа итерације обавља се над вредношћу кључа итерације и вредношћу стања након операције мешања колона у свим итерацијама осим у последњој, када се користи вредност стања након операције померања редова. Са леве стране екрана приказана је хексадецимална вредност улазног стања у виду матрице 4x4 у којој свако поље представља по један бајт стања. У средини екрана приказана је хексадецимална вредност кључа итерације у виду матрице 4x4 у којој свако поље представља по један бајт кључа. Са десне стране екрана приказана је хексадецимална вредност резултата операције ексклузивно ИЛИ примењене над улазним стањем и кључем итерације на нивоу бајта у виду матрице 4x4 у којој свако поље представља по један бајт стања након операције додавања кључа итерације. Позиционирањем миша на било које поље једне од три матрице на



Слика 5.35. Екран за приказ операција додавање кључа итерације код AES алгоритма у COALA систему

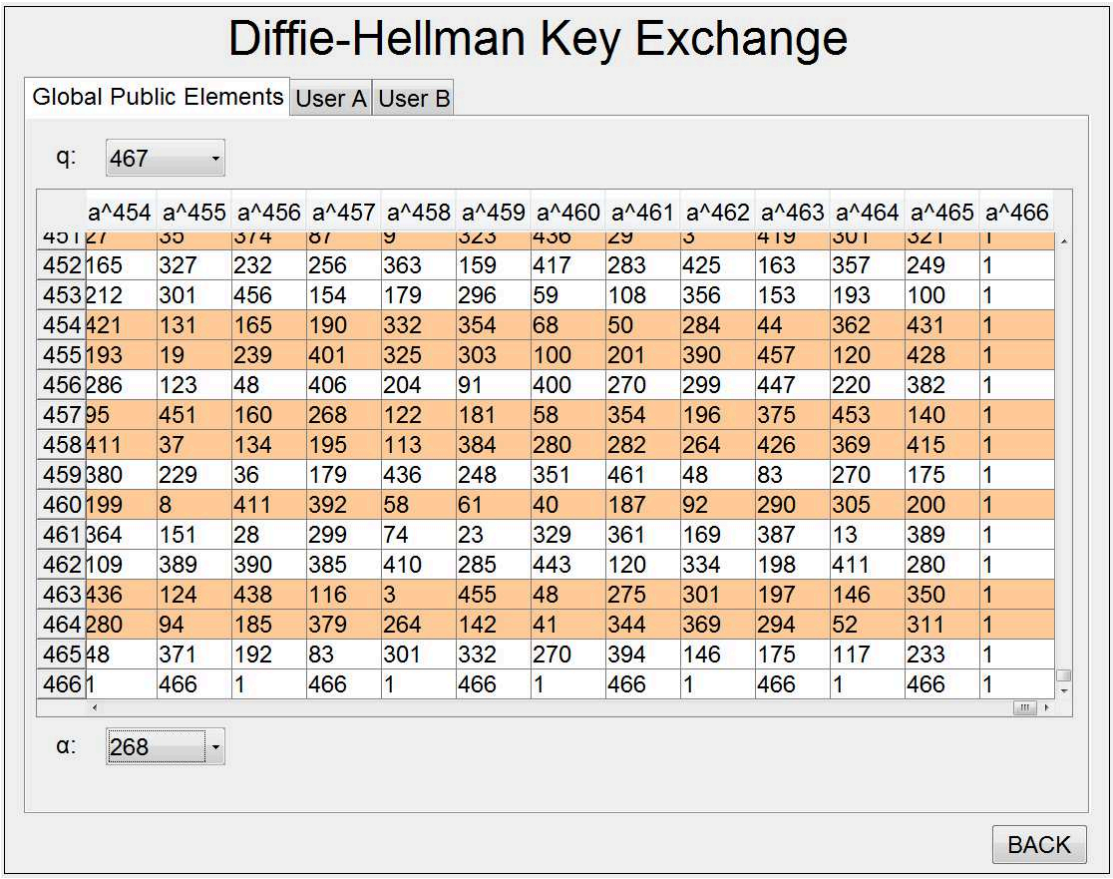
екрану бојом се означава начин формирања резултујућег бајта на позицији одабраног бајта. У случају да се миш позиционира на неко поље резултујуће матрице добија се и текстуално појашњење извршене операције ексклузивно ИЛИ на нивоу бита. Након операције додавања кључа итерације последње итерације AES алгоритма добија се шифрована порука.

5.5. АЛГОРИТМИ СА ЈАВНИМ КЉУЧЕМ

У овом поглављу се разматра визуелна репрезентација криптографских алгоритама са јавним кључем подржаних у COALA систему: *Diffie-Hellman* алгоритам и *RSA* алгоритам, као што је приказано на слици 5.1.

5.5.1. *DIFFIE-HELLMAN* АЛГОРИТАМ

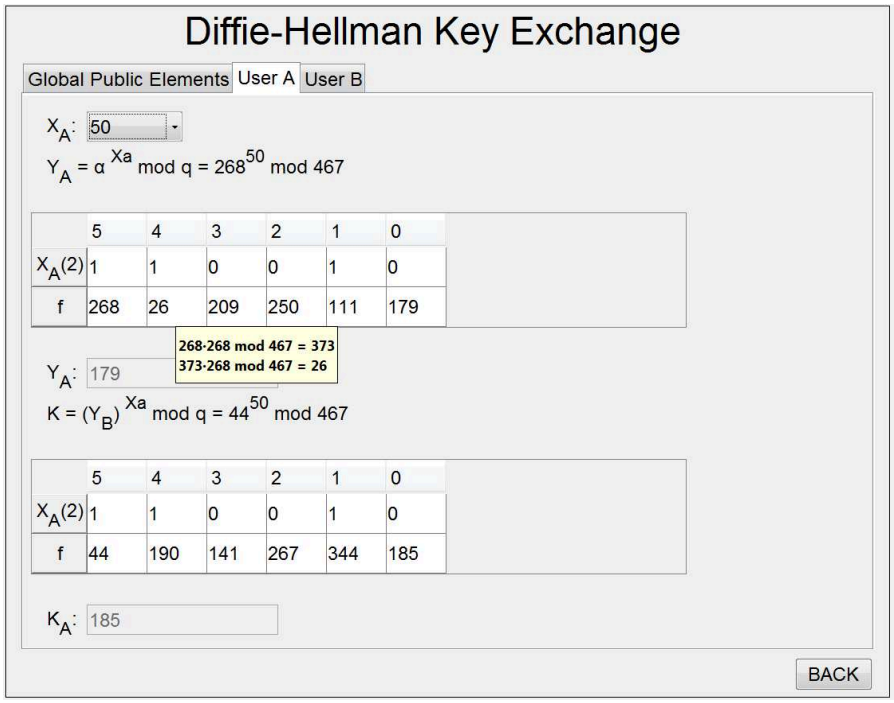
Када корисник изабере визуелну репрезентацију *Diffie-Hellman* алгоритма на почетном екрану, најпре се појављује екран на коме треба да се одаберу јавно доступне вредности. Када корисник одабере прост број q приказује се табела која помаже да се одреде сви могући прости корени за одабрану вредност q, односно све вредности које долазе у обзир да буду α (Слика 5.36). Након тога корисник бира и неку од понуђених вредности за α. Приказана табела садржи по врстама



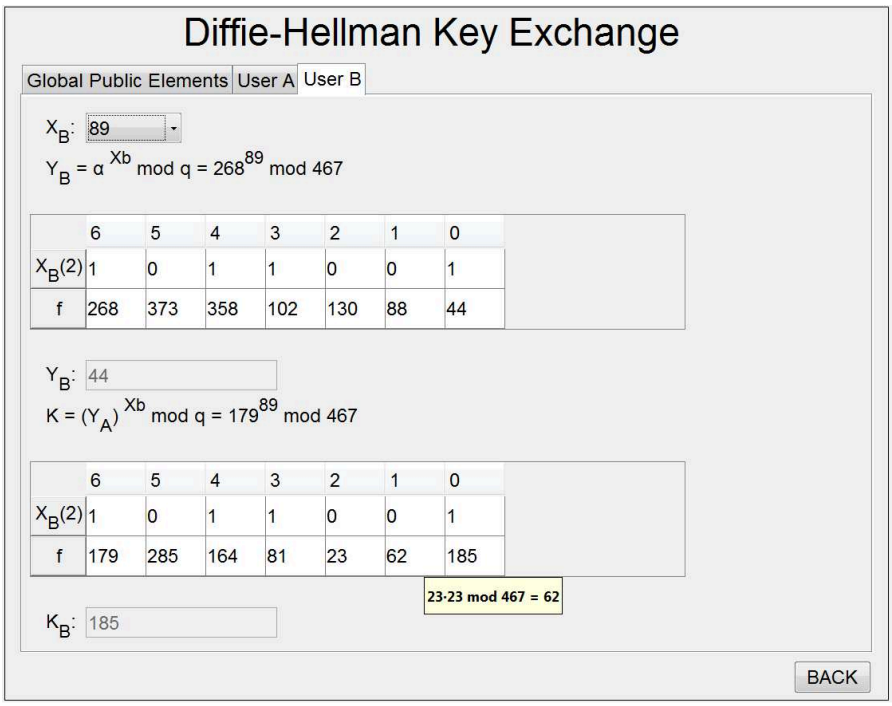
Слика 5.36. Екран за одабир јавно доступних вредности код *Diffie-Hellman* алгоритма у COALA систему

све целе позитивне бројеве мање од одабране вредности за q, а по колонама све степене сваког од бројева почевши од 1 до $q-1$ по модулу q. За сваку врсту у табели бојом је означена најдужа секвенца без понављања. Само врсте у којима је читава секвенца без понављања, односно приликом степеновања по модулу q се добија 1 тек код експонента $q-1$, представљају просте корене за q и могу бити одабране као вредности за α. Само те вредности су и понуђене кориснику у падајућој листи за одабир вредности α.

Након одабира јавно доступних вредности потребно је одабрати тајне вредности за корисника А и корисника Б. За кориснике А и Б посебно је приказан процес формирања заједничке тајне вредности (слика 5.37а и 5.37б, респективно). Када се изабере тајна вредност за корисника А, види се на који начин је формирана јавна вредност корисника А. Приказан је и пример коришћења алгоритма за брзу експонентизацију. Исти случај је и за корисника Б. Када се одаберу тајне вредности за оба корисника, за сваког корисника се приказује како је формирана заједничка тајна вредност. И овде је демонстрирано коришћење ал–



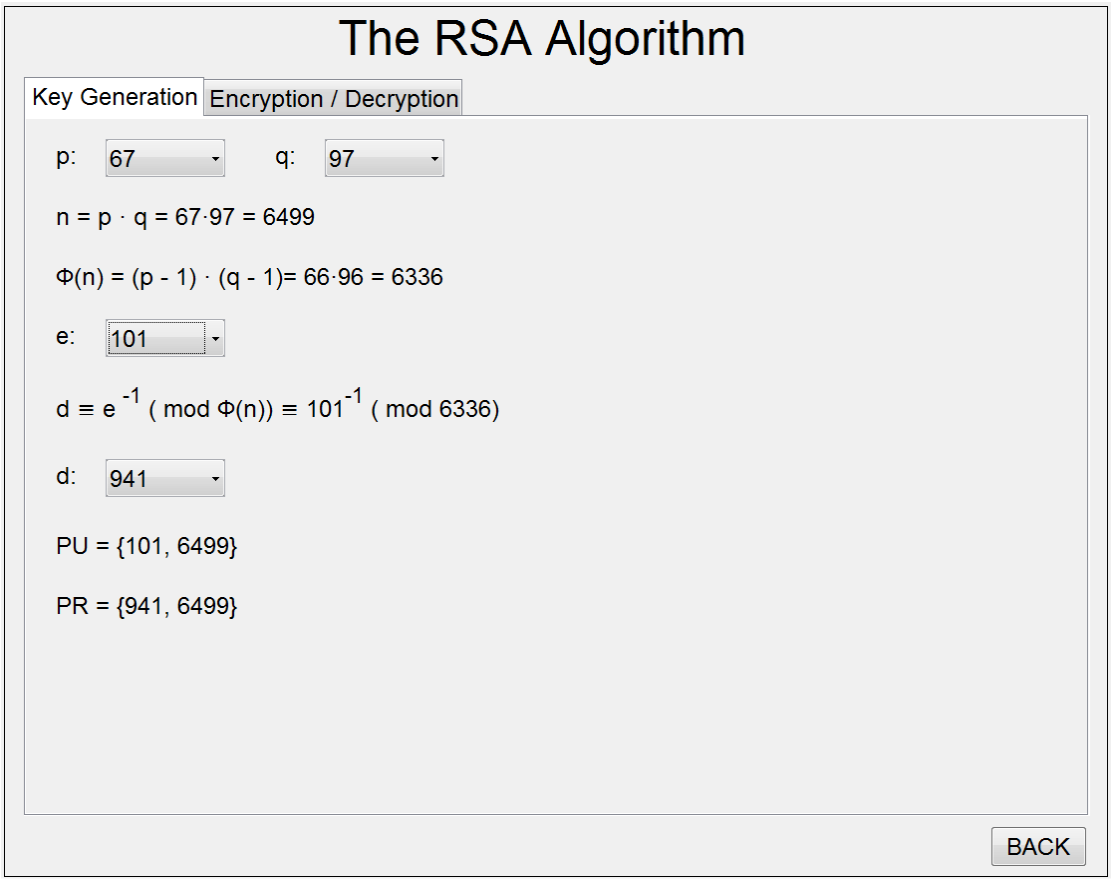
а)



б)

Слика 5.37. Екрани за израчунавање заједничке тајне вредности за кориснике А и Б код *Diffie-Hellman* алгоритма у COALA систему

горитма за брзу експонентизацију.

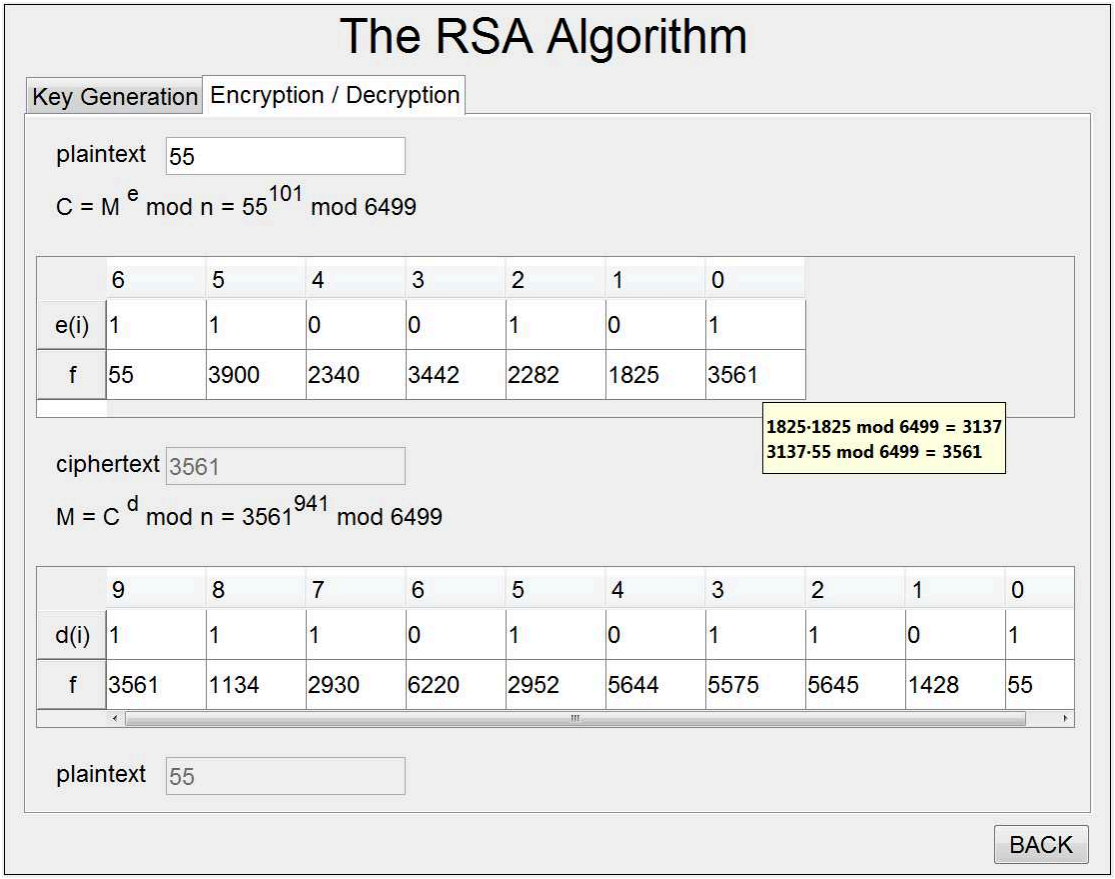


Слика 5.38. Екран за формирање пара кључева код RSA алгоритма у COALA систему

5.5.2. RSA АЛГОРИТАМ

Када корисник изабере визуелну репрезентацију RSA алгоритма на почетном екрану, најпре се појављује екран на коме треба да се формира пар кључева (слика 5.38). Када корисник одабере просте бројеве *p* и *q* приказује се израчуната вредност за n, као и за Φ(n). Након тога се у падајућој листи за *e* појављују све вредности које испуњавају задате услове. Када корисник одабере вредност за *e*, у падајућој листи се појављују вредности које испуњавају задате услове за *d*. На крају, корисник бира и вредност за *d*, чиме је поступак формирања пара кључева завршен и приказани су формирани приватни и јавни кључ.

Након формирања пара кључева могуће је видети поступак шифровања и дешифровања за произвољно одабрану вредност оригиналне поруке у складу са правилом RSA алгоритма (слика 5.39). На истом екрану приказан је и поступак шифровања и поступак дешифровања, уз приказ употребе алгоритма за брзу експонентизацију.



Слика 5.39. Екран за приказ шифровања и дешифровања код RSA алгоритма у COALA систему

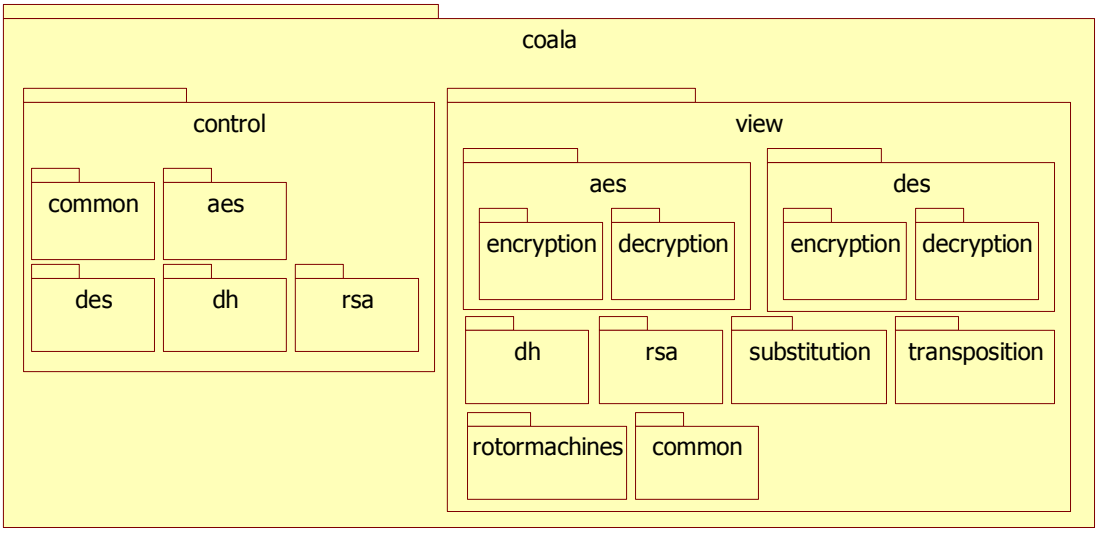
6. ИМПЛЕМЕНТАЦИЈА COALA СИСТЕМА

У овом поглављу биће објашњени детаљи везани за софтверску имплементацију COALA система. Прво ће бити дат преглед технологија коришћених за софтверску имплементацију. Након тога биће објашњена структура софтверског решења. Потом ће бити дати детаљи неопходних проширења стандардног API-ја (*Application Programming Interface*) која су направљена. На крају ће бити објашњен начин извршавања алгоритама подржаних у COALA систему.

6.1. КОРИШЋЕНА ТЕХНОЛОГИЈА

С обзиром на то да је COALA софтверски систем за визуелну репрезентацију криптографских алгоритама, приликом имплементације најпре је било потребно одабрати програмски језик у коме ће систем бити реализован. Данас је у употреби велики број различитих програмских језика од којих сваки има своје предности и недостатке. За имплементацију COALA система одабран је програмски језик *Java* [80] из неколико разлога. Прво, програми написани у програмском језику *Java* су платформски независни, па самим тим имају висок степен портабилности. Друго, програмски језик *Java* је бесплатан за коришћење и постоји велики број различитих развојних окружења који подржавају језик *Java* и који су и сами бесплатни за коришћење. Треће, аутор има доста искуства у програмирању у програмском језику *Java*.

Како је у COALA систему акценат на визуелној репрезентацији криптографских алгоритама, након одабира програмског језика било је потребно одабрати и развојно окружење које ће бити коришћено за имплементацију. Велики број различитих развојних окружења који подржавају програмски језик *Java* имају и добру подршку за развој апликација са графичким корисничким интерфејсом (енгл. *GUI applications*). За имплементацију COALA система одабрано је *NetBeans* развојно окружење [81] које је бесплатно за коришћење и има добар интерфејс за једноставан развој апликација са графичким корисничким интерфејсом. Коришће–



Слика 6.1. Структура пакета у софтверској имплементацији COALA система

не су различите верзије овог окружења, закључно са верзијом 7.0.1. За развој графичког корисничког интерфејса коришћен је *Swing* API, који обезбеђује велики број разноврсних компоненти потребних за развој графичког корисничког интерфејса. У коришћеним верзијама *NetBeans* развојног окружења *Swing* API је подржан и уз помоћ *Swing Application Framework* оквира који омогућава једноставнији развој апликација са графичким корисничким интерфејсом.

6.2. СТРУКТУРА СОФТВЕРСКОГ РЕШЕЊА

Класе написане у програмском језику *Java* које су коришћене за софтверску имплементацију COALA система подељене су у пакете чија је структура приказана на слици 6.1.

Главни пакет назван је *coala* и у овај пакет су смештени сви остали пакети, као и главна класа којом се покреће апликација: *COALAApp*. Све остале класе подељене су у две групе: оне које се односе на извршавање самих алгоритама и оне које се односе на визуелну репрезентацију алгоритама. Због тога су направљена два пакета унутар главног пакета *coala* и то: пакет *control* за прву групу класа и пакет *view* за другу групу класа.

6.2.1. ПАКЕТ *CONTROL*

У пакету *control* налази се пет пакета: *common*, *aes*, *des*, *dh* и *rsa* који обухватају све класе које су неопходне за извршавање алгоритама. За класичне криптографске алгоритме, који су сами по себи једноставни, није било неопходно

раздвајати део са извршавањем алгоритма од дела са визуелном репрезентацијом алгоритма.

Пакет *coala.control.common* садржи класе које представљају проширења стандардног API-ја која су била неопходна како би алгоритми могли бити коректно имплементирани:

- *BinaryValue* – класа која служи за представљање бинарних података,
- *State* – класа која служи за бележење појединачног стања у току извршавања код симетричних блок алгоритама,
- *VectorState* – класа која служи за чување свих међустања у току извршавања код симетричних блок алгоритама и
- *Constants* – класа која служи за чување константних вредности које се користе приликом извршавања различитих алгоритама.

Пакет *coala.control.aes* садржи класе које омогућавају извршавање AES алгоритма:

- *KeyExpansion* – класа која омогућава извршавање операције експанзије кључа,
- *SBox* – класа која омогућава извршавање операције замене бајтова,
- *ShiftRows* – класа која омогућава извршавање операције померања редова и
- *MixColumns* – класа која омогућава извршавање операције мешања колона.

Пакет *coala.control.des* садржи класу која омогућава извршавање DES алгоритма:

- *Function* – класа која омогућава извршавање свих операција DES алгоритма.

Пакет *coala.control.dh* садржи класу која омогућава извршавање *Diffie-Hellman* алгоритма:

- *KeyExchange* – класа која омогућава извршавање свих операција *Diffie-Hellman* алгоритма.

Пакет *coala.control.rsa* садржи класу која омогућава извршавање RSA

алгоритма:

- *Key* – класа која представља структуру која се користи као кључ код RSA алгоритма и
- *RSA* – класа која омогућава извршавање свих операција RSA алгоритма.

6.2.2. ПАКЕТ *VIEW*

У пакету *view* налази се осам пакета: *common*, *aes*, *des*, *dh*, *rsa*, *substitution, transposition, rotormachines* који обухватају све класе које су неопходне за визуелну репрезентацију алгоритама. Поред тога ту је и класа која служи за приказ почетног екрана COALA система: *COALAMainView*.

Пакет *coala.view.common* садржи класе које представљају проширења *Swing* API-ја која су била неопходна како би алгоритми могли бити коректно имплементирани:

- *NLabel* – класа која служи за генерисање лабела неопходних за приказ дијаграма код симетричних блок алгоритама,
- *IPanel* – класа која служи за генерисање панела који имају слику у позадини,
- *LabelMatrix* – класа која служи за исцртавање матрице лабела које могу бити међусобно повезане и
- *FixedTable* – класа која служи за приказ табела на одговарајући начин код алгоритама са јавним кључем.

Пакет *coala.view.aes* садржи класе које су неопходне за визуелну репрезентацију AES алгоритма заједничке за шифровање и дешифровање:

- *COALAAESMainView* – класа која служи за приказ екрана за избор операције,
- *AESDataAndKeyInputView* – класа која служи за приказ екрана за унос улазних параметара,
- *AESExpandKeyDetailWithGView, AESExpandKeyDetailWithoutGView* – класе које служе за приказ екрана са детаљима операције експанзије кључа и
- *AESNLabelEvents* и сродне класе – класе које служе за приказ дела за

навигацију код симетричних блок алгоритама.

Пакет *coala.view.aes.encryption* садржи класе које су неопходне за визуелну репрезентацију AES алгоритма специфичне за шифровање:

- *AESEncryptionFirstRoundView* – класа која служи за приказ прве итерације алгоритма,
- *AESEncryptionNRoundView* – класа која служи за приказ свих итерација алгоритма осим прве и последње,
- *AESEncryptionLastRoundView* – класа која служи за приказ последње итерације алгоритма,
- *AESEncryptionSBoxView* – класа које служи за приказ табеле замене и
- *AESEncryptionMixColumnsDetailView* – класа које служи за приказ детаља операције мешања колона.

Пакет *coala.view.aes.decryption* садржи класе које су неопходне за визуелну репрезентацију AES алгоритма специфичне за дешифровање:

- *AESDecryptionFirstRoundView* – класа која служи за приказ прве итерације алгоритма,
- *AESDecryptionNRoundView* – класа која служи за приказ свих итерација алгоритма осим прве и последње,
- *AESDecryptionLastRoundView* – класа која служи за приказ последње итерације алгоритма и
- *AESDecryptionSBoxInverseView* – класа које служи за приказ табеле замене.

Пакет *coala.view.des* садржи класе које су неопходне за визуелну репрезентацију DES алгоритма заједничке за шифровање и дешифровање:

- *COALADESMainView* – класа која служи за приказ екрана за избор операције,
- *DESDataAndKeyInputView* – класа која служи за приказ екрана за унос улазних параметара,
- *DESSBox* – класа која служи за приказ екрана са детаљима операције замене,

- *DESNLabelEvents* и сродне класе – класе које служе за приказ дела за навигацију код симетричних блок алгоритама,
- *LabelMatrixEvents* и сродне класе – класе које служе за приказ делова који користе *LabelMatrix* компоненте.

Пакет *coala.view.des.encryption* садржи класе које су неопходне за визуелну репрезентацију DES алгоритма специфичне за шифровање:

- *DESEncryptionFirstRoundView* – класа која служи за приказ прве итерације алгоритма,
- *DESEncryptionNRoundView* – класа која служи за приказ свих итерација алгоритма осим прве и последње и
- *DESEncryptionLastRoundView* – класа која служи за приказ последње итерације алгоритма.

Пакет *coala.view.des.decryption* садржи класе које су неопходне за визуелну репрезентацију DES алгоритма специфичне за дешифровање:

- *DESDecryptionFirstRoundView* – класа која служи за приказ прве итерације алгоритма,
- *DESDecryptionNRoundView* – класа која служи за приказ свих итерација алгоритма осим прве и последње и
- *DESDecryptionLastRoundView* – класа која служи за приказ последње итерације алгоритма.

Пакет *coala.view.dh* садржи класе које су неопходне за визуелну репрезентацију *Diffie-Hellman* алгоритма:

- *COALADHMainView* – класа која служи за приказ свих операција *Diffie-Hellman* алгоритма.

Пакет *coala.view.rsa* садржи класе које су неопходне за визуелну репрезентацију RSA алгоритма:

- *COALARSAMainView* – класа која служи за приказ свих операција RSA алгоритма.

Пакет *coala.view.substitution* садржи класе које су неопходне за визуелну репрезентацију супституционих алгоритама:

- *COALACaesarMainView* – класа која служи за приказ извршавања *Caesar* алгоритма,
- *COALACaesarAnalysisView* – класа која служи за приказ криптоанализе *Caesar* алгоритма,
- *COALAMonoalphabeticMainView* – класа која служи за приказ извршавања моноалфабетског алгоритма,
- *COALAPlayfairMainView* – класа која служи за приказ извршавања *Playfair* алгоритма и
- *COALAVigenereMainView* – класа која служи за приказ извршавања *Vigenere* алгоритма.

Пакет *coala.view.transposition* садржи класе које су неопходне за визуелну репрезентацију транспозиционих алгоритама:

- *COALARailFenceMainView* – класа која служи за приказ извршавања *Rail Fence* алгоритма и
- *COALARowTranspositionMainView* – класа која служи за приказ извршавања *Row Transposition* алгоритма.

Пакет *coala.view.rotormachines* садржи класу која је неопходна за визуелну репрезентацију продукционих алгоритама:

- *COALARotorMachineMainView* – класа која служи за приказ извршавања *Rotor Machine* алгоритма.

6.3. ПРОШИРЕЊА КОРИШЋЕНИХ API-ЈА

У оквиру имплементације COALA система јавила се потреба за новим класама и новим компонентама које нису дефинисане у стандардном *Java* API-ју, односно у *Swing* API-ју. Класе које проширују стандардни *Java* API користиле су се у имплементацији сложених алгоритама за складиштење и обраду специфичних структура података које се користе у алгоритмима. Класе које проширују *Swing* API користиле су се за прављење нових компонената графичког корисничког интерфејса неопходних за визуелну репрезентацију алгоритама.

6.3.1. ПРОШИРЕЊА СТАНДАРДНОГ *JAVA* API-ЈА

Класе које представљају проширења стандардног API-ја смештене у пакету

```
public String getBinaryValue(int nBits){
    String format="";
    String bin = Long.toBinaryString(value);
    if(nBits > bin.length()){
        for(int i=0;i<(nBits-bin.length());i++){
            format+="0";
        }
    }
    return format+bin;
}
```

Слика 6.2. Метода *getBinaryValue* класе *BinaryValue*

coala.control.common су: *BinaryValue*, *State*, *VectorState* и *Constants*.

BinaryValue класа је осмишљена да омогући складиштење и рад са бинарним подацима. Ова класа омогућава да се сачувана вредност прикаже на одговарајући начин у бинарној репрезентацији, а такође омогућава и да се изврше одређене операције над бинарним подацима. Вредност која представља сачувани податак је типа *long* и може се поставити и дохватити као *long* или као *int* помоћу метода *setLongValue*, *getLongValue*, *setIntValue* и *getIntValue*, респективно. Сачувана вредност се помоћу дефинисаних метода може приказати на различите начине у зависности од потребе алгоритма у коме се примењује.

Неке од најзначајнијих метода класе *BinaryValue* су:

- *getBinaryValue* (слика 6.2) – метода која враћа вредност као *String* фиксне дужине који представља њену бинарну репрезентацију допуњену водећим нулама по потреби,
- *getBinaryValueNoExtension* – метода која враћа вредност као *String* који представља њену бинарну репрезентацију са минималним неопходним бројем бита (без водећих нула),
- *getBinaryValueShiftLeftOneBit* – метода која враћа вредност као *String* фиксне дужине који представља њену бинарну репрезентацију допуњену водећим нулама по потреби и померену за један бит у лево,
- *isHighestWeightBitOne* – метода која за бинарну репрезентацију вредности фиксне дужине враћа да ли је водећа цифра јединица,
- *getHexValue* – метода која враћа вредност као *String* фиксне дужине који представља њену хексадецималну репрезентацију допуњену водећим нулама по потреби,
- *shiftLeftByOneBit* – метода која мења вредност извршавањем операције

```
public State addRoundKey(State iterationKey){
    BinaryValue[][] res = new BinaryValue[4][4];
    for(int i=0;i<4;i++)
        for(int j=0;j<4;j++){
            res[i][j] = new BinaryValue();
            res[i][j].setIntValue(
                this.state[i][j].getIntValue()
                ^
                iterationKey.getBinaryValue(i, j).getIntValue());
        }
    return new State(res);
}
```

Слика 6.3. Метода *addRoundKey* класе *State*

померања у лево за један бит на бинарном нивоу и

- *xor* – метода која извршава бинарну ексклузивно ИЛИ операцију текуће вредности и друге бинарне вредности која се прослеђује као улазни параметар.

State класа омогућава да се за симетричне блок алгоритме (DES и AES) забележи међурезултат сваке појединачне операције у оквиру сваке итерације алгоритма. Како се подаци на различит начин обрађују код ова два алгоритма, класа *State* има два атрибута, од којих је један матрица *BinaryValue* објеката (користи се код AES алгоритма), а други је низ *BinaryValue* објеката (користи се код DES алгоритма).

Неке од најзначајнијих метода класе *State* су:

- *getBinaryValue* – метода која има два различита потписа: у првом случају прихвата две *int* вредности које представљају позицију у матрици *BinaryValue* објеката са које треба дохватити *BinaryValue* објекат, док у другом случају прихвата једну *int* вредност која представља позицију у низу *BinaryValue* објеката са које треба дохватити *BinaryValue* објекат,
- *length* – метода која враћа дужину низа *BinaryValue* објеката и
- *addRoundKey* (слика 6.3) – метода која омогућава извршавање *addRoundKey* операције код AES алгоритма, а заправо представља бајтовску ексклузивно ИЛИ операцију над матрицом *BinaryValue* објеката и одговарајућом матрицом *BinaryValue* објеката из објекта типа *State* који је прослеђен као улазни параметар.

VectorState класа је осмишљена да за симетричне блок алгоритме (DES и AES)

```
public void addState(int position, State state){
    vectorState[position] = state;
    setLastStatePosition(position);
}
```

Слика 6.4. Метода *addState* класе *VectorState*

```
public static final int[][] INITIAL_PERMUTATION = new int[8][8];
static{
   for (int i=0; i<4;i++)
    {
        INITIAL_PERMUTATION[i][0] = i*2+57;
        INITIAL_PERMUTATION[i][1] = i*2+49;
        INITIAL_PERMUTATION[i][2] = i*2+41;
        INITIAL_PERMUTATION[i][3] = i*2+33;
        INITIAL_PERMUTATION[i][4] = i*2+25;
        INITIAL_PERMUTATION[i][5] = i*2+17;
        INITIAL_PERMUTATION[i][6] = i*2+9;
        INITIAL_PERMUTATION[i][7] = i*2+1;

        INITIAL_PERMUTATION[(i*8+32)/8][0] = i*2+56;
        INITIAL_PERMUTATION[(i*8+33)/8][1] = i*2+48;
        INITIAL_PERMUTATION[(i*8+34)/8][2] = i*2+40;
        INITIAL_PERMUTATION[(i*8+35)/8][3] = i*2+32;
        INITIAL_PERMUTATION[(i*8+36)/8][4] = i*2+24;
        INITIAL_PERMUTATION[(i*8+37)/8][5] = i*2+16;
        INITIAL_PERMUTATION[(i*8+38)/8][6] = i*2+8;
        INITIAL_PERMUTATION[(i*8+39)/8][7] = i*2;
    }
}
```

Слика 6.5. Пример константне таблице *INITIAL_PERMUTATION* класе *Constants*

забележи све међурезултате приликом извршавања алгоритма. Због тога као атрибут ова класа садржи низ објеката типа *State*, као и једну *int* вредност која служи да покаже која је позиција последњег додатог објекта у низу.

Неке од најзначајнијих метода класе *VectorState* су:

- *addState* (слика 6.4) – метода која додаје нови објекат типа *State* у низ на позицији која се задаје као улазни параметар,
- *getLastStatePosition* – метода која враћа вредност атрибута *lastStatePosition*,
- *getState* – метода која враћа објекат типа *State* из низа са позиције која је одређена вредношћу добијеном као улазни параметар и
- *setLastStatePosition* – метода помоћу које се може поставити вредност атрибута *lastStatePosition*.

Constants класа је пројектована тако да на једном месту буду све константе вредности које се користе у осталим класама. На слици 6.5 приказан је пример таблице која се користи за иницијалну пермутацију код DES алгоритма.

6.3.2. ПРОШИРЕЊА *SWING* API-ЈА

Класе које представљају проширења *Swing* API-ја смештене у пакету *coala.view.common* су: *NLabel*, *IPanel*, *LabelMatrix* и *FixedTable*. Ове класе заправо представљају проширења компонената графичког корисничког интерфејса које постоје у оквиру *Swing* API-ја, али помоћу којих би било јако тешко имплементирати визуелну репрезентацију. *NetBeans* окружење омогућава да се ове нове компоненте могу користити на исти начин као већ постојеће у оквиру дизајнера графичког корисничког интерфејса.

NLabel класа проширује компоненту *JLabel* из *Swing* API-ја (javax.swing.JLabel) тако да нова компонента представља активну лабелу која се може користити приликом дизајнирања дела за навигацију код DES и AES алгоритама. Нова компонента најпре решава проблем подешавања параметара лабеле, с обзиром да се у делу за навигацију користи велики број ових компоненти, јер алгоритми имају велики број итерација и у оквиру итерација велики број операција, а за сваку операцију сваке итерације би требало да постоји по једна компонента. Коришћењем *NLabel* класе сви параметри су постављени само једанпут. Поред тога, лабеле које ће се користити у делу за навигацију код DES и AES алгоритама треба да реагују на прелазак показивача миша преко лабеле и клик дугмета миша на лабелу. Коришћењем *NLabel* класе могуће је само на једном месту додати ослушкиваче (енгл. *listeners*) за ове врсте догађаја. Како би било могуће да се конкретне акције које лабела треба да изврши у оквиру ослушкивача буду накнадно дефинисане, с обзиром да неће увек бити исте, направљен је интерфејс *NLabelEvents* који само садржи потписе метода које касније треба дефинисати. Осим ових погодности које је креирање класе *NLabel* донело, битна ствар је и додавање два нова атрибута *roundNumber* (број итерације) и *operationNumber* (број операције у оквиру итерације) који служе да означе позицију објекта типа *NLabel* у оквиру дела за навигацију конкретног алгоритма. На слици 6.6 приказана је метода *createLabel* која дефинише све заједничке параметре компоненте типа *NLabel.*

IPanel класа проширује компоненту *JPanel* из *Swing* API-ја (javax.swing.JPanel) тако да нова компонента представља панел у чијој позадини се може приказати слика. Ова компонента се може користити приликом дизајнирања дела за навига–

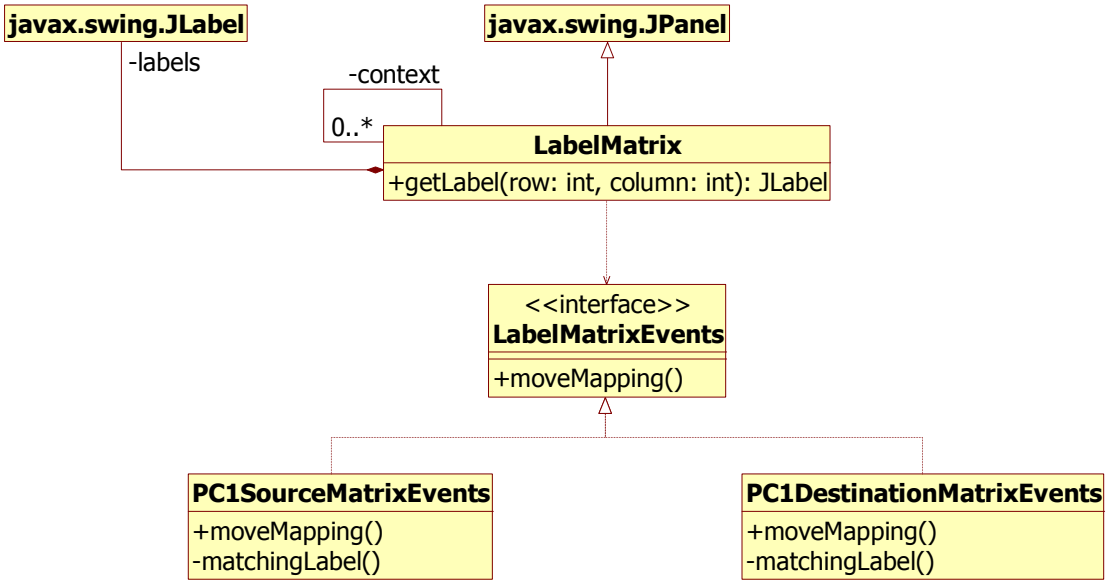
```
public void createLabel(){

    this.setFont(resourceMap.getFont("label.font"));
    this.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    this.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
    this.setMaximumSize(new java.awt.Dimension(LABEL_WIDTH, LABEL_HEIGTH));
    this.setMinimumSize(new java.awt.Dimension(LABEL_WIDTH, LABEL_HEIGTH));
    this.setOpaque(true);
    this.setPreferredSize(new java.awt.Dimension(LABEL_WIDTH, LABEL_HEIGTH));
    if (isHasMouseMovedEvent()){
                this.addMouseListener(new java.awt.event.MouseAdapter(){
                @Override
                public void mouseEntered(java.awt.event.MouseEvent evt) {
                    getEvents().mouseMotion(true, (NLabel) evt.getSource());
                }
                @Override
                public void mouseExited(java.awt.event.MouseEvent evt) {
                    getEvents().mouseMotion(false, (NLabel) evt.getSource());
                }
                @Override
                public void mouseClicked(java.awt.event.MouseEvent evt) {
                    getEvents().mouseClicked(roundNumber, operationNumber, (NLabel) evt.getSource());
                }
                }
                );
    }
}
```

Слика 6.6. Метода *createLabel* класе *NLabel*

цију код DES и AES алгоритама како би био приказан статички део алгоритма.

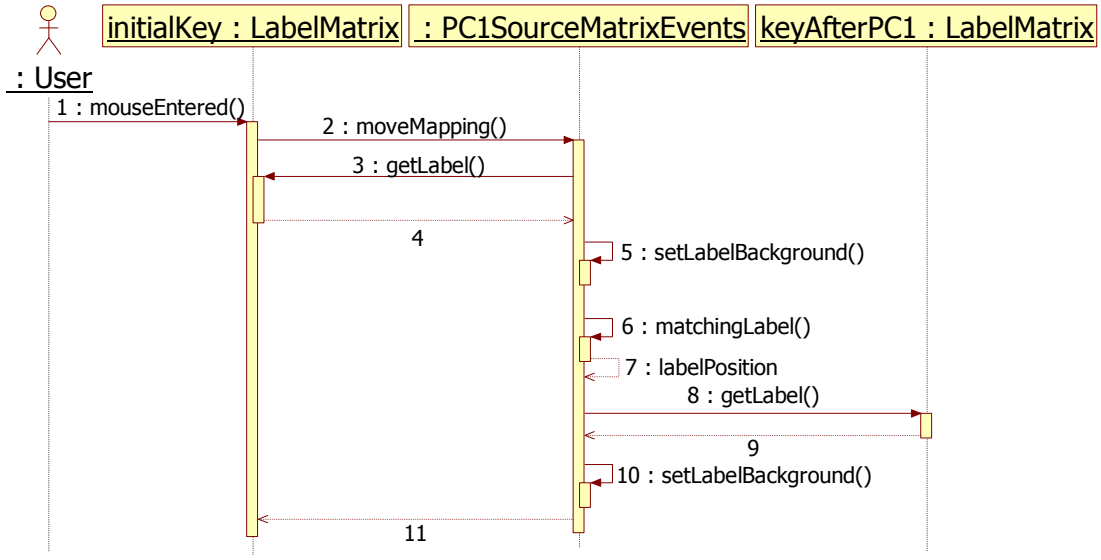
LabelMatrix класа проширује компоненту *JPanel* из *Swing* API-ја (javax.swing.JPanel) тако да нова компонента представља панел који садржи матрицу лабела са одређеним новим могућностима. Нова компонента се користи за приказ стања код симетричних блок алгоритама (DES и AES). Како се код AES алгоритма операције извршавају на нивоу бајта, стања и кључеви итерације се приказују подељени на бајтове у виду матрице величине 4x4 бајта. Код DES алгоритма је одређено да се подаци и кључеви итерације такође приказују у виду матрице, али на нивоу бита, с тим што код овог алгоритма величина матрице није фиксна. *LabelMatrix* компонента је флексибилна по питању величине матрице која ће бити приказана. Класа *LabelMatrix* доноси све оне предности као што смо видели и код класе *NLabel* по питању једноставнијег дизајнирања графичког корисничког интерфејса.

На слици 6.7 дат је класни дијаграм који приказује пример коришћења *LabelMatrix* класе за приказ података у оквиру једне операције DES алгоритма. Класа *LabelMatrix* наслеђује класу *JPanel*. Димензије матрице лабела представљене објектом класе *LabelMatrix* одређују се приликом креирања тог објекта. Свакој од лабела у оквиру матрице може се приступити помоћу индекса врсте и колоне којој лабела припада. Поред лабела за приказ бајтова, односно би–



Слика 6.7. Класни дијаграм примера коришћења класе *LabelMatrix*

това, стања објекат класе *LabelMatrix* садржи и лабелу за приказ хексадецималне вредности целокупног податка. Видљивост овог податка није обавезна, а уколико је укључена, хексадецимална вредност ће бити исписана у дну матрице. Да би се ефикасније објаснила веза између битова, односно бајтова, у матрицама које представљају операнде и матрици која представља резултат одређене операције алгоритма, било је потребно детектовати покрете миша корисника и селектовање одговарајућих битова, односно бајтова, у складу са логиком операције. Како би се избегло писање метода за обраду догађаја покрета миша (енгл. *handler*) за сваку лабелу понаособ, осмишљено је да се ове методе аутоматски генеришу приликом креирања матрице. У оквиру метода за обраду догађаја покрета миша позива се метода *moveMapping* интерфејса *LabelMatrixEvents*. Да би се приказала повезаност матрица у оквиру операција алгоритама за различите операције је било потребно направити различите имплементације интерфејса *LabelMatrixEvents*. Објекту класе *LabelMatrix* се приликом креирања додељује одговарајући објекат који имплементира овај интерфејс. Пошто се матрица најчешће налази у вези са бар још једном или већим бројем матрица, да би се у оквиру обраде покрета мишем могло приступити објектима са којима је дата матрица спрегнута, објекат класе *LabelMatrix* садржи низ објеката који је потребно попунити објектима класе *LabelMatrix* спрегнутих матрица. Генерисање метода за обраду догађаја покрета миша је опционо, а да ли ће нека матрица реаговати на покрете миша или не одре–



Слика 6.8. Дијаграм секвенце за догађај померања миша у оквиру матрице иницијалног кључа на табу операције *Permuted Choice One* код DES алгоритма

ђује се приликом креирања матрице. На слици 6.7 приказане су само две класе које се користе за обраду догађаја код операције *Permuted Choice One* код DES алгоритма.

На слици 6.8 приказан је дијаграм секвенце за догађај померања миша у оквиру матрице иницијалног кључа на табу операције *Permuted Choice One*. Објекат класе *PC1SourceMatrixEvents* везује се за објекат класе *LabelMatrix* којим је приказан иницијални кључ. Када корисник помери показивач миша преко површине неког бајта иницијалног кључа, окида се догађај чији је метод за обраду догађаја креиран аутоматски приликом креирања самог објекта *initialKey*. У оквиру методе *moveMapping* најпре се детектује ред и колона лабеле којом је дати бајт представљен, затим се лабела дохвата методом *getLabel* и мења се њена боја. Затим се проналазе ред и колона лабеле којом је дати бајт представљен након што се примени операција *Permuted Choice One*. Ово мапирање се обавља помоћу матрице константних вредности дефинисане самим алгоритмом. Да би се обојила одговарајућа лабела у матрици која представља кључ након операције *Permuted Choice One* приступа се низу *LabelMatrix* објеката са којима је објекат *initialKey* спрегнут и затим се приступа одговарајућој лабели, којој се потом мења боја.

FixedTable класа проширује компоненту *JTable* из *Swing* API-ја (javax.swing.JTable) тако да нова компонента представља табелу код које су колоне фиксиране и не може им се мењати величина ни редослед, као ни садржај поља.

Ова компонента се може користити приликом дизајнирања дела за приказ рада алгоритма за брзу експонентизацију код алгоритама са јавним кључем (Diffie-Hellman и RSA).

6.4. НАЧИН ИЗВРШАВАЊА АЛГОРИТАМА

За све подржане алгоритме имплементација алгоритама у COALA систему написана је од нуле, иако је за већину алгоритама могуће наћи готове имплементације за разне програмске језике на Интернету. Два су разлога из којих је имплементација алгоритама писана испочетка. Први разлог је што је имплементација морала да одговара спецификацији алгоритама која се користи на предавањима и вежбама, док у готовим решењима обично има доста импровизација. Други, важнији разлог, је то што је имплементација морала да буде прилагођена одабраном начину визуелне репрезентације, односно да омогући да се сви детаљи у извршавању алгоритама могу приказати. У свим постојећим имплементацијама омогућено је само да се за улазне параметре добије излазна вредност. Код класичних криптографских алгоритама (супституциони, транспозициони и продукциони) није било великих изазова приликом имплементације, с обзиром да су и сами алгоритми једноставни. У наставку су објашњени проблеми који су се јавили приликом имплементације симетричних блоковских алгоритама и алгоритама са јавним кључем.

6.4.1. НАЧИН ИЗВРШАВАЊА СИМЕТРИЧНИХ БЛОК АЛГОРИТАМА

Код симетричних блок алгоритама (DES и AES) изазов је био имплементирати извршавање алгоритама тако да буде омогућен приказ сваког појединачног корака у алгоритму. Како су ово итеративни алгоритми и одлучено је да је у једном тренутку потребно приказати једну итерацију, извршавање се одвија итерацију по итерацију. У свакој итерацији чувају се међурезултати свих операција у оквиру итерације и приказују кориснику на начин описан у поглављу 4.2.

Код DES алгоритма извршавање операција самог алгоритма дефинисано је у класи *Function* која се налази у пакету *coala.control.des*. Неке од значајнијих метода ове класе су:

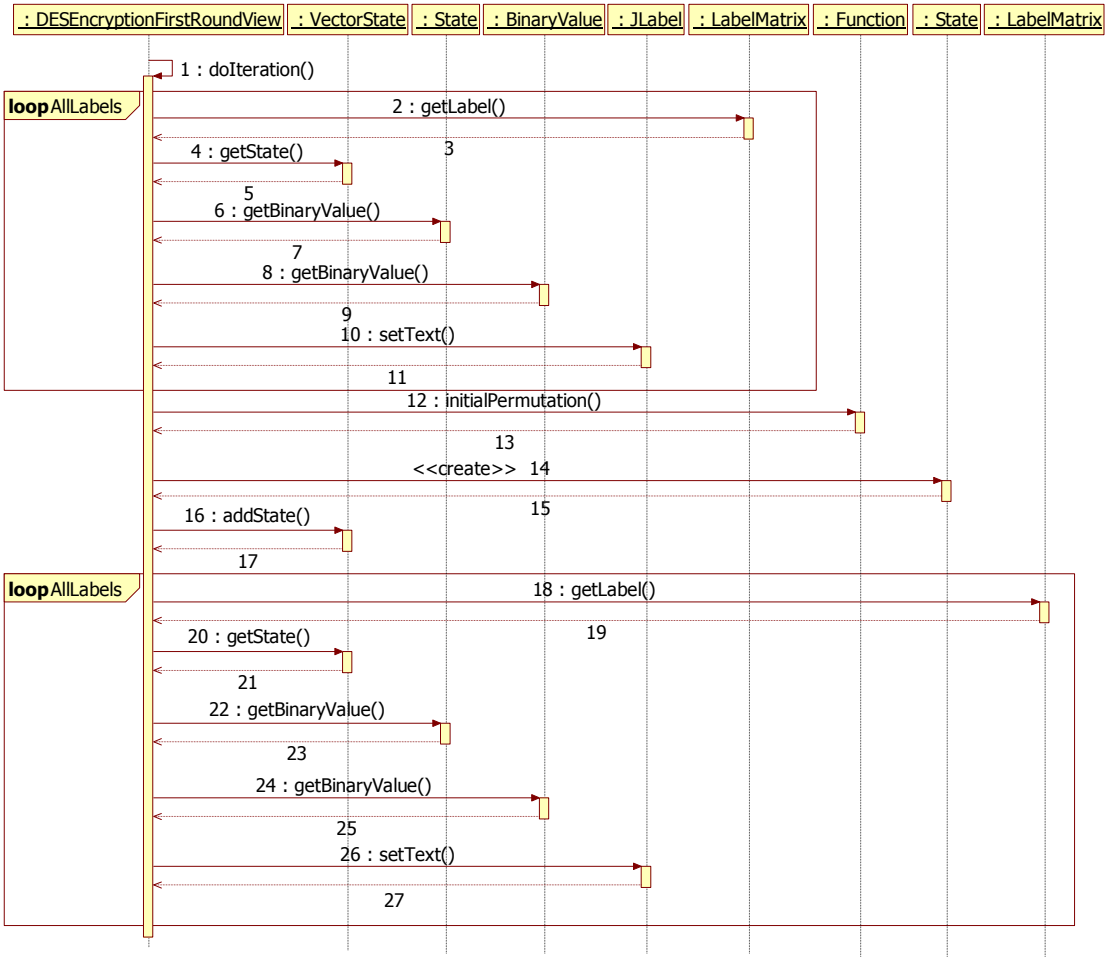
- *initialPermutation* – метода која извршава операцију иницијалне пермутације над улазним параметром типа *State*,

```
//56 bit input, 48 bit output
public BinaryValue[] permutedChoiceTwo(State input){
    BinaryValue[] output = new BinaryValue[48];
    for(int i=0; i<output.length; i++)
        output[i] = new BinaryValue();
    for (int i=0; i<8; i++)
        for (int j=0; j<6; j++)
            copy(output[i*6+j],
                 input.getBinaryValue(Constants.PERMUTED_CHOICE_TWO[i][j]-1));
    return output;
}
```

Слика 6.9. Метода *permutedChoiceTwo* класе *Function*

- *inverseInitialPermutation* – метода која извршава операцију инверзне иницијалне пермутације над улазним параметром типа *State*,
- *expansionPermutation* – метода која извршава операцију пермутације са проширивањем (E) над улазним параметром типа *State*,
- *permutationFunction* – метода која извршава операцију пермутације (P) над улазним параметром типа *State*,
- *xor* – метода која извршава операцију ексклузивно ИЛИ над два улазна параметра типа *State*,
- *sBox* – метода која извршава операцију замене над улазним параметром типа *State*,
- *permutedChoiceOne* – метода која извршава операцију пермутације PC1 над улазним параметром типа *State*,
- *permutedChoiceTwo* (слика 6.9) – метода која извршава операцију пермутације PC2 над улазним параметром типа *State*,
- *swap* – метода која извршава операцију замене леве и десне половине над улазним параметром типа *State*,
- *leftShift* – метода која извршава операцију кружног померања улево над улазним параметром типа *State* за одређен број бита у зависности од броја итерације који је такође улазни параметар и
- *calculateRoundKeys* – метода која извршава израчунавање свих кључева итерације на основу улазног параметра типа *State* који представља иницијални кључ.

Спрега између класа из пакета *coala.view.des* које служе за визуелну репрезентацију алгоритма и класе *Function* приказана је на слици 6.10 где је приказан дијаграм секвенце извршавања дела прве итерације алгоритма. Када ко–



Слика 6.10. Дијаграм секвенце дела извршавања прве итерације DES алгоритма у COALA систему

рисник одабере да започне шифровање DES алгоритмом први корак је уношење оригиналног блока за шифровање и кључа. Након уноса улазних параметара корисник започиње шифровање и креира се објекат типа *DESEncryptionFirstRoundView*. Приликом креирања овог објекта у конструктору се најпре обаве све радње неопходне за исцртавање самог прозора и на крају се позива метода *doIteration* (слика 6.10-1), која служи за извршавање прве итерације алгоритма и попуњавање свих међурезултата који ће бити приказани. Део ове методе који извршава иницијалну пермутацију и попуњава одговарајући део приказа у визуелној репрезентацији је демонстриран на слици 6.10. Најпре се креира петља која пролази кроз све *JLabel* објекте који се налазе у *initialData* објекту, који је типа *LabelMatrix*. Овај објекат заправо представља иницијалну матрицу која садржи унети оригинални блок података. У петљи се најпре дохвата *JLabel* објекат позивом методе *getLabel* (слика 6.10-2) у оквиру *initialData* објекта

са одговарајућим индексом врсте и колоне. Затим се дохвата одговарајуће стање позивом *getState* методе (слика 6.10-4) у оквиру *vectorState* објекта, који је *VectorState* типа. За добијени објекат типа *State* позива се метод *getBinaryValue* (слика 6.10-6) са индексом који је кореспондентан са позицијом дохваћеног *JLabel* објекта у матрици и добија се одговарајући *BinaryValue* објекат. На крају петље се позива метода *getBinaryValue* (слика 6.10-8) у оквиру добијеног *BinaryValue* објекта која враћа *String* репрезентацију одговарајуће вредности овог објекта која се затим поставља као текст *JLabel* објекта позивом методе *setText* (слика 6.10-10) у оквиру тог објекта. Након што се петља заврши матрица која је приказана као иницијална вредност оригиналног блока података је попуњена одговарајућим вредностима. Након тога извршава се операција иницијалне пермутације позивом методе *initialPermutation* (слика 6.10-12) у оквиру објекта типа *Function*. Ова метода враћа низ *BinaryValue* објеката који користимо да направимо нови објекат типа *State* (слика 6.10-14). Затим новонастали објекат типа *State* додајемо у објекат *vectorState* позивом методе *addState* (слика 6.10-16) овог објекта и на тај начин смо извршили операцију иницијалне пермутације. На крају је потребно приказати и вредност након извршене операције иницијалне пермутације на екрану. Прави се нова петља за попуњавање вредности матрице са резултатом операције иницијалне пермутације. У петљи се најпре дохвата објекат типа *JLabel* позивом методе *getLabel* (слика 6.10-18) у оквиру *dataAfterIP* објекта са одговарајућим индексом реда и колоне. Након тога дохвата се одговарајуће стање позивом методе *getState* (слика 6.10-20) у оквиру *vectorState* објекта, који је типа *VectorState*. За добијени објекат типа *State* позива се метод *getBinaryValue* (слика 6.10-22) са индексом који је кореспондентан са позицијом дохваћеног *JLabel* објекта у матрици и на тај начин се добија објекат типа *BinaryValue*. На крају петље се позива метода *getBinaryValue* (слика 6.10-24) у оквиру добијеног *BinaryValue* објекта која враћа *String* репрезентацију одговарајуће вредности овог објекта која се затим поставља као текст *JLabel* објекта позивом методе *setText* (слика 6.10-26) у оквиру тог објекта.

Код AES алгоритма извршавање операција самог алгоритма дефинисано је у класама *KeyExpansion*, *SBox*, *ShiftRows* и *MixColumns* које се налазе у пакету *coala.control.aes*, док је операција додавања кључа итерације подржана у класи

```
public State expandKey(State previousKey, int round){//round = 1..10
    BinaryValue[][] res = new BinaryValue[4][4];
    SBox sBox = new SBox();

    for(int i=0;i<4;i++){//0. rec
        res[i][0] = sBox.subBytes(previousKey.getBinaryValue((i+1)%4, 3));
        res[i][0].xor(Rcon[round-1][i]);
        res[i][0].xor(previousKey.getBinaryValue(i, 0));
    }

    for(int i=1;i<4;i++)//ostale reci
        for(int j=0;j<4;j++){
            res[j][i] = new BinaryValue();
            res[j][i].setIntValue(res[j][i-1].getIntValue());
            res[j][i].xor(previousKey.getBinaryValue(j, i));
        }

    return new State(res);
}
```

Слика 6.11. Метода *expandKey* класе *KeyExpansion*

State која се налази у пакету *coala.control.common*.

Неке од значајнијих метода класе *KeyExpansion* су:

- *expandKey* (слика 6.11) – метода која извршава експанзију наредног кључа итерације на основу текућег кључа итерације и
- *expandAllKeysForDecryption* – метода која врши експанзију свих кључева итерације на основу иницијалног кључа.

Неке од значајнијих метода класе *SBox* су:

- *doSubBytes* – метода која извршава операцију замене бајтова за објекат типа *State* који је прослеђен као улазни параметар и
- *doInverseSubBytes* – метода која извршава инверзну операцију замене бајтова за објекат типа *State* који је прослеђен као улазни параметар.

Неке од значајнијих метода класе *ShiftRows* су:

- *shiftRows* – метода која извршава операцију померања редова за објекат типа *State* који је прослеђен као улазни параметар и
- *inverseShiftRows* – метода која извршава инверзну операцију померања редова за објекат типа *State* који је прослеђен као улазни параметар.

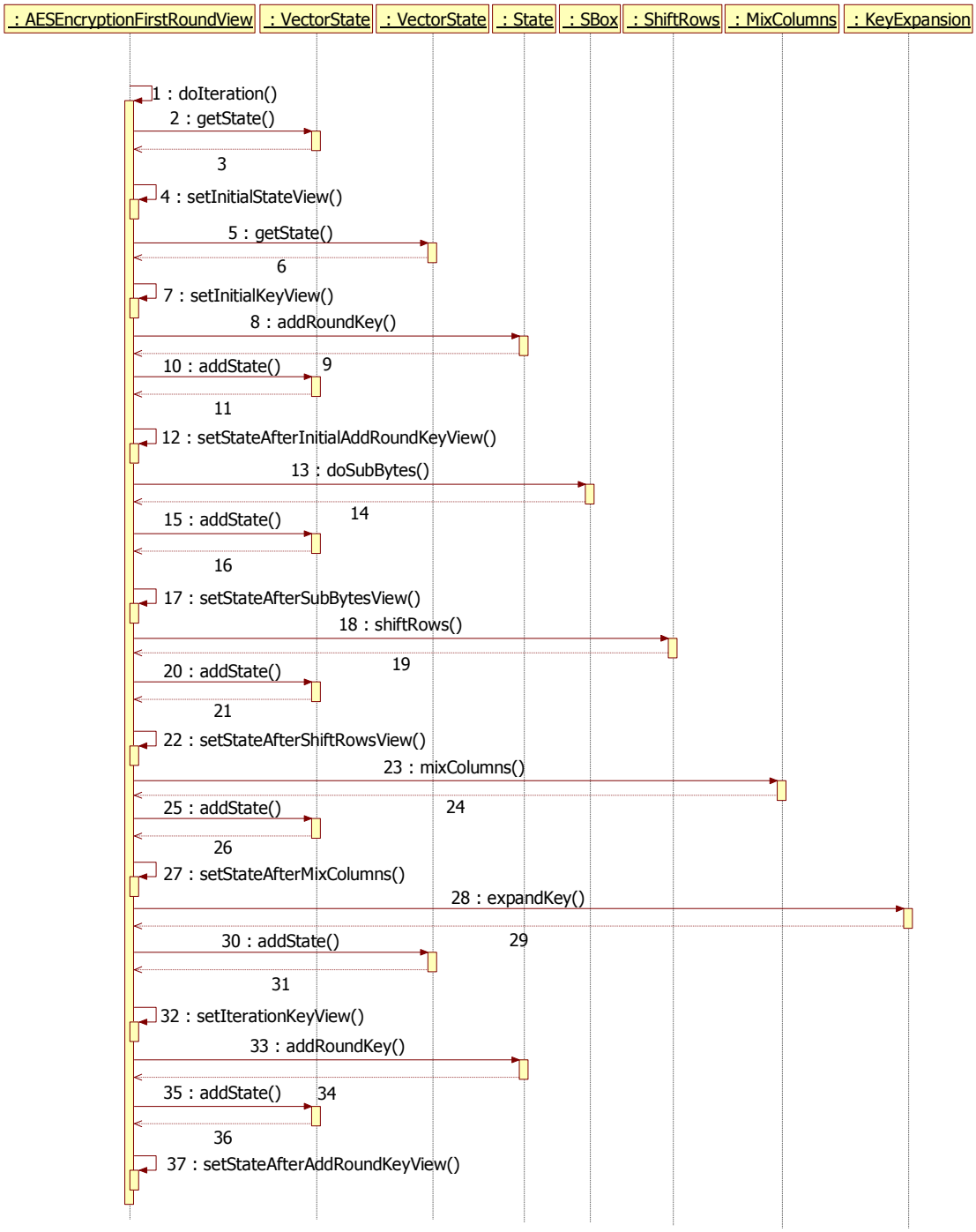
Неке од значајнијих метода класе *MixColumns* су:

- *mixColumns* – метода која извршава операцију мешања колона за објекат типа *State* који је прослеђен као улазни параметар и
- *inverseMixColumns* – метода која извршава инверзну операцију мешања колона за објекат типа *State* који је прослеђен као улазни параметар.

Метода класе *State* која је од значаја за извршавање AES алгоритма је:

- *addRoundKey* – метода која извршава операцију ексклузивно ИЛИ над текућим стањем и објектом типа *State* који је прослеђен као улазни параметар.

Спрега између класа из пакета *coala.view.aes* које служе за визуелну репрезентацију алгоритма и претходно описаних класа које служе за извршавање алгоритма приказана је на слици 6.12 где је приказан дијаграм секвенце извршавања прве итерације алгоритма. Када корисник одабере да започне шифровање AES алгоритмом први корак је уношење оригиналног блока за шифровање и кључа. Након уноса улазних параметара корисник започиње шифровање и креира се објекат типа *AESEncryptionFirstRoundView*. Приликом креирања овог објекта у конструктору се најпре обаве све радње неопходне за исцртавање самог прозора и на крају се позива метода *doIteration* (слика 6.12-1), која служи за извршавање прве итерације алгоритма и попуњавање свих међурезултата који ће бити приказани. Најпре се позивом методе *getState* објекта *vectorState* (слика 6.12-2), који је типа *VectorState*, дохвата иницијално стање, односно оригинални блок података на основу кога се попуњавају матрице на екрану које се односе на овај податак позивом методе *setInitialStateView* (слика 6.12-4). Након тога се позивом методе *getState* објекта *vectorKey* (слика 6.12-5), који је типа *VectorState*, дохвата иницијални кључ на основу кога се попуњавају матрице на екрану које се односе на овај податак позивом методе *setInitialKeyView* (слика 6.12-7). Потом се за објекат типа *State* који је дохваћен из објекта *vectorState* позива метода *addRoundKey* (слика 6.12-8) којој се као улазни параметар прослеђује објекат типа *State* који је дохваћен из објекта *vectorKey*. Метода *addRoundKey* извршава операцију иницијалног додавања кључа и њен резултат се позивом методе *addState* објекта *vectorState* (слика 6.12-10) додаје у вектор у коме се чувају међурезултати. Резултат операције иницијалног додавања кључа се користи и да се попуне матрице на екрану које се односе на овај податак позивом методе *setStateAfterInitialAddRoundKeyView* (слика 6.12-12). Након тога се за објекат типа *SBox* позива метода *doSubBytes* (слика 6.12-13) којој се као улазни параметар прослеђује објекат типа *State* који је последњи додат у објекат *vectorState*. Метода *doSubBytes* извршава операцију замене бајтова и њен резултат



Слика 6.12. Дијаграм секвенце извршавања прве итерације AES алгоритма у COALA систему

се позивом методе *addState* објекта *vectorState* (слика 6.12-15) додаје у вектор у коме се чувају међурезултати. Резултат операције замене бајтова се користи и да се попуне матрице на екрану које се односе на овај податак позивом методе *setStateAfterSubBytesView* (слика 6.12-17). Затим се за објекат типа *ShiftRows* позива метода *shiftRows* (слика 6.12-18) којој се као улазни параметар прослеђује

објекат типа *State* који је последњи додат у објекат *vectorState*. Метода *shiftRows* извршава операцију померања редова и њен резултат се позивом методе *addState* објекта *vectorState* (слика 6.12-20) додаје у вектор у коме се чувају међурезултати. Резултат операције померања редова се користи и да се попуне матрице на екрану које се односе на овај податак позивом методе *setStateAfterShiftRowsView* (слика 6.12-22). Потом се за објекат типа *MixColumns* позива метода *mixColumns* (слика 6.12-23) којој се као улазни параметар прослеђује објекат типа *State* који је последњи додат у објекат *vectorState*. Метода *mixColumns* извршава операцију мешања колона и њен резултат се позивом методе *addState* објекта *vectorState* (слика 6.12-25) додаје у вектор у коме се чувају међурезултати. Резултат операције мешања колона се користи и да се попуне матрице на екрану које се односе на овај податак позивом методе *setStateAfterMixColumnsView* (слика 6.12-27). Након тога се за објекат типа *KeyExpansion* позива метода *expandKey* (слика 6.12-28) којој се као улазни параметар прослеђује објекат типа *State* који је последњи додат у објекат *vectorKey*. Метода *expandKey* извршава операцију експанзије наредног кључа итерације и њен резултат се позивом методе *addState* објекта *vectorKey* (слика 6.12-30) додаје у вектор у коме се чувају кључеви итерација. Резултат операције експанзије наредног кључа итерације се користи и да се попуне матрице на екрану које се односе на овај податак позивом методе *setIterationKeyView* (слика 6.12-32). На крају се за објекат типа *State* који је последњи додат у објекат *vectorState* позива метода *addRoundKey* (слика 6.12-33) којој се као улазни параметар прослеђује објекат типа *State* који је последњи додат у објекат *vectorKey*. Метода *addRoundKey* извршава операцију додавања кључа итерације и њен резултат се позивом методе *addState* објекта *vectorState* (слика 6.12-35) додаје у вектор у коме се чувају међурезултати. Резултат операције додавања кључа итерације се користи и да се попуне матрице на екрану које се односе на овај податак позивом методе *setStateAfterAddRoundKeyView* (слика 6.12-37).

6.4.2. НАЧИН ИЗВРШАВАЊА АЛГОРИТАМА СА ЈАВНИМ КЉУЧЕМ

Код алгоритама са јавним кључем (*Diffie-Hellman* и RSA) било је потребно имплементирати извршавање алгоритама тако да буде омогућен приказ сваког детаља у алгоритму и промена у извршавању које се дешавају са променама улазних параметара, као што је дефинисано визуелном репрезентацијом у погла–

```
//a^b mod n
public long fastModularExponentiation(long a, long b, long n){
    long result = 1;
    BinaryValue bBin = new BinaryValue();
    bBin.setLongValue(b);
    String bBinString = bBin.getBinaryValueNoExtension();
    for(int i=0; i<bBinString.length()-1; i++){
        result = (result*result) % n;
        if (Integer.parseInt(bBinString.substring(i, i+1), 2) == 1)
            result = (result*a) % n;
    }
    result = (result*result) % n;
    if (Integer.parseInt(bBinString.substring(bBinString.length()-1), 2) == 1)
        result = (result*a) % n;

    return result;
}
```

Слика 6.13. Метода *fastModularExponentiation* класе *KeyExchange*

вљу 4.3. За разлику од симетричних алгоритама ово нису итеративни алгоритми па није било проблема који су се тамо јавили.

Код *Diffie-Hellman* алгоритма извршавање операција самог алгоритма дефинисано је у класи *KeyExchange* која се налази у пакету *coala.control.dh*. Неке од значајнијих метода ове класе су:

- *optionsForAlpha* – метода која за задату целобројну вредност *q* враћа листу вредности мањих од *q* које представљају просте корене за *q*,
- *isPrimitiveRoot* – метода која проверава да ли је прва од две улазне целобројне вредности прост корен за другу унету вредност,
- *gcd* – метода која проналази највећи заједнички делилац за два унета цела броја,
- *Fi* – метода која израчунава Ојлерову фи функцију за унету целобројну вредност,
- *fastModularExponentiation* (слика 6.13) – метода која извршава алгоритма за брзу експонентизацију за унете целобројне вредности,
- *calculateYa* – метода која израчунава јавну вредност корисника А,
- *calculateYb* – која израчунава јавну вредност корисника Б и
- *calculateKey* – метода која израчунава заједничку тајну вредност.

Код RSA алгоритма извршавање операција самог алгоритма дефинисано је у класама *Key* и *RSA* које се налазе у пакету *coala.control.rsa*. При томе класа *Key* само дефинише одговарајућу структуру података за креирање кључева који се користе код RSA алгоритма.

```
public Key[] keyGeneration(long p, long q){
    this.p = p;
    this.q = q;

    this.n = p * q;
    this.Fin = (p-1) * (q-1);

    for(e=2; e < Fin; e++)
        if (gcd(Fin, e) == 1) break;
    for(d=2; d < Fin; d++)
        if (((d*e) % Fin) == 1) break;
    PU = new Key(e, n);
    PR = new Key(d, n);
    Key[] result = new Key[2];
    result[0] = PU;
    result[1] = PR;

    return result;
}
```

Слика 6.14. Метода *keyGeneration* класе *RSA*

Неке од значајнијих метода класе *RSA* су:

- *keyGeneration* (слика 6.14) – метода која на основу унетих целобројних вредности за *p* и *q* извршава поступак генерисања кључа код RSA алгоритма и као резултат даје пар објеката типа *Key*,
- *optionsForE* – метода која за дефинисане *p* и *q* враћа све вредности које долазе у обзир да буду одабрани као *e*,
- *optionsForD* – метода која за дефинисану вредност за *e* враћа све вредности које долазе у обзир да буду одабрани као *d*,
- *recalculateKeys* – метода која омогућава замену пара кључева када се промене улазни параметри,
- *encryptRSA* – метода која за дефинисан пар кључева и унету целобројну вредност оригиналне поруке даје одговарајућу шифровану поруку и
- *decryptRSA* – метода која за дефинисан пар кључева и унету целобројну вредност шифроване поруке даје одговарајућу оригиналну поруку.

7. ПРИМЕНА COALA СИСТЕМА

У овом поглављу биће описан начин коришћења реализованог система за визуелну репрезентацију криптографских алгоритама у оквиру курса Заштита података на Електротехничком факултету у Београду.

7.1. НАЧИН ПРИМЕНЕ СИСТЕМА

Реализовани систем се користи на курсу Заштита података на Електротехничком факултету у Београду. Овај курс је обавезан на четвртој години основних студија на одсеку за софтверско инжењерство, а изборни је на четвртој години основних студија на одсеку за рачунарску технику и информатику и изборни је на мастер студијама на одсеку за рачунарску технику и информатику. Курс је у понуди на студијском програму софтверско инжењерство од 2007/2008. школске године, а на студијском програму рачунарска техника и информатика, као и на мастер студијама од 2008/2009. школске године.

На основним студијама курс има фонд од два часа предавања, два часа вежби на табли и један час лабораторијских вежби недељно, док на мастер студијама нема часове лабораторијских вежби. Из тог разлога реализовани систем за визуелну репрезентацију криптографских алгоритама је уведен у наставу само за групу студената на основним студијама, с тим што је проценат студената са одсека за рачунарску технику и информатику који бирају предмет на основим студијама занемарљив, па је тако акценат на групи студената са одсека за софтверско инжењерство. Мали број студената одсека за рачунарску технику и информатику који бирају предмет на основним студијама је лако објаснити, наиме ови студенти у семестру у коме се држи предмет бирају један од четрнаест предмета у понуди, а могу га касније одабрати и на мастер студијама.

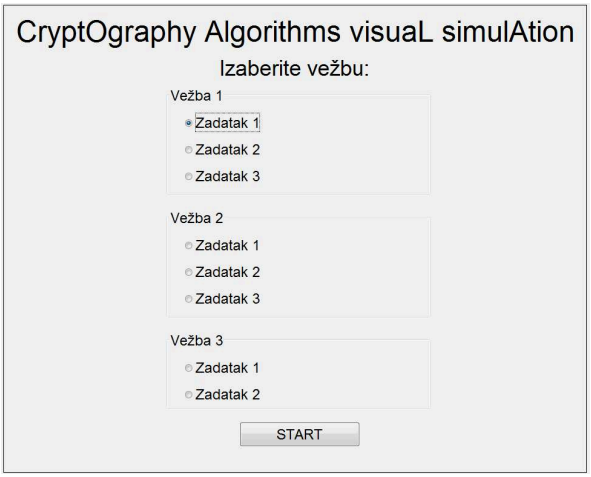
Дакле, на предмету постоје две релевантне групе студената: студенти основних студија одсека за софтверско инжењерство (ОС СИ) и студенти мастер студија одсека за рачунарску технику и информатику (МС РТИ). Систем је уведен у наставу за прву групу студената (ОС СИ) 2011/2012. школске године. Осмишљене су лабораторијске вежбе на којима је омогућено студентима да користе COALA систем како би боље савладали градиво предмета које се односи на криптографске

алгоритме.

Направљене су три лабораторијске вежбе: (1) DES алгоритам, (2) AES алгоритам и (3) RSA и Diffie-Hellman алгоритми. Свака лабораторијска вежба траје 135 минута. Студенти се припремају за лабораторијске вежбе тако што науче градиво из области лабораторијске вежбе које је предавано на часовима предавања и часовима вежби на табли. На лабораторијској вежби студенти могу да користе COALA систем и добијају сет питања (између 25 и 40 питања у зависности од вежбе) на која треба да одговоре. Одговоре дају на папиру, њихови одговори се касније прегледају и оцењују и скалирају на број поена који носи лабораторијска вежба. Све вежбе носе по 5 поена тако да лабораторијске вежбе улазе у финалну оцену на испиту као 15% оцене. Лабораторијске вежбе нису обавезне да би се положио испит, али се поени освојени на лабораторијским вежбама не могу надокнадити на други начин. Један део питања на лабораторијским вежбама захтева познавање концепата на којима се алгоритми заснивају, као и разумевање алгоритама на вишем нивоу апстракције, док је други део питања фокусиран на праћење извршавања алгоритама у COALA систему са конкретним вредностима. Питања су осмишљена тако да су студенти усмерени да испрате комплетан процес шифровања и дешифровања за сваки алгоритам, као и да обрате пажњу на специфичности сваког од алгоритама. Од 2013/2014. школске године питања су пребачена у софтверски систем *Moodle* [82], тако да студенти сада одговарају на питања на рачунару и одмах по завршетку вежбе добијају број освојених поена. У COALA систему су унапред конфигурисани сценарији извршавања алгоритма за сваки алгоритам и сваку од ситуација које се прате на лабораторијским вежбама, па је уводни екран система прилагођен потребама лабораторијских вежби (слика 7.1).

7.2. ПРВА ЛАБОРАТОРИЈСКА ВЕЖБА

Прва лабораторијска вежба односи се на праћење извршавања DES алгоритма и проверу познавања овог алгоритма од стране студената. Лабораторијска вежба се састоји из три дела: а) DES шифровање, б) DES шифровање са промењеним оригиналним блоком података за 1 бит у односу на шифровање под а) и в) DES дешифровање. Питања на првој лабораторијској вежби су приказана на слици 7.2. Питања су подељена у три групе и односе се на задатке 1, 2 и 3 из прве вежбе у



Слика 7.1. Уводни екран COALA система на лабораторијским вежбама

Покренути симулатор и одабрати вежбу 1 – *задатак 1 (DES шифровање)*. Одговорити на следећа питања:

1. (**1п**) Која је хексадецимална вредност податка који се шифрује?
2. (**1п**) Која је хексадецимална вредност кључа који се користи за шифровање?
3. (**5п**) Скицирати изглед DES алгоритма приликом шифровања?
4. (**3п**) На основу вредности из симулатора, нацртати изглед таблице која дефинише PC1 пермутацију?
5. (**1п**) Која је хексадецимална вредност након PC1 пермутације?
6. (**2п**) Која је хексадецимална вредност након Left Shift Key ротације кључа? За колико бита је извршено померање и зашто?
7. (**3п**) На основу вредности из симулатора, нацртати изглед таблице која дефинише PC2 пермутацију?
8. (**1п**) Која је хексадецимална вредност након PC2 пермутације? Шта представља ова вредност?
9. (**3п**) На основу вредности из симулатора, нацртати изглед таблице која дефинише иницијалну пермутацију?
10. (**1п**) Која је хексадецимална вредност након иницијалне пермутације?
11. (**5п**) Скицирати изглед једне итерације DES алгоритма.
12. (**3п**) На основу вредности из симулатора, нацртати изглед таблице која дефинише E пермутацију?
13. (**1п**) Која је хексадецимална вредност након E пермутације?
14. (**1п**) Која је хексадецимална вредност након XOR операције са кључем итерације?
15. (**2п**) Која је хексадецимална вредност након супституције? Објаснити како је добијена ова вредност.
16. (**3п**) На основу вредности из симулатора, нацртати изглед таблице која дефинише P пермутацију?
17. (**1п**) Која је хексадецимална вредност након P пермутације?
18. (**2п**) Која је хексадецимална вредност резултат функције прве итерације? Како је добијена ова вредност?
19. (**1п**) Која је хексадецимална вредност након прве итерације?
20. (**10п**) Направити табелу која за сваку итерацију приказује хексадецималну вредност кључа итерације и хексадецималну вредност након извршавања итерације.

Итерација	**Кључ итерације**	**Вредност након итерације**

21. (**1п**) Која је хексадецимална вредност након 32-bit swap?
22. (**3п**) На основу вредности из симулатора, нацртати изглед таблице која дефинише инверзну иницијалну пермутацију?
23. (**2п**) Која је хексадецимална вредност након инверзне иницијалне пермутације? Шта представља ова вредност?

Покренути симулатор и одабрати вежбу 1 – *задатак 2 (DES шифровање)*. Одговорити на следећа питања:

24. (**1п**) Која је хексадецимална вредност податка који се шифрује?
25. (**1п**) Која је хексадецимална вредност кључа који се користи за шифровање?
26. (**5п**) За сваку итерацију упоредити хексадецималну вредност кључа итерације и хексадецималну вредност након извршавања итерације са вредностима у табели у питању 20. Шта се може уочити?
27. (**1п**) Која је хексадецимална вредност након инверзне иницијалне пермутације? Шта представља ова вредност?
28. (**1п**) За колико бита се разликује кључ из овог примера од кључа употребљеног у претходном задатку?
29. (**1п**) За колико бита се разликује порука која се шифрује из овог примера од поруке која се шифровала у претходном задатку?
30. (**1п**) За колико бита се разликује шифрована порука из овог примера од шифроване поруке у претходном задатку?
31. (**2п**) Како се назива појава која се може уочити анализом одговора на питања 28, 29 и 30?

Покренути симулатор и одабрати вежбу 1 – *задатак 3 (DES дешифровање)*. Одговорити на следећа питања:

32. (**1п**) Која је хексадецимална вредност шифроване поруке која се дешифрује?
33. (**1п**) Која је хексадецимална вредност кључа који се користи за дешифровање?
34. (**2п**) По чему се разликује DES алгоритам приликом дешифровања од алгоритма приликом шифровања? Због чега?
35. (**5п**) За сваку итерацију упоредити хексадецималну вредност кључа итерације и хексадецималну вредност након извршавања итерације са вредностима у табели у питању 20. Шта се може уочити?
36. (**2п**) Која је хексадецимална вредност након инверзне иницијалне пермутације? Шта представља ова вредност?

Слика 7.2. Питања на првој лабораторијској вежби

COALA систему (слика 7.1), а одговарају ситуацијама а), б) и в), респективно.

Циљ првог дела лабораторијске вежбе је да студенти испрате извршавање алгоритма приликом шифровања до најситнијег детаља, пратећи притом сваку појединачну операцију. Студенти најпре прате прву итерацију шифровања и имају задатак да за сваку операцију уоче шта се добија као резултат и на који начин је тај резултат добијен од улазних вредности дате операције. Поред тога постоје и питања која проверавају концептуално познавање алгоритма од стране студената. На крају овог дела вежбе студенти праве табелу у којој попуњавају кључ итерације и вредност након итерације за све итерације алгоритма и затим бележе вредност шифрованог блока података.

Други део лабораторијске вежбе служи да омогући студентима да уоче ефекат лавине код DES алгоритма. Ефекат лавине представља пожељно својство криптографских алгоритама, а подразумева промену што више бита шифроване поруке са променом само једног бита оригиналне поруке или кључа коришћеног за шифровање. У овом делу вежбе студенти најпре треба да уоче да је кључ коришћен за шифровање идентичан као у претходном делу вежбе, а да се оригинални блок података за шифровање разликује за тачно 1 бит у односу на оригинални блок података из првог дела вежбе. Затим студенти треба да испрате цео поступак шифровања и да упореде кључеве итерације и вредности добијене након итерације за све итерације алгоритма са вредностима које су добили у првом делу лабораторијске вежбе. Закључак треба да буде да су кључеви итерације идентични у оба случаја, а да се вредности након итерације разликују све више са сваком следећом итерацијом да би на крају разлика била на око половине од укупног броја бита. Студенти треба да препознају да се овакво понашање алгоритма назива ефекат лавине.

У трећем делу лабораторијске вежбе студенти треба да испрате поступак дешифровања код DES алгоритма. Као шифровани блок података узима се вредност која је добијена као резултат шифровања у првом делу лабораторијске вежбе, а као кључ за дешифровање узима се исти кључ који је коришћен за шифровање у првом делу лабораторијске вежбе. На тај начин студенти могу да верификују да ће алгоритам коректно вратити оригинални блок података када се користи одговарајући кључ за дешифровање. Затим студенти могу да анализирају

Покренути симулатор и одабрати вежбу 2 – *задатак 1 (AES шифровање)*. Одговорити на следећа питања:
1. (**1п**) Која је хексадецимална вредност податка који се шифрује?
2. (**1п**) Која је хексадецимална вредност кључа који се користи за шифровање?
3. (**5п**) Скицирати изглед AES алгоритма приликом шифровања?
4. (**1п**) Која је хексадецимална вредност стања након почетног AddRoundKey?
5. (**2п**) Због чега се извршава почетно AddRoundKey?
6. (**2п**) Како се извршава AddRoundKey операција?
7. (**1п**) Која је хексадецимална вредност стања након SubstituteBytes у првој итерацији?
8. (**2п**) Чему служи SubstituteBytes операција?
9. (**2п**) Како се извршава SubstituteBytes операција?
10. (**1п**) Која је хексадецимална вредност стања након ShiftRows у првој итерацији?
11. (**2п**) Чему служи ShiftRows операција?
12. (**2п**) Како се извршава ShiftRows операција?
13. (**1п**) Која је хексадецимална вредност стања након MixColumns у првој итерацији?
14. (**3п**) На примеру првог бајта стања након MixColumns у првој итерацији, објаснити како се израчунава вредност сваког бајта стања након MixColumns.
15. (**2п**) Чему служи MixColumns операција?
16. (**2п**) Како се извршава MixColumns операција?
17. (**1п**) Која је хексадецимална вредност кључа прве итерације?
18. (**2п**) Како је добијена вредност речи W4 проширеног кључа?
19. (**2п**) Како су добијене вредности речи W5, W6, W7 проширеног кључа?
20. (**3п**) Како се извршава проширивање кључа?
21. (**1п**) Која је хексадецимална вредност стања након прве итерације?
22. (**10п**) Направити табелу која за сваку итерацију приказује хексадецималну вредност кључа итерације и хексадецималну вредност након извршавања итерације.

Итерација	**Кључ итерације**	**Вредност након итерације**

Покренути симулатор и одабрати вежбу 2 – *задатак 2 (AES шифровање)*. Одговорити на следећа питања:
23. (**1п**) Која је хексадецимална вредност податка који се шифрује?
24. (**1п**) Која је хексадецимална вредност кључа који се користи за шифровање?
25. (**5п**) За сваку итерацију упоредити хексадецималну вредност кључа итерације и хексадецималну вредност након извршавања итерације са вредностима у табели у питању 22. Шта се може уочити?
26. (**1п**) За колико бита се разликује кључ из овог примера од кључа употребљеног у претходном задатку?
27. (**1п**) За колико бита се разликује порука која се шифрује из овог примера од поруке која се шифровала у претходном задатку?
28. (**1п**) За колико бита се разликује шифрована порука из овог примера од шифроване поруке у претходном задатку?
29. (**2п**) Како се назива појава која се може уочити анализом одговора на питања 26, 27 и 28?

Покренути симулатор и одабрати вежбу 2 – *задатак 3 (AES дешифровање)*. Одговорити на следећа питања:
30. (**1п**) Која је хексадецимална вредност шифроване поруке која се дешифрује?
31. (**1п**) Која је хексадецимална вредност кључа који се користи за дешифровање?
32. (**2п**) По чему се разликује AES алгоритам приликом дешифровања од алгоритма приликом шифровања? Због чега?
33. (**2п**) Да ли постоји разлика између AddRoundKey код шифровања и код дешифровања? Зашто?
34. (**2п**) Која је разлика између ShiftRows и InverseShiftRows операција?
35. (**2п**) Која је разлика између SubstituteBytes и InverseSubstituteBytes операција?
36. (**2п**) Која је разлика између MixColumns и InverseMixColumns операција?
37. (**5п**) За сваку итерацију упоредити хексадецималну вредност кључа итерације и хексадецималну вредност након извршавања итерације са вредностима у табели у питању 22. Шта се може уочити?
38. (**2п**) Која је хексадецимална након AddRoundKey у десетој итерацији? Шта представља ова вредност?

Слика 7.3. Питања на другој лабораторијској вежби

на који начин се код симетричног алгоритма који користи Feistel структуру, као што је случај код DES алгоритма, исти алгоритам користи и за шифровање и за дешифровање само са примењеним кључевима итерације у инверзном редоследу.

7.3. ДРУГА ЛАБОРАТОРИЈСКА ВЕЖБА

Друга лабораторијска вежба односи се на праћење извршавања AES алгоритма и проверу познавања овог алгоритма од стране студената. Лабораторијска вежба се састоји из три дела, слично као и прва лабораторијска вежба: а) AES шифровање, б) AES шифровање са промењеним оригиналним блоком података за 1 бит у односу на шифровање под а) и в) AES дешифровање. Питања на другој

лабораторијској вежби су приказана на слици 7.3. Питања су подељена у три групе и односе се на задатке 1, 2 и 3 из друге вежбе у COALA систему (слика 7.1), а одговарају ситуацијама а), б) и в), респективно.

Циљ првог дела лабораторијске вежбе је да студенти испрате извршавање алгоритма приликом шифровања до најситнијег детаља, пратећи притом сваку појединачну операцију. Студенти најпре прате прву итерацију шифровања и имају задатак да за сваку операцију уоче шта се добија као резултат и на који начин је тај резултат добијен од улазних вредности дате операције. Поред тога постоје и питања која проверавају концептуално познавање алгоритма од стране студената. На крају овог дела вежбе студенти праве табелу у којој попуњавају кључ итерације и вредност након итерације за све итерације алгоритма и затим бележе вредност шифрованог блока података.

Други део лабораторијске вежбе служи да омогући студентима да уоче ефекат лавине код AES алгоритма. У овом делу вежбе студенти најпре треба да уоче да је кључ коришћен за шифровање идентичан као у претходном делу вежбе, а да се оригинални блок података за шифровање разликује за тачно 1 бит у односу на оригинални блок података из првог дела вежбе. Затим студенти треба да испрате цео поступак шифровања и да упореде кључеве итерације и вредности добијене након итерације за све итерације алгоритма са вредностима које су добили у првом делу лабораторијске вежбе. Закључак треба да буде да су кључеви итерације идентични у оба случаја, а да се вредности након итерације разликују све више са сваком следећом итерацијом да би на крају разлика била на око половине од укупног броја бита. Студенти треба да препознају да се овакво понашање алгоритма назива ефекат лавине.

У трећем делу лабораторијске вежбе студенти треба да испрате поступак дешифровања код AES алгоритма. Као шифровани блок података узима се вредност која је добијена као резултат шифровања у првом делу лабораторијске вежбе, а као кључ за дешифровање узима се исти кључ који је коришћен за шифровање у првом делу лабораторијске вежбе. На тај начин студенти могу да верификују да ће алгоритам коректно вратити оригинални блок података када се користи одговарајући кључ за дешифровање. Затим студенти могу да анализирају на који начин се код симетричног алгоритма који не користи Feistel структуру,

Покренути симулатор и одабрати вежбу 3 – *задатак 1 (RSA)*. Одговорити на следећа питања:
1. (**5п**) Набројати кораке за генерисање пара кључева код *RSA* алгоритма?
2. (**1п**) На који начин се бирају вредности за p и q?
3. (**2п**) Одабрати вредности 199 и 163 за p и q, респективно. Која је вредност за n и како је добијена?
4. (**2п**) Колика може бити вредност поруке која се шифрује помоћу *RSA* алгоритма у односу на n? Која је максимална вредност поруке у овом случају?
5. (**3п**) Која је вредност $\Phi(n)$? Како је добијена ова вредност? Шта представља ова вредност?
6. (**1п**) По ком правилу је одређен скуп потенцијалних вредности за e?
7. (**2п**) Одабрати вредност 21253 за e. Који је скуп могућих вредности за d? Како је добијен овај скуп вредности?
8. (**2п**) Која је вредност јавног кључа? Како је добијена ова вредност?
9. (**2п**) Која је вредност приватног кључа? Како је добијена ова вредност?
10. (**4п**) Скицирати како се добијени пар кључева може користити за остваривање тајности?
11. (**2п**) Која шифрована вредност се добија, шифровањем оригиналне вредности 10? Како је добијена ова вредност?
12. (**4п**) У симулатору је приказана употреба алгоритма експонентизације, који на ефикасан начин врши експонентизацију. Објаснити функционисање овог алгоритма на примеру шифровања оригиналне вредности 10.
13. (**2п**) Која оригинална вредност се добија, дешифровањем шифроване вредности 861? Како је добијена ова вредност?
14. (**4п**) Скицирати како се добијени пар кључева може користити за остваривање аутентикације?

Покренути симулатор и одабрати вежбу 3 – *задатак 2 (Diffie Hellman)*. Одговорити на следећа питања:
15. (**1п**) На који начин се код *Diffie Hellman* алгоритма бира q?
16. (**1п**) Одабрати вредност 997 за q. На који начин се бира α?
17. (**3п**) Шта значи да је α прост корен за q?
18. (**2п**) На који начин табела која је приказана помаже да утврдимо који су све прости корени за 997? Одабрати вредност 968 за α.
19. (**1п**) Из ког скупа вредности корисници А и В могу одабрати своје тајне вредности X_A и X_B?
20. (**2п**) Одабрати вредност 660 за X_A. Која је вредност за Y_A? Како је добијена ова вредност?
21. (**2п**) Одабрати вредност 856 за X_B. Која је вредност за Y_B? Како је добијена ова вредност?
22. (**2п**) Која је заједничка тајна вредност коју добија корисник А? Како је добијена ова вредност?
23. (**2п**) Која је заједничка тајна вредност коју добија корисник В? Како је добијена ова вредност?
24. (**4п**) Доказати да корисници А и В увек добијају исту заједничку тајну вредност, иако је рачунају на различите начине?
25. (**1п**) Шта недостаје овом алгоритму да би се избегла претња од man-in-the-middle напада?
26. (**1п**) Да ли се овај алгоритам може користити за шифровање?

Слика 7.4. Питања на трећој лабораторијској вежби

као што је случај код AES алгоритма, различити алгоритми користе за шифровање и за дешифровање при чему се кључеви итерације примењују у инверзном редоследу. Студенти би требали да уоче да је код AES алгоритма дешифровање значајно мање ефикасно него шифровање.

7.4. ТРЕЋА ЛАБОРАТОРИЈСКА ВЕЖБА

Трећа лабораторијска вежба односи се на праћење извршавања RSA и Diffie-Hellman алгоритама и проверу познавања ових алгоритама од стране студената. Лабораторијска вежба се састоји из два дела: а) RSA алгоритам и б) Diffie-Hellman алгоритам. Питања на трећој лабораторијској вежби су приказана на слици 7.4. Питања су подељена у две групе и односе се на задатке 1 и 2 из треће вежбе у COALA систему (слика 7.1), а одговарају ситуацијама а) и б), респективно.

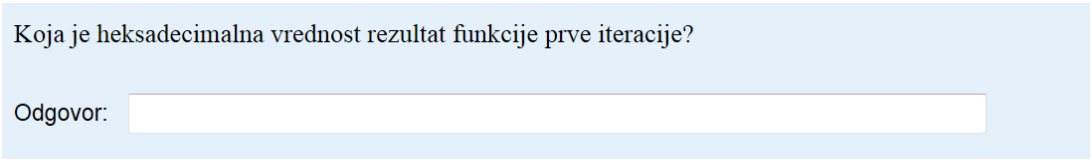
Циљ првог дела лабораторијске вежбе је да студенти испрате целокупан поступак генерисања пара кључева, шифровања и дешифровања код RSA алгоритма. Студенти најпре треба да покажу да су теоријски научили алгоритам кроз концептуална питања о алгоритму. Након тога студенти пролазе најпре кроз поступак генерисања пара кључева са конкретним вредностима. Иако вредности

које се користе нису реда величина које се користе у реалним применама (по неколико стотина децималних цифара имају бројеви у реалним применама алгоритма), ипак су одабране вредности довољне да демонстрирају све битне детаље алгоритма. У поступку генерисања кључева студенти имају прилику да виде у свакој фази које су то конкретне вредности које испуњавају дефинисане услове према алгоритму. Приликом поступка шифровања и дешифровања где се јављају изрази са експонентима по модулу са коришћеним великим бројевима демонстриран је алгоритам за брзу експонентизацију који је објашњен студентима на часовима. Студенти имају могућност да испрате детаље извршавања овог алгоритма корак по корак и да уоче на који начин вредност експонента утиче на ефикасност израчунавања.

Други део лабораторијске вежбе служи да студенти испрате поступак извршавања Diffie-Hellman алгоритма за сигурну размену заједничке тајне вредности. И овде постоји одређени број питања који проверавају да ли су студенти научили концепте алгоритма. Након тога студенти могу да прођу кроз читав поступак извршавања алгоритма са конкретним примерима (као и код RSA и овде вредности нису реалних димензија из истих разлога). Код Diffie-Hellman алгоритма студенти најпре виде на који начин се бирају глобални јавни елементи. Визуелно је приказано како се могу одредити прости корени неког одабраног простог броја. Затим студенти генеришу тајне вредности за сваког од два учесника у комуникацији, уочавају на који начин су на основу тајних вредности генерисане јавне вредности и како сваки корисник сам за себе израчунава заједничку тајну вредност. С обзиром да се и овде користе изрази са експонентима по модулу и овде је приказан рад алгоритма за брзу експонентизацију.

7.5. НАЧИН ИСПИТИВАЊА НА ЛАБОРАТОРИЈСКИМ ВЕЖБАМА

Прве две године коришћења система на лабораторијским вежбама студенти су имали на располагању COALA систем на рачунару у лабораторији и питања у pdf



Слика 7.5. Пример питања где студент треба да упише одговор у електронском систему за испитивање

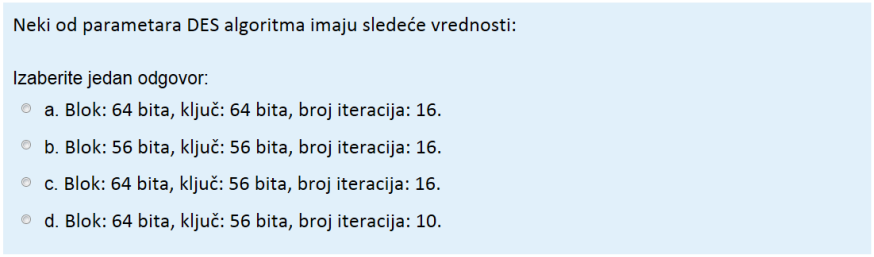
формату. У лабораторији је био забрањен приступ Интернету и онемогућено коришћење било каквих материјала (ни електронских ни штампаних). Студенти нису унапред добијали питања за лабораторијску вежбу. Студенти су на питања одговарали на папиру и након одржавања вежбе њихови радови су оцењивани и резултати скалирани на број поена који је одређен за сваку вежбу.

У трећој години коришћења система на лабораторијским вежбама све је остало исто што се тиче извођења лабораторијских вежби, осим што су питања пребачена у електронски систем за испитивање *Moodle*. Структура и циљеви лабораторијских вежби, као и појединих делова сваке од лабораторијских вежби су остали исти. Питања су прилагођена могућностима коришћеног електронског система за испитивање, али се није изгубила суштина ових питања, мада је за нека питања морала да буде промењена форма. Увођењем електронског система за испитивање смањено је време које су студенти трошили за одговарање на питања, чиме је повећано време које студенти могу да посвете пажљивом праћењу извршавања алгоритама. Коришћењем електронског система за испитивања студенти тренутно добијају резултате лабораторијске вежбе чим предају своје одговоре.

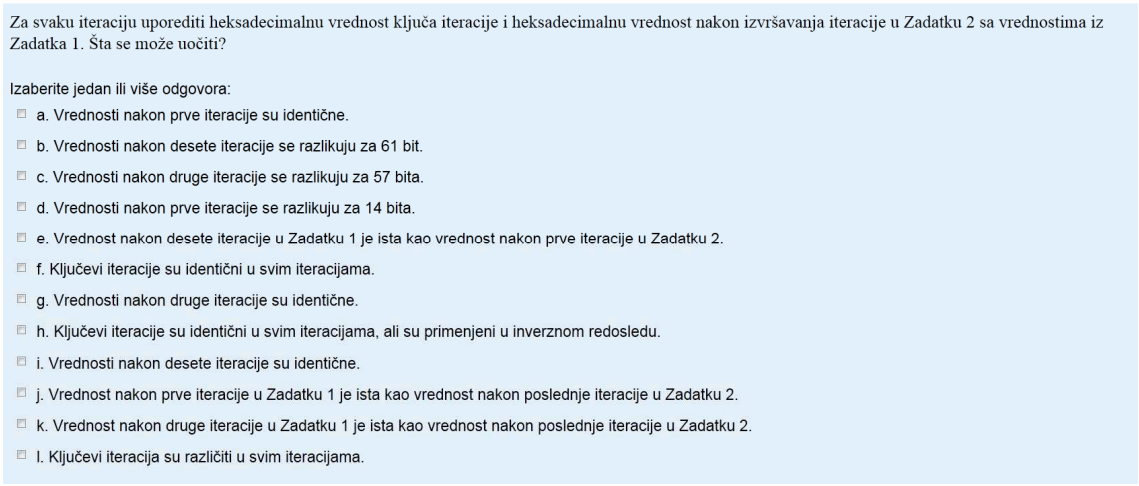
Врсте питања које су коришћене у електронском систему за испитивање су следеће: питања где студент треба да упише одговор, питања са више понуђених одговора, питања са упаривањем и питања са одговором тачно или нетачно.

Питања где студент треба да упише одговор (слика 7.5) су коришћена у ситуацијама где студенти приликом праћења рада алгоритма треба да очитају неку вредност из COALA система, као и у концептуалним питањима где се одговор може формулисати у виду броја.

Код питања са више понуђених одговора коришћене су две варијанте: где се бира само један одговор (слика 7.6а) и где се бира један или више понуђених одговора (слика 7.6б). Ова група питања коришћена је најчешће за ситуације када

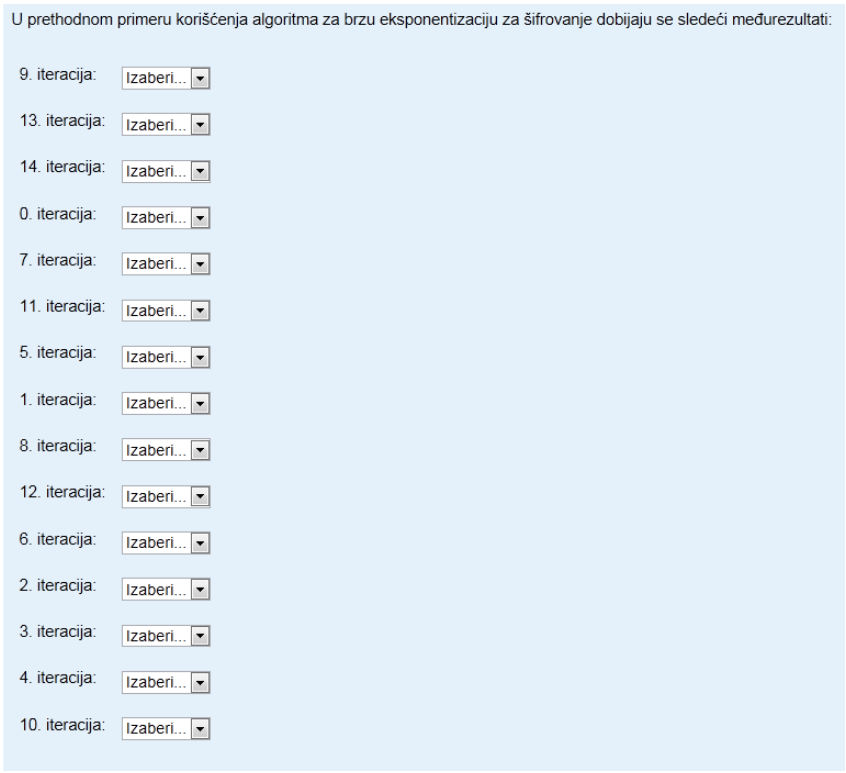


а)

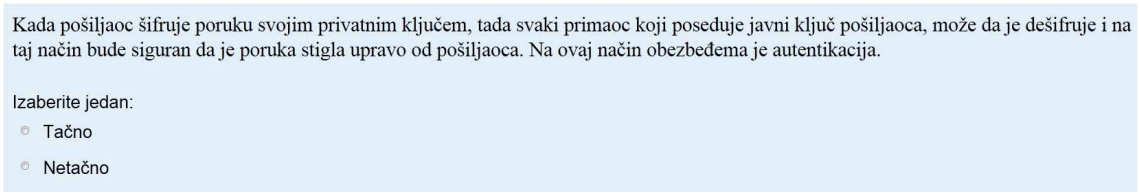


б)

Слика 7.6. Примери питања са више понуђених одговора у електронском систему за испитивање



Слика 7.7. Пример питања са упаривањем у електронском систему за испитивање



Слика 7.8. Пример питања са одговором тачно или нетачно у електронском систему за испитивање

је потребно објаснити како функционише нека операција у алгоритму, како је добијена нека вредност и за проверу познавања концепата алгоритама.

Питања са упаривањем (слика 7.7) су погодна као замена за питања код којих је на папиру требало правити табелу уз помоћ које се пратио ток извршавања неког од итеративних алгоритама кроз итерације.

Питања са одговором тачно или нетачно (слика 7.8) коришћена су да замене питања у којима се на папиру тражило од студената да донесу неки закључак. Било би боље да су оваква питања могла да буду направљена тако да студенти сами испишу закључке, али би то нарушило форму одговарања на питања коришћењем електронског система за испитивање, јер би тада та питања морала да буду прегледана од стране наставника. Са друге стране коришћењем овог типа питања студенти морају да препознају да ли тврдња коју прочитају одговара њиховом закључку.

8. ЕВАЛУАЦИЈА COALA СИСТЕМА

У овом поглављу биће описана евалуација COALA система са аспекта ефикасности приликом примене на курсу Заштита података на Електротехничком факултету у Београду. Систем се користи у настави у претходне три школске године (уведен је у наставу 2011/2012. школске године). Анализа резултата коришћења система подељена је на три дела. Први део обухвата анализу ефеката увођења система у наставу прве школске године коришћења система. Други део анализира напредак у коришћењу између прве и друге школске године у којима је систем био коришћен. Трећи део анализе представља анализу утиска студената о коришћењу система кроз упитник који су попуњавали.

8.1. ЕФЕКТИ УВОЂЕЊА COALA СИСТЕМА У НАСТАВУ

На Електротехничком факултету у Београду на већини предмета, па тако и на предмету Заштита података, студенти се оцењују независно у односу на друге на основу процента градива које су савладали. Оцене се формирају на следећи начин: мање од 51% оцена 5 (студент није положио испит), од 51% до 60% оцена 6, од 61% до 70% оцена 7, од 71% до 80% оцена 8, од 81% до 90% оцена 9 и од 91% до 100% оцена 10. Када је COALA систем уведен у наставу на предмету Заштита података, одлучено је да лабораторијске вежбе на којима се користи носе одређени проценат поена који улази у формирање финалне оцене на испиту. Свака лабораторијска вежба носи по 5 поена, што у укупном збиру даје 15% оцене на испиту. Ова чињеница мотивише студенте да иако лабораторијске вежбе нису обавезне ипак одлуче да их похађају. Осим што могу да зараде одређени број поена на лабораторијским вежбама, ове вежбе им такође помажу да се боље припреме за колоквијуме и испит.

Чињеница да постоје две групе студената које имају могућност да слушају предмет Заштита података (ОС СИ и МС РТИ) омогућила је да се ефикасност увођења новог система у наставу измери и упореди са ситуацијом када систем није уведен у наставу. Иако су студенти из МС РТИ групе годину дана старији од

Табела 8.1. Број полагања на испиту у току школске године и број полагања у првом испитном року у школској години

	2007/2008.	2008/2009.	2009/2010.	2010/2011.	2011/2012.
ОС СИ	23 (23)	26 (20)	38 (32)	28 (13)	45 (24)
МС РТИ	-	29 (17)	20 (13)	28 (14)	28 (9)

студената из ОС СИ групе, ова чињеница не утиче на резултате мерења, с обзиром да су обе групе студената на истом нивоу претходног знања из области Заштите података (без претходног знања) и самим тим уче исто градиво. COALA систем је уведен у наставу за ОС СИ групу студената 2011/2012. школске године, док за групу студената МС РТИ није уведен у наставу. Како би се измерила ефикасност увођења новог система у наставу мерено је и процедурално и концептуално знање, како је то описано у секцији 2. Процедурално знање је мерено на лабораторијским вежбама, док је концептуално знање проверавано на колоквијумима, односно испиту. За мерење стеченог знања коришћена је метода тест након коришћења (енгл. post-test method).

Табела 8.1 приказује број студената који су полагали испит из Заштите података за обе групе студената и за школске године од када се предмет држи на тој групи студената до школске године у којој је систем уведен на предмет за групу ОС СИ. Број у загради означава колико студената је полагало предмет у првом испитном року у школској години.

Неколико нумеричких индикатора је одабрано како би се квантификовао ефекат увођења COALA система у наставу. Ти индикатори су: проценат успешних полагања испита у току читаве школске године, просечна оцена постигнута на испиту у току читаве школске године, проценат успешних полагања испита у првом испитном року у школској години, просечна оцена постигнута на испиту у првом испитном року у школској години.

Проценат успешних полагања испита у току читаве школске године (табела 8.2) приказује константан пад за обе групе студената све до 2011/2012. школске године када је COALA систем уведен у наставу за студенте ОС СИ групе. Тада за групу ОС СИ овај параметар бележи повећање, док се за групу МС РТИ пад на–

Табела 8.2. Проценат успешних полагања испита у току читаве школске године

	2007/2008.	2008/2009.	2009/2010.	2010/2011.	2011/2012.
ОС СИ	91.7	85.19	78.72	62.5	76.09
МС РТИ	-	88.9	76.92	77.42	63.3

Табела 8.3. Просечна оцена постигнута на испиту у току читаве школске године

	2007/2008.	2008/2009.	2009/2010.	2010/2011.	2011/2012.
ОС СИ	8.58	7.67	7.23	6.59	7.24
МС РТИ	-	8.67	7.54	7.16	7

Табела 8.4. Проценат успешних полагања испита у првом испитном року у школској години

	2007/2008.	2008/2009.	2009/2010.	2010/2011.	2011/2012.
ОС СИ	91.3	100	81.25	100	91.7
МС РТИ	-	94.13	69.23	85.71	66.7

ставља и даље.

Просечна оцена постигнута на испиту у току читаве школске године (табела 8.3) приказује константан пад за обе групе студената све до 2011/2012. школске године када је COALA систем уведен у наставу за студенте ОС СИ групе. Тада за групу ОС СИ овај параметар почиње да се опоравља, док се за групу МС РТИ пад наставља и даље.

Проценат успешних полагања испита у првом испитном року у школској години (табела 8.4) приказује константно високу вредност за ОС СИ групу студената и овај тренд се наставља и у 2011/2012. школској години када је COALA систем уведен у наставу за ову групу студената (91.7/100). Овај параметар за МС РТИ групу студената има непредвидив тренд током година, који је настављен и у 2011/2012. школској години.

Просечна оцена постигнута на испиту у првом испитном року у школској години (табела 8.5) приказује константно високу вредност за обе групе студената

Табела 8.5. Просечна оцена постигнута на испиту у првом испитном року у школској години

	2007/2008.	2008/2009.	2009/2010.	2010/2011.	2011/2012.
ОС СИ	8.57	7.95	7.31	7.85	7.75
МС РТИ	-	9.06	7.31	7.57	7.78

у свим школским годинама.

Анализа показује да су за студенте ОС СИ групе који су имали прилику да користе COALA систем прва два параметра показала раст, док су код друге групе студената МС РТИ који нису користили систем ови параметри показали наставак негативног тренда. Да би се установили разлози за претходно настали негативан тренд ових параметара морала би се направити посебна анализа многих различитих фактора који нису у домену ове дисертације. Важно је напоменути да током анализираног временског периода није било других промена на курсу, осим увођења COALA система за ОС СИ групу студената (исти наставници, исто градиво и исти начин испитивања сваке године). Овакав резултат приликом увођења система у наставу је бољи него што је очекивано, јер се очекивало да ће увођење система наићи на отпор студената с обзиром да су нови систем и лабораторијске вежбе захтевале додатан напор са њихове стране. Очекује се да ће овај резултат бити све бољи из године у годину.

Проценат успешних полагања испита у првом испитном року у школској години и просечна оцена постигнута на испиту у првом испитном року у школској години нам показује да студенти који су полагали испит у првом испитном року у школској години имају мање проблема од осталих студената да положе испит успешно и са високом оценом. Такође, анализа ових параметара показује да увођење новог система није значајно утицало на промену ових параметара за ОС СИ групу студената, као што ни не увођење система није утицало значајно на промену ових параметара за МС РТИ групу студената. Студенти који испит полажу у првом испитном року у школској години су типично студенти који редовно посећују часове предавања и вежби на табли током семестра, немају проблема да на време заврше све своје предиспитне обавезе и иначе полажу испите са високим оценама. Овај део евалуације COALA система показује да је

ова група свакако имала користи од увођења новог система у наставу, али да коришћење новог система није значајно утицало на њихов и иначе натпросечан успех на испитима (ОС СИ), као што ни не увођење није утицало за МС РТИ групу студената. Резултати овог дела анализе показују да су највећу корист од увођења новог система у наставу имали студенти који иначе одлажу тренутак у коме полажу испит за касније испитне рокове у току школске године како би имали више времена да се припреме за испит. То су типично студенти који не присуствују часовима предавања и вежби и често прескачу даураде предиспитне обавезе уколико нису обавезне. Ови студенти имају тенденцију да градиво уче самостално. Пре увођења COALA система у наставу ови студенти су имали великих потешкоћа да науче градиво из области криптографских алгоритама, што представља једну од кључних области на предмету Заштита података. Из тог разлога они нису успевали да положе испит или су успевали да га положе, али са ниским оценама. Из анализе се види да је увођење COALA система на предмет помогло овој групи студената да разумеју градиво и буду успешнији на испиту.

8.2. ПОРЕЂЕЊЕ ПРВЕ И ДРУГЕ ГОДИНЕ КОРИШЋЕЊА СИСТЕМА

Као што смо видели у табели 8.1 број студената у групама ОС СИ и МС РТИ је био прилично уједначен у школским годинама које су анализиране у секцији 8.1. У 2012/2013. школској години број студената у групи ОС СИ је порастао на 57, док се број студената у групи МС РТИ смањио на свега 13 студената. Због тога група студената МС РТИ није била адекватна за даљу анализу и уместо тога акценат је стављен на поређење резултата примене COALA система код ОС СИ групе студената у две сукцесивне школске године.

Табела 8.6 приказује како изгледају параметри дефинисани у претходној секцији у 2011/2012. и 2012/2013. школској години за ОС СИ групу студената. Колоне редом садрже вредности за параметре: (1) број студената који су полагали испит из Заштите података (број у загради означава колико студената је полагало предмет у првом испитном року у школској години), (2) проценат успешних полагања испита у току читаве школске године, (3) просечна оцена постигнута на испиту у току читаве школске године, (4) проценат успешних полагања испита у првом испитном року у школској години и (5) просечна оцена постигнута на

Табела 8.6. Параметри дефинисани у секцији 6.1 у 2011/2012. и 2012/2013. школској години за ОС СИ групу студената

ОС СИ	1	2	3	4	5
2011/2012.	45 (24)	76.09	7.24	91.7	7.75
2012/2013.	57 (38)	81.82	7.27	86.84	7.63

Табела 8.7. Проценат резултата ОС СИ студената изнад 90% на испиту током једне школске године

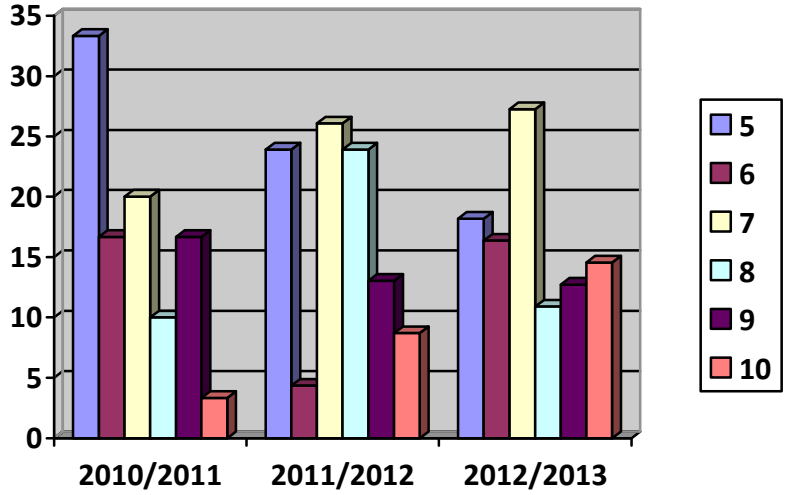
	2007/2008.	2008/2009.	2009/2010.	2010/2011.	2011/2012.	2012/2013.
ОС СИ	42	23.33	8.7	3.33	8.7	14.55

испиту у првом испитном року у школској години.

Из табеле 8.6 се види да је проценат успешних полагања испита у току читаве школске године (2) наставио да расте у односу на претходну школску годину, просечна оцена постигнута на испиту у току читаве школске године (3) бележи незнатан пораст у односу на претходну школску годину, док су проценат успешних полагања испита у првом испитном року у школској години (4) и просечна оцена постигнута на испиту у првом испитном року у школској години (5) и у 2012/2013. школској години конзистентни са претходно установљеним трендом. Проценат успешних полагања испита у току читаве школске године је у 2012/2013. школској години достигао највећу вредност у претходне четири школске године. Просечна оцена постигнута на испиту у току читаве школске године је у 2012/2013. школској години такође забележила највећу вредност у претходне четири школске године, али је повећање у односу на претходну школску годину занемарљиво. Из тог разлога је утицај увођења система у наставу истражен мало детаљније.

Дистрибуција оцена студената на испиту остварених током читаве школске године за 2010/2011, 2011/2012. и 2012/2013. школске године за групу студената ОС СИ је приказана на слици 8.1.

Проценат студената који су остварили више од 90% поена на испиту током читаве школске године за групу студената ОС СИ за 2007/2008, 2008/2009, 2009/2010, 2010/2011, 2011/2012. и 2012/2013. приказан је у табели 8.7.



Слика 8.1. Дистрибуција оцена студената ОС СИ на испиту током једне школске године

Са слике 8.1 види се да проценат студената који не успевају да положе испит (оцена 5) константно опада од 2011/2012. школске године када је COALA систем уведен у наставу. Са друге стране, проценат студената који су оцењени најбољим оценама (оцене 9 и 10) је у порасту од 2011/2012. школске године када је COALA систем уведен у наставу. Табела 8.7 показује да је проценат студената који успевају да освоје максималну оцену на испиту (изнад 90% освојених поена на испиту) у значајном порасту из године у годину почевши од 2011/2012. школске године када је COALA систем уведен у наставу.

Иако је закључак првог дела анализе био да је увођење COALA система у наставу највише користи донело студентима који су претходно имали проблема да положе испит, овај закључак би на основу другог дела анализа требало проширити са опсервацијом да је увођење система у наставу донело користи и најбољим студентима да побољшају своје оцене.

У другој години коришћења система на лабораторијским вежбама могуће је упоредити и резултате студената постигнуте на самим лабораторијским вежбама у две сукцесивне школске године. Табела 8.8 приказује просечан проценат освојених поена на лабораторијским вежбама у једној школској години (1), као и проценат студената који су постигли резултат изнад 90% освојених поена на лабораторијским вежбама (2).

Из табеле 8.8 се види да су студенти постигли боље резултате у другој

Табела 8.8. Просечан проценат освојених поена на лабораторијским вежбама у једној школској години и проценат студената који су постигли резултат изнад 90% освојених поена на лабораторијским вежбама

ОС СИ	1	2
2011/2012.	62.2	24.57
2012/2013.	75	53.63

Табела 8.9. Резултати анкете спроведене 2012/2013. школске године

	1	2	3	4	5
13	1	8	12	19	13
14	5	6	16	16	10
15	0	7	11	14	21
16	2	8	12	12	19
17	2	4	10	12	25
20	1	6	13	21	12

школској години у којој је COALA систем коришћен. Слабији резултати 2011/2012. школске године се могу приписати проблемима који се уобичајено јављају приликом транзиционе фазе код увођења новог система у наставу и неопходно периода навикавања за студенте и наставнике.

8.3. УПИТНИК

Поред емпиријске анализе ефеката COALA система на резултате студената на испиту спроведене на основу мерљивих резултата школске 2012/2013. спроведена је и анкета међу студентима који су користили систем како би се установило колико студенти сматрају да им је коришћење система користило. Анкета је била анонимна и у анкети је учествовало 54 студента који су користили систем на лабораторијским вежбама. Анкета (слика 8.2) се састојала из две групе питања: питања са понуђеним одговорима (базирана на Likert-scale приступу) и питања за

DES					
1. Колико Вам је систем помогао да схватите алгоритам?	1	2	3	4	5
2. Колико Вам је времена требало да се припремите за лаб. вежбу?					
3. Колико Вам је коришћење система значило приликом припреме за колоквијум/испит?	1	2	3	4	5
4. Да ли постоји нешто што би побољшало употребу система за ову вежбу?					
AES					
5. Колико Вам је систем помогао да схватите алгоритам?	1	2	3	4	5
6. Колико Вам је времена требало да се припремите за лаб. вежбу?					
7. Колико Вам је коришћење система значило приликом припреме за колоквијум/испит?	1	2	3	4	5
8. Да ли постоји нешто што би побољшало употребу система за ову вежбу?					
RSA, Diffie-Hellman					
9. Колико Вам је систем помогао да схватите алгоритме?	1	2	3	4	5
10. Колико Вам је времена требало да се припремите за лаб. вежбу?					
11. Колико Вам је коришћење система значило приликом припреме за колоквијум/испит?	1	2	3	4	5
12. Да ли постоји нешто што би побољшало употребу система за ову вежбу?					
Општа оцена					
13. Колико Вам је систем помогао да схватите градиво?	1	2	3	4	5
14. Колико Вам је систем помогао да се припремите за колоквијуме/испит?	1	2	3	4	5
15. Колико сте задовољни корисничким интерфејсом система?	1	2	3	4	5
16. Колико питања на вежбама омогућавају да се испрате детаљи алгоритама?	1	2	3	4	5
17. Колико начин визуелне репрезентације алгоритама омогућава да се испрате детаљи алгоритама?	1	2	3	4	5
18. Да ли би и шта требало променити у самом систему?					
19. Да ли би и шта требало променити у питањима на лаб. вежбама?					
20. Оцените укупно искуство коришћења система.	1	2	3	4	5

Слика 8.2. Изглед анкетног листића који су попуњавали студенти

слободне одговоре.

Вредности скале која је коришћена за питања са понуђеним одговорима су: 1 – веома незадовољан, 2 – незадовољан, 3 – ни задовољан ни незадовољан, 4 – задовољан, 5 – веома задовољан.

Табела 8.9 приказује распоред оцена које су студенти дали у анкети на питања 13, 14, 15, 16, 17 и 20 која се односе на општу оцену коришћења система.

Просечне вредности које се добијају по анализираним питањима су 3.66, 3.38, 3.92, 3.72, 4.02, 3.70, респективно. Из резултата се види да су студенти дали веома високе оцене на питања 15 и 17 што значи да су били задовољни начином дизајна

COALA система. На питање 14 студенти су дали најлошије оцене, а највероватнији разлог је тај што су алгоритми који се изучавају на лабораторијским вежбама само део градива које се полаже на колоквијумима, односно испиту, па тако систем не може да помогне студентима да се у потпуности припреме за колоквијум или испит. Резултати одговора студената на питања 13 и 16 показују да је COALA систем успешно испунио основни циљ, а то је да студенти боље разумеју градиво и детаље алгоритама који су покривени овим системом. Питање 20 говори о генералном утиску студената који су користили систем и овакав резултат одговора на питање говори да је већина студената била задовољна.

Питања на која су могли да дају слободне одговоре студенти су углавном искористили да изнесу замерке на рачун система или организације лабораторијских вежби уколико их је било. Неки од студената су изнели замерку да је фонт који је коришћен у COALA систему сувише мали и да то представља проблем када се користи систем. Неколико студената је сугерисало да би им од користи било када би у COALA систему постојало више објашњења алгоритама, јер би тада лакше могли да испрате извршавање алгоритама. Велики број студената се жалио да су нека од питања на лабораторијским вежбама превише напорна јер захтевају да се доста текста испише у оквиру одговора на питање. Ови студенти су предложили да се број таквих питања смањи.

9. ЗАКЉУЧАК

У овом раду описан је нови систем за визуелну репрезентацију криптографских алгоритама (COALA). Главни циљ развијеног система је да се помогне студентима који су нови у области заштите података да лакше и боље савладају градиво из области криптографских алгоритама које представља основу на коју се надовезују сложеније теме. На Електротехничком факултету у Београду постоји курс под називом Заштита података на коме се најпре изучавају криптографски алгоритми на којима се заснивају сигурносни протоколи и апликације који служе да обезбеде сигурносне сервисе, а након тога се изучавају апликације које обезбеђују мрежну сигурност, док се на крају улази у проблематику сигурности рачунарских система (хакери, вируси, итд.). Праћењем резултата студената на испиту примећено је да студенти имају проблема да савладају градиво из области криптографских алгоритама, што се касније негативно одражава на њихов свеукупан успех на испиту. Анализом разлога који су довели до оваквог стања дошло се до закључка да би увођење система за визуелну репрезентацију криптографских алгоритама у оквиру лабораторијских вежби на предмету могло бити једно од решења проблема. Коришћење оваквих система је уобичајено на предметима из области рачунарске технике и информатике који изучавају софтвер, односно прецизније алгоритме. Пре развоја новог система, направљена је анализа постојећих решења како би се утврдило да ли већ постоји систем који би испунио потребе предмета и како би се установило које су то карактеристике које би тај систем требао да има да би се успешно користио у настави. Установљено је да иако постоји велики број система за визуелну репрезентацију алгоритама, релативно мали број њих омогућава визуелну репрезентацију криптографских алгоритама, док међу анализираним системима не постоји систем за визуелну репрезентацију криптографских алгоритама који у потпуности задовољава потребе предмета Заштита података на Електротехничком факултету у Београду. На основу систематизације постојећих система за визуелну репрезентацију алгоритама са посебним освртом на системе који се могу користити за визуелну репрезентацију криптографских алгоритама дефинисана је методологија развоја

оваквих система. Анализом литературе која се бави системима који се користе за помоћ у учењу дефинисан је начин коришћења развијеног система у настави. Реализовани систем омогућава праћење извршавања свих криптографских алгоритама који су објашњени на часовима предавања и вежби, приказује све детаље извршавања за сваки од подржаних алгоритама, за поједине алгоритме где је то од важности систем ради са реалним величинама параметара алгоритма које се користе у пракси и сви параметри алгоритама су конфигурабилни како би могли лако и брзо да се прикажу различити примери извршавања алгоритама.

У раду је направљен преглед области система за помоћ у учењу, преглед области визуелне репрезентације алгоритама, као и преглед области визуелне репрезентације криптографских алгоритама. Дат је опис криптографских алгоритама који су подржани у COALA систему. Систем има могућност визуелне репрезентације пет врста криптографских алгоритама: супституционих алгоритама (Цезар алгоритам, моноалфабетски алгоритам, *Playfair* алгоритам и *Vigenere* алгоритам), транспозиционих алгоритама (*Rail Fence* алгоритам и *Row Transposition* алгоритам), продукционих алгоритама (*Rotor Machine* алгоритам), симетричних блок алгоритама (DES алгоритам и AES алгоритам) и алгоритама са јавним кључем (*Diffie-Hellman* алгоритам и *RSA* алгоритам). Посебан акценат је стављен на објашњење детаља везаних за визуелну репрезентацију алгоритама коришћену у COALA систему. Приказан је и детаљан функционални опис COALA система за визуелну репрезентацију криптографских алгоритама. У раду су објашњени и детаљи везани за софтверску имплементацију COALA система. Дат је преглед технологија коришћених за софтверску имплементацију, објашњена је структура софтверског решења и дати су детаљи направљених проширења *Java* и *Swing* API-ја, као и начин извршавања алгоритама подржаних у COALA систему. Начин коришћења реализованог система за визуелну репрезентацију криптографских алгоритама у оквиру предмета Заштита података на Електротехничком факултету у Београду описан је кроз преглед осмишљених лабораторијских вежби које користе COALA систем. У школској години 2013/2014. уведена је још једна новина на предмету, а то је коришћење система *Moodle* за испитивање и оцењивање студената у оквиру лабораторијских вежби, па је и то описано. У раду је приказана и евалуација COALA система са аспекта

ефикасности приликом примене на предмету Заштита података на Електротехничком факултету у Београду. Систем се користи у настави у претходне три школске године (уведен је у наставу 2011/2012. школске године). Резултати примене су мерени за прве две године коришћења система, с обзиром да је трећа школска година у којој се систем користи још увек у току. Евалуација је подељена на три дела. Најпре су мерени ефекти увођења COALA система у наставу на основу успешности студената на предмету у првој школској години, затим су упоређени резултати коришћења из прве и друге године коришћења, а трећи део анализе обухватио је утисак студената на основу упитника који су попуњавали. Резултати евалуације су показали да су највећу корист од увођења новог система у наставу имали студенти који иначе одлажу тренутак у коме полажу испит за касније испитне рокове у току школске године како би имали више времена да се припреме за испит и који не успевају да положе испит или га положе али са ниском оценом, као и најбољи студенти којима је коришћење система помогло да побољшају своје оцене. Анализа анкете коју су студенти попуњавали је показала да је COALA систем успешно испунио основни циљ, а то је да студенти боље разумеју градиво и детаље алгоритама који су покривени овим системом.

У оквиру дисертације постигнути су следећи научни доприноси:

- генерални преглед области алата за помоћ у учењу (енгл. *e-learning tools*),
- генерални преглед области визуелне репрезентације алгоритама (енгл. *algorithm visualization*) уз осврт на постојећу систематизацију и класификацију ове области и са посебним акцентом на применљивости и ефикасности у образовном процесу,
- систематизација и класификација постојећих решења у области визуелне репрезентације криптографских алгоритама,
- генерализација функционалности постојећих система за визуелну репрезентацију криптографских алгоритама,
- формирање нове методе за визуелну репрезентацију криптографских алгоритама на основу класификације и анализе постојећих решења из ове области,
- предлог и имплементација оригиналног софтверског система за визуелну

репрезентацију криптографских алгоритама који примењује осмишљену методу,

- предлог начина примене имплементираног софтверског система у образовном процесу на основу анализе применљивости и ефикасности оваквих система,
- евалуација ефикасности имплементираног софтверског система приликом примене у образовном процесу.

Евалуација COALA система ће бити настављена и у наредним годинама и очекују се још бољи резултати. Увођењем система *Moodle* за испитивање и оцењивање студената у оквиру лабораторијских вежби у текућој школској години време израде појединачних лабораторијских вежби је скраћено, па се сада јавља простор да се у оквиру расположивог фонда часова предвиђеног за лабораторијске вежбе на предмету Заштита података уведу и нове лабораторијске вежбе. Приликом истраживања за овај рад примећен је велики број система за помоћ у учењу који су намењени за област заштите података, али који се не односе на криптографске алгоритме. У будућности би било потребно направити преглед постојећих система за помоћ у учењу из области заштите података. Анализом таквог прегледа и тема из области за које се на предмету Заштита података установи да су критичне требало би донети одлуку о увођењу додатног система за помоћ у учењу на лабораторијске вежбе. То би могао бити неки од постојећих система или ако се установи да не постоји систем који би покрио критичне теме на предмету у том случају би требало осмислити и реализовати нови систем.

Овај рад отвара неколико нових праваца истраживања која би у будућности могла бити спроведена. Један правац даљег истраживања би могао да буде развој и имплементација алата за генерисање конфигурабилних система за визуелну репрезентацију симетричних криптографских блок алгоритама заснованих на *Feistel* структури алгоритма. Принципи дизајна симетричних криптографских блок алгоритама се доста добро могу објаснити коришћењем *Feistel* структуре алгоритма, иако актуелни алгоритам који се користи као стандард (AES) не користи ову структуру. Пројектовање оваквог алата би захтевало решавање два проблема. Први проблем би био обезбедити могућност да се конфигуришу сви параметри алгоритма заснованог на *Feistel* структури: величина блока, величина

кључа, број итерација, функција итерације и функција за генерисање кључева итерације. Други проблем би био обезбедити механизам за динамичко пројектовање визуелне репрезентације алгоритма. Код оваквог алата могли би се установити и критеријуми на основу којих би се могао мерити квалитет генерисаног алгоритма, па би се тако могли поредити различити алгоритми. Један од критеријума би могао бити степен постигнутог ефекта лавине у алгоритму. Како је у COALA систему подржан DES алгоритам који је заснован на *Feistel* структури алгоритам, управо би овај део система могао бити полазна тачка у имплементацији описаног алата. Овакав алат би се могао користити у оквиру домаћих задатака, где би се од студената очекивало да осмисле и помоћу алата креирају и испробају сопствени симетрични криптографски блок алгоритам који би морао да испуни одређене унапред дефинисане критеријуме. Други правац истраживања би био да се COALA систем искористи као основа за даљу имплементацију визуелне репрезентације одређених сигурносних протокола или апликација. Нпр. могла би се направити визуелна репрезентација PGP (енгл. *Pretty Good Privacy*) апликације за мрежну безбедност. Ова шема користи комбинацију симетричних и асиметричних криптографских алгоритама да би обезбедила тајност и аутентикацију код електронске поште. Визуелна репрезентација би могла да користи делове COALA система за приказ детаља добијања одређених резултата код PGP-а. Ово би могао бити пројекат који би студенти сами радили или би могло бити имплементирано и коришћено у оквиру лабораторијских вежби.

ЛИТЕРАТУРА

1. Stallings, W.: *Cryptography and Network Security*, Fourth Edition, Prentice Hall, 2005.

2. Jovanović, Z., Stanisavljević, Ž.: *Materijali za predavanja i vežbe iz Zaštite podataka*, dostupno na: http://rti.etf.bg.ac.rs/rti/ir4zp/materijali/index.html, poslednji pristup: april 2014.

3. Stanisavljevic, Z., Pavlovic, V., Nikolic, B., Djordjevic, J.: *SDLDS—System for Digital Logic Design and Simulation*, IEEE Transactions on Education, Vol. 56, No. 2, 2013, pp. 235-245.

4. Stanisavljevic, Z., Nikolic, B., Djordjevic, J.: *A Module for Automatic Assessment and Verification of Students' Work in Digital Logic Design*, ECBS 2012, Novi Sad 2012, pp. 275-282.

5. Pavlovic, V., Stanisavljevic, Z., Nikolic, B., Djordjevic, J.: *Digital Logic Simulator*, ECBS-EERC 2011, Bratislava 2011, pp. 155-156.

6. Stanisavljević, Ž., Pavlović, V., Đorđević, J., Nikolić, B.: *Vizuelni simulator prekidačkih mreža*, Zbornik radova sa konferencije ETRAN 2010, Donji Milanovac 2010, RT1.1.

7. Stanisavljevic, Z., Stanisavljevic, J.: *Softverski sistem za vizuelnu reprezentaciju klasičnih kriptografskih algoritama*, INFO M, Vol. 48, 2013, pp. 21-28.

8. Stanisavljevic, Z., Stanisavljevic, J.: *Softverski sistem za vizuelnu reprezentaciju Advanced Encryption Standard algoritma*, Zbornik radova sa konferencije TELFOR 2011, Beograd 2011, str. 1364-1367.

9. Stanisavljevic, Z., Stanisavljevic, J., Vuletic, P., Jovanovic, Z.: *COALA - System for Visual Representation of Cryptography Algorithms*, IEEE Transactions on Learning Technologies, Vol. 7, No. 2, pp. 178-190, April-June 2014.

10. Cooper, S., Dann, W., Pausch, R.: *Alice: a 3-D tool for introductory programming concepts*, Journal of Computing Sciences in Colleges, Vol. 15, No. 5, 2000, pp. 107-116.

11. Kölling, M., Quig, B., Patterson, A., Rosenberg, J.: *The BlueJ system and its pedagogy*, Computer Science Education, Vol. 13, No. 4, 2003, pp. 249-268.

12. Beer, R. D., Chiel, H. J., Drushel, R. F.: *Using autonomous robotics to teach science and engineering*, Communications of the ACM, Vol. 42, No. 6, 1999, pp. 85-92.

13. Yu, L., Harrison, L., Lu, A., Li, Z., Wang, W.: *3D Digital Legos for Teaching Security Protocols*, IEEE Transactions on Learning Technologies, Vol. 4, No. 2, 2011, pp. 125-137.

14. Yin, J., Wang, Y., Hsu, D.: *Digital violin tutor: an integrated system for beginning violin learners*, In Proceedings of the 13th annual ACM international conference on Multimedia, 2005, pp. 976-985.

15. Ardito, C., Buono, P., Costabile, M. F., Lanzilotti, R., Pederson, T., Piccinno, A.: *Experiencing the past through the senses: an m-learning game at archaeological parks*, IEEE Multimedia, Vol. 15, No. 4, 2008, pp. 76-81.

16. Hsi, S., Fait, H.: *RFID enhances visitors' museum experience at the Exploratorium*, Communications of the ACM, Vol. 48, No. 9, 2005, pp. 60-65.

17. Oshima, C., Nishimoto, K., Suzuki, M.: *Family ensemble: a collaborative musical edutainment system for children and parents*, In Proceedings of the 12th annual ACM international conference on Multimedia, 2004, pp. 556-563.

18. Toennies, J. L., Burgner, J., Withrow, T. J., Webster, R. J.: *Toward haptic/aural touchscreen display of graphical mathematics for the education of blind students*, In IEEE World Haptics Conference (WHC), 2011, pp. 373-378.

19. Harders, M., Bajka, M., Spaelter, U., Tuchschmid, S., Bleuler, H., Szekely, G.: *Highly-realistic, immersive training environment for hysteroscopy*, Studies in health technology and informatics, Vol. 119, 2005, pp. 176-181.

20. Chang, C. W., Lee, J. H., Chao, P. Y., Wang, C. Y., Chen, G. D.: *Exploring the Possibility of Using Humanoid Robots as Instructional Tools for Teaching a Second Language in Primary School*, Journal of Educational Technology & Society, Vol. 13, No. 2, 2010, pp. 13-24.

21. Farbood, M. M., Pasztor, E., Jennings, K.: *Hyperscore: a graphical sketchpad for novice composers*, IEEE Computer Graphics and Applications, Vol. 24, No. 1, 2004, pp. 50-54.

22. Ullrich, C., Shen, R., Tong, R., Tan, X.: *A mobile live video learning system for large-scale learning—system design and evaluation*, IEEE Transactions on Learning Technologies, Vol. 3, No. 1, 2010, pp. 6-17.

23. Klassner, F., Anderson, S. D.: *Lego MindStorms: Not just for K-12 anymore*, IEEE Robotics & Automation Magazine, Vol. 10, No. 2, 2003, pp. 12-18.

24. ACM/IEEE: *Computing curriculum 2001*, Available at: http://www.acm.org/education/curric_vols/cc2001.pdf, Accessed: April 2014.

25. Lee, S. H., Choi, J., Park, J. I.: *Interactive e-learning system using pattern recognition and augmented reality*, IEEE Transactions on Consumer Electronics, Vol. 55, No. 2, 2009, pp. 883-890.

26. Barab, S., Thomas, M., Dodge, T., Carteaux, R., Tuzun, H.: *Making learning fun: Quest Atlantis, a game without guns*, Educational Technology Research and Development, Vol. 53, No. 1, 2005, pp. 86-107.

27. Kanda, T., Ishiguro, H.: *Communication robots for elementary schools*, In Proc. AISB'05 Symposium Robot Companions: Hard Problems and Open Challenges in Robot-Human Interaction, 2005, pp. 54-63.

28. Ferguson, S., Vande Moere, A., Cabrera, D.: *Seeing sound: Real-time sound visualisation in visual feedback loops used for training musicians*, In Proceedings IEEE Ninth International Conference on Information Visualisation, 2005, pp. 97-102.

29. Fleck, M., Frid, M., Kindberg, T., Rajani, R., O'Brien-Strain, E., Spasojevic, M.: *From informing to remembering: Deploying a ubiquitous system in an interactive science museum*. IEEE pervasive computing, Vol. 1, No. 2, 2002, pp. 13-21.

30. Mayer, R. E., Moreno, R.: *Animation as an aid to multimedia learning*, Educational psychology review, Vol. 14, No. 1, 2002, pp. 87-99.

31. Mayer, R. E.: *Cognitive theory of multimedia learning*, The Cambridge handbook of multimedia learning, 2005, pp. 31-48.

32. Moreno, R., Mayer, R.: *Interactive multimodal learning environments*, Educational Psychology Review, Vol. 19, No. 3, 2007, pp. 309-326.

33. Mayer, R. E., Johnson, C. I.: *Revising the redundancy principle in multimedia learning*, Journal of Educational Psychology, Vol. 100, No. 2, 2008, pp. 380-386.

34. Churchill, D.: *Towards a useful classification of learning objects*, Educational Technology Research and Development, Vol. 55, No. 5, 2007, pp. 479-497.

35. Ainsworth, S.: *How should we evaluate multimedia learning environments?*, In Understanding multimedia documents, 2008, pp. 249-265.

36. Ellis, T.: *Animating to Build Higher Cognitive Understanding: A Model for Studying Multimedia Effectiveness in Education*, Journal of Engineering Education, Vol. 93, No. 1, 2004, pp. 59-64.

37. Mayer, R. E., Moreno, R.: *A Cognitive Theory of Multimedia Learning: Implications for Design Principles*, In annual meeting of the ACM SIGCHI Conference on Human Factors in Computing Systems, 1998, pp. 1-8.

38. Urquiza-Fuentes, J., Velazquez-Iturbide, J. A.: *A survey of successful evaluations of program visualization and algorithm animation systems*, ACM Transactions on Computing Education (TOCE), Vol. 9, No. 2, article no. 9, 2009, pp. 1-24.

39. Price, B., Baecker, R., Small, I.: *An introduction to software visualization*, In J. Stasko, J. Domingue, M. Brown, B. Price (Eds.) Software Visualization, Cambridge, MA: MIT Press, 1998, pp. 3–27.

40. Kulyk, O., Kosara, R., Urquiza-Fuentes, J., I., W.: *Human-Centered Visualization Environments*, chap. Human-Centered Aspects, Springer-Verlag, 2007, pp. 13–75.

41. Hundhausen, C. D., Douglas, S. A., Stasko, J. T.: *A meta-study of algorithm visualization effectiveness*, Journal of Visual Languages & Computing, Vol. 13, No. 3, 2002, pp. 259-290.

42. Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Velázquez-Iturbide, J. Á.: *Exploring the role of visualization and engagement in computer science education*, ACM SIGCSE Bulletin, Vol. 35, No. 2, 2002, pp. 131-152.

43. Hundhausen C. D.: *Toward efective algorithm visualization artifacts: designing for participation and communication in an undergraduate algorithms course*, Unpublished Ph.D. dissertation, Department of Computer and Information Science, University of Oregon, 1999.

44. Mayer, R. E., Anderson, R. B.: *Animations need narrations: an experimental test of a dual-coding hypothesis*, Journal of Educational Psychology, Vol. 83, 1991, pp. 484-490.

45. Cooper, C.: *Individual Differences*, Oxford Illustrated Press, Oxford, 1997.

46. Resnick, L. B.: *Introduction*, In: Knowing, Learning, and Instruction: Essays in Honor of RobertGlaser (L. B. Resnick, ed.), Erlbaum, Hillsdale, NJ, 1989, pp. 1-24.

47. Shaffer, C. A., Cooper, M. L., Alon, A.J.D., Akbar, M., Stewart, M., Ponce, S., Edwards, S.H.: *Algorithm Visualization: The state of the field*, ACM Transactions on Computing Education (TOCE), Vol. 10, No. 3, article no. 9, 2010, pp. 1-22.

48. Rosling, G., Schuer, M., Freisleben, B.: *The ANIMAL algorithm animation tool*, In ACM SIGCSE Bulletin, Vol. 32, No. 3, 2000, pp. 37-40.

49. Hundhausen, C. D., Brown, J. L.: *What You See Is What You Code: A "live" algorithm development and visualization environment for novice learners*, Journal of Visual Languages & Computing, Vol. 18, No.1, 2007, pp. 22-47.

50. Sánchez-Torrubia, M. G., Torres-Blanc, C., López-Martínez, M. A.: *PathFinder: A Visualization eMathTeacher for Actively Learning Dijkstra's Algorithm*, Electronic Notes in Theoretical Computer Science, Vol. 224, 2009, pp. 151–158.

51. Vrachnos, E., Jimoyiannis, A.: *Dave: A dynamic algorithm visualization environment for novice learners*, ICALT'08. Eighth IEEE International Conference on Advanced Learning Technologies, 2008, pp. 319-323.

52. Dershem, H. L., McFall, R. L., Uti, N.: *Animation of Java linked lists*. ACM SIGCSE Bulletin, Vol. 34, No. 1, 2002, pp. 53-57.

53. Geschke, A., Kortenkamp, U., Lutz-Westphal, B., Materlik, D.: *Visage—Visualization of algorithms in discrete mathematics*, ZDM, Vol. 37, No. 5, 2005, pp. 395-401.

54. Moreno, A., Myller, N., Sutinen, E., Ben-Ari, M.: *Visualizing programs with Jeliot 3,* Proceedings of the 8th International Working Conference on Advanced Visual Interfaces, 2004, pp. 373–376.

55. Naps, T. L.: *Jhave: Supporting algorithm visualization*, IEEE Computer Graphics and Applications, Vol. 25, No. 5, 2005, pp. 49-55.

56. Karavirta, V., Korhonen, A., Malmi, L., Stålnacke, K.: *MatrixPro-A tool for on-the-fly demonstration of data structures and algorithms*. Proceedings of the Third Program Visualization Workshop, 2004, pp. 26-33.

57. Giordano, J. C., Carlisle, M.: *Toward a more effective visualization tool to teach novice programmers*, Proceedings of the 7th ACM conference on Information technology education, 2006, pp. 115-122.

58. Baloukas, T.: *JAVENGA: JAva-based Visualization Environment for Network and Graph Algorithms*, Computer Applications in Engineering Education, Vol. 20, No. 2, 2012, pp. 255-268.

59. Akingbade, A., Finley, T., Jackson, D., Patel, P., Rodger, S. H.: *JAWAA: easy web-based animation from CS 0 to advanced CS courses*, ACM SIGCSE Bulletin, Vol. 35, No. 1, 2003, pp. 162-166.

60. Schweitzer, D., Brown, W.: *Interactive visualization for the active learning classroom*, ACM SIGCSE Bulletin, Vol. 39, No. 1, 2007, pp. 208-212.

61. Cattaneo, G., De Santis, A., Ferraro Petrillo, U.: *Visualization of cryptographic protocols with GRACE*, Journal of Visual Languages & Computing, Vol. 19, No. 2, 2008, pp. 258-290.

62. Schweitzer, D., Baird, L.: *The design and use of interactive visualization applets for teaching ciphers*, IEEE Information Assurance Workshop, 2006, pp. 69-75.

63. Tao, J., Ma, J., Mayo, J., Shene, C. K., Keranen, M.: *DESvisual: a visualization tool for the DES cipher*, Journal of Computing Sciences in Colleges, Vol. 27, No. 1, 2011, pp. 81-89.

64. Tao, J., Ma, J., Keranen, M., Mayo, J., Shene, C. K.: *ECvisual: a visualization tool for elliptic curve based ciphers*, Proceedings of the 43rd ACM technical symposium on Computer Science Education, 2012, pp. 571-576.

65. Tao, J., Ma, J., Keranen, M., Mayo, J., Shene, C. K., Wang, C: *RSAvisual: A Visualization Tool for the RSA Cipher*, In Proceedings of ACM Technical Symposium on Computer Science Education, 2014.

66. Anane, R., Purohit, K., Theodoropoulos, G.: *An Animated Cryptographic Learning Object,* IEEE Fifth International Conference on Computer Graphics, Imaging and Visualisation, 2008, pp. 61-68.

67. Schweitzer, D., Baird, L., Collins, M., Brown, W., Sherman, M.: *GRASP: A visualization tool for teaching security protocols*, Proceedings of the 10th Colloquium for Information Systems Security Education, 2006, pp. 75-81.

68. Schweitzer, D., Gibson, D., Collins, M.: *Active learning in the security classroom*, 42nd Hawaii International Conference on System Sciences, 2009, pp. 1-8.

69. National Institute of Standards and Technology: *FIPS PUB 46-3. Data Encryption Standard (DES)*, 1999, Available at: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, Accessed: April 2014.

70. National Institute of Standards and Technology: *FIPS PUB 197. Advanced Encryption Standard*, 2001, Available at: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, Accessed: April 2014.

71. Daemen, J., Rijmen, V.: *The design of Rijndael: AES-the advanced encryption standard*, Springer, 2002.

72. National Institute of Standards and Technology: *NIST Special Publication 800-67 Version 1.1.*, 2012, Available at: http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf, Accessed: April 2014.

73. Burwick, C., Coppersmith, D., D'Avignon, E., Gennaro, R., Halevi, S., Jutla, C., Matyas, S., O'Connor, L., Peyravian, M., Safford, N., Zunic, N.: *MARS-a candidate cipher for AES*, NIST AES Proposal, 1998.

74. Rivest, R., Robshaw, M., Sidney, R.: *The RC6 Block Cipher*, NIST AES Proposal, 1998.

75. Daemen, J., Rijmen, V.: *AES Proposal: Rijndael*, NIST AES Proposal, 1998.

76. Anderson, R., Biham, E., Knudsen, L.: *Serpent: A Proposal for the Advanced Encryption Standard*, NIST AES Proposal, 1998.

77. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: *Twofish: A 128-bit Block Cipher*, NIST AES Proposal, 1998.

78. Diffie, W., Hellman, M. E.: *New directions in cryptography*, IEEE Transactions on Information Theory, Vol. 22, No. 6, 1976, pp. 644-654.

79. Rivest, R. L., Shamir, A., Adleman, L.: *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, Vol. 21, No. 2, 1978, pp. 120-126.

80. *Java*. Available at: http://www.java.com/en/, Accessed: April 2014.

81. *Netbeans*. Available at: https://netbeans.org/, Accessed: April 2014.

82. *Moodle*. Available at: http://moodle.com/, Accessed: April 2014.

БИОГРАФИЈА АУТОРА

Жарко Станисављевић, мастер инжењер електротехнике и рачунарства, рођен је 21.09.1984. године у Београду, Република Србија, од оца Славка и мајке Марине. Основну школу „Стеван Синђелић“ и средњу школу „Никола Тесла“ завршио је у Београду као један од најбољих ученика.

Електротехнички факултет Универзитета у Београду, одсек Рачунарска техника и информатика, уписао је 2003. године. Све обавезе је одрадио у предвиђеном року и дипломирао у августу 2007. године са просечном оценом 8,89 и оценом 10 на дипломском. Дипломске академске студије - мастер је уписао 2007. године на Електротехничком факултету Универзитета у Београду, одсек Рачунарска техника и информатика, и завршио их је 2008. године са просечном оценом 9,67 и оценом 10 на мастер раду. Докторске академске студије је уписао у јануару 2009. године на Електротехничком факултету Универзитета у Београду, одсек Рачунарска техника и информатика.

Од септембра 2007. године до марта 2008. године био је запослен у фирми Ариус у Београду на позицији софтверског инжењера за развој интернет апликација. Од марта 2008. године ангажован је хонорарно као сарадник у настави на Електротехничком факултету Универзитета у Београду и на пројектима Електротехничког факултета под руководством проф. др Зорана Јовановића, а од јануара 2010. године је запослен на Електротехничком факултету Универзитета у Београду као асистент где је тренутно ангажован на предметима: Заштита података, Основи рачунарске технике 1, Архитектура и организација рачунара, Организација рачунара и Архитектура и организација рачунара 1.

Коаутор је два рада у међународним часописима са *impact* фактором са SCI листе од којих је један у директној вези са дисертацијом, једног рада у домаћем часопису, два рада на међународним конференцијама и три рада на домаћим конференцијама.

Прилог 1.

Изјава о ауторству

Потписани Жарко Станисављевић

број индекса 2008/5050

Изјављујем

да је докторска дисертација под насловом

Визуелна репрезентација криптографских алгоритама

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

У Београду, 27.08.2014.

Потпис докторанда

Станисављевић

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Жарко Станисављевић

Број индекса 2008/5050

Студијски програм Рачунарска техника и информатика

Наслов рада Визуелна репрезентација криптографских алгоритама

Ментор проф. др Зоран Јовановић

Потписани Жарко Станисављевић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, 27.08.2014.

Станисављевић

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Визуелна репрезентација криптографских алгоритама

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, 27.08.2014.

Станисављевић М.

1. Ауторство - Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.