SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Roko Milošević

NAPADI NA MODERNU SIMETRIČNU KRIPTOGRAFIJU

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Roko Milošević

Matični broj: 0016141820

Studij: Informacijski sustavi

NAPADI NA MODERNU SIMETRIČNU KRIPTOGRAFIJU

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Nikola Ivković

Varaždin, rujan 2022.

Roko Milošević

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Cilj ovog završnog rada je simulirati napad na modernu simetričnu kriptografiju. Rad se sastoji od teorijskog dijela u kojem su ukratko opisani najbitniji dijelovi duge povijesti kriptografije te su detaljno opisani odabrani algoritmi i metode kriptografije. Opisane su supstitucijske šifre, transpozicijske šifre, jednokratni ključevi te moderna kriptografija u obliku simetrične i asimetrične kriptografije. Rad pobliže opisuje AES algoritam te razloge njegovog nastanka i korištenja. Nadalje, rad opisuje motivaciju te načine napada na kriptografiju. Praktični dio rada se sastoji od programskog koda čiji cilj je šifriranje i dešifriranje poruka korištenjem AES algoritma, simulacija napada na isti te prikaz podataka o napadu. Naposlijetku se putem dobivenih podataka iz simulacije napada donosi zaključak o sigurnosti AES algoritma.

Ključne riječi: simetrična, kriptografija, algoritam, napad, AES

Sadržaj

1. Uvod

Nitko ne želi da njegove tajne budu javno dostupne pa je kriptografija kao znanstvena disciplina od velike važnosti svima, od individualnih osoba pa sve do multinacionalnih kompanija vrijednih milijarde dolara. Zbog navedene vrijednosti kriptografije ona se često nalazi na meti osoba s malicioznim namjerama kojima je u cilju razotkriti nečije tajne radi nekog razloga – često materijalnog. Tema ovog završnog rada je moderna simetrična kriptografija kao jedna od podvrsta kriptografije te napad na istu.

Većina svjetske populacije se koristi raznim uređajima te dosta vremena provodi ispred ekrana. Za svakog pojedinog individualca je iznimno bitno biti upućen u sustave koji se nalaze u pozadini moderne tehnologije, primarno zbog vlastite sigurnosti. Ovo posebno vrijedi za mlade ljudi koji su odrasli uz korištenje modernih tehnologija te im je stoga lakše razumjeti način na koji iste funkcioniraju i to razumijevanje predočiti i prenijeti starijima i onima koji nisu u stanju pohvatati konce moderne tehnologije. Kriptografija je odličan prvi korak ka razumijevanju šire teme sigurnosti na internetu i sigurne komunikacije te bi svatko tko koristi modernu tehnologiju morao biti u mogućnosti baratati elementarnim znanjem kriptografije.

Ovaj rad će se baviti poviješću kriptografije od njenog začeća pa sve do modernog informacijskog doba, pobliže će pojašnjavati određene pojmove, kriptografske koncepte, metode, algoritme i načine napada na kriptografiju. Nadalje, praktično će prikazati odabranu vrstu napada na kriptografiju te analizirati učinkovitost iste.

2. Kriptografija

Prije no što možemo početi razgovarati o kriptografiji kao znanstvenoj disciplini potrebno je sagledati razloge zbog kojih je kriptografija relevantna. Komunikacija između individualaca je jedan od, ako ne i najvažniji faktor koji je pogodovao razvoju moderne civilizacije te moderno društvo kao takvo ne bi moglo postojati bez razine organizacije i koordinacije koju omogućava komunikacija. S pojavom organiziranog društva i formiranjem distinktnih socijalnih grupa, koje su odvojene od te u nekim slučajevima i otvoreno suprotstavljene drugim sličnim no ipak različitim grupama, javlja se pojava malicioznog djelovanja na međuljudskoj razini. Komunikacija je važan alat no u krivim rukama ona postaje opasno oruđe kojim zlonamjerni individualci mogu štetno djelovati na društvo.

Komunikacija „licem i lice“ ima ugrađeni mehanizam koji sprječava većinu zlokobnih djelovanja. Taj mehanizam je naravno činjenica da se u takvoj komunikaciji, kao što naziv sugerira, nalazite licem u lice s vašim sugovornikom te je iznimno teško lažno se predstaviti, presresti privatnu poruku ili zlokobno djelovati na bilo kakav drugi način. Stoga možemo zaključiti da se prvi problemi javljaju otkrićem i širenjem pisanog jezika, oko 3000. godine prije nove ere.

Stupanjem pisanog jezika tj. pisma na scenu otvaraju se nove mogućnosti komunikacije između individualaca te organiziranih skupina. Razvijaju se i rastu se trgovina, knjigovodstvo, itd. te se pisana komunikacija isprepleće sa sponama društva i postaje kamen temeljac civilizacije kakvu poznajemo danas. Ovaj eksponencijalni rast i razvitak komunikacije uzrokuje pojavu prvih zlonamjernih metoda iskorištavanja komunikacije – prisluškivanja, lažnog predstavljanja, presretanja poruka, promjene sadržaja poruke, itd. Pojava metoda manipuliranja komunikacijom uzrokuje pojavu načina borbe protiv istih te tu na scenu stupa kriptografija.

Kriptografija je znanstvena disciplina koja se bavi metodama sigurne komunikacije pisanim porukama čiji cilj je da poruku mogu razumjeti samo pošiljatelj i primatelj. Navedena ekskluzivnost razumijevanja poruke se postiže prevođenjem razgovijetnih podataka u nerazgovijetne podatke na način da primatelj uz pomoć tajnog ključa može obrnuti proces i od nerazgovijetnih podataka doći do razgovijetnih [1]. Riječ „kriptografija“ dolazi od grčkog „*kriptós* “ što znači „tajno“ i „*gráfo* “ što znači „pisanje“ [2].

2.1. Osnovni pojmovi i svojstva kriptografije

2.1.1. Sigurna komunikacija

Glavna svrha kriptografije je da osigura sigurnu komunikaciju. Kurose i Ross [3] definiraju četiri svojstva sigurne komunikacije: povjerljivost, integritet poruke, autentičnost krajnje točke i operativnu sigurnost.

Povjerljivost se odnosi na prije spomenuto svojstvo da sadržaj poruke moraju moći razumjeti samo pošiljatelj i primatelj poruke te je ona prva stavka koja mnogima padne na pamet kada se spomene pojam kriptografije i sigurnosti u komunikaciji a samim time je i žarišna točka ovog rada. No povjerljivost naravno nije jedini problem u sigurnosti tj. nesigurnosti komunikacije.

Integritet poruke je svojstvo koje se odnosi na nepromijenjenost poruke kroz čitav tok komunikacije. „Osiguran“ integritet poruke znači da poruka definitivno nije promijenjena, bilo to namjerno ili slučajno.

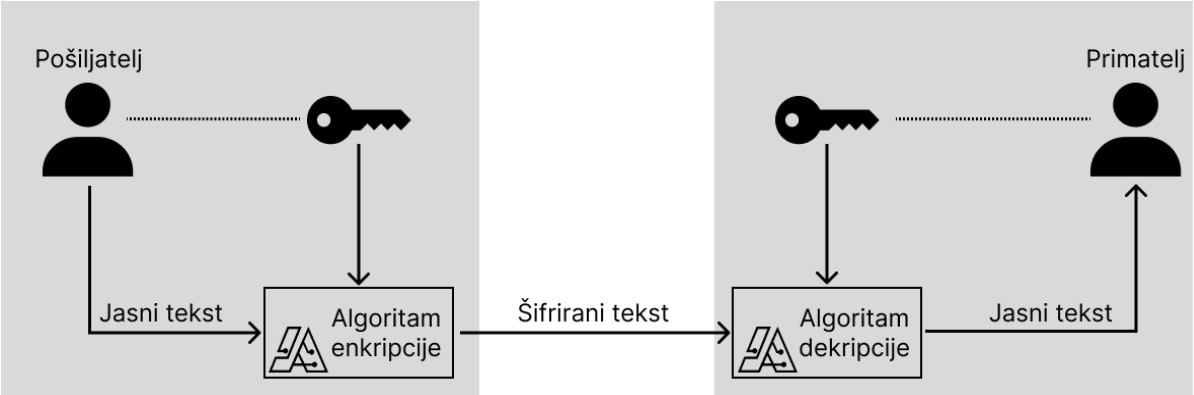
Autentičnost krajnje točke je svojstvo koje osigurava da svi sudionici razgovora mogu potvrditi identitet ostalih sudionika. Možda zvuči banalno no prisluškivanjem i ubacivanjem na kanale komunikacije je izrazito lako predstaviti se kao netko drugi.

Operativna sigurnost je posljednje navedeno svojstvo te se više odnosi na periferiju nego na samu komunikaciju. Govori kako je osiguravanje implementacije kriptografskih metoda jednako važno, ako ne i važnije, od osiguravanja same metode. Dobra analogija bi bila sljedeća: u želji za osiguranjem kuće od provalnika osoba se odluči za kupnju teških metalnih protuprovalnih vrata s šest neovisnih lokota i brava ali za instalaciju vrata odabere najjeftinijeg majstora koji će loše obaviti posao ili vrata instalira na štok od iverice i/ili montažni zid koji se lako mogu probiti. Sama vrata su iznimno sigurna, no njihova okolina omogućuje provalnicima da ih lako zaobiđu.

2.1.2. Osnovni pojmovi

Kada govorimo o razmjeni poruka između dvije stranke, spominjemo pojmove poput „algoritama enkripcije i dekripcije“, „jasnog teksta“, „šifriranog teksta“ te „ključa“. Pošiljatelj poruke zapisuje poruku u jasnom tekstu tj. na standardnom pismu odabranog jezika na način da je bilo tko može pročitati. Zatim se taj jasan tekst unosi u algoritam enkripcije te se odabire ili nasumično generira ključ. Algoritam enkripcije uz pomoć ključa šifrira jasni tekst te ga pretvara u šifrirani tekst. Šifrirani tekst se potom šalje primatelju poruke koji posjeduje svoj ključ. Ključ koji primatelj posjeduje može a i ne mora biti jednak ključu pošiljatelja – ovisno o vrsti kriptografije o kojoj je riječ. Primatelj uz pomoć algoritma dekripcije i vlastitog ključa

dešifrira šifrirani tekst te dobiva jasni tekst koji može pročitati [3]. Čitav upravo opisani postupak je prikazan na slici (Slika 1).



Slika 1. Osnovni kriptografski pojmovi (Prema: Kurose i Ross, 2007)

2.1.3. Kerckhoffsov princip

Kerckhoffsov princip ili Kerckhoffsovo načelo je svojevrsno pravilo koje je pripisano i nazvano po nizozemskom tj. flamanskom vojnom kriptografu Auguste-u Kerckhoff-u. U jednom od svojih radova 1883. Kerckkhoff tvrdi kako svi algoritmi u kriptografiji moraju biti u potpunosti javni te samo ključevi moraju biti tajni [2]. U današnje doba većina, ako ne i svi kriptografi se drže ovog jednostavnog načela.

2.2. Povijest kriptografije

Kao što je već spomenuto u uvodnom dijelu poglavlja, prva upotreba tajnih poruka se javlja odmah nakon prve upotrebe pisma. Nekoliko tisućljeća prije nove ere stari Egipćani su koristili određene primitivne metode kriptografije a šifriranje se također može pronaći i u Bibliji. Ipak, prvi dobro dokumentirani primjeri potječu iz antičke Grčke.

2.2.1. Antička Grčka

Gotovo sve što znamo o staroj Grčkoj dolazi od nekolicine povjesničara, jedan od kojih je Herodot - također poznat i kao „otac povijesti“. U svom dijelu „*Povijesti*“ Herodot opisuje dva odvojena slučaja korištenja tajnih poruka u svrhu političkih spletki i ratovanja. U prvoj priči Herodot opisuje način na koji je jedan Grk poslao poruku svojim sunarodnjacima u Grčku tako da je njegovi perzijski čuvari nisu otkrili. Naime on uzima drvene pločice premazane voskom, koje su u antičko doba služile za pisanje, te struže sav vosak s njih. Potom urezuje svoju tajnu

poruku upozorenja Grcima na drvo i ponovno primjenjuje voštani sloj preko drveta. Primatelj poruke je s tablica ponovno ostrugao sav vosak te pročitao poruku [4].

Druga Herodotova priča opisuje sličan način prijenosa tajne poruke. Glasniku koji prenosi poruku je obrijana glava te mu je na tjeme napisana tajna poruka. Glasnik tada čeka da mu ponovno naraste kosa te tek onda kreće na put prema željenom odredištu. Nakon dolaska na odredište, glasnik brije svoju glavu te otkriva tajnu poruku napisanu na tjemenu [4].

Obje Herodotove priče govore o primjerima komunikacije gdje je tajnost poruke izuzetno važna no ipak to nisu primjeri kriptografije već znanosti koja je blisko srodna kriptografiji, a to je steganografija. Kriptografija se bavi skrivanjem pravog značenja poruke na način da se promijeni značenje poruke uz mogućnost dobivanja inicijalnog značenja ako primatelj posjeduje ključ. U drugu ruku, steganografija se bavi skrivanjem postojanja same poruke. U antičko doba steganografija se provodila na jednostavne ali domišljate načine kao što je prikazano u spomenutim primjerima. Iako obje metode imaju isti cilj – skrivanje informacija od neželjenih stranaka, nisu iste u svojoj izvedbi.

Jedan od prvih dokumentiranih slučajeva korištenja kriptografije nas vodi sedamdesetak godina nakon prve Herodotove priče – na sam kraj petog stoljeća prije nove ere. Riječ je o spartanskom alatu naziva *„skutálē“* ili „skital“ tj. „palica“ ili „cilindar“. Radi se o drvenoj palici koja nije u potpunosti okrugla već ima ravnine. Kriptiranje se odvija na način da se tanka traka kože ili papira namota oko palice te se slova upisuju na dijelove kože ispod kojih su ravne površine. Nakon pisanja poruke koža ili papir se odmotaju te se prenose do primatelja. Primatelj posjeduje palicu jednakog obujma kao i pošiljatelj te poruku dekriptira tako da kožu ili papir namota na svoju palicu. U slučaju da netko pokuša pročitati poruku direktno sa kože/papira ili da cilindar nije identičnog obujma kao onaj korišten za šifriranje, poruka nije čitljiva i izgleda kao nasumičan niz slova [4].

2.2.2. Rim

Spartanski skital opisan u poglavlju 2.2.1. je primjer tehnike šifriranja koja se naziva „transpozicija“. Osim transpozicije, u kriptografiji se može pojaviti i „supstitucija“ [4]. Više o supstituciji i transpoziciji u poglavlju o vrstama kriptografije.

Cezarova šifra je transpozicijska šifra koja je ime dobila po svom kreatoru – Gaju Juliju Cezaru, rimskom političaru i vojskovođi. Naime Cezar je u svojim vojnim pohodima u Galiji, današnjoj Francuskoj, prvi upotrijebio supstituciju za prenošenje tajnih poruka. U svojoj knjizi *Galski Rat*, Cezar opisuje način na koji je šifrirao poruku koju je poslao svom generalu Ciceru koji se našao u okruženju i pod opsadom. Cezar je jednostavno svako slovo latinske abecede zamijenio srodnim slovom grčke abecede te tako osigurao da Gali neće moći pročitati poruku ako im dospije u ruke [4].

Iako je primjer šifre koja supstituira latinska slova grčkima jako zanimljiv i domišljat, to ipak nije najpoznatiji način na koji je Cezar šifrirao svoje poruke te se ne misli na njega kada se upotrebljava naziv „Cezarova šifra“. Daleko poznatiji način na koji je Cezar svoje poruke šifrirao supstitucijom, te onaj koji je po njemu i dobio ime opisuje Suetonius u svom dijelu *O životu dvanaest careva*: Cezar je jednostavno zamijenio slovo abecede sa slovom tri mjesta nakon njega [4]. U primjeni Cezarove šifre na hrvatsku abecedu bi tako slovo A postalo slovo Č, slovo B postalo slovo Ć, itd. dok se ne popuni čitava abeceda. Kada dođemo do kraja abecede naravno počinjemo ispočetka te slovo Z postaje slovo B a slovo Ž postaje slovo C. Na sljedećoj slici (Slika 2) možemo vidjeti čitavu hrvatsku abecedu pokraj šifrirane abecede s pomakom vrijednosti 3, tj. koristeći Cezarovu šifru.

Hrvatska abeceda	A	B	C	Č	Ć	D	Dž	Đ	E	F	G	H	I	J	K	L	Lj	M	N	Nj	O	P	R	S	Š	T	U	V	Z	Ž
Šifrirana abeceda	Č	Ć	D	Dž	Đ	E	F	G	H	I	J	K	L	Lj	M	N	Nj	O	P	R	S	Š	T	U	V	Z	Ž	A	B	C

Slika 2: Hrvatska abeceda u Cezarovoj šifri (Prema: Singh, 1999)

Naravno, šifriranje hrvatske abecede Cezarovom šifrom može predstaviti nekoliko problema poput npr. što učiniti sa dvoznačnim slovima. Na sljedećoj slici (Slika 3) možemo vidjeti jednu kratku poruku šifriranu Cezarovom šifrom.

Jasni tekst: Kriptografija

Šifrirani tekst: MTLŠZSJTČILLjČ

Slika 3: Primjer šifriranja Cezarovom šifrom [autorski rad]

Singh [4] navodi kako se jasni tekst piše uobičajenim pravilima velikog i malog slova a šifrirani tekst se piše isključivo velikim slovima. Ovdje se vidi problem koji nastaje korištenjem hrvatske abecede – kod slova Nj i Lj je nemoguće pisati oba znaka velikim slovom jer to mijenja značenje poruke.

Cezar je u svojim tajnim porukama koristio pomak vrijednosti 3 te se stoga supstitucijska šifra s pomakom vrijednosti 3 naziva po njemu, no moguće je koristiti i broj drugih vrijednosti pomaka [4]. Broj mogućih vrijednosti pomaka naravno ovisi o broju slova u abecedi pa tako u hrvatskoj abecedi možemo koristiti vrijednosti od 1 do 29, jer postoji 30 slova (a pomak od 30 ili 0 bi značio da jasni tekst i šifrirani tekst budu identični).

2.2.3. Islamsko zlatno doba

Arapi svoj doprinos kriptografiji nisu dali direktno poput Rimljana i Grka prije njih, već indirektno tako što su se prvi počeli intenzivno baviti kriptoanalizom. Jedna od najzanimljivijih metoda kriptoanalize kojim su se Arapi koristili u svojim pokušajima probijanja šifri je ona koju opisuje tzv. „arapski filozof" Al-Kindī. On predlaže da, ako znamo u kojem jeziku je pisan šifrirani tekst, uzmemo bilo koji drugi jasni tekst pisan u tom jeziku koji je približno jednake duljine te prebrojimo ponavljanja pojedinih slova. Isto učinimo za kriptirani tekst (prebrojimo broj znakova) pa usporedimo s jasnim tekstom i zamijenimo znakove sa slovima iz jasnog teksta koja imaju približno jednaku frekvenciju ponavljanja [4]. Ovaj način razbijanja šifri se pokazao učinkovit kod razbijanja dužih tekstova pisanih jednostavnim jezikom no ne i kod kraćih tekstova ili tekstova s jako specifičnim temama.

2.2.4. Srednji vijek i renesansa

Već pred kraj srednjeg vijeka je bilo jasno da su monoalfabetske supstitucijske šifre stvar povijesti te su pojedinci nastojali osmisliti novi način šifriranja koji neće rezultirati dospijevanjem poruka u krive ruke. Rješenje za ovaj problem je bila polialfabetska šifra. „Poli" od starog grčkog *„polýs"* nosi značenje „više" ili „mnogo" te se u ovom slučaju odnosi na više abeceda korištenih u šifriranju iste poruke. Teorijsku osnovu za ovaj novi tip šifre je postavio Leon Battista Alberti u šezdesetim godinama 15. stoljeća [4]. Rješenje je iznimno jednostavno a u isto vrijeme genijalno: umjesto korištenja jedne šifrirane abecede za pretvaranje jasnog teksta u šifrirani tekst, koristi se više abeceda naizmjence. Na primjer uzimaju se dvije abecede te se poruka šifrira na način da je prvo slovo šifrirano po prvoj abecedi, drugo slovo po drugoj, treće ponovno po prvoj, itd. Korištenjem više abeceda dolazi se u situaciju gdje jedan znak jasnog teksta može biti označen sa više znakova u šifriranom tekstu i jedan znak šifriranog teksta može značiti više znakova u jasnom tekstu. Ova metoda šifriranja znatno otežava dešifriranje po staroj arapskoj metodi frekvencijske analize.

Alberti je postavio teorijsku osnovu no nije sam i razvio šifru već je njegov rad nastavljen od strane nekolicine intelektualaca poput Johannesa Trithemiusa, Giovannia Porte i Blaisea de Vignèrea. Posljednji od njih, Vignère, je šifru doveo u njeno konačno stanje te je stoga ona i nazvana po njemu – Vignèreova šifra [4]. Vignèreova šifra funkcionira na način da za svako slovo abecede postoji zasebna Cezarova šifra s pomakom za jednim više od prethodnog slova. Tako bi za hrvatsku abecedu Vignèreova šifra imala 30 različitih Cezarovih šifri. Poanta šifre je da se ne koristi ista Cezarova šifra za cijeli tekst (jer tada je to najobičnija monoalfabetska šifra) već da se za znakove redom koriste različite šifre. Ne moraju sve monoalfabetske šifre biti iskorištene ali poanta je da se koristi više od jedne a po mogućnosti barem tri, četiri ili više

šifri. Praćenje koje šifre se koriste se izvodi na način da se uz šifrirani tekst primatelju šalje i ključna riječ uz pomoć koje primatelj može saznati koji red tj. koja monoalfabetska šifra se koristi za koji znak šifriranog teksta. Vrijedno je još jednom spomenuti da se Vignèreova šifra sastoji od X redova gdje svaki red sadrži monoalfabetsku šifru s pomakom za jedan većim od prošlog reda. Prvi red tako ima pomak jedan, drugi red dva, itd. što za posljedicu ima činjenicu da svaki red počinje s različitim slovom – prvi red sa slovom B, drugi sa C, itd. Način na koji primatelj zna koji red tj. koju šifru koristiti za dešifriranje kojeg znaka je sljedeći: primatelj od pošiljatelja prima ključnu riječ te je ispisuje iznad šifriranog teksta tako da jedan znak ključne riječi bude iznad jednog znaka teksta. Ovaj proces ponavlja sve dok ne dođe do kraja teksta. Tada primatelj gleda slovo ključne riječi iznad znaka teksta te pronalazi red u Vignèreovoj šifri koji počinje s tim slovom i tako zna da taj znak mora dešifrirati koristeći taj red. Na primjer: uzmimo da je ključna riječ „paun“ a da je šifrirani tekst „a…“. Možemo vidjeti da je prvi znak „a“ te da je iznad njega slovo „p“ što bi značilo da znak a dešifriramo onim redom tj. šifrom Vignèreove šifre čiji pomak je takav da abeceda počinje sa slovom „p“. To bi u ovom slučaju bio 21. red. Na sljedećoj slici (Slika 4) je prikazano prvih pet redova Vignèreove šifre s hrvatskom abecedom.

	A	B	C	Č	Ć	D	Dž	Đ	E	F	G	H	I	J	K	L	Lj	M	N	Nj	O	P	R	S	Š	T	U	V	Z	Ž
1	B	C	Č	Ć	D	Dž	Đ	E	F	G	H	I	J	K	L	Lj	M	N	Nj	O	P	R	S	Š	T	U	V	Z	Ž	A
2	C	Č	Ć	D	Dž	Đ	E	F	G	H	I	J	K	L	Lj	M	N	Nj	O	P	R	S	Š	T	U	V	Z	Ž	A	B
3	Č	Ć	D	Dž	Đ	E	F	G	H	I	J	K	L	Lj	M	N	Nj	O	P	R	S	Š	T	U	V	Z	Ž	A	B	C
4	Ć	D	Dž	Đ	E	F	G	H	I	J	K	L	Lj	M	N	Nj	O	P	R	S	Š	T	U	V	Z	Ž	A	B	C	Č
5	D	Dž	Đ	E	F	G	H	I	J	K	L	Lj	M	N	Nj	O	P	R	S	Š	T	U	V	Z	Ž	A	B	C	Č	Ć

Slika 4: Prvih pet redova Vignèreove šifre s hrvatskom abecedom (Prema: Singh, 1999)

Nedostatak Vignèreove šifre je njena kompliciranost [4]. Činjenica je da je velika većina populacije nepismena što se tiče kriptografije te stoga nije bilo potrebe koristiti kompliciranе polialfabetske šifre koje zahtijevaju puno truda i vremena već su se za šifriranje svakodnevnih poruka koristile monoalfabetske šifre unatoč slabijoj sigurnosti.

2.2.5. Industrijska revolucija

U 18. stoljeću šifriranje je pratilo trend vremena pa je industrijalizacijom uvelike ubrzano razbijanje jednostavnih šifri. Kriptoanalitičari su još jednom bili korak ispred kriptografa te je to kriptografe napokon nagnalo na to da prestanu koristiti monoalfabetske šifre i prebace se na korištenje sigurnijih no vremenski zahtjevnijih polialfabetskih šifri, među kojima je najpopularnija bila Vignèreova šifra [4].

1753. godine pojavljuje se ideja o sklopu koji ima mogućnost slanja poruke preko velike udaljenosti u nekoliko sekundi [4]. Iako ideja neće zaživjeti do 19. stoljeća, ovo je prvi spomen telegrama – tehnologije koja će vladati domenom komunikacije sve do adventa telefona. Verzija telegrama razvijena u Europi tj. Engleskoj, tzv. Wheatstone-Cooke sistem telegrama – nazvan po njegovim izumiteljima koji su uzeli originalnu ideju iz 1753. te je primijenili u praksi, je popularizirana sredinom 19. stoljeća . Istovremeno u Americi nastaje i populariziran je Morseov telegram koji je za slanje poruka koristio Morseov kod – abecedu koja svako slovo zamjenjuje serijom crtica i točkica. Problem kod korištenja telegrama i Morseovog koda za slanje poruka je naravno taj da Morseov kod nije šifra već najobičnija abeceda te kao takav može biti prisluškivan od strane bilo koga. Singh [4] navodi kako je ovaj problem riješen jednostavno: prije slanja poruke Morseovim kodom poruka je šifrirana korištenjem jedne od već poznatih polialfabetskih šifri,najčešće Vignèreovom šifrom, koja je u to vrijeme smatrana neslomljivom te je čak dobila i prikladan nadimak - *le chiffre indéchiffrable*, francuski za „neprobojna šifra“.

Povijest nas uči da ništa ne traje vječno. Isto naravno vrijedi i za takozvanu „neprobojnu šifru“ koja naposlijetku nije bila toliko neprobojna koliko je javnost smatrala. Naime, Vignèreovu šifru je probio Englez Charles Babbage, poznat po svojem doprinosu računarstvu. Kao što je već spomenuto Vignèreova šifra ovisi o ključnoj riječi koja diktira koji od redaka će se koristiti za šifriranje kojeg znaka. Babbage je odlučio ovo iskoristiti kako bi probio šifru na način da je pronalazio ponavljanja u uzorcima šifriranog teksta te bi tako otkrio duljinu ključne riječi. Jednom kada je duljina ključne riječi poznata, Babbage je zaključio da je broj znakova u ključnoj riječi ujedno i broj redova u poliafabetskoj šifri tj. broj monoalfabetskih šifri korištenih za šifriranje. Ovo mu je omogućilo da poruku podijeli u X monoalfabetskih šifri (gdje je X naravno upravo spomenuti broj znakova u ključnoj riječi) te poznatom metodom analiziranja frekvencije slova uspije dobiti ključnu riječ [4]. Nakon toga je proces probijanja svake od monoalfabetskih šifri trivijalan.

2.2.6. Svjetski ratovi

Arthur Zimmermann je bio njemački ministar vanjskih poslova za vrijeme Prvog svjetskog rata. Početkom 1917. je ishod Prvog svjetskog rata još uvijek bio neizvjestan. Zimmermann je, kao i ostatak Njemačkog vodstva, smatrao da će Sjedinjene Američke Države prije ili kasnije ući u rat na stranu saveznika te je stoga odlučio djelovati preventivno i stvoriti tajni savez s Meksikom i Japanom. Plan mu je bio da Meksiku pošalje tajni brzojav u kojem im objašnjava situaciju i obećava novčanu pomoć i teritorije u slučaju ulaska Sjedinjenih Američkih Država u rat. Kako su njemački podmorski kabeli bili uništeni još na početku rata, Zimmermann je bio primoran svoj brzojav slati preko neutralnih švedskih kablova koje su Britanci prisluškivali. Unutar nekoliko sati od njegovog slanja brzojav je djelomično dešifriran i

njegova tajna poruka otkrivena [4]. Razbijanjem Zimmermannovog brzojava je pobuđena inicijativa kriptografa diljem svijeta. Još od razbijanja Vignèreove šifre su razbijači kodova bili dva koraka ispred kriptografa [4], no to će se uskoro promijeniti jer krajem rata u Americi se razvija nova metoda šifriranja. Ova nova metoda će se svojevremeno razviti u modernu metodu koju danas poznajemo kad „One-time pad“ ili „OTP“ metodu tj. metodu „jednokratnog ključa“.

2.3. Vrste kriptografije

Cilj ovog poglavlja je pobliže opisati odabrane kriptografske metode. Bitno je napomenuti razliku između pojmova „šifra“ i „kod“. Naime, iako se često koriste kao sinonimi, šifra i kod nisu jedna te ista stvar. Tanenbaum i Wetherall [2] tvrde kako šifra označava zamjenu znakova po principu znak za znak, tj. jedan znak jasnog teksta se pretvara u jedan znak šifriranog teksta i to bez ikakvog obzira za lingvističku strukturu poruke. U drugu ruku, za kod smatraju da zamjenjuje jednu riječ sa simbolom ili drugom riječi.

2.3.1. Supstitucijske šifre

Supstitucijska šifra je pojam koji obuhvaća sve šifre u kojima se odvija proces supstitucije, tj. zamjene. Slova ili skupina slova iz jasnog teksta se zamjenjuje slovima ili skupinom slova [2] u kriptiranom tekstu po unaprijed određenom pravilu.

Za primjer supstitucijske šifre uzima se Cezarova šifra koja je ujedno jedna od, ako ne i najpopularnija supstitucijska šifra. Cezarova šifra funkcionira na način da se svako slovo abecede zamjenjuje, tj. supstituira slovom koje se nalazi tri mjesta iza njega. Tako bi, na primjeru hrvatske abecede, slovo „a“ postalo slovo „č“, slovo „e“ postalo slovo „h“, slovo „m“ postalo slovo „o“ i tako dalje. Iako je Cezar striktno koristio pomak od 3 slova to naravno ne mora uvijek biti tako već se može koristiti pomak u rasponu od jedan do broja slova u abecedi minus jedan.

Cezarova šifra te njene varijacije s različitih pomacima su na samom dnu što se tiče sigurnosti. Naime one imaju toliko malen raspon ključeva da ih je moguće probiti u nekoliko minuta. Sigurnija varijanta supstitucijske šifre je ona koja ne koristi pomak za određivanje zamjene znakova već nasumično generira znak zamjene za svaki znak abecede. Tako na primjer slovo „a“ može biti bilo koje slovo abecede te slovo „b“ iza njega je potpuno neovisno te također može biti bilo koje slovo abecede (osim onog koje je već zauzeto od strane slova

„a“). Dok Cezarova šifra ima mogućih 25 ido 35 ključeva, ovakva unaprijeđena verzija šifre ima mogućih 25! do 35! ključeva, što je ogromna razlika [2].

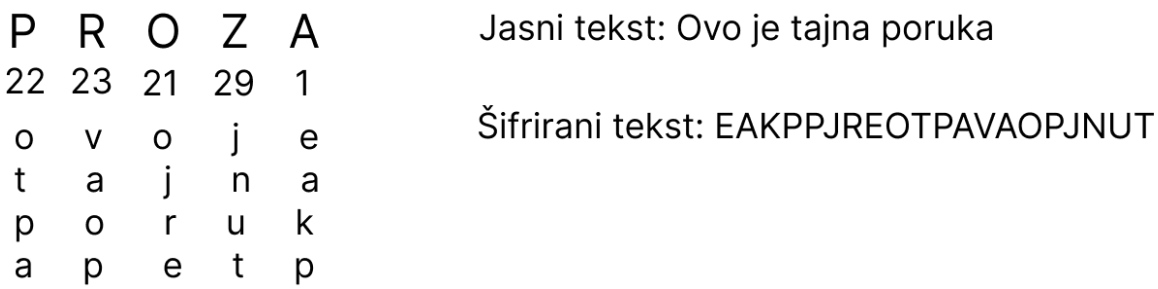
Kada je postalo jasno da dotadašnje supstitucijske šifre ne zadovoljavaju sigurnosne potrebe, pokušalo se preći sa monoalfabetskih šifri na polialfabetske šifre. Monoalfabetske šifre su šifre koje koriste samo jednu abecedu za šifriranje poruka dok su polialfabetske one koje koriste više njih. U monoalfabetskim šiframa jedan znak može biti supstituiran samo jednim drugim znakom, dok u polialfabetskim šiframa jedan znak može biti supstituiran s više znakova – ovisno o tome koliko abeceda se koristi. U pravilu se kod polialfabetskih šifri iterira kroz uzorak korištenja različitih abeceda pa tako na primjer prvo slovo šifriramo prvom abecedom, drugo drugom, treće trećom, četvrto opet prvom, peto drugom, itd.

Iz navedenog se može zaključiti da su supstitucijske šifre jedna od najelementarnijih metoda kriptografije te se stoga, čak i nakon brojnih promjena i percipiranih unaprjeđenja, ne koriste u bilo koje ozbiljne svrhe.

2.3.2. Transpozicijske šifre

Ideja iza transpozicijski šifri je da se tekst ostavi u originalnom, nešifriranom obliku ali da se „transpozicionira“ tj. da mu se promijeni redoslijed te tako sakrije originalno značenje. Primjer transpozicijske šifre je spartanski „skital“ koji je pobliže opisan u poglavlju 2.2.1.

Tanenbaum i Wetherall [2] navode jednostavan primjer transpozicijske šifre uz uporabu ključne riječi ili fraze. Važno je da ključna riječ ili fraza ne sadrži ponavljajuća slova. Tako na primjer riječ „kuća“ može biti ključ ali riječ „papuča“ ne može. Za početak se u prvi red napiše ključna riječ. Nakon toga se svakom slovu ključne riječi pridoda numeričku vrijednost ovisno o tome na kojem mjestu se nalazi u abecedi. Tako na primjer slovo „a“ ima vrijednost 1, slovo „b“ ima vrijednost 2, itd. Nadalje je potrebno ispisati punu poruku u redove ispod ključne riječi te posljednji red ispuniti dodatnim slovima tako da čitava poruka bude ispisana te svaki stupac ima jednak broj znakova u sebi. Sada kreće premještanje slova i to na sljedeći način: uzima se stupac s najmanjom numeričkom vrijednošću i ispisuju se svi znakovi koji se nalaze u tom stupcu jedan iza drugoga. Nakon toga se uzima stupac sa sljedećom najmanjom vrijednošću i postupak se ponavlja sve dok ne ispišemo sve stupce. Na sljedećoj slici (Slika 5) je prikazan primjer ove jednostavne metode transpozicije.



Slika 5: Primjer transpozicijske šifre (Prema: Tanenbaum i Wetherall, 2011)

Kao što je prikazano, ključna riječ je „proza“ te su ispod svakog od njenih slova napisane numeričke vrijednosti. Poruka u jasnom tekstu je tada zapisana u stupce ispod ključne riječi. Na kraju nakon riječi „poruka“ postoji praznina od četiri znaka do kraja retka koju je potrebno popuniti dodatnim slovima ili riječima, u ovom slučaju odabrana je riječ „pet“ te se ona ponavlja dok svi stupci nisu popunjeni. Stupac s najnižom vrijednosti je stupac ispod slova „a“ pa je on prvi ispisan. Nakon toga slijede stupci ispod slova „o“, „p“, „r“ i naposlijetku „z“. Dešifriranje bi se moglo obaviti na način da se izračuna koliko slova se nalazi u svakom stupcu (uz pomoć ključne riječi koja je naravno poznata) te se stupce zapiše na jednak način kao i kod šifriranja.

Transpozicijske šifre ne pružaju veću razinu sigurnosti u usporedbi sa supstitucijskim. Dapače, moglo bi se zaključiti da su dosta nesigurnije od supstitucijskih šifri zbog činjenice da ne skrivaju znakove originalne poruke već ih samo premještaju te se stoga mogu smatrati glorificiranom verzijom dječjih slagalica.

2.3.3. Jednokratni ključ

Nakon prvog svjetskog rata u Americi se pojavila metoda jednokratnog ključa (eng. *One-time pad*). Metoda jednokratnog ključa je prva kriptografska metoda čije probijanje je matematički nemoguće a funkcionira na sljedeći način: jasni tekst se pretvori u niz bitova te se odabere nasumični niz bitova jednake duljine kao onaj koji je nastao pretvaranjem jasnog teksta. Nakon što imamo dva niza bitova jednake duljine potrebno je izvršiti operaciju isključive disjunkcije tj. „XOR“ operaciju nad njima [2]. Dobiveni niz bitova predstavlja šifrirani tekst. U slučaju da šifrirani tekst želimo dešifrirati, potrebno je izvršiti istu operaciju isključive disjunkcije na nizu bitova šifriranog teksta i nizu bitova ključa koji smo koristili za šifriranje. Na slici (Slika 6) je prikazan primjer šifriranja korištenjem jednokratnog ključa.

Jasni tekst:	PAS		
Bitovi poruke:	01010000	01000001	01010011
Bitovi ključa:	01000010	01001111	01001100
Šifrirani bitovi:	00010010	00001110	00011111

Slika 6: Šifriranje metodom jednokratnog ključa (Prema: Singh, 1999)

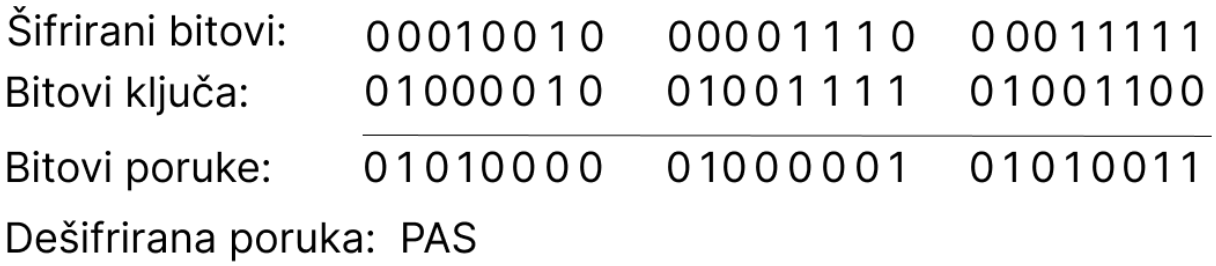
U prikazanom primjeru šifrira se poruka koja sadrži samo jednu riječ – „PAS“. Prvo se tekst pretvara u bitove korištenjem ASCII tablice pa tako slovo „P“ postaje „01010000“, slovo „A“ postaje „01000001“ a slovo „S“ postaje „01010011“. Nakon toga se odabire ključ kojim će jasni tekst biti šifriran. Ključ može biti nasumično generiran ali može biti i smislen. Nakon što je ključ odabran bitovi teksta i bitovi ključa se zapisuju jedni ispod drugih te se obavlja operacija isključive disjunkcije i to na sljedeći način: uzima se prvi bit iz niza bitova poruke te prvi bit iz niza bitova ključa te se oni uspoređuju putem tablice za isključivu disjunkciju (Slika 7).

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Slika 7: Tablica isključive disjunkcije [autorski rad]

U primjeru prva dva bita su „0“ i „0“ te oni isključivom disjunkcijom daju bit „0“. Druga dva bita su „1“ i „1“ te oni također daju „0“. Isto vrijedi i za dva bita na trećem mjestu dok se na četvrtom mjestu nalaze bitovi „1“ i „0“ te oni daju bit „1“. Postupak se ponavlja za ostatak bitova u poruci i ključu.

Dešifriranje kod metode jednokratnog ključa se odvija na gotovo identičan način kao i šifriranje. Jedina razlika je ta da primatelj ima šifrirane bitove i bitove ključa te je u mogućnosti od njih dobiti bitove poruke na način da obavi operaciju isključive disjunkcije između šifriranih bitova i bitova ključa. Dešifriranje poruke je prikazano na sljedećoj slici (Slika 8).



Slika 8: Dešifriranje metodom jednokratnog ključa (Prema: Singh, 1999)

Iako metoda jednokratnog ključa izgleda vrlo korisno sa svojom sto postotnom sigurnošću od probijanja, ona ima i svojih mana. Sam naziv metode već aludira na prvu, prilično veliku manu – jednokratnost ključeva. Naime ključevi se u metodi jednokratnih ključeva smiju koristiti samo jednom ako se želi sačuvati razina sigurnosti koja je obećana. Sljedeća mana metode jednokratnih ključeva je to da jednostavno nije dovoljno praktična za svakodnevno korištenje. Još jedna nepraktičnost korištenja ove metode šifriranja je ta da je ključ dug koliko i poruka te stoga u slučaju slanja velikih poruka i ključ mora biti jednako velik. Ove mane su dovele do toga da se metoda jednokratnog ključa rijetko koristi u svakodnevnoj komunikaciji no razina sigurnosti koju pruža znači da se ipak koristi u nekim specifičnim situacijama. Jedan od najpoznatijih primjera modernog korištenja metode jednokratnog ključa možemo pronaći kod autentifikacije u dva koraka korištene u raznim mobilnim aplikacijama, od kojih su najpoznatije aplikacije za mobilno bankarstvo. Tako praktički svaka osoba koja ima bankovni račun i telefon ili token koristi metodu jednokratnog ključa da bi pristupila svom računu i/ili odobrila plaćanja i slično.

2.3.4. Simetrična kriptografija

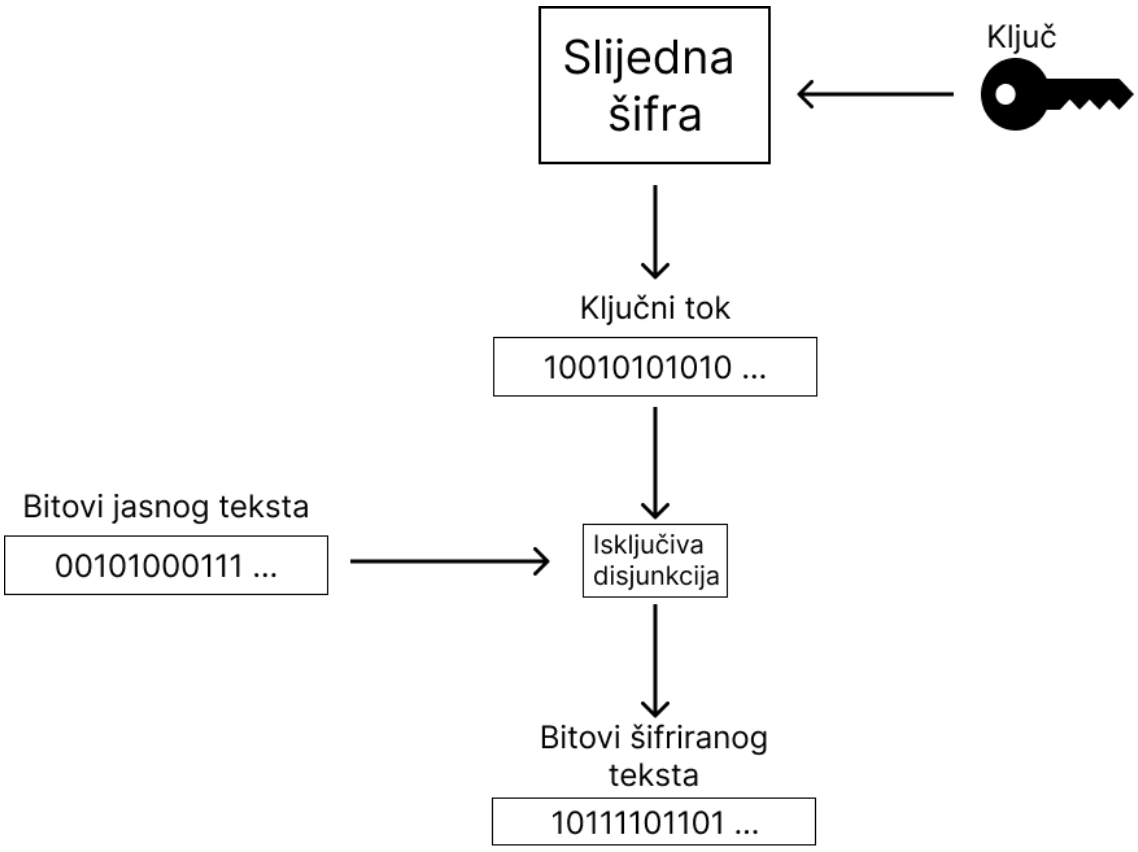
Simetrična kriptografija se naziva „simetričnom“ iz razloga što za šifriranje i dešifriranje koristi isti ključ. Kurose i Ross [3] navode kako postoje dvije različite vrste tehnika simetrične kriptografije: slijedne šifre i blok šifre.

2.3.4.1. Slijedne šifre

Slijedne šifre su donekle slične prethodno pojašnjenim jednokratnim ključevima na nekoliko načina. Prva sličnost se nalazi u operacijama šifriranja i dešifriranja koje se, kao i kod jednokratnih ključeva, obavljaju pomoću operacije isključive disjunkcije i to tako da se disjunkcijom jasnog teksta i ključa dobije šifrirani tekst a disjunkcijom šifriranog teksta i ključa dobije jasni tekst. Sljedeća sličnost je ta da se korištenjem istog ključa nad istim tekstom uvijek dobije isti šifrirani tekst, što je svojstvo koje je moguće iskoristiti u maliciozne svrhe.

Sličnosti između slijednih šifri i jednokratnih ključeva ne staju tu. Dapače, za slijedne šifre bi se moglo reći to da one donekle pokušavaju replicirati jednokratne ključeve do te mjere da je konačan cilj slijedne šifre taj da aproksimira jednokratni ključ. U poglavlju 3.3.3. koje se bavi jednokratnim ključevima je kao jedan od nedostataka jednokratnih ključeva navedena činjenica da oni moraju biti jednake duljine kao i poruka koju šifriraju. Slijedne šifre pokušavaju taj problem riješiti na način da umjesto korištenja dugog ključa, one koriste kratki ključ čiji zadatak je da generira dugi ključ koji se zatim koristi za šifriranje poruke. Neki stvarni primjeri slijednih šifri su A5/1, A5/2, RC4, HC-256, Grain, ChaCha itd. te se zbog svoje jednostavnosti, brzine i činjenice da nisu ograničene na podatkovne blokove fiksne duljine, što je slučaj kod blok šifri, koriste i u moderno doba.

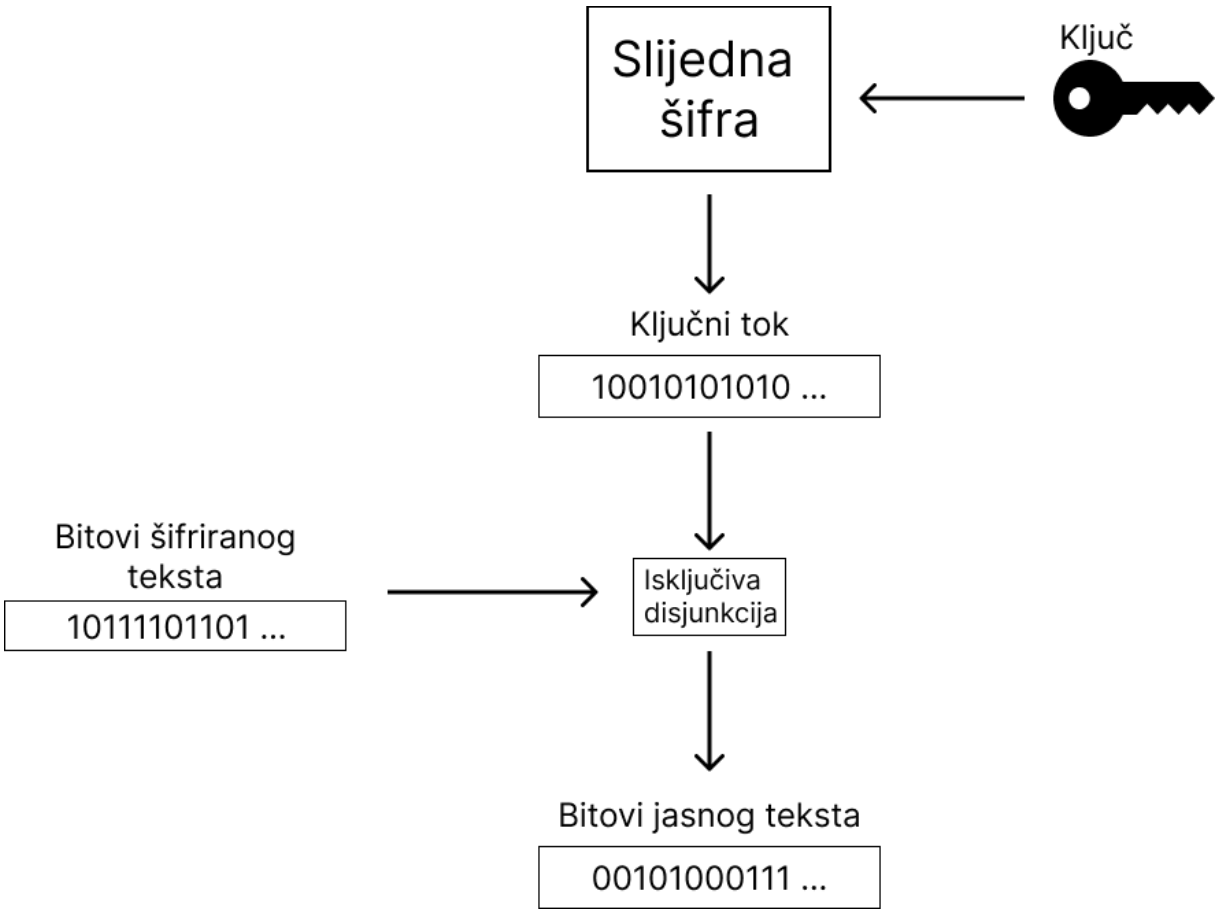
Slijedne šifre za svrhu šifriranja poruka ne koriste direktno ključ, već uz pomoć ključa generiraju tzv. „ključni tok“ – veliku sekvencu izlaznih bitova [2] koja nije fiksne duljine već se generira sve dok ne pokrije duljinu jasnog teksta kojeg je potrebno šifrirati. Kada se dobije ključni tok postupak je identičan postupku kod šifriranja jednokratnim ključem – isključivom disjunkcijom se dobije šifrirani tekst. Proces šifriranja slijednom šifrom je prikazan na sljedećoj slici (Slika 9).



Slika 9: Šifriranje slijednom šifrom [autorski rad]

Proces dešifriranja je gotovo identičan procesu šifriranja. S obzirom da se radi o metodi simetrične kriptografije, primatelj ima identičan ključ kao i pošiljatelj no on naravno nema

jasan tekst već ima šifrirani tekst koji je primio od pošiljatelja. Primatelj uz pomoć svog ključa algoritmom slijedne šifre generira ključni tok koji je identičan ključnom toku generiranom od strane pošiljatelja pri šifriranju poruke. Zaključujemo da jedan te isti ključ generira jedan te isti ključni tok. Nakon generiranja ključnog toka pošiljatelj obavlja operaciju isključive disjunkcije između ključnog toka i bitova šifriranog teksta kojeg je primio od pošiljatelja. Rezultat operacije isključive disjunkcije su bitovi jasnog teksta. Opisani proces je prikazan na sljedećoj slici (Slika 10).



Slika 10: Dešifriranje slijednom šifrom [autorski rad]

2.3.4.2. Blok šifre

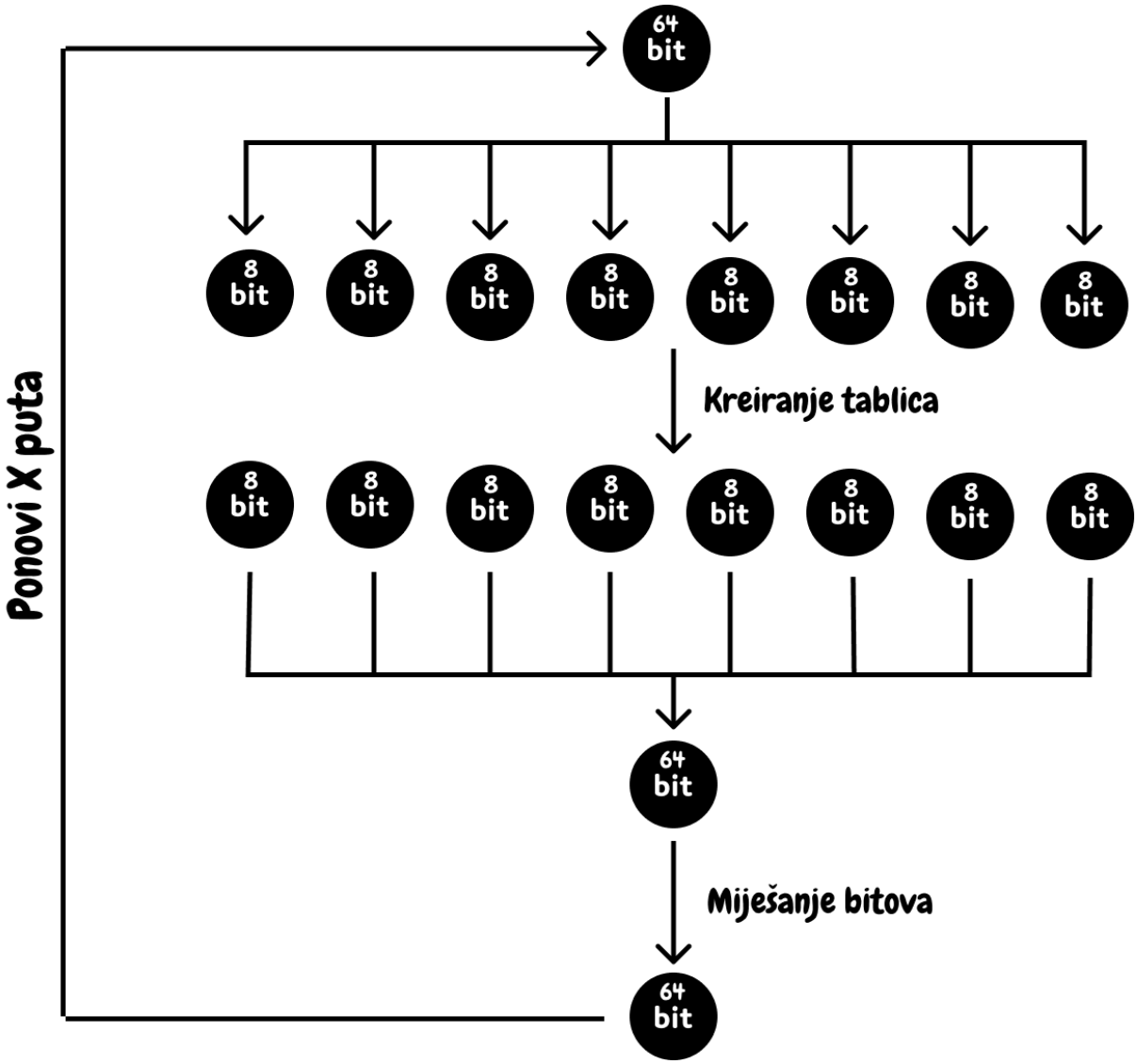
Druga vrsta tehnike u simetričnoj kriptografiji se naziva „blok šifra“. Kao što naziv sugerira, blok šifra šifrira podatke u blokovima fiksne veličine. Fiksna veličina blokova podataka kao posljedicu ima to da je nemoguće šifrirati količinu podataka koja je manja ili veća od veličine bloka koji se koristi. U slučaju da je veličina bloka podataka manja obavlja se postupak „punjenja“ (eng. *padding*) bloka s nasumičnim podacima dok se u slučaju da je

veličina bloka podataka veća podaci moraju rascjepkati u blokove propisne veličine. Ključevi kod blok šifri također moraju biti fiksne veličine. Bitno je napomenuti da veći ključevi i blokovi kod blok šifri znače bolju sigurnost no ujedno i više vremena koje potrebnog za šifriranje i dešifriranje te korisnik stoga mora dobro razmisliti što mu je važnije.

Blok šifra koristi preslikavanje jedan na jedan između blokova jasnog teksta i blokova šifriranog teksta [3], što znači da svaki ulazni blok ima točno jedan pripadajući izlazni blok te da jedan te isti ulazni tekst, uz korištenje jednog te istog ključa, uvijek daje jedan te isti šifrirani tekst. Parovi blokova jasnog teksta i šifriranog teksta su pohranjeni u tablici te postaje jasno da su blok šifre zapravo varijacija na temu supstitucijskih šifri. Dok se u jednostavnim supstitucijskim šiframa svaki znak abecede zamjenjuje drugim znakom, kod blok šifri se svaki blok zamjenjuje nekim drugim blokom podataka tj. bitova – jedinica i nula. Kurose i Ross [3] tako zaključuju da šifra čija je veličina bloka jednaka 3, tj. čiji blokovi se sastoje od skupina od 3 bita, ima pripadajuću supstitucijsku tablicu od osam unosa. Do ovog podatka dolaze sljedećom formulom:

$$X = 2^Y$$

gdje je X broj zapisa u supstitucijskoj tablici a Y veličina bloka podataka. Kurose i Ross [3] također dolaze od zaključka da je takvu tablicu moguće rasporediti na $Y!$ načina gdje je Y i dalje veličina bloka podataka, te da je svaki od tih rasporeda tablice zapravo jedinstveni ključ blok šifre. Tako blok šifra s veličinom bloka od 6 bitova ima $2^6 = 64$ jedinstvena ključa dok blok šifra s veličinom bloka od 64 bita ima $2^{64} = 18{,}446{,}744{,}073{,}709{,}551{,}616$ (oko osamnaest i pol trilijuna) ključeva. Uzevši u obzir da moderne blok šifre imaju blokove od 64, 128 ili više bitova, jasno je da tablica sa tolikim brojem unosa jednostavno fizički nije moguća te da je potrebno drugo rješenje. Stoga blok šifre ne koriste čitave tablice sa svim unosima već uz pomoć algoritama simuliraju velike tablice. Kurose i Ross [3] iznose primjer jednog algoritma koji simulira nasumično generirane tablice i to na sljedeći način: algoritam prvo razbija blok od 64 bita na 8 manjih blokova. Zatim algoritam procesira manje blokove tablicama gdje, prema već utvrđenim pravilima, svaka tablica ima $2^8 = 256$ unosa. Nakon toga su manji blokovi ponovno spojeni u blok veličine 64 bita koji zatim prolazi proces „miješanja“ svojih bitova tj. zamjene njihovih mjesta kao kod transpozicijskih šifri. Proces sada može završiti te se u tom slučaju ključ sastoji od onih 8 tablica koje sadrže unose za 8 manjih blokova, no to ne mora biti tako već se dobiveni „izmiješani“ blok može ponovno provesti kroz čitav proces. Svaki od ovih ciklusa prolaženja kroz navedeni proces se naziva „runda“ te svaka dodatna runda dodatno otežava probijanje poruke. Opisani postupak je ilustriran na sljedećoj slici (Slika 11).

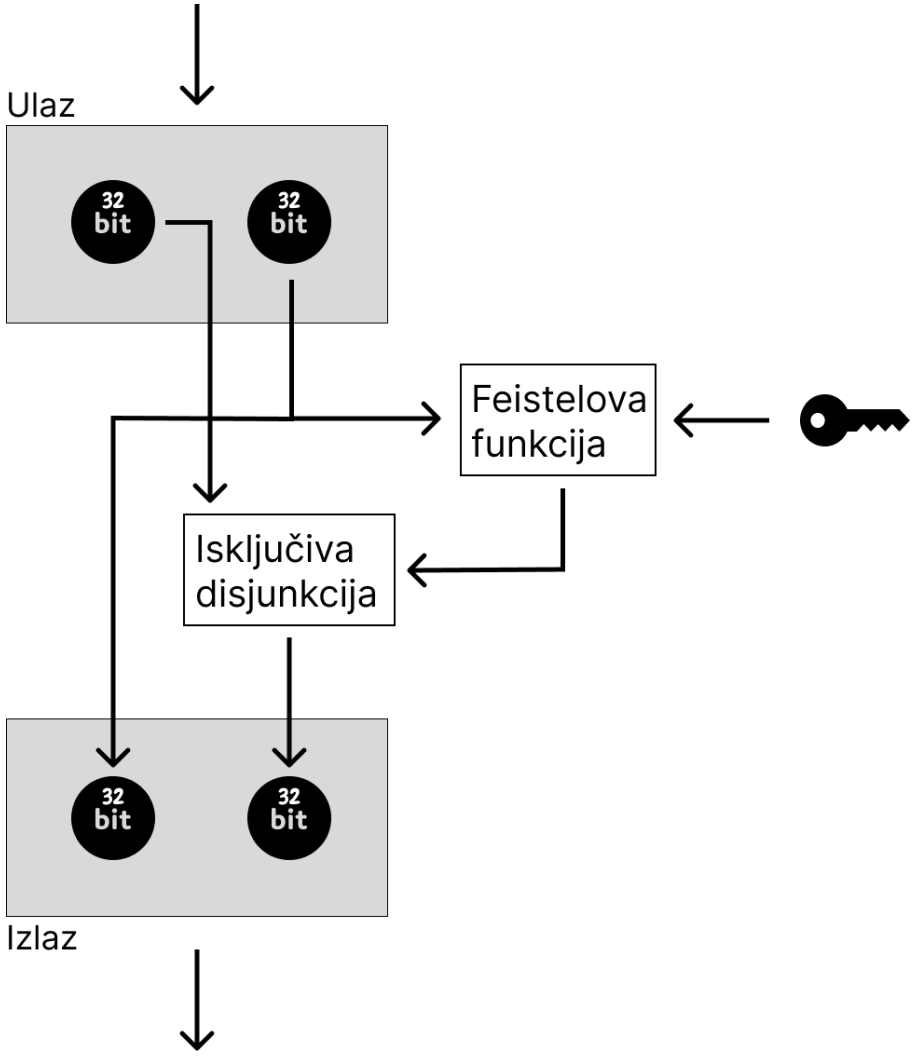


Slika 11: Primjer algoritma za simulaciju tablice blok šifre (Prema: Kurose i Ross, 2007)

Blok šifre su neke od najpopularnijih šifri u uporabi te se koriste u čitavoj industriji, uključujući i za potrebe mnogih svjetskih vlada. Neke od najpoznatijih blok šifri su DES, 3DES i AES. DES je šifra koja je u javnost puštena 1976. godine od strane IBM-a. Američkoj vladi nije dugo trebalo da prepozna vrijednost DES-a te su ga već u siječnju 1977. preuzeli kao službeni standard za svoje podatke koji nisu povjerljivi a ostatak industrije je ubrzo slijedio njihov primjer [2].

DES je šifra koja radi s blokovima podataka veličine 64 bita i ključevima veličine 56 bitova. Tanenbaum i Wetherall [2] iznose kako je inicijalna veličina ključa DES-a bila 128 bitova, što je sličnije modernim šiframa, ali je IBM odlučio smanjiti ključ na veličinu od samo 56 bitova. Ovaj potez, skupa s činjenicom da IBM prvotno nije objavio dizajn DES-a, je mnoge ljude nagnao na vjerovanje kako je američka vlada zatražila da IBM unazadi vlastitu šifru kako bi je doveli u stanje u kojem samo američka vlada ima dovoljno resursa da razbije šifru.

DES šifrira podatke korištenjem procesa od čak 19 koraka. Prvi korak DES-a je korak jednostavne transpozicije jasnog teksta dok je posljednji korak zrcalno suprotan. Korak prije posljednjeg je jednostavna zamjena prvih 32 bita i zadnjih 32 bita [2]. Iako DES prima blok podataka veličine 64 bita, on u šifriranju uglavnom koristi dva manja bloka od 32 bita. Tanenbaum i Wetherall [2] za ostalih 16 koraka tvrde da su jednaki u funkcionalnosti ali različiti u parametrima. Svaki od ovih koraka na ulazi prima dva 32-bitna bloka podataka te ih obrađuje i na izlaz šalje dva 32-bitna bloka podataka. Desni ulazni blok se kopira i postaje lijevi izlazni blok bez ikakvih promjena ali se također šalje u funkciju skupa s pripadajućim ključem za odabrani korak. Ovu funkciju je dizajnirao Horst Feistel, jedan od članova IBM-ovog tima koji je razvio DES. Bitno je napomenuti da funkcija za ulaz uzima desni blok podataka te ključ koji nije identičan glavnom ključu ali je izveden iz njega i koji je različit za svaki od 16 koraka. Izlaz iz funkcije se zatim podliježe operaciji isključive disjunkcije skupa s lijevim ulaznim blokom podataka te rezultat postaje desni izlazni blok podataka. Ovaj proces je ilustriran na sljedećoj slici (Slika 12).



Slika 12: Jedan od 16 ponavljajućih koraka DES-a (Prema: Tanenbaum i Wetherall, 2011)

Čitava kompleksnost algoritma DES-a se nalazi u Feistelovoj funkciji te je to skupa s činjenicom da je ključ maksimalne veličine samo 56 bitova dovelo do toga da je DES ubrzo nakon početka korištenja razbijen. Već 1977. par znanstvenika sa Stanforda su dizajnirali stroj čija svrha je razbijanje DES-a te su ocijenili da je stroj u stanju napadnom grube sile probiti DES šifru u samo jedan dan i da je za njegovu izgradnju potrebno „samo“ 20 milijuna američkih dolara. Tri godine nakon prvog razvijanja DES-a već je bilo jasno da je relativno malena veličina ključa ogroman problem te da algoritam neće ostati siguran duže vremena pa stoga IBM razvija rješenje – Triple DES [2].

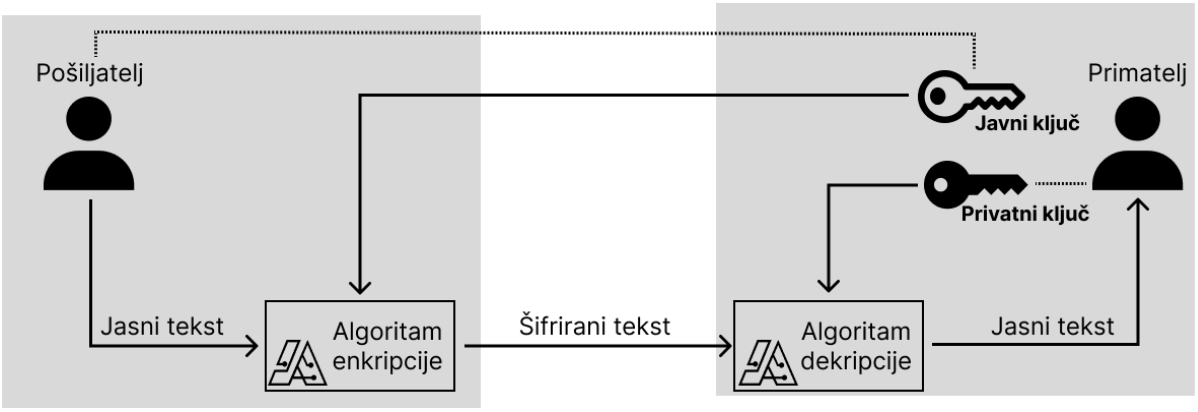
„Triple DES“ ili skraćeno „3DES“ je IBM-ovo „flaster“ rješenje za problem nastao kao posljedica smanjivanja veličine ključa DES-a. Kao što naziv sugerira, 3DES je u biti postupak trostruke enkripcije tj. ponavljanje DES-a tri puta da bi se povećala sigurnost i to na način da se poruka šifrira DES-om, dešifrira DES-om i zatim ponovno šifrira DES-om. U postupku se koriste dva ključa, jedan od kojih služi za dva šifriranja a drugi za dešifriranje između njih. Tanenbaum i Wetherall [2] postavljaju i daju odgovore na dva bitna pitanja: Zašto koristimo samo dva ključa a ne tri te zašto koristimo šifriranje, dešifriranje pa opet šifriranje a ne tri šifriranja? Odgovor na prvo pitanje, vezano za broj ključeva, je relativno jednostavan – redundantnost. Većina kriptografa vjeruje da su dva ključa od 56 bitova više nego dovoljna za svakodnevno komercijalno korištenje te da bi dodavanje još jednog 56-bitnog ključa samo zakompliciralo čitav proces prenošenja i upravljanja ključem koji bi u tom slučaju bio veličine 168 bitova. Drugo pitanje je pak dosta zanimljivije i na prvu pomisao ne predstavlja očit odgovor. Naime, razlog za korištenje „EDE“ (*„Encrypt Decrypt Encrypt“*) principa umjesto „EEE“ (*„Encrypt Encrypt Encrypt“*) principa je kompatibilnost s ranijim verzijama, tj. „običnim“ DES-om. Već smo spomenuli da se 3DES u biti sastoji od dva šifriranja i jednog dešifriranja običnim DES-om uz dva ključa – jedan za šifriranja a drugi za dešifriranje. Prema tome možemo zaključiti da ako koristimo isti ključ za šifriranje i dešifriranje, prva dva koraka 3DES-a se poništavaju i ostaje samo posljednji korak – šifriranje. Tako 3DES korištenjem jednog ključa umjesto dva različita ključa u biti postaje najobičniji DES. Ovo svojstvo je pomoglo postepenom uvođenju 3DES-a kao zamjeni za DES što je stavka uvelike važna industriji [2].

Nakon dvadesetak godina intenzivnog korištenja postalo je evidentno da s prelaskom u 21. stoljeće DES više nije dovoljno siguran standard i da je potrebna promjena. Ubrzo će ga kao standard moderne kriptografije zamijeniti algoritam danas poznat kao AES – Advanced Encryption Standard.

2.3.5. Asimetrična kriptografija

Sve do sada spomenute kriptografske metode imaju jednu stvar zajedničku – identične ključeve pošiljatelja i primatelja tajne poruke. Ovaj pristup šifriranja gdje pošiljatelj poruku šifrira ključem a primatelj poruku dešifrira istim ključem na prvi pogled i ima najviše smisla jer je postupak dešifriranja u biti samo obrnuti postupak šifriranja. Kurose i Ross [3] pak navode jednu veliku manu korištenja istog ključa u obliku jednostavnog pitanja: Kako razmijeniti ključ potreban za uspostavljanje sigurne komunikacije ako već nemamo siguran komunikacijski kanal? U prošlosti je ovo možda bio manji problem jer su osobe koje su razmjenjivale tajnu komunikaciju vjerojatno bile jako bliske te su se mogli sastati lice u lice i tako razmijeniti ključeve. Nažalost u modernom digitalnom svijetu to više nije uvijek moguće već postoje situacije gdje individualci koji se nikad prije nisu upoznali žele komunicirati sigurnim kanalom.

Rješenje ovog problema su prvo predstavili Diffie i Hellman 1976. godine [3] i ono je sljedeće: Umjesto korištenja dva jednaka ključa na strani primatelja i pošiljatelja, primatelj poruke ima dva vlastita ključa – jedan javni i jedan privatni. Privatni ključ primatelja je poznat samo njemu dok je javni ključ poznat svima. Pošiljatelj uzima javni ključ primatelja i šifrira poruku njime te je šalje primatelju koji je zatim dešifrira korištenjem svoj privatnog ključa. Javni ključ ne može dešifrirati poruku koju je šifrirao javni ključ već takvu poruku može dešifrirati samo privatni ključ. Postupak se naziva asimetričnim jer javni i privatni ključevi nisu identični kao što je to slučaj kod simetrične kriptografije i prikazan je na sljedećoj slici (Slika 13).



Slika 13: Postupak asimetrične kriptografije (Prema: Kurose i Ross, 2007)

Postupak naravno funkcionira i u obratu. U slučaju da privatni ključ šifrira poruku, javni ključ je može dešifrirati. Na prvu ovo svojstvo ne zvuči toliko korisno jer ovakva komunikacija uopće nije sigurna s obzirom da bilo tko s javnim ključem može dešifrirati poruku a javni ključ je dostupan baš svima, no tajnost komunikacije nije ni svrha šifriranja privatnim ključem. Uzmimo u obzir sljedeću situaciju: Osoba A želi osobi B poslati poruku. Osoba A uzima javni ključ osobe B i šifrira poruku te je šalje osobi B. Osoba B prima poruku te je dešifrira svojim

privatnim ključem. Do sada je sve u redu i komunikacija na prvu izgleda sasvim sigurno ali kako osoba B može znati da je ta poruka došla baš od osobe A? Potpis u tekstu poruke kaže da je osoba A napisala poruku ali što ako to nije istina? Što ako je osoba C presrela poruku osobe A te ubacila svoju poruku u kojoj se potpisala kao osoba A? Tu na scenu stupa gore spomenuta mogućnost šifriranja privatnim ključem i dešifriranja javnim ključem. Naime ovaj proces može poslužiti kao svojevrstan „potpis“ poruke. U slučaju da osoba A šifrira svoju poruku svojim privatnim ključem a zatim i javnim ključem osobe B te je takvu pošalje osobi B, osoba B onda može poruku dešifrirati prvo svojim privatnim ključem a zatim i javnim ključem osobe A te tako može biti 100% sigurna da je poruka došla baš od osobe A jer ona jedina ima pristup svom privatnom ključu.

Kurose i Ross [3] tvrde kako postoji više algoritama koji su građeni po principu asimetrične kriptografije no jedan od njih je praktički postao sinonim za asimetričnu kriptografiju. Algoritam u pitanju je „RSA“, razvijen 1978. na M.I.T.-u [2] ime je dobio po inicijalima svojih kreatora – Ronu Rivestu, Adiju Shamiru i Leonardu Adlemanu. Tanenbaum i Wetherall [2] govore kako se RSA temelji na teoriji brojeva te ukratko opisuju proces šifriranja i dešifriranja. Prije šifriranja i dešifriranja je potrebno izračunati javni i privatni ključ i to na sljedeći način: odabiru se dva velika prosta broja „p“ i „q“, najčešće veličine 1024 bita. Izračunava se $n = p \times q$ i $z = (p-1) \times (q-1)$. Odabire se broj „d“ koji je relativno prost prema „z“ i pronalazi se broj „e“ tako da vrijedi $e \times d = 1 \, mod \, z$. Javni ključ je par (e, n) a privatni ključ je par (d, n). Proces šifriranja se obavlja prema sljedećoj formuli:

$$C = P^e \ (mod \, n)$$

gdje je C šifrirani tekst, P jasni tekst i gdje su (e, n) parametri javnog ključa. Proces dešifriranja se obavlja prema sljedećoj formuli:

$$P = C^d \ (mod \, n)$$

gdje je P jasni tekst, C šifrirani tekst i gdje su (d, n) parametri privatnog ključa [2].

Zbog činjenice da ključevi kod algoritma RSA dosežu prilično velik broj bitova jasno je da vrijeme potrebno za izvršavanje šifriranja i dešifriranja također raste u usporedbi s nekim drugim kriptografskim metodama. Kurose i Ross [3] navode kako se RSA stoga često koristi u kombinaciji sa simetričnom kriptografijom i to na način da osobe uspostave sigurnu komunikaciju korištenjem RSA, a zatim nastave komunicirati putem algoritama simetrične kriptografije poput DES-a ili AES-a koji su nekoliko puta brži.

2.3.6. AES

Krajem prošlog stoljeća javnost je uvidjela kako tadašnji standardni algoritmi nisu dovoljno sigurni za daljnju uporabu zbog broja faktora poput malene veličine ključa, netransparentnosti kreatora, sumnji na ugrađene zakulisne načine za neovlašteni ulazak, itd. Zbog navedenih razloga javnost je sve manje i manje koristila tadašnji standard – DES, te se pojavila potreba za razvojem novog standarda [2].

U siječnju 1997. godine, NIST je odlučio povući dotad neviđeni potez. Odlučili su sponzorirati potpuno javno natjecanje kriptografskih algoritama u nadi da će pobjednik postati novi industrijski standard, naziva „Advanced Encryption Standard“ skraćeno „AES“ [2]. U pokušaju sprječavanja nastajanja problema koji su mučili DES, NIST je postavio nekoliko pravila čija svrha je bila osiguravanje transparentnosti i kompatibilnosti novog algoritma. Tanenbaum i Wetherall [2] ih navode:

- Algoritam mora biti simetrična blok šifra
- Dizajn algoritma mora biti u potpunosti objavljen za javnost
- Algoritam mora podržavati veličine ključa od 128, 192 i 256 bitova
- Algoritam mora podržavati softversku i hardversku implementaciju
- Algoritam mora biti javan ili licenciran korisnicima na nediskriminirajućoj osnovi

Natječaj je trajao tri i pol godine te se sastojao od više rundi. Prvotno je zaprimljeno petnaest prijava od kojih je pet ušlo u finalni odabir. Naposlijetku je kao pobjednik odabran algoritam „Rijndael“, kreiran od strane Vincenta Rijmena i Joana Daemena. Naziv algoritma je izveden iz njihovih prezimena: „Rij(men) + Dae(men)“ [2].

2.3.6.1. Rijndael

Za razliku od gotovo svih do sada spomenutih algoritama koji su podacima baratali u obliku niza bitova, Rijndael taj niz bitova pretvara u matricu od 4x4 bajtova. Svaki bajt naravno sadrži 8 bitova pa tako Rijndael barata sa $4 \times 4 \times 8 = 128$ bitova. Rijndael sam po sebi može imati veličinu bloka veću od 128 bitova, sve do 256 bitova, no AES standard definira veličinu bloka kao 128 bitova [2] pa se držimo toga. Veličina ključa po standardu može biti 128, 192 ili 256 bitova no praktična potreba za korištenjem ičeg većeg od 128 bitova gotovo ni ne postoji pa industrija stoga gotovo isključivo koristi blokove i ključeve veličine 128 bitova, što je više nego duplo veće od ključa kojeg koristi DES. Snaga Rijndael algoritma dakle leži u veličini njegovog ključa.

Kada rastavimo cjeloviti postupak na proste faktore možemo vidjeti da Rijndael nije uveo ništa revolucionarno i do sada neviđeno u postupak šifriranja i dešifriranja. Dapače, Rijndael koristi sada već prastare metode poput supstitucije i transformacije te naravno –

isključivu disjunkciju tj. „XOR“ operaciju. Algoritam ima, kao i sve ostale blok šifre, više rundi kroz koje provlači podatke da bi povećao razinu sigurnosti. Broj rundi ovisi o veličini ključa pa tako veličina ključa od 128 bitova uvjetuje 10 rundi, 192 bita uvjetuju 12 rundi a maksimalnih 256 bitova znači 14 rundi. Tanenbaum i Wetherall [2] opisuju algoritam na sljedeći način:

Prvi korak Rijndael algoritma se ne dotiče podataka već samog ključa koji će se koristiti u šifriranju. Još jedan od razloga zašto je Rijndael toliko siguran je postupak tzv. „proširivanja“ ključeva u kojem Rijndael uzima ključ od 128 bitova te uz pomoć isključive disjunkcije i rotacije bitova od njega stvara 11 novih ključeva duljine 128 bitova. Jedan od tih ključeva se koristi prije početka kalkulacija dok se ostalih deset koriste svaki po jednoj rundi.

U sljedećem koraku Rijndael kopira 128-bitni blok podataka jasnog teksta u blok koji Tanenbaum i Wetherall [2] nazivaju „stanje“. Blok stanja je strukturom identičan svim drugim blokovima u algoritmu po tome što je on 4x4 matrica bajtova. Sljedeća slika prikazuje jedan blok u Rijndael algoritmu (Slika 14).

B1	B5	B9	B13
B2	B6	B10	B14
B3	B7	B11	B15
B4	B8	B12	B16

Slika 14: Blok u Rijndael algoritmu (Prema: Daemen i Rijmen, 2007)

B1 do B16 označavaju 16 bajtova koji se nalaze u jednom 128-bitnom bloku. Jasni tekst se u blok stanja prepisuje na način koji je prikazan na slici: tako da prva četiri bajta ulaze u prvi stupac, druga četiri u drugi i tako dalje. Nakon što je jasni tekst prepisan u blok stanja, algoritam uzima prvi od 11 ključeva nastalih od glavnog ključa te obavlja operaciju isključive disjunkcije između tog ključa i bloka stanja. Rezultat isključive disjunkcije postaje novi blok stanja. Algoritam je sada spreman izvršiti spomenutih deset rundi.

Prvi korak svake od rundi se sastoji od već poznate supstitucije, kao kod starih monoalfabetskih šifri. Jedan bajt bloka stanja se supstituira s drugim bajtom bloka stanja uz poštivanje nekih posebnih pravila čija svrha je kompliciranje čitavog procesa, na primjer: svaki bajt uvijek mora biti promijenjen i bajt ne može biti promijenjen na način da se vrijednosti bitova jednostavno obrnu.

Drugi korak rotira svaki red ulijevo i to na sljedeći način: prvi red se okreće 0 mjesta ulijevo tj. ostaje na mjestu, drugi red se okreće 1 mjesto ulijevo, treći 2 mjesta ulijevo a četvrti tri mjesta ulijevo. Sljedeća slika prikazuje stanje bajtova na kraju drugog koraka (Slika 15).

B1	B5	B9	B13
B6	B10	B14	B2
B11	B15	B3	B7
B16	B4	B8	B12

Slika 15: Blok stanja nakon drugog koraka algoritma [autorski rad]

Treći korak koristi matematički koncept konačnih polja kako bi vektorskim množenjem matrice izmiješao svaki od stupaca bloka stanja neovisno o ostalim stupcima. Iako na prvu zvuči komplicirano, Daemen i Rijmen [5] tvrde kako se ovaj korak izvršava uz pomoć nekoliko elementarnih supstitucijskih tablica i operacija isključive disjunkcije.

Četvrti i posljednji korak je najjednostavniji. Potrebno je izvršiti operaciju isključive disjunkcije između ključa pojedine runde i bloka stanja. Prisjetimo se, ključ unesen kao parametar za šifriranje ili dešifriranje se na početku procesa proširuje na 11 ključeva. Krajem četvrtog koraka završava i proces šifriranja. Proces dešifriranja se obavlja jednostavnim obratom svih navedenih koraka [5].

2.3.6.2. Metoda lančanih blokova

Tanenbaum i Wetherall [2] napominju kako čak i uz svu njegovu kompleksnost i sigurnost, AES je naposlijetku i dalje najobičnija supstitucijska šifra koja umjesto individualnih bitova koristi čitave blokove. Ovo znači da za isti ulaz AES uvijek daje isti izlaz što se može protumačiti kao slabost i iskoristiti kao osnova za napad. Jedan mogući način na koji ovakav šifrirani tekst može biti napadnut je jednostavnom zamjenom blokova, bila ona ciljana ili potpuno nasumična tekst svakako gubi smisao.

Jedan od načina na koji se ovakav napad može spriječiti je korištenjem metode lančanih blokova. Metoda lančanih blokova je relativno jednostavna: prije nego što se šifrira,

nad svakim blokom jasnog teksta se provodi operacija isključive disjunkcije s prethodnim blokom šifriranog teksta. Ovo kao posljedicu ima to da ako netko zamjeni dva bloka, dešifriranje više neće biti moguće već će se na mjestu zamijenjenog bloka nalaziti hrpa nesmislenih podataka. Povlači se pitanje: Ako svaki blok prije šifriranja mora proći isključivu disjunkciju sa šifriranim prethodnim blokom, kako onda šifriramo prvi blok? Prvi blok naravno ispred sebe nema niti jedan blok pa stoga on isključivu disjunkciju obavlja s nasumično odabranim blokom podataka kojeg nazivamo Inicijacijski vektor. Inicijacijski vektor se šalje u obliku jasnog teksta skupa s šifriranim tekstom, najčešće umetanjem na sam početak poruke tako da ga je lakše pronaći kod dešifriranja.

Osim prednosti zaštite protiv navedene vrste napada, metoda lančanih blokova također kao posljedicu ima to da jedan te isti blok jasnog teksta ne daje uvijek jedan te isti blok šifriranog teksta što dalje otežava kriptoanalizu [2].

3. Napadi na kriptografiju

Glavni razlog postojanja kriptografije je želja individualaca za presretanje komunikacije te korištenje informacija u maliciozne svrhe. Bilo kakav pokušaj čitanja šifrirane poruke može se smatrati napadom na kriptografiju.

Postoji nekoliko različitih podjela napada na kriptografiju. Tako na primjer razlikujemo direktne i indirektne napade. Direktni napadi su oni napadi koji ciljaju direktno na šifru dok indirektni napadi, još poznati i kao *„side-channel“* napadi, ciljaju na implementaciju i okolinu same šifre.

Nadalje, napade možemo dijeliti na pasive i aktivne napade. Pasivni napadi su oni napadi gdje napadač ne pokušava aktivno promijeniti ili oštetiti sadržaj šifrirane poruke te su samim time manje opasni od aktivnih napada. Neki od primjera pasivnih napada su prisluškivanje i njuškanje podataka. Aktivni napadi u drugu ruku označavaju napade gdje se napadač aktivno trudi presresti, probiti i izmijeniti šifriranu poruku. Ostatak ovog poglavlja govori isključivo o aktivnim napadima.

3.1. Napadi grubom silom

Kad osobi u ruke stavite snop ključeva na lancu i stavite ih pred vrata te im naredite da prođu kroz ta vrata, prva stvar koju će velika većina ljudi napraviti je ta da će redom isprobavati jedan po jedan ključ sa snopa te tako pokušati otkriti koji od ključeva otključava vrata. Isto je s kriptografijom. Najintuitivniji način probijanja nekakve šifre je taj da jednostavno isprobamo

svaki ključ iz domene za taj algoritam. Ovaj postupak se još naziva i napad grubom silom tj. „brute-force“ napad.

Napadi grubom silom se u moderno doba smatraju naj elementarnijom, primitivnijom ili „glupljom“ metodom napada na kriptografiju no kroz povijest su svakako imali svojih primjena. Monoalfabetske supstitucijske šifre su na primjer jako podložne razbijanju napadima grubom silom zbog činjenice da su njihove domene ključeva iznimno malene. Cezarova šifra na primjer ima samo 20-ak do 30-ak mogućih ključeva te je ovakvim napadom moguće razbiti unutar nekoliko minuta čak i bez uporabe ikakvih računala.

Glavna zaštita koju šifre imaju protiv ovakvog napada je naravno veličina same domene ključeva tj. broj mogućih ključeva. Tako su šifre koje koriste veće ključeve sigurnije od šifri koje koriste ključeve iz manje domene. Uzmimo za primjer ranije spomenute algoritme simetrične kriptografije – DES i AES. DES koristi ključeve velike 56 bita što znači da u njegovoj domeni postoji 2^{56} ključeva, što evidentno nije bilo dovoljno za potrebe industrije te je stoga uveden novi standard – AES koji koristi ključeve veličine 128 bita i više pa se njegova domena sastoji od 2^{128} ključeva ili više.

Moderne kriptografske metode su više-manje sigurno od čistog napada grubom silom no bitno je napomenuti kako napadi grubom silom gotovo nikad nisu korišteni samostalno već su oni samo jedan od alata u arsenalu kriptoanalitičara.

3.2. Kriptoanaliza

Singh [4] kriptoanalizu definira kao znanost koja se bavi dešifriranjem tajnih poruka bez poznavanja ključa. U poglavlju o povijesti kriptografije smo spomenuli da se kriptoanaliza prvo javlja u osmom stoljeću na Bliskom istoku gdje su je razvili Arapski učenjaci.
Kurose i Ross [3] definiraju nekoliko različitih vrsta napada na kriptografiju kojima se služe kriptoanalitičari te ih razlikuju po količini informacija koju kriptoanalitičar poznaje, a to su napadi isključivo šifriranim tekstom, napadi poznatim jasnim tekstom i napadi odabranim jasnim tekstom.

3.2.1. Napad isključivo šifriranim tekstom

Napad isključivo šifriranim tekstom (eng. *Ciphertext-only attack - COA*) je vrsta napada na kriptografiju u kojem kriptoanalitičar poznaje samo šifrirani tekst. Bitno je napomenuti da se podrazumjeva da kod svih napada kriptoanalitičar također poznaje i algoritam korišten u kriptiranju. Kod ove vrste napada kriptoanalitičar ne može birati točno

koji dio šifriranog teksta poznaje već mu je on nasumično dodijeljen. Napadač na primjer prisluškuje komunikacijski kanal te uspije presresti šifriranu poruku te istu pokušava dešifrirati bez ključa. Ako ova vrsta napada zvuči teško za izvesti, to je zato što i je! Napad uz poznavanje samo šifriranog teksta je najzahtjevnija vrsta napada i šifra koja može biti probijena ovakvim napadom definitivno nije sigurna šifra.

3.2.2. Napad poznatim jasnim tekstom

Napad poznatim jasnim tekstom (eng. *Known-plaintext attack - KPA*) je vrsta napada na kriptografiju u kojem kriptoanalitičar poznaje šifrirani tekst te nekoliko parova šifriranog i jasnog teksta. Preciznije, napadač posjeduje čitavu šifriranu poruku, kao i kod napada isključivo šifriranim tekstom (eng. *Ciphertext-only attack – CPA*) no u ovom slučaju napadač posjeduje i nekoliko isječaka jasnog teksta te zna točno gdje se oni nalaze u šifriranom tekstu. Napadaču je u cilju analizom parova šifriranog i jasnog teksta otkriti koji je ključ korišten pri šifriranju te pomoću toga dešifrirati čitavu poruku.

Jedan dobro poznat primjer ovakvog napada je savezničko razbijanje njemačke „Engima“ šifre u drugom svjetskom ratu. Nakon neuspjeha njihove kriptografije u Prvom svjetskom ratu, Nijemci su odlučili ozbiljnije shvatiti kriptografiju u sljedećem sukobu te su stoga svaki dan mijenjali ključ korišten za šifriranje poruka. Engleski kriptograf Alan Turing je nakon nekog vremena primijetio kako njemački dokumenti sadrže određene fraze koje se svakodnevno ponavljaju. Prvi slučaj ovakvog ponavljanja je bila riječ „wetter“, što je njemački za „vrijeme“, koja se ponavljala na istom mjestu u poruci koja je svakog dana slana u šest sati i pet minuta ujutro [4]. Nijemci su također velik broj poruka završavali s riječima „Heil Hitler“ no glavni krivac je svakako bila fraza „Ništa za prijaviti“ koju su Nijemci svakodnevno slali u ogromnom broju ponavljanja. Engleski kriptografi su dakle imali velik broj šifriranih poruka uz nekoliko parova šifriranog i jasnog teksta te su tako svaki dan bili u mogućnosti otkriti koji ključ neprijatelj koristi.

3.2.3. Napad odabranim jasnim tekstom

Napad odabranim jasnim tekstom (eng. *Chosen-plaintext attack – CPA*) napad je vrsta napada na kriptografiju u kojem kriptoanalitičar ima na raspolaganju čitav šifrirani tekst te čitav jasni tekst. Cilj napadača u ovakvom napadu je naravno otkrivanje ključa korištenog za šifriranje poruka. Napadač do šifriranog teksta može doći na način da presretne poruku, kao i kod prijašnja dva napada, a do jasnog teksta može doći prevarom žrtve. Alternativno napadač možda ima pristup stroju koji šifrira poruku ali nema pristup tajnom ključu koji je već

unesen u stroj pa tako može sam odabrati jasni tekst i dobiti pripadajući šifrirani tekst. Ovakva vrsta napada napadaču osigurava više informacija od prethodne dvije te samim time možemo zaključiti da bi trebala biti uspješnija. U praktičnom dijelu ovog rada se u svrhu demonstracije napada na modernu simetričnu kriptografiju koristi baš ovakav tip napada zajedno s napadom grubom silom.

4. Aplikacija za simulaciju napada

U praktičnom dijelu rada simuliran je napad na algoritam AES uz poznavanje bloka jasnog i šifriranog teksta. Jasni i šifrirani test su u ovom slučaju nisu bili korišteni u svrhe kriptoanalize već isključivo za potvrđivanje uspješnosti izvršenog napada. Simulirani napad je stoga napad grubom silom i ne uključuje elemente kriptoanalize.

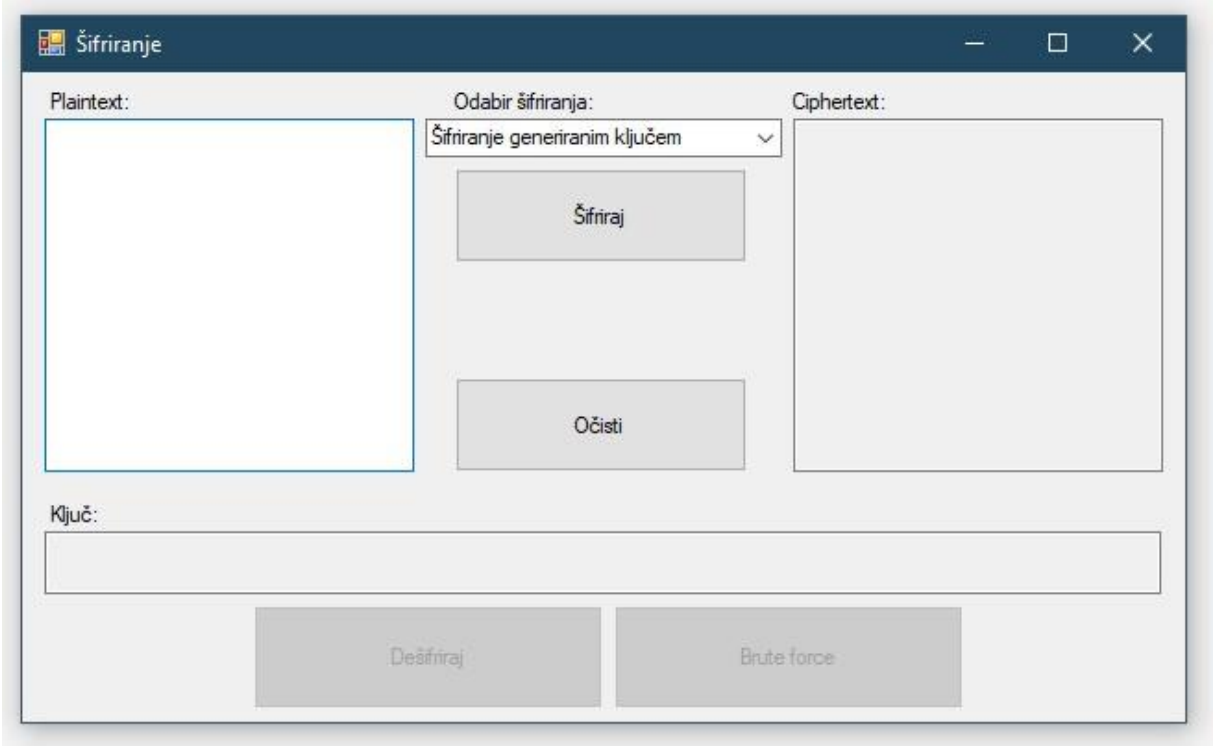
Za izradu programskog koda simulacije napada je korišten programski jezik C# u Microsoft Visual Studio 2019 integriranom razvojnom okruženju. Aplikacija za simulaciju napada je izgrađena u obliku Windows formi u .NET okosnici. Razlog odabiru ovog načina izrade simulacije je pristupačnost i bolja vizualizacija simulacije i njenih rezultata u usporedbi s onima koje bi generirala simulacija u npr. konzolnoj aplikaciji. Aplikacija za simulaciju se sastoji od nekoliko različitih sučelja, svako sa svojom svrhom. Tako postoje sučelja za šifriranje, dešifriranje i izvršavanje samog napada.

U aplikaciji je moguće izvršiti šifriranje i napad na dva različita načina tj. korištenjem dvije različite domene ključeva. Prva opcija je šifriranje korištenjem ključa veličine 128 bita iz čitave domene ključeva koju inače koristi AES algoritam s tom veličinom ključa. U ovom slučaju se nasumično odabire tj. generira jedan od 2^{128} ključeva. Kod izvršavanja napada se također iterira kroz čitavu domenu mogućih ključeva. Alternativno, korisnik aplikacije je u mogućnosti odabrati šifriranje i kasnije napad na smanjeni broj ključeva. U ovom slučaju se ključ generira na način da su samo prva dva bajta 128-bitnog tj. 16-bajtnog ključa nasumična a svi ostali bajtovi su nula i to na način da prvi bajt pokriva sve kombinacije bitova dok drugi bajt pokriva samo četvrtinu kombinacija bitova. Kao rezultat toga, druga opcija šifriranja i napadanja koristi domenu ključeva u kojoj se nalazi samo 2^{14}, tj. 16,384 ključeva. U sljedećim potpoglavljima će kodovi obiju metoda biti prikazani te pobliže objašnjeni.

4.1. Šifriranje

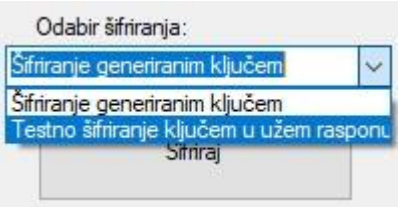
4.1.1. Sučelje

Sučelje za šifriranje se sastoji od sljedećih elemenata: kućice namijenjene za upisivanje jasnog teksta, kućice za prikazivanje šifriranog teksta, kućice za prikazivanje generiranog ključa, padajućeg izbornika za odabir načina šifriranja, gumbova za šifriranje, uklanjanje unosa te gumbova za otvaranje sučelja za dešifriranje i simulaciju napada. Sučelje za šifriranje je prikazano na sljedećoj slici (Slika 16).



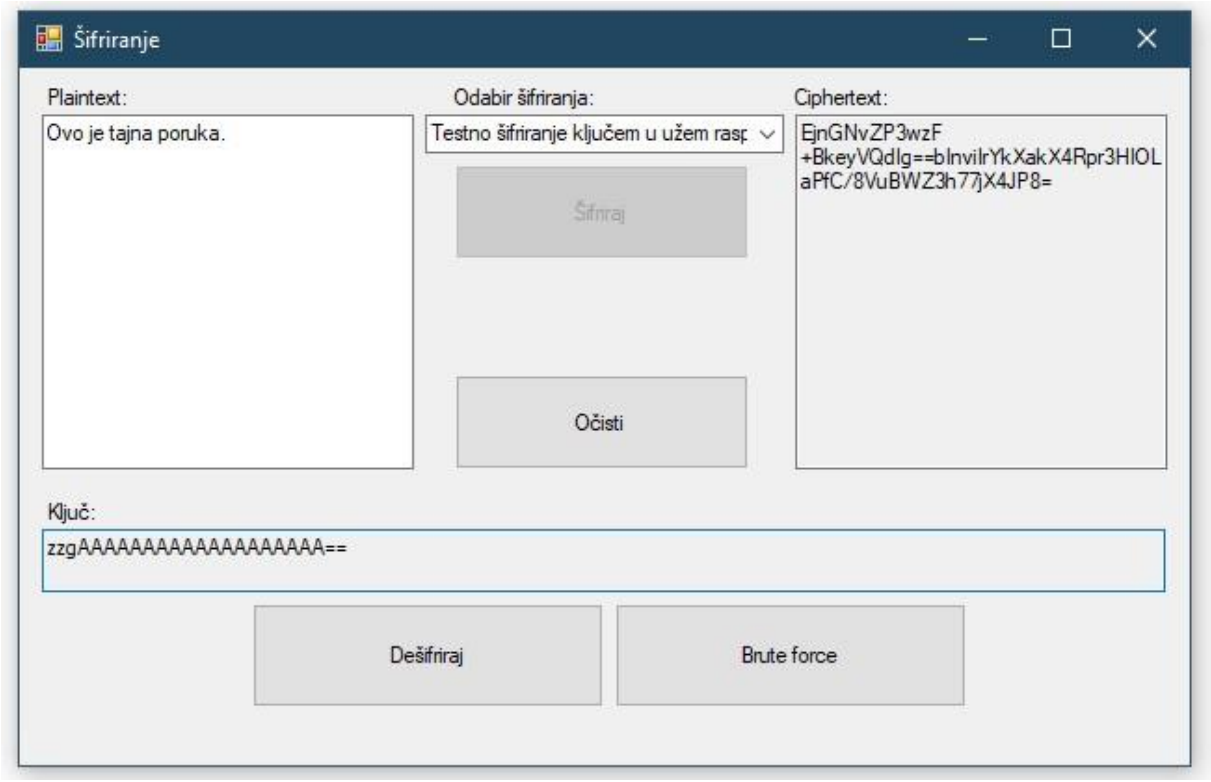
Slika 16: Sučelje šifriranja [autorski rad]

Slučaj korištenja opisanog sučelja je sljedeći: korisnik u kućicu za jasni tekst (eng. *„plaintext“*) upisuje poruku koju želi šifrirati. Korisnik zatim odabire način šifriranja koji želi koristiti tj. želi li koristiti čitavu domenu ili ograničenu domenu u testne svrhe. Razlike između dva načina šifriranja su objašnjene u prethodnom poglavlju. Padajući izbornik je prikazan na sljedećoj slici (Slika 17).



Slika 17: Padajući izbornik načina šifriranja [autorski rad]

Nakon upisivanja poruke i odabira načina šifriranja, korisnik klikom na gumb „Šifriraj“ šifrira poruku te se u kućici namijenjenoj za šifrirani tekst (eng. *„ciphertext“*) isti pojavljuje dok se u isto vrijeme u kućici „Ključ“ pojavljuje novonastali ključ koji se koristio u algoritmu šifriranja. Sučelje nakon šifriranja poruke je prikazano na sljedećoj slici (Slika 18).



Slika 18: Sučelje za šifriranje nakon šifriranja poruke [autorski rad]

Nakon procesa šifriranja na sučelju gumbovi „Dešifriraj“ i „Brute force“ postaju aktivni te je moguće pritiskom na njih otvoriti sučelja za dešifriranje i simulaciju napada. Bilo kada u čitavom procesu je moguće poništiti šifriranje te vratiti sučelje u početno stanje klikom na gumb „Očisti“.

4.1.2. Programski kod

U programskom kodu sučelja za šifriranje glavnu riječ vodi funkcija „buttonSifriraj_Click“ koja se izvršava klikom na gumb „Šifriraj“ i čiji zadatak je šifriranje poruke i prikazivanje rezultata na sučelju kao što je prikazano na Slici 18. Programski kod ove funkcije je sljedeći:

```
using System.Security.Cryptography;
/*
Funkcija koja pri pritisku na gumb "Šifriraj" uzima jasni tekst iz kućice
na formi, generira ključ i IV uz pomoć kojih šifrira jasni tekst
te postavlja ključ i šifrirani tekst u Base64 obliku u kućice na formi
*/
private void buttonSifriraj_Click(object sender, EventArgs e)
{
	using (Aes aes = Aes.Create())
	{
		byte[] sifriraniTekstBajtovi = null;
		// Ako je odabran testni način šifriranja
		if(comboBoxSifriranje.SelectedIndex == 1)
		{
			Random rand = new Random();
			int b1 = rand.Next(0, 255);
			int b2 = rand.Next(0, 63);
			byte[] kljuc = {(byte)b1, (byte)b2, 0, 0, 0, 0, 0, 0, 0,
			0, 0, 0, 0, 0, 0, 0 };
			aes.Key = kljuc;
		}

		ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key,
		aes.IV);
		using (MemoryStream mse = new MemoryStream())
		{
			using (CryptoStream cse = new CryptoStream(mse,
			encryptor, CryptoStreamMode.Write))
				{
				using (StreamWriter swe = new StreamWriter(cse))
				{
					swe.Write(textBoxJasniTekst.Text);
				}
				sifriraniTekstBajtovi = mse.ToArray();
			}
		}

		// Inicijaciski vektor nam je potreban za dešifriranje pa ga
		stoga dodajemo na početak šifriranog teksta prije nego tekst
		pošaljemo na formu
		textBoxSifriraniTekst.Text = Convert.ToBase64String(aes.IV) +
		Convert.ToBase64String(sifriraniTekstBajtovi);
		textBoxKljuc.Text = Convert.ToBase64String(aes.Key);
		buttonSifriraj.Enabled = false;
		buttonDesifriraj.Enabled = true;
		buttonBruteForce.Enabled = true;
	}
}
```

Sam kod već sadrži komentare koji pobliže opisuju neke zbunjujuće dijelove koda no neće škoditi i ukratko opisati čitav kod. Na početku se stvara objekt AES klase koja se nalazi u „System.Security.Cryptography“ biblioteci te se taj objekt koristi kroz čitav proces šifriranja. AES objekt sam generira ključ i inicijacijski vektor no u slučaju da je korisnik odabrao testni način šifriranja potrebno je generirati novi ključ iz smanjenog obujma domene ključeva i to na sljedeći način: generiraju se dva nasumična broja i to tako da je prvi u rasponu 0 – 255 a drugi u rasponu 0 – 63. Uz pomoć generiranih brojeva se izrađuje novi ključ i to tako da definiramo varijablu „kljuc“ kao polje bajtova te na mjesta prva dva bajta umetnemo dva nasumična broja. Naposlijetku postavimo varijablu „kljuc“ kao vrijednost „Key“ objekta „aes“.

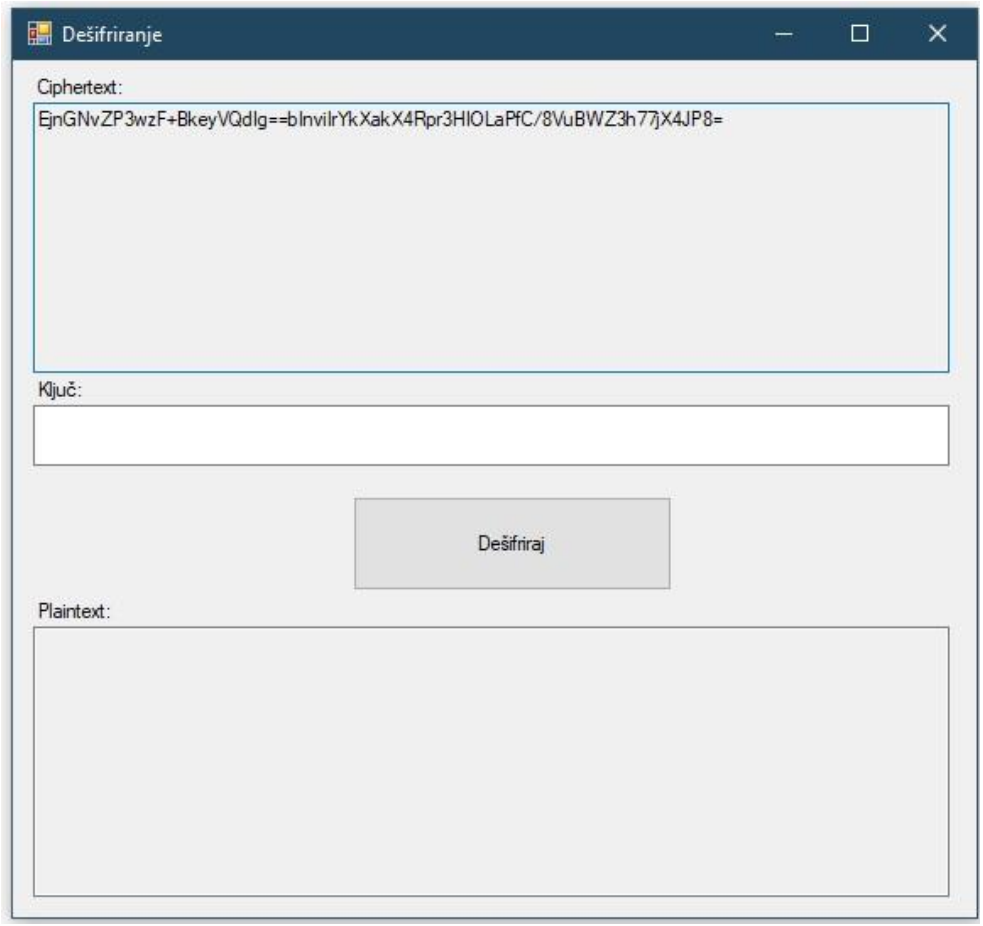
U sljedećem bloku koda se kreira nekoliko objekata čiji zadatak je izvođenje samog šifriranja. Kreirani su objekti klasa „ICryptoTransform“, „MemoryStream“, „CryptoStream“ i „StreamWriter“. Objekt klase „CryptoStream“ je zadužen za samo šifriranje dok ostali služe kao potpora. Bitno je da objekt klase „ICryptoTransform“ bude tipa „encryptor“. Obradom jasnog teksta spomenutim objektima se dolazi do bajtova šifriranog teksta koji se pohranjuju u varijablu „sifriraniTekstBajtovi“.

U posljednjem bloku koda rezultati šifriranja se formatiraju i pohranjuju. Bajtovi inicijacijskog vektora, ključa i šifriranog teksta se pretvaraju u tip podataka „string“ koristeći funkciju „ToBase64String“ te se upisuju u kućice na sučelju. Inicijacijski vektor se dodaje na početak šifriranog teksta zbog razloga spomenutih u poglavlju „3.3.6.2. Metoda lančanih blokova“.

4.2. Dešifriranje

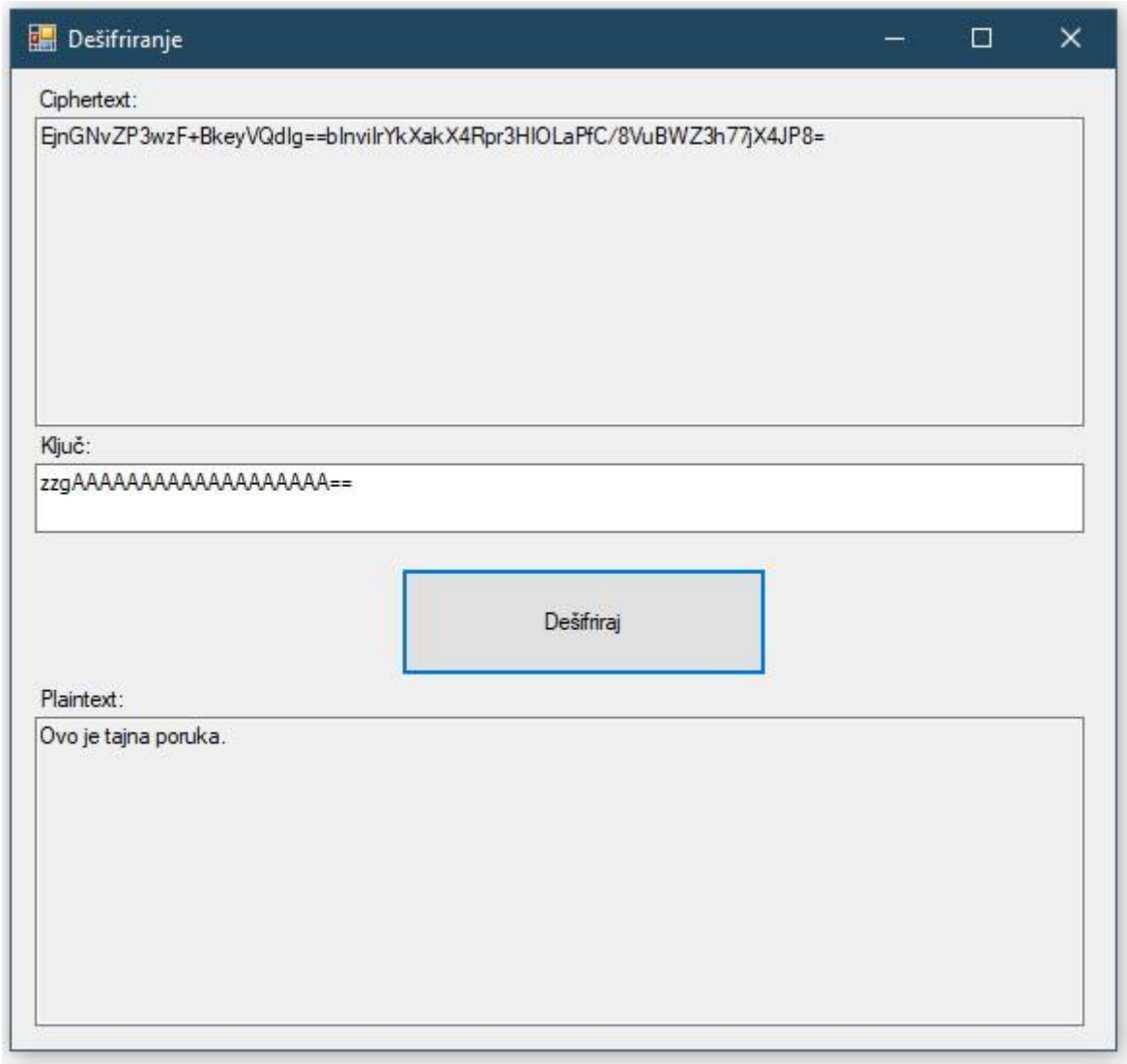
4.2.1. Sučelje

Sučelje za dešifriranje je podosta jednostavnije od sučelja za šifriranje pa se tako sastoji samo od kućice za šifrirani tekst, kućice za ključ, kućice za javni tekst te gumba za dešifriranje. Sučelje je prikazano na sljedećoj slici (Slika 19).



Slika 19: Sučelje dešifriranja [autorski rad]

Prilikom otvaranja sučelja šifrirani tekst se automatski prosljeđuje iz sučelja za šifriranje pa je stoga potrebno samo unijeti ključ korišten za šifriranje te kliknuti na gumb „Dešifriraj“. Algoritam potom dešifrira poruku te jasni tekst ispisuje u kućicu za jasni tekst. Sučelje nakon opisanog procesa je prikazano na sljedećoj slici (Slika 20).



Slika 20: Sučelje za dešifriranje nakon dešifriranja poruke [autorski rad]

4.2.2. Programski kod

Programski kod sučelja za dešifriranje se sastoji samo od konstruktora i funkcije koja se izvršavam prilikom klika na gumb „Dešifriraj“:

```
public FrmDesifriraj(string sifriraniTekst)
{
	InitializeComponent();
	textBoxSifriraniTekst.Text = sifriraniTekst;
}

private void buttonDesifriraj_Click(object sender, EventArgs e)
{
	if(textBoxKljuc.Text=="")
	{
		MessageBox.Show("Unesite ključ!");
	}
```

```
        else
        {
            try
            {
                byte[] kljuc =
                Convert.FromBase64String(textBoxKljuc.Text);
                byte[] inicijaciskiVektor =
                Convert.FromBase64String(textBoxSifriraniTekst.Text.Subst
                ring(0,24));
                byte[] sifriraniTekst =
                Convert.FromBase64String(textBoxSifriraniTekst.Text.Subst
                ring(24));
                using (Aes aes = Aes.Create())
                {
                    ICryptoTransform decryptor =
                    aes.CreateDecryptor(kljuc, inicijaciskiVektor);
                    using (MemoryStream msd = new
                    MemoryStream(sifriraniTekst))
                    {
                        using (CryptoStream csd = new
                        CryptoStream(msd, decryptor,
                        CryptoStreamMode.Read))
                        {
                            using (StreamReader srd = new
                            StreamReader(csd))
                            {
                                textBoxJasniTekst.Text =
                                srd.ReadToEnd();
                            }
                        }
                    }
                }
            }
            catch
            {
                MessageBox.Show("Greška!");
            }
        }
}
```

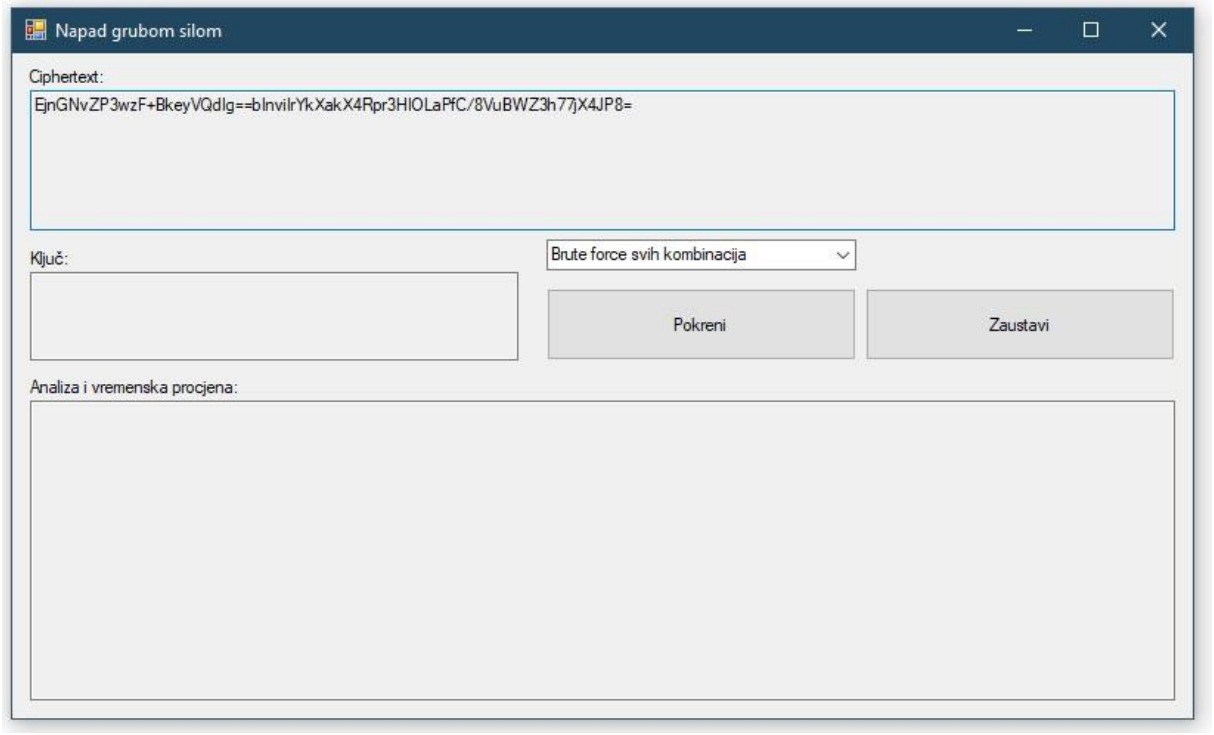
U konstruktoru forme naziva „FrmDesifriraj“ se preuzima te u prikladnu kućicu upisuje šifrirani tekst. Funkcija za dešifriranje prvo provjerava je li u kućici za ključ išta uneseno te upozorava korisnika da nešto unese u slučaju da nije. U slučaju da je ključ već unesen funkcija pokušava izvesti dešifriranje i to na način suprotan procesu šifriranja. Prvo se ključ, inicijacijski vektor i šifrirani tekst pretvaraju natrag iz teksta u polja bajtova te se zatim kreiraju objekti istih klasa kao i kod šifriranja. Razlika je u tome što je u slučaju dešifriranja objekt klase „ICryptoTransform“ tipa „decryptor“ umjesto „encryptor“ te što se ne koristi klasa „StreamWriter“ već klasa „StreamReader“. Šifrirani tekst se obrađuje spomenutim objektima te se kao rezultat dobiva jasni tekst koji upisujemo u kućicu za jasni tekst. U slučaju bilo kakve greške, npr. unosa netočnog ključa, program vraća grešku.

4.3. Simulacija napada

Treća i posljednja funkcionalnost aplikacije je funkcionalnost simulacije napada na kriptografski algoritam.

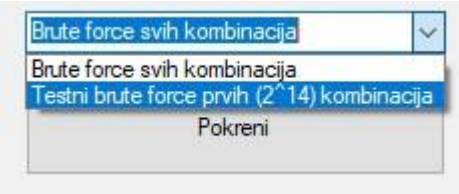
4.3.1. Sučelje

Sučelje za simulaciju napada se sastoji od sljedećih elemenata: kućice za prikaz šifriranog teksta, kućice za prikaz ključa, kućice za prikaz analize i vremenske procjene, padajućeg izbornika za odabir načina izvršavanja napada i gumba za pokretanje izvršavanja napada. Sučelje je prikazano na sljedećoj slici (Slika 21).



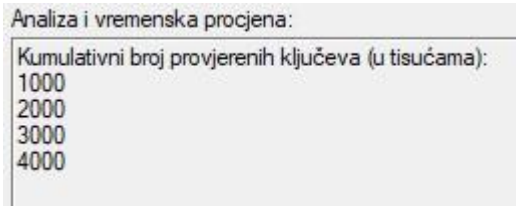
Slika 21: Sučelje simulacije napada [autorski rad]

Prilikom otvaranja sučelja na njemu je već prikazan šifrirani tekst koji je, kao i u slučaju dešifriranja, preuzet od sučelja šifriranja. Prije pokretanja napada korisnik odabire način izvršavanja napada analogan načinu šifriranja koji je odabrao prilikom šifriranja. Tako u svrhu testiranja odabiremo testni način koji iterira kroz smanjenu testnu domenu ključeva. Opcije padajućeg izbornika za odabir načina izvršavanja napada su prikazane na sljedećoj slici (Slika 22).



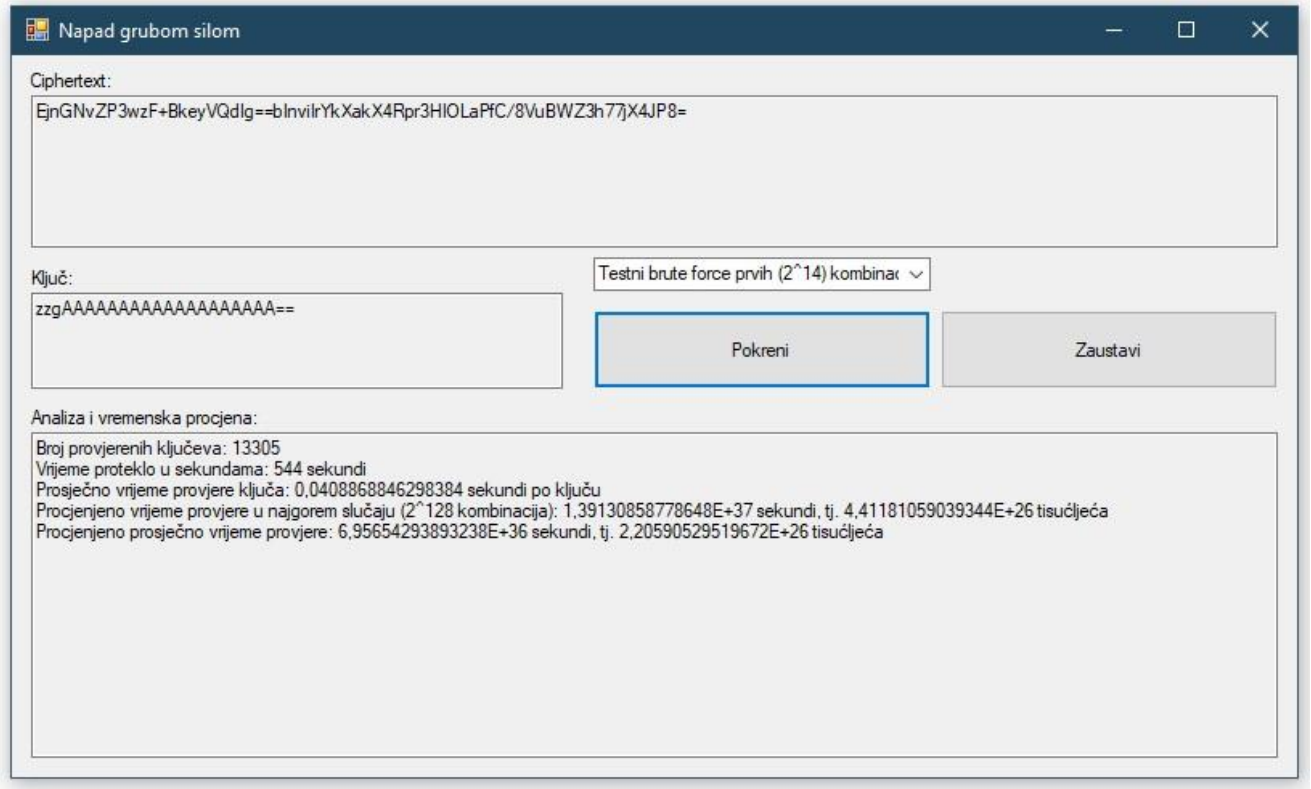
Slika 22: Padajući izbornik načina izvršavanja napada [autorski rad]

Nakon odabira načina izvršavanja napada korisnik pokreče napad klikom na gumb „Pokreni“. Aplikacija napad pokreće u novoj dretvi te stoga dozvoljava korištenje drugih akcija za vrijeme izvođenja te aktivno ažurira status napada na način da prikazuje broj svakog tisućitog provjerenog broja. Tako se u kućici „Analiza i vremenska procjena“ za vrijeme izvršavanja redom prikazuju brojevi 1000, 2000, itd. što je prikazano na sljedećoj slici (Slika 23.)



Slika 23: Poruka o napretku simulacije napada [autorski rad]

Napad završava kada aplikacija pronađe ključ koji je korišten za šifriranje poruke te isti prikaže korisniku tako što ga upiše u za to namijenjenu kućicu na sučelju. Aplikacija također prikaže određene informacije o provedenom napadu. Sučelje nakon izvršenog napada je prikazano na sljedećoj slici (Slika 24).



Slika 24: Sučelje za izvršavanje napada nakon izvršenog napada [autorski rad]

Obratimo pozornost na kućicu naslova „Analiza i vremenska procjena“ u kojoj je prikazana, kao što naziv sugerira, kratka vremenska analiza provedenog napada te procjena vremena potrebno za izvršavanje punog napada. U prikazanom primjeru je u 544 sekunde provjereno 13305 ključeva što znači da je za provjeru jednog ključa u prosjeku bilo potrebno otprilike 0.0409 sekundi. Na prvi pogled ovo se možda čini brzo no ostatak vremenske analize nam jasno prikazuje da to nije tako. Naime, program množi prosječno vrijeme provjere ključa s ukupnim brojem ključeva koji moraju biti provjereni te tako dobiva vrijeme potrebno za provjeru svih ključeva u domeni. Posljednji dio analize se odnosi na procjenu prosječnog vremena provjere svih ključeva koja se dobiva tako da jednostavno podijelimo procjenu vremena provjere svih ključeva s 2. Obje dobivene brojke su uistinu astronomske te je evidentno da ovaj način izvršavanja napada na AES algoritam nema gotovo pa nikakvog smisla.

4.3.2. Programski kod

Kod dijela programskog koda zaduženog za šifriranje su prikazana dva načina šifriranja: jedan prezentacijski koji koristi čitavu domenu ključeva za generiranje ključa te jedan testni koji koristi smanjenu domenu u svrhu testiranja. Kod šifriranja razlika između

programskih kodova ovih dvaju načina je bila nekoliko linija koda te uvelike zanemariva no u slučaju simulacije napada ta razlika je dovoljno naglašena da opravdava odvajanje koda u dvije različite funkcije.

Program za simulaciju napada koristi novu dretvu koja je odvojena od glavne dretve i to uz pomoć „BackgroundWorker“ klase. Prilikom klika na gumb „Pokreni“ program pokreće pozadinsku dretvu koja izvršava sljedeći kod:

```
int brojKljuceva = 0;
double protekloVrijeme;
double prosjecnoVrijeme;
Rezultat rezultat = new Rezultat();
Stopwatch stoperica = new Stopwatch();


stoperica.Start();
if ((int)e.Argument==0)
{
      BruteForce();
}
else if ((int)e.Argument == 1)
{
      BruteForceTest();
}
stoperica.Stop();

protekloVrijeme = stoperica.ElapsedMilliseconds / 1000;
prosjecnoVrijeme = protekloVrijeme / brojKljuceva;

rezultat.analiza =
      $"Broj provjerenih ključeva: {brojKljuceva}" + Environment.NewLine +
      $"Vrijeme proteklo u sekundama: {protekloVrijeme} sekundi" +
      Environment.NewLine +
      $"Prosječno vrijeme provjere ključa: {prosjecnoVrijeme} sekundi po
      ključu" + Environment.NewLine +
      $"Procjenjeno vrijeme provjere u najgorem slučaju (2^128
      kombinacija): {prosjecnoVrijeme * Math.Pow(2, 128)} sekundi, tj.
      {((((((prosjecnoVrijeme * Math.Pow(2, 128)) / 60) / 60) / 24) / 365) /
      1000} tisućljeća" + Environment.NewLine +
      $"Procjenjeno prosječno vrijeme provjere: {((prosjecnoVrijeme *
      Math.Pow(2, 128)) / 2)} sekundi, tj. {((((((prosjecnoVrijeme *
      Math.Pow(2, 128)) / 2) / 60) / 60) / 24) / 365) / 1000} tisućljeća";

e.Result = rezultat;
```

Vrijeme potrebno za provođenje čitavog napada se mjeri uz pomoć „Stopwatch“ klase „System.Diagnostics“ biblioteke i to na način da se štoperica pokreće prije pozivanja funkcije „BruteForceTest()“ ili „BruteForce()“ te zaustavlja nakon njenog izvršavanja. Program računa vrijeme koje je proteklo te ga pretvara u sekunde a zatim računa prosječno vrijeme potrebno za provjeru jednog ključa tako da podijeli proteklo vrijeme s brojem provjerenih ključeva. Rezultati se zatim prikazuju u pripadajućoj kućici na sučelju.

Za zapis i vraćanje podataka potrebnih za vremensku analizu i ključa iz pozadinske dretve u glavnu dretvu se koristi struktura „Rezultat“ koja se sastoji od dva podatka tipa „string“:

```
public class Rezultat
{
      public string analiza;
      public string kljuc;
}
```

U slučaju zaustavljanja programa prije pronalaska ključa obje funkcije prikazuju koliko ključeva je isprobano do zaustavljanja te pripadajuću vremensku analizu. Vrijednost ključa naravno ne mogu vratiti jer ona nije pronađena pa stoga prikazuju prikladnu poruku. Kod za prijevremeno zaustavljanje procesa je identičan u obje funkcije a izgleda ovako:

```
if (backgroundWorker1.CancellationPending)
{
      stoperica.Stop();
      protekloVrijeme = stoperica.ElapsedMilliseconds / 1000;
      prosjecnoVrijeme = protekloVrijeme / brojKljuceva;

      rezultat.kljuc = "Ključ nije pronađen!";
      rezultat.analiza =
      $"Broj provjerenih ključeva: {brojKljuceva}" + Environment.NewLine +
      $"Vrijeme proteklo u sekundama: {protekloVrijeme} sekundi" +
      Environment.NewLine +
      $"Prosječno vrijeme provjere ključa: {prosjecnoVrijeme} sekundi po
      ključu" + Environment.NewLine +
      $"Procjenjeno vrijeme provjere u najgorem slučaju (2^128
      kombinacija): {prosjecnoVrijeme * Math.Pow(2, 128)} sekundi, tj.
      {(((((prosjecnoVrijeme * Math.Pow(2, 128)) / 60) / 60) / 24) / 365) /
      1000} tisućljeća" + Environment.NewLine +
      $"Procjenjeno prosječno vrijeme provjere: {((prosjecnoVrijeme *
      Math.Pow(2, 128)) / 2)} sekundi, tj. {((((((prosjecnoVrijeme *
      Math.Pow(2, 128)) / 2) / 60) / 60) / 24) / 365) / 1000} tisućljeća";

      e.Result = rezultat;
      return;
}
```

4.3.2.1. Prezentacijski način

Prvi od dva načina je prezentacijski način koji koristi čitavu domenu ključeva za generiranje ključa. Prisjetimo se da je za 128-bitni AES moguće generirati bilo koji od 2^{128} ključeva. Vrijeme potrebno za iteraciju čitave domene je stoga astronomski veliko te ovaj način nije praktičan za izvođenje bilo kakvih simulacija napada pa je stoga uključen u aplikaciju isključivo u prezentacijske svrhe. Funkcija za izvršavanje ovog načina napada koristi 16 ugniježđenih for petlji u svrhu generiranja ključa – po jedna petlja za svaki bajt ključa. U sljedećem kodu su prikazane prve 3 od 16 petlji:

```
for (int b1 = 0; b1 < 256; b1++)
      for (int b2 = 0; b2 < 256; b2++)
            for (int b3 = 0; b3 < 256; b3++)
```

Unutar ugniježđenih petlji se uz već spomenuti kod za prijevremeno zaustavljanje nalazi i sljedeći kod:

```
brojKljuceva++;
backgroundWorker1.ReportProgress(brojKljuceva);
byte[] generiraniKljuc = {(byte)b1, (byte)b2, (byte)b3, (byte)b4,
                          (byte)b5, (byte)b6, (byte)b7, (byte)b8,
                          (byte)b9, (byte)b11, (byte)b11, (byte)b12,
                          (byte)b13, (byte)b14, (byte)b15, (byte)b16};

byte[] inicijaciskiVektor =
Convert.FromBase64String(textBoxSifriraniTekst.Text.Substring(0, 24));
byte[] sifriraniTekst =
Convert.FromBase64String(textBoxSifriraniTekst.Text.Substring(24));

try
{
      using (Aes aes = Aes.Create())
      {
            ICryptoTransform decryptor =
            aes.CreateDecryptor(generiraniKljuc, inicijaciskiVektor);
            using (MemoryStream msd = new MemoryStream(sifriraniTekst))
            {
                  using (CryptoStream csd = new CryptoStream(msd,
                  decryptor, CryptoStreamMode.Read))
                  {
                        using (StreamReader srd = new StreamReader(csd))
                        {
                              desifriraniTekst = srd.ReadToEnd();
                        }
                  }
            }
      }
      if (String.Compare(jasniTekst, desifriraniTekst) == 0)
      {
            rezultat.kljuc = Convert.ToBase64String(generiraniKljuc);
            MessageBox.Show("Ključ pronađen!");
            return;
      }
}
catch
{
}
```

Svaka iteracija počinje inkrementom broja isprobanih ključeva te slanjem tog broja funkciji praćenja procesa pozadinske dretve koja je zadužena za prikazivanje poruka napretka procesa. Kod funkcije je sljedeći:

```
private void backgroundWorker1_ProgressChanged(object sender,
ProgressChangedEventArgs e)
{
      if (e.ProgressPercentage % 1000 == 0 && e.ProgressPercentage != 0)
      {
            textBoxAnaliza.Text += $"{e.ProgressPercentage}" +
            Environment.NewLine;
      }
}
```

Funkcija praćenja procesa jednostavno provjerava je li primljeni broj djeljiv sa tisuću te ga prikazuje na formi u slučaju da je.

Funkcija za izvršavanje napada nastavlja s generiranjem ključa te odvajanjem inicijacijskog vektora od šifriranog teksta. Inicijacijski vektor i šifrirani tekst se pretvaraju iz čitljivog oblika natrag u polja bajtova. Program zatim pokušava provesti dešifriranje na način koji je već prikazan u prethodnom poglavlju. Pri završetku postupka dešifriranja u varijabli „desifriraniTekst“ se nalazi tekstualna vrijednost koju program zatim uspoređuje sa tekstualnom vrijednošću jasnog teksta (prisjetimo se da su napadaču zbog vrste provedenog napada dostupna oba teksta). U slučaju da su tekstualne vrijednosti iste funkcija uspoređivanja vraća vrijednost 0 te je pronađen ključ korišten za šifriranje. U slučaju da vrijednosti nisu iste program nastavlja sa iteracijom kroz petlje sve dok ne pronađe ključ.

4.3.2.2. Testni način

Drugi način simulacije napada ne koristi čitavu domenu ključeva AES algoritma već koristi uvelike smanjenu domenu koja sadrži „samo“ 2^{14}, tj. 16384 ključeva te je stoga pogodniji za testnu simulaciju napada. Kod funkcije testnog načina simulacije napada je sljedeći:

```
for (int b1 = 0; b1 < 256; b1++)
      for (int b2 = 0; b2 < 64; b2++)
      {
            brojKljuceva++;
            backgroundWorker1.ReportProgress(brojKljuceva);
            byte[] generiraniKljuc = { (byte)b1, (byte)b2, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0 };
            byte[] inicijaciskiVektor =
            Convert.FromBase64String(textBoxSifriraniTekst.Text.Substring(0
            , 24));
            byte[] sifriraniTekst =
            Convert.FromBase64String(textBoxSifriraniTekst.Text.Substring(2
            4));
            try
            {
                  using (Aes aes = Aes.Create())
                  {
```

```
                        ICryptoTransform decryptor =
                        aes.CreateDecryptor(generiraniKljuc,
                        inicijaciskiVektor);
                        using (MemoryStream msd = new
                        MemoryStream(sifriraniTekst))
                        {
                              using (CryptoStream csd = new
                              CryptoStream(msd, decryptor,
                              CryptoStreamMode.Read))
                              {
                                    using (StreamReader srd = new
                                    StreamReader(csd))
                                    {
                                          desifriraniTekst =
                                          srd.ReadToEnd();
                                    }
                              }
                        }
                  }
                  if (String.Compare(jasniTekst, desifriraniTekst) == 0)
                  {
                        rezultat.kljuc =
                        Convert.ToBase64String(generiraniKljuc);
                        MessageBox.Show("Ključ pronađen!");
                        return;
                  }
            }
            catch
            {
            }
      }
```

U svrhu jednostavnosti je iz koda izostavljena već opisana funkcija za prijevremeno zaustavljanje napada. Najočitija razlika između programskog koda testnog načina i programskog koda prezentacijskog načina je ta da se u ovom slučaju koriste samo dvije for petlje u usporedbi s 16 petlji korištenih u prethodnom načinu simulacije. Razlog tome je naravno taj da su u testnom načinu simulacije samo prva dva bajta ključa nasumična dok je ostalih 14 bajtova uvijek nula.

5. Zaključak

Ovaj rad je prošao kroz gotovo čitavu povijest kriptografije i ljudskog društva, od izuma pisma pa sve do 21. stoljeća. Opisane su neke od najpoznatijih šifri korištenih u povijesti poput Cezarove šifre te načini na koje su one probijene. Dočarana je utrka između onih koji stvaraju šifre te onih koji iste razaraju te je dan detaljan opis modernih šifri koje su danas u upotrebi. Na jednostavan način su prikazane funkcionalnosti modernih šifri te što izbjegavati pri korištenju istih. Prikazane su i opasnosti pri korištenju šifri te napadi na kriptografiju. Naposlijetku je prikazan jedan rudimentarni napad na modernu kriptografiju uz pomoć kojeg je ilustrirana važnost korištenja sigurnih algoritama i kanala komunikacije.

Opisom nekoliko različitih vrsta algoritama šifriranja koji se koriste u moderno doba je dočarana razina sigurnosti potrebna da bi se algoritam smatrao standardom. Detaljnim opisom algoritma Rijndael, koji uključuje motivaciju njegovog nastanka te način na koji funkcionira, je prikazana važnost njegova postojanja te opravdavan njegov status industrijskog standarda. Nadalje, praktičnim prikazom simulacije napada na Rijndael tj. AES je prikazano koliko vremenski zahtjevan bi bio napad na isti te je time potvrđena pretpostavka da baš taj algoritam daje najbolji omjer sigurnosti i jednostavnosti pristupa.

Aplikacija koja je kreirana u svrhu prikazivanja simulacije napada na simetričnu kriptografiju je intuitivna i laka za korištenje te pruža mogućnost šifriranja poruke, dešifriranja poruke te simulacije napada grubom silom na AES. Aplikacija koristi više dretvi te je kao takva u stanju izvršavati simulaciju napada u pozadini dok se korisnik bavi nečim drugim unutar ili izvan aplikacije.

Za kraj je bitno zaključiti kako je u današnje doba tajni, krađe privatnih informacija i izrabljivanja individualne privatnosti svakog pojedinca od strane malicioznih sila najvažnije biti obrazovan i dobro upućen u bitne teme poput kriptografije.

Popis literature

[1] “kriptografija | Hrvatska enciklopedija.” https://enciklopedija.hr/Natuknica.aspx?ID=33988 (accessed Aug. 01, 2022).

[2] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks, 5th Edition*, 2011.

[3] J. Kurose and W. Keith, *K. Ross Computer networking: a top down approach*. 2007.

[4] S. Singh, “The Code Book: The Science of Secrecy from Ancient Egypt To Quantum Cryptography. Anchor Books,” *EDICION, ISBN*, 1999.

[5] J. Daemen and V. Rijmen, “The Design of Rijndael,” *New York*, 2002.

Popis slika