**Treća gimnazija Sarajevo**

 **MATURSKI RAD IZ SOFTVER INŽINJERINGA**

**Tema: Asimetrična Enkripcija u JavaScript-u**

Predmetni profesor:

*Admir Demir, prof.*

Učenik:

*Ismail Mujanović, IV2*

**Sarajevo, 2023.**

Sadržaj

[1. UVOD 2](#_Toc129209008)

[2. OSNOVNI KONCEPTI 3](#_Toc129209009)

[2.1. Šifra (cipher) 3](#_Toc129209010)

[2.2. Cezarova šifra 3](#_Toc129209011)

[2.2.1. Rješavanje Cezarove šifre 3](#_Toc129209012)

[2.3. Modularna (satna) aritmetika 4](#_Toc129209013)

[2.4. Jednokratni blok (one-time pad) 4](#_Toc129209014)

[2.5. Čista slučajnost (pure randomness) 4](#_Toc129209015)

[2.6. Simetrična i asimetrična enkripcija 5](#_Toc129209016)

[2.7. Dva tipa asimetrične enkripcije 5](#_Toc129209017)

[2.7.1. Šifriranje javnog ključa; POVJERLJIVOST 5](#_Toc129209018)

[2.7.2. Digitalni potpisi; AUTENTIFIKACIJA 5](#_Toc129209019)

[3. HISTORIJA 6](#_Toc129209020)

[3.1. Tajno otkriće 6](#_Toc129209021)

[3.2. Javno otkriće 6](#_Toc129209022)

[4. NAJBITNIJI PRINCIPI I ALGORITMI 7](#_Toc129209023)

[4.1. Razmjena ključeva 7](#_Toc129209024)

[4.2. Merkleove zagonetke 7](#_Toc129209025)

[4.3. Modularna eksponencijacija 8](#_Toc129209026)

[4.4. Jednosmjerna funkcija 8](#_Toc129209027)

[4.5. Diffie–Hellman razmjena ključeva 10](#_Toc129209028)

[4.6. RSA (kriptosistem) 11](#_Toc129209029)

[4.6.1. Kako RSA radi? 11](#_Toc129209030)

[4.7. Hibridni kriptosistem 12](#_Toc129209031)

[4.7.1. Mehanizam enkapsulacije ključa 13](#_Toc129209032)

[4.7.2. Shema enkapsulacije podataka 13](#_Toc129209033)

[4.7.3. Primjer hibridnog kriptosistema 13](#_Toc129209034)

[5. IMPLEMENTACIJA U JAVASCRIPTU 14](#_Toc129209035)

[6. ZAKLJUČAK 18](#_Toc129209036)

[LITERATURA 19](#_Toc129209037)

# UVOD

U današnjem digitalnom dobu, sigurnost i zaštita podataka su postali ključni elementi poslovanja i svakodnevnog života. Asimetrična enkripcija igra ključnu ulogu u osiguravanju sigurne razmjene podataka. U ovom maturskom radu, detaljno ću istražiti asimetričnu enkripciju (enkripciju javnog ključa), njezinu primjenu u modernom digitalnom dobu kao i šta se to JavaScriptom može napraviti u području enkripcije pomoću vlastitog projekta - RSA algoritma i njegovog interfejsa.

Izabrao sam ovu temu jer me mnogo interesovao pojam kriptografije, kako funkcioniše sam internet i sigurnost na njemu, na koji način je zagarantovano da informacije ostanu tajne te kako da je moguće da pomoću ovako jakih mašina koje danas imamo podaci opet budu sigurni. Naravno, asimetrična enkripcija nije jedini pojam u internet sigurnosti, ali je definitivno jedan od bitniji.

U prvom dijelu rada, definirati ću osnovne koncepte kriptografije i enkripcije, objasniti razliku između simetrične i asimetrične enkripcije, te opisati kako asimetrična enkripcija koristi javne i privatne ključeve za enkripciju i dekripciju podataka. U drugom dijelu, izučiti ću historiju asimetrične enkripcije, kako i gdje je nastala te ko su bili ti brilijantni ljudi koji su ovo izumili. U trećem dijelu, detaljno ću istražiti najbitnije principe i algoritme asimetrične enkripcije, kako oni rade i zbog čega postoje. U četvrtom ću prikazati implementaciju RSA kriptoalgoritma u JavaScriptu i vizuelno ga prikazati.

Konačno, u zaključku ću sažeti glavne karakteristike i prednosti asimetrične enkripcije, te razmotriti njezinu budućnost i očekivanu primjenu u digitalnom društvu. Ovaj rad će pružiti temeljno razumijevanje asimetrične enkripcije i njezine primjene, te će biti koristan za sve one koji žele bolje razumjeti ovu ključnu tehnologiju sigurnosti podataka.

I na kraju, završio bih ovaj uvod citatom osnivača RSA algoritma, Clifford Cocksa. Ovaj citat ukratko opisuje način na koji je on nadošao na ovu inovativnu ideju koja je toliko bitna u savremenom vremenu. Prvi koncept enkripcije javnog ključa:

"My thinking was that you need something that is easy to do, and difficult to undo, so I thought of the product of two primes. It's essential that you have a one-way function which can only be inverted if you know the factors — and raising numbers to a power with the product as modulus is something you can unpick only if you know the factors."[[1]](#footnote-1)

# OSNOVNI KONCEPTI

Kriptografija je znanost o prikrivanju poruka tajnim kodom. Njena svrha je da oteža otkrivanje istine. Enkripcija je način šifriranja podataka, odnosno proces pretvaranja običnog teksta (**plaintext**) u šifriran tekst (**ciphertext**), koji se može pročitati samo pomoću dekripcije (ciphertext u plaintext).

## Šifra (cipher)

U kriptografiji, šifra je algoritam za izvođenje enkripcije ili dekripcije

Postoji više vrsta šifri:

* **supstituciona** (zamjenska) **šifra** - plaintext se zamenjuje ciphertext-om, na definisan način, uz pomoć ključa
* **transpoziciona** **šifra** - šifrira pozicije znakova bez promjene samih znakova
* polialfabetske zamjenske šifre - korištenje višestrukih alfabeta za šifriranje i dešifriranje
* **jednokratni blok** (one-time pad) - kreiranje potpuno slučajnog, jednokratnog ključa za šifriranje i slanje istog ključa drugoj osobi, u tajnosti, za dešifriranje poruke
* **algoritmi simetričnog ključa** - isti ključ za enkripciju i dekripciju
* **algoritmi asimetričnog ključa** - različiti ključevi za enkripciju i dekripciju

## Cezarova šifra

Jedna od najpoznatijih i najjednostavnijih tehnika šifriranja je Cezarova šifra, koja predstavlja jednu od prvih istinskih metoda enkripcije. Ova šifra koristi se kao vrsta supstitucijske šifre, gdje se svako slovo u plaintext-u zamjenjuje slovom na određenoj poziciji u abecedi, s pomakom u desno ili lijevo za određeni broj mjesta. Na primjer, s pomakom ulijevo za tri mjesta, slovo Č zamijenjuje se slovom A, Ć postaje B, itd. Ova metoda je dobila ime po Juliju Cezaru, koji ju je koristio za svoju privatnu komuninkaciju.

### Rješavanje Cezarove šifre

Postoje dvije situacije koje se mogu razmotriti:

1. napadač zna ili pretpostavlja da se koristi neka vrsta jednostavne supstitucijske šifre, ali nije siguran da je riječ o Cezarovoj šifri
2. napadač zna da se koristi Cezarova šifra, ali ne zna vrijednost pomaka

U prvom slučaju, šifra se može razbiti korištenjem istih tehnika koje se koriste za opću jednostavnu supstitucijsku šifru, poput analize frekvencije ili analize uzorka riječi.

U drugom slučaju, razbijanje šifre je još jednostavnije. Budući da postoji samo ograničen broj mogućih smjena (30 u slučaju bosanskog jezika), napadač može isprobati svaku moguću vrijednost pomoću napada grube sile (brute force attack).

## Modularna (satna) aritmetika

U matematici, modularna aritmetika je sistem aritmetike za cijele brojeve, gdje se brojevi "omotavaju" kada dostignu određenu vrijednost, nazvanu modul.

**Primjer:**

Modularna aritmetika se često primjenjuje u 12-časnom satu gdje je dan podijeljen na dva perioda od 12 sati. Na primjer, ako je trenutno 7:00 sati, tada će biti 3:00 nakon 8 sati. Iako bi se jednostavnim sabiranjem dobilo 7 + 8 = 15, sati se “omotavaju” svakih 12 sati, pa se ponovo počinje brojati od nule nakon broja 12. U matematičkom smislu, ovo se naziva aritmetičkim modulom 12. Prema definiciji, broj 15 je kongruentan broju 3 po modulu 12, što znači da bi "15:00" na 24-satnom satu bio prikazan kao "3:00" na 12-satnom satu.

## Jednokratni blok (one-time pad)

Jednokratni blok (OTP) je tehnika šifriranja koja se smatra nesavladivom, ali zahtijeva korištenje tajnog ključa za jednokratnu upotrebu koji je barem jednake veličine kao i poruka koja se šalje. U ovoj tehnici, plaintext se kombinira sa slučajno generisanim ključem (poznatim i kao jednokratni blok). Svaki bit ili znak plaintext-a se zatim šifrira kombiniranjem sa odgovarajućim bitom ili znakom iz ključa korištenjem modularnog sabiranja.

Rezultirajući šifrirani tekst (ciphertext) bit će nemoguće dešifrirati ili razbiti ako su ispunjena sljedeća četiri uslova:

1. Ključ mora biti dugačak najmanje koliko i otvoreni tekst.
2. Ključ mora biti nasumičan, u potpunosti uzorkovan iz nealgoritamskog, haotičnog izvora kao što je hardverski generator slučajnih brojeva; bez uzorka.
3. Ključ se nikada ne smije ponovo koristiti u cijelosti ili djelomično.
4. Strane koje komuniciraju moraju držati ključ u potpunoj tajnosti.

## Čista slučajnost (pure randomness)

Visokokvalitetni slučajni brojevi su teško generisani jer funkcije za generisanje slučajnih brojeva u većini biblioteka programskih jezika nisu prikladne za kriptografske svrhe. Čak i generatori koji su prikladni za normalnu kriptografsku upotrebu, kao što su /dev/random i mnogi hardverski generatori slučajnih brojeva, ne koriste se uvijek za kriptografske svrhe jer mogu uključivati kriptografske funkcije čija sigurnost nije dokazana. Primjer tehnike za generiranje čiste slučajnosti je mjerenje radioaktivnih emisija.

## Simetrična i asimetrična enkripcija

**Razlika:**

Simetrična enkripcija je metoda u kojoj se isti ključ koristi i za enkripciju i dekripciju, što je čini vrlo brzom i efikasnom. Međutim, ključ mora biti bezbijedno podjeljen između pošiljaoca i primaoca, što može biti izazov u nekim situacijama.

Asimetrična enkripcija, koristi dva različita ključa za šifriranje i dešifriranje. Jedan ključ se koristi za šifriranje, a drugi ključ se koristi za dešifriranje. Ključ za šifriranje je javno dostupan, a ključ za dešifriranje se čuva privatnim. Ovo omogućava sigurnu komunikaciju preko interneta, jer samo osoba koja ima privatni ključ može dešifrirati podatke. Asimetrična enkripcija se smatra sigurnijom od simetrične jer eliminiše potrebu za dijeljenjem tajnog ključa, ali je i sporija i manje efikasna.

Simetrična enkripcija je brža i efikasnija, ali zahtijeva sigurno dijeljenje ključeva, dok je asimetrična enkripcija sporija i manje efikasna, ali pruža bolju sigurnost.

## Dva tipa asimetrične enkripcije

### Šifriranje javnog ključa; POVJERLJIVOST

Pakuje s javnim ključem, otvara privatnim ključem; U sistemu šifriranja s javnim ključem, svako ko ima javni ključ može šifrirati poruku, dajući šifrirani tekst, ali samo oni koji znaju odgovarajući privatni ključ mogu dešifrirati šifrirani tekst kako bi dobili originalnu poruku.

**Primjer:**

Novinar može objaviti javni ključ za šifriranje na web stranici novinske organizacije, kako bi izvori mogli slati tajne poruke u šifriranom obliku. Samo novinar koji posjeduje odgovarajući privatni ključ može dešifrirati te poruke i saznati njihov sadržaj. To osigurava da prisluškivač koji čita e-poštu na putu do novinara ne može dešifrirati šifrirane poruke.

### Digitalni potpisi; AUTENTIFIKACIJA

Pakuje s privatnim ključem, otvara javnim ključem; U sistemu digitalnog potpisa, pošiljalac može koristiti privatni ključ zajedno sa porukom za kreiranje potpisa. Svako sa odgovarajućim javnim ključem može provjeriti da li potpis odgovara poruci, ali krivotvoritelj koji ne poznaje privatni ključ ne može pronaći nijedan par poruka/potpis koji će proći verifikaciju sa javnim ključem.

**Primjer:**

Izdavač softvera može kreirati par ključeva za potpis i uključiti javni ključ u softver instaliran na računarima. Nakon toga, izdavač može poslati ažuriranje softvera koje je potpisano pomoću privatnog ključa. Svako računalo koje primi ažuriranje može provjeriti je li autentično tako što će provjeriti potpis koristeći javni ključ.

# HISTORIJA

U ranim danima kriptografije, dvije strane su koristile ključ koji bi razmijenile na bezbjedan, ali nekriptografski način, kao što su susreti uživo ili kuriri od povjerenja. Taj ključ bi se morao čuvati u potpunoj tajnosti da bi se mogao koristiti za razmjenu šifriranih poruka. Međutim, postoji niz praktičnih poteškoća sa ovim pristupom distribuciji ključeva.

### Tajno otkriće

James H. Ellis, britanski kriptograf koji je radio za vladine komunikacije, zamislio je koncept "netajne enkripcije" (današnje ime kriptografije javnog ključa ili asimetrične enkripcije) 1970. godine, ali nije imao praktičan način za njegovu implementaciju. Njegov kolega Clifford Cocks implementirao je **RSA enkripcijski algoritam** 1973. godine, koji je omogućio praktičan način za netajnu enkripciju, dok je matematičar i kriptograf Malcolm J. Williamson razvio **"Diffie-Hellman razmjenu ključeva"** 1974. godine. Ove ideje su bile proslijeđene američkoj Agenciji za nacionalnu sigurnost i britanskoj vladi, ali su bile ograničene vojnim fokusom i ograničenom računalnom snagom, pa nije bilo praktične primjene. Zbog toga su ostale nepoznate javnosti 27 godina, sve dok britanska vlada nije uklonila tajnost istraživanja 1997. godine.

### Javno otkriće

Godine 1976., Whitfield Diffie i Martin Hellman su razvili asimetrični kriptosistem ključeva uz pomoć Ralpha Merklea, koji je otkrio metodu dogovora javnog ključa. Ova metoda razmjene ključeva je poznata kao Diffie-Hellman razmjena ključeva, i predstavljala je prvu praktičnu metodu za uspostavljanje zajedničkog tajnog ključa preko autentificiranog komunikacijskog kanala bez prethodne zajedničke tajne.

Merkleova tehnika je postala poznata kao **Merkleove zagonetke** i objavljena je 1978., iako je izumljena već 1974. Ovo je učinilo asimetričnu enkripciju relativno novim poljem u kriptografiji, koja inače datira iz više od 2000 godina.

Godine 1977., **Ron Rivest, Adi Shamir i Leonard Adleman** nezavisno su generalizirali Cocks-ovu shemu, objavivši svoj rad 1978. Algoritam je postao poznat kao RSA, prema njihovim inicijalima. Od 1970-ih godina, razvijen je veliki broj i raznovrsnost enkripcije, digitalnog potpisa, dogovora ključeva i drugih tehnika, uključujući Rabin kriptosistem, ElGamal enkripciju, DSA i kriptografiju eliptičke krivulje.

# NAJBITNIJI PRINCIPI I ALGORITMI

## Razmjena ključeva

Razmjena ključeva je metoda u kriptografiji kojom se kriptografski ključevi razmjenjuju između dvije strane, što omogućava korištenje kriptografskog algoritma.

Ako pošiljalac i primalac žele razmjenjivati šifrirane poruke, svaki mora biti opremljen za šifriranje poruka koje se šalju i dešifriranje primljenih poruka.

Razmjena ključeva je problem koji se odnosi na način razmjene svih potrebnih ključeva ili drugih informacija za uspostavljanje sigurnog komunikacijskog kanala, tako da ne postoji mogućnost da ih drugi pronađu. Prije pronalaska asimetrične kriptografije (kriptografije s javnim ključem), simetrična kriptografija je koristila jedan ključ za šifriranje i dešifriranje poruka. Za uspostavu sigurne komunikacije, strane su morale razmijeniti tajni ključ kako bi šifrirale poruke prije slanja i dešifrirale primljene poruke. Ovaj proces se naziva razmjena ključeva.

## Merkleove zagonetke

U kriptografiji, Merkleove zagonetke su rana konstrukcija kriptosistema s javnim ključem, protokol koji je osmislio Ralph Merkle 1974. i objavljen 1978. Omogućuje dvije strane da se dogovore o zajedničkoj tajni razmjenom poruka, čak i ako nemaju tajni uobičajeno unaprijed.

**Primjer:**

Alisa i Bob žele uspostaviti komunikaciju, pa Bob stvara veliki broj šifriranih poruka u obliku zagonetki, svaka sa umjerenim nivoom težine, tako da Alisa može riješiti zagonetku s umjerenom količinom računarskog napora. Ključevi za šifriranje su kratki, što omogućava napad grubom silom. Bob šalje sve zagonetke Alisi, a ona odabire jednu i rješava je. Kada riješi zagonetku, dobije zajednički ključ, tako da Alisa može vratiti Bobu koju je zagonetku riješila. Obje strane sada imaju zajednički ključ; Alisa, jer je rešila zagonetku, a Bob, jer je poslao zagonetku. Svaki prisluškivač (recimo Eva) ima teži zadatak — ne zna koju je zagonetku Alisa riješila. Njena najbolja strategija je riješiti sve zagonetke, ali pošto ih ima toliko, ovo je računski skuplje za Evu nego za Alisu.

Kvadratna složenost nije dovoljno sigurna protiv napadača za praktične kriptografske primjene u stvarnom svijetu. Ova shema ima i jednu posebnost, a to je jedan od prvih primjera kriptografije s javnim ključem. Također, ova shema je bila inspiracija za Diffie-Hellman protokol za razmjenu ključeva koji ima znatno veću složenost, jer se oslanja na problem diskretnog logaritma.

## Modularna eksponencijacija

Modularno eksponenciranje je postupak eksponenciranja koji se obavlja preko modula i koristi se u računarstvu, posebno u kriptografiji javnog ključa. Ova tehnika je korisna u Diffie-Hellman protokolu za razmjenu ključeva i u RSA javnom/privatnom ključu.

Modularna eksponencijacija je ostatak kada se cijeli broj b (baza) podigne na stepen e (eksponent) i podijeli pozitivnim cijelim brojem m (modul); to jest, *c* = *be* mod *m*. Iz definicije dijeljenja slijedi da je 0 ≤ *c* < *m*.

Funkcija modularne eksponencijacije je učinkovita u izračunavanju čak i vrlo velikih cijelih brojeva, dok se računanje modularnog diskretnog logaritma - što uključuje pronalaženje eksponenta e uz dane vrijednosti b, c i m - smatra teškim. Ovaj jednosmjerni aspekt funkcije čini modularnu eksponencijaciju privlačnom za upotrebu u kriptografskim algoritmima.

## Jednosmjerna funkcija

U informatici, jednosmjerna funkcija je funkcija koju je lako izračunati na svakom ulazu, ali je teško vratiti u prvobitno stanje s obzirom na sliku slučajnog ulaza.

Postojanje takvih jednosmjernih funkcija još uvijek nije dokazano. Ako bi se dokazalo postojanje jednosmjernih funkcija, to bi pokazalo da složenost klasa P i NP nisu jednake, što bi riješilo jedno od najvažnijih neriješenih pitanja teorijske računarske nauke.

Sudoku primjer **P protiv NP problema**:

U igri Sudoku, gdje igrač dobiva djelomično ispunjenu mrežu brojeva i pokušava je dovršiti slijedeći određena pravila. Za svaku nepotpunu Sudoku mrežu, postoji li barem jedno legalno rješenje? Svako predloženo rješenje lako se provjerava, a vrijeme potrebno za provjeru rješenja raste polako (polinomno) kako se mreža povećava. Međutim, svi poznati algoritmi za pronalaženje rješenja zahtijevaju eksponencijalno vrijeme za teške primjere, što znači da Sudoku pripada u NP klasu (brzo provjerljivo), ali se čini da ne pripada u P klasu (brzo rješivo). Hiljade drugih problema slično izgledaju, jer se brzo provjeravaju, ali sporo rješavaju. Istraživači su dokazali da mnogi problemi u NP klasi imaju dodatnu osobinu da se brzo rješenje bilo kojeg od njih može koristiti za izgradnju brzog rješenja bilo kojeg drugog problema u NP klasi, a ovo svojstvo se zove NP-potpunost. Nakon decenija istraživanja, nije pronađeno brzo rješenje nijednog od ovih problema, pa većina znanstvenika sumnja da se nijedan od ovih problema ne može riješiti brzo.

Ako se ispostavi da je P ≠ NP, što je široko rasprostranjeno vjerovanje, to bi značilo da u NP postoje problemi koje je teže izračunati nego provjeriti: oni se ne mogu riješiti u polinomskom vremenu, ali se odgovor može provjeriti u polinomskom vremenu.

Problem je nazvan najvažnijim otvorenim problemom u informatici. Osim što je važan problem u teoriji računarstva, dokaz bi u svakom slučaju imao duboke implikacije za matematiku, kriptografiju, istraživanje algoritama, umjetnu inteligenciju, teoriju igara, multimedijsku obradu, filozofiju, ekonomiju i mnoga druga polja.

To je jedan od sedam zadataka Milenijumske nagrade koje je izabrao Clay Mathematics Institute, od kojih svaki nosi nagradu od 1.000.000 USD za prvo ispravno rješenje.

U primjenjenim kontekstima, termini "lako" i "teško" se obično tumače u odnosu na neki specifični računarski entitet; tipično "dovoljno jeftino za korisnike" i "previše skupo za zlonamjerne agente".

Jednosmjerne funkcije su ključni alati za mnoge primjene sigurnosti podataka, kao što su kriptografija, autentifikacija, i lična identifikacija. Iako je još uvijek otvoreno pitanje da li postoje jednosmjerne funkcije, postoji nekoliko kandidata koji su izdržali decenije intenzivnog ispitivanja.. Mnogi od tih kandidata se koriste u različitim sistemima širom svijeta, uključujući telekomunikacije, e-trgovinu i e-bankarstvo.

## Diffie–Hellman razmjena ključeva

Diffie-Hellman Razmjena ključeva je matematička metoda sigurne razmjene kriptografskih ključeva preko javnog kanala i jedan je od prvih protokola s javnim ključem koji je zamislio Ralph Merkle i nazvan po Whitfieldu Diffiju i Martinu Hellmanu. DH je jedan od prvih primjera razmjene javnih ključeva koji se praktično primjenjuju u kriptografiji. Godine 1976. Diffie i Hellman su objavili ovaj javno poznati rad koji je predložio koncept privatnog i odgovarajućeg javnog ključa.

|  |  |  |
| --- | --- | --- |
| Do sada, sigurna šifrirana komunikacija između dvije strane bila je moguća samo ako bi prethodno razmijenile ključeve pomoću sigurnih fizičkih sredstava, poput papirnih listova ključeva koje prenosi pouzdani kurir. Međutim, Diffie-Hellman metoda razmjene ključeva omogućava dvjema stranama koje nemaju prethodno znanje jedne o drugoj da uspostave zajednički tajni ključ putem nesigurnog kanala. Tada se taj ključ može koristiti za šifriranje buduće komunikacije koristeći simetrični ključ.  Protokol Diffie-Hellman koristi multiplikativnu grupu cijelih brojeva modulo p, gdje je p prost broj, a g primitivni korijen modulo p. Ova jednostavna i originalna implementacija je dizajnirana tako da omogući stvaranje zajedničkog tajnog ključa koji može poprimiti bilo koju vrijednost od 1 do p-1. | Slika 1 |  |

Evo primjera protokola, s javnim vrijednostima u plavoj, a tajnim vrijednostima u crvenoj:

1. Alisa i Bob se javno slažu da koriste modul *p* = 23 i bazu  *g* = 5 (što je primitivni korijen po modulu 23).
2. Alisa bira tajni cijeli broj ***a*** = 4, a zatim šalje Bobu *A* = *g****a*** mod *p*
   * *A* = 5**4** mod 23 = 4 ((u ovom primjeru i A and a imaju istu vrijednost 4, ali to obično nije slučaj)
3. Bob bira tajni cijeli broj ***b*** = 3, a zatim šalje Alisa *B* = *g****b*** mod *p*
   * *B* = 5**3** mod 23 = 10
4. Alisa izračunava ***s*** = *B****a*** mod *p*
   * ***s*** = 10**4** mod 23 = 18
5. Bob izračunava ***s*** = *A****b*** mod *p*
   * ***s*** = 4**3** mod 23 = 18
6. Alisa i Bob sada dijele tajnu (broj 18) (ilustrativno predstavljeno na slici 1)

Kada Alisa i Bob stvore zajedničku tajnu, oni mogu iskoristiti tu tajnu kao ključ za šifriranje poruka, koji je poznat samo njima, kako bi slali poruke putem istog javnog komunikacijskog kanala.

Metodu je ubrzo nakon toga slijedio RSA, implementacija kriptografije s javnim ključem korištenjem asimetričnih algoritama.

## RSA (kriptosistem)

RSA je široko rasprostranjen algoritam za kriptografiju javnog ključa. Bio je jedan od prvih praktičnih kriptosustava koji su se koristili za siguran prijenos podataka. Ovaj algoritam su osmislili **Ron Rivest, Adi Shamir i Leonard Adleman** 1977. godine te je dobio ime po njihovim inicijalima. RSA se koristi široko u raznim aplikacijama koje zahtijevaju sigurnost podataka.

U RSA, proces generisanja ključeva uključuje kreiranje dva ključa: javnog ključa i privatnog ključa. Javni ključ se koristi za šifriranje podataka, dok se privatni ključ koristi za dešifriranje podataka. Sigurnost RSA zasnovana je na matematičkim svojstvima velikih prostih brojeva i poteškoće faktoringa proizvoda dva velika prosta broja.

### Kako RSA radi?

Sve promjenjive spadaju u skup Z, odnosno skup cijelih brojeva (integer).

1. *Odaberite dva velika prosta broja:* Prvi korak u generiranju RSA para ključeva je odabir dva velika prosta broja, p i q. Ovi prosti brojevi su obično dugi najmanje 300 bita i biraju se nasumično.
2. *Izračunajte n = p \* q:* koristi se za definisanje veličine ključa. Broj n mora biti veći od veličine poruke.
3. *Izračunajte* *φ(n) = (p-1) \* (q-1):* φ(n) (čita se: pi od en) je totijent broja n. Totijent je broj pozitivnih cijelih brojeva koji nije veći od navedenog cijelog broja koji su relativno prosti prema njemu. Relativno prost broj M prema nekom broju N znači da M i N nemaju istih faktora. Npr: broj 9 je relativno prost broju 14 jer ne ih ne dijele isti faktori.
4. *Odaberite javni eksponent e:* e je cijeli broj koji je relativno prost sa n i sa φ(n) i koji mora biti 1 < e < φ(n).
5. *Izračunajte privatni eksponent d:* Nakon što je izabran javni eksponent, može se izračunati privatni eksponent. Formula za ovu vrijednost je d\*e mod φ(n) = 1. Privatni eksponent se čuva u tajnosti i koristi se za dešifriranje šifriranih podataka.
6. *Generirajte javni ključ (e,n):* Javni ključ se sastoji od modula n i javnog eksponenta e. Ove vrijednosti mogu biti javno dostupne jer se koriste za šifriranje podataka. Enkripcija se vrši: Plaintext^e mod n
7. *Generirajte privatni ključ (d,n):* Privatni ključ se sastoji od modula n i privatnog eksponenta d. Ovaj ključ se čuva u tajnosti i koristi se za dešifriranje šifriranih podataka. Dekripcija se vrši: Ciphertext^d mod n

## Hibridni kriptosistem

U kriptografiji, hibridni kriptosistem je onaj koji kombinuje pogodnost kriptosistema sa javnim ključem sa efikasnošću kriptosistema sa simetričnim ključem.

Kriptosistemi s javnim ključem pogodni su po tome što ne zahtijevaju od pošiljaoca i primaoca da dijele zajedničku tajnu kako bi bezbedno komunicirali. Oni često koriste kompleksne matematičke izračune i stoga su generalno mnogo sporiji od sličnih simetričnih kriptosistema.

U mnogim aplikacijama, visoka cijena šifriranja dugih poruka u kriptosistemu s javnim ključem može biti previsoka. Ovo rješavaju hibridni sistemi korištenjem kombinacije oba.

Hibridni kriptosistem se može konstruisati korišćenjem bilo koja dva odvojena kriptosistema:

* *mehanizam enkapsulacije ključa*, koji je kriptosistem sa javnim ključem, i
* *shema enkapsulacije podataka*, koja je kriptosistem sa simetričnim ključem.

Važno je napomenuti da kod veoma dugih poruka većinu posla u procesu šifriranja/dešifriranja obavlja efikasnija simetrična shema ključa, dok se sporija asimetrična shema koristi samo za šifriranje/dešifriranje kratkih ključeva.

Sve praktične primjene asimetrične kriptografije danas koriste hibridni sistem koji kombinuje oba tipa kriptografije. Primjeri uključuju TLS protokol i SSH protokol, koji koriste mehanizam javnog ključa za razmjenu ključeva (kao što je Diffie-Hellman) i mehanizam simetričnog ključa za enkapsulaciju podataka (kao što je AES).

### Mehanizam enkapsulacije ključa

U kriptografskim protokolima, *mehanizam enkapsulacije ključa* se koristi za osiguranje materijala pomoću simetričnog ključa, te za prijenos istog koristeći asimetrične algoritme. U praksi, asimetrični sistemi su nepraktični za prijenos dugih poruka. Umjesto toga, često se koriste za razmjenu simetričnih ključeva, koji su relativno kratki. Simetrični ključ se tada koristi za šifriranje duže poruke.

### Shema enkapsulacije podataka

|  |  |
| --- | --- |
| U kontekstu računarskih mreža, enkapsulacija je metoda dizajniranja modularnih komunikacijskih protokola. Ova metoda omogućuje da se logički odvojene funkcije mreže isključuju iz svojih osnovnih struktura kroz dodavanje ili skrivanje informacija unutar objekata višeg nivoa. Drugim riječima, enkapsulacija "uzima informacije iz višeg sloja i dodaje mu zaglavlje, tretirajući informacije višeg sloja kao podatke". | Slika 2 |

Enkapsulacija korisničkih podataka u UDP steku u Unix stilu podrazumijeva dodavanje novih slojeva informacija koji uključuju podatke iz prethodnih slojeva, ali bez mogućnosti identificiranja kojim dijelom podataka pripada zaglavlje iz prethodnog sloja. Ovim se postiže efektivno skrivanje (inkapsuliranje) informacija iz nižih slojeva (slika 2).

### Primjer hibridnog kriptosistema

Da bi šifrirao poruku upućenu Alisi u hibridni kriptosistem, Bob radi sljedeće:

1. Dobiva Alisin javni ključ.
2. Generira svježi simetrični ključ za shemu enkapsulacije podataka.
3. Šifruje poruku prema shemi enkapsulacije podataka, koristeći upravo generirani simetrični ključ.
4. Šifruje simetrični ključ prema shemi enkapsulacije ključa, koristeći Alisin javni ključ.
5. Šalje oba ova šifrirana teksta Alisi.

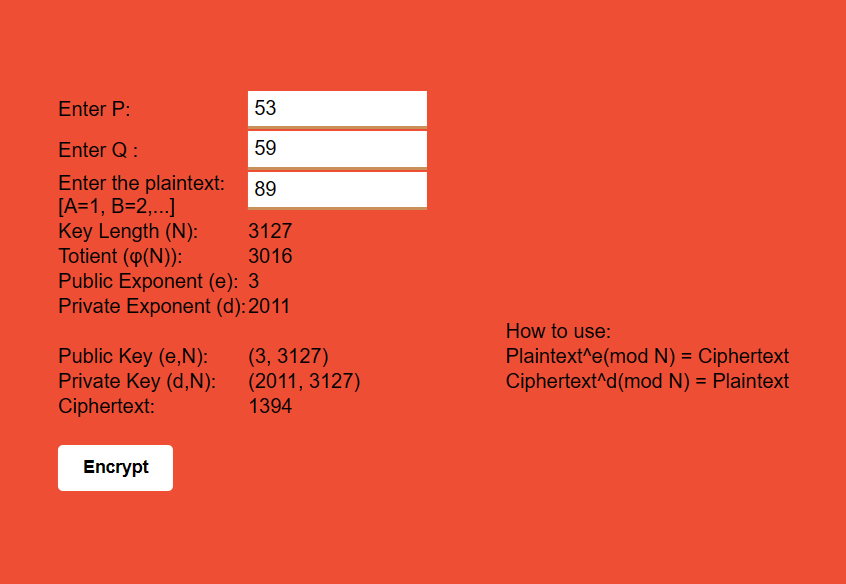
Da bi dešifrovala ovaj hibridni šifrovani tekst, Alisa radi sledeće:

1. Koristi svoj privatni ključ za dešifriranje simetričnog ključa sadržanog u segmentu enkapsulacije ključa.
2. Koristi ovaj simetrični ključ za dešifriranje poruke sadržane u segmentu enkapsulacije podataka.

# IMPLEMENTACIJA U JAVASCRIPTU

Za svoj projekat sam napravio implementaciju RSA algoritma u JavaScriptu. Na slici broj 3 možete vidjeti interfejs ove web aplikacije.

U JavaScriptu postoje mnoge biblioteke kako bi se lakše implementirao RSA algoritam, ali ja ih nisam koristio već sam napisao svoj program od nule za formiranje ključeva i pretvaranje plaintexta u ciphertext. Da bi se biblioteke koristile, potreban je Node.js i npm te odgovarajući paket. Ovo je dokaz da je JavaScript veoma pogodan za enkripciju te da se može napraviti puno toga sa njim, ali sa malim brojevima. JavaScript nije dobar za profesionalnu upotrebu jer koristi interpreter i dosta je sporiji od jezika sa pravim kompajlerima. Ipak se tu nalaze ogromne kalkulacije i brojevi koji su najmanje dugi 300 cifara. Programski jezik koji je napravljen samo da bi se u njemu moglo raditi dobro sa velikim brojevima i koji je namijenjen specifično za kriptografiju je Cryptol.



Slika 3

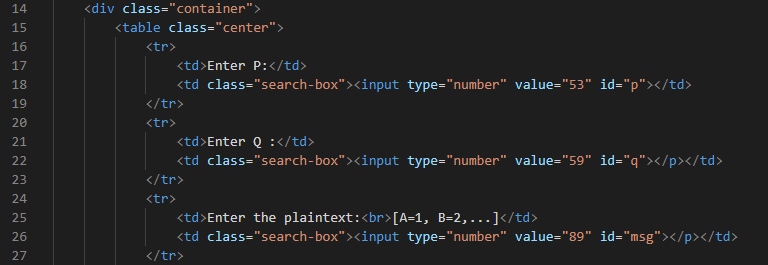
Korisnik unosi proste brojeve P i Q koje drži tajnim. Po potrebi, oni bi trebali biti čisto slučajni i mnogo veliki. Njihova veličina utiče na N od kojeg zavisi koliko karaktera plaintext može sadržavati. Zatim unosi svoju tajnu poruku (plaintext); u obliku brojeva, može koristiti ili ASCII tabelu ili da je svako abecedno slovo jednako broju koji odgovara poziciji tom slovu u abecedi.

Za to kako je svaka varijabla dobijena pogledajte poglavlje 4.6.1. Kako RSA radi?

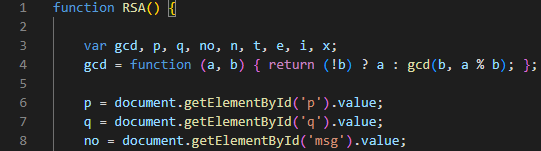
Program će korisniku dati sve moguće informacije i varijable koje su korištene za enkripciju a na dnu će ispisati i šifrovani tekst (ciphertext) što je i bit samog programa. Također su date najbitnije formule, odnosno kako šifrirati tekst ako imamo sve varijable.

Kao što možete primijetiti u primjeru na slici broj 3, javni eksponent je veoma mali, što čini plaintext laganim za šifriranje. Međutim, vidimo da je tajni eksponent mnogo veći što znači da će dekripcija ciphertexta biti toliko puta teža. Iako su P i Q još uvijek dvocifreni brojevi, da bi se ciphertext dešifrovao, morao bi se dići na stepen od 2011. P i Q za profesionalnu upotrebu imaju minimalno 300 cifara. To govori koliko je RSA algoritam moćan i kako je blizu da bude jednosmjerna funkcija. Mala računarska snaga treba za enkripciju, velika za dekripciju.

**Objašnjenje koda:**



Slika 4



Slika 5

Prvo, definiraju se neke varijable koje će se koristiti u postupku. Varijabla "gcd" je funkcija koja računa najveći zajednički djelitelj dva broja (slika 5). To je korisno za kasnije u postupku jer ćemo koristiti dva velika prosta broja i moramo provjeriti da su međusobno djeljivi samo s 1.

Varijabla "p" (slika 5) se postavlja na vrijednost unesenu u element s ID-om "p" u HTML dokumentu (slika 4). Ova vrijednost predstavlja jedan od dva velika prosta broja koji će biti korišteni za generiranje RSA ključa.

Varijabla "q" (slika 5) se postavlja na vrijednost unesenu u element s ID-om "q" u HTML dokumentu (slika 4). Ova vrijednost predstavlja drugi veliki prosti broj koji će biti korišten za generiranje RSA ključa.

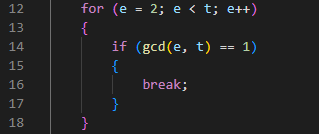
Varijabla "no" (slika 5) se postavlja na vrijednost unesenu u element s ID-om "msg" u HTML dokumentu (slika 4). Ova vrijednost predstavlja poruku koju želimo kriptirati pomoću RSA algoritma.



Slika 6

Nakon što su dobijeni prosti brojevi p i q, slijedi izračunavanje njihovog proizvoda kako bi se dobilo n, veličina ključa (slika 6).

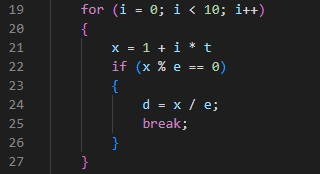
Nakon toga, varijabla "t" se izračunava kao proizvod (p-1) i (q-1) (slika 6). Ta vrijednost je tocijent od n. Za detaljnije objašnjenje pogledajte poglavlje 4.6.1. Kako RSA radi?, korak 3.



Slika 7

Nakon što je izračunata vrijednost "t", u ovom dijelu koda se provodi postupak odabira broja "e", javnog eksponenta.

Petlja "for" se izvodi za vrijednosti "e" od 2 do "t", pokušavajući pronaći broj koji je relativno prost s "t". U petlji se poziva funkcija "gcd" koja računa najveći zajednički djelitelj između brojeva "e" i "t". Ako je taj najveći zajednički djelitelj jednak 1, to znači da su brojevi relativno prosti i prekidamo petlju pomoću naredbe "break" (slika 7).



Slika 8

Nakon što je odabran javni ključ (e,n), slijedi postupak generisanja privatnog eksponenta "d".

Petlja "for" se izvodi za vrijednosti "i" od 0 do 9, pokušavajući pronaći broj "x" koji zadovoljava uvjet da je djeljiv s "e". Prvo se računa "x" kao 1 + i \* t. Ako je "x" djeljiv s "e", to znači da postoji broj "d" koji je privatni eksponent. Vrijednost "d" se dobiva dijeljenjem "x" s "e". Petlja se prekida pomoću naredbe "break" kada se pronađe prvi broj "x" koji zadovoljava uvjete (slika 8).

Čitav ovaj algoritam je povezan sa formulom d\*e mod φ(n) = 1 pomoću koje se nalazi "d".

Sada imamo i privatni ključ (d,n).



Slika 9

Nakon što su generisani javni i privatni ključevi, može se kriptirati poruka.

U ovom dijelu koda, ulazna poruka "no" se kriptira koristeći javni ključ "e" i modul "n". Naredba "Math.pow(no, e)" računa poruku na "e"-tu potenciju. Funkcija "toFixed(0)" zatim zaokružuje tu vrijednost na najbliži cijeli broj. Zatim se računa kriptirana poruka "ct" pomoću operatora modulo (%), koji računa ostatak dijeljenja "ctt" s "n" (slika 9).

Konačno, vrijednost "ct" predstavlja kriptiranu verziju ulazne poruke "no" i to je ono što se može sigurno poslati preko javnog kanala, jer bez privatnog ključa "d" nije moguće dekriptirati ovu poruku.

Na kraju, imamo još kod za ispis koji nije toliko kompleksan pa ga neću prikazati. Koristio sam GetElementById kako bi iz HTML-a mogao uzeti sve varijable koje mi trebaju i zatim ispisati ih na web stranici.

# ZAKLJUČAK

Kako privodim ovaj rad kraju, moje znanje se znatno proširilo te sam čak dobio veliki interes za kriptografiju općenito i sada mi je mnogo jasnije na čemu je većina internetske sigurnosti zasnovana. Sada ću ukratko ponoviti najbitnije što se tiče asimetrične enkripcije u JavaScriptu.

Asimetrična enkripcija je ključna tehnologija za sigurnu razmjenu podataka u modernom digitalnom dobu. Zaštita povjerljivosti, integriteta i autentičnosti podataka putem javnih i privatnih ključeva je neophodna u različitim sektorima, uključujući financijski sektor, zdravstvo, e-trgovinu i državne institucije.

Ključna prednost asimetrične enkripcije je upotreba javnog ključa za enkripciju podataka, što znači da se privatni ključ koristi samo za dekripciju podataka. Ovo sprječava da se privatni ključ otkrije tijekom prijenosa podataka, što bi ugrozilo sigurnost podataka.

Uz to, važno je naglasiti da je implementacija asimetrične enkripcije samo jedan dio cjelokupne sigurnosne strategije. Zaštita od napada i zlonamjernih aktivnosti zahtijeva integraciju sigurnosnih mjera na više razina, uključujući upravljanje identitetom i pristupom, detekciju i prevenciju napada, te zaštitu od softverskih ranjivosti.

Sve u svemu, asimetrična enkripcija igra ključnu ulogu u osiguravanju privatnosti i sigurnosti digitalnih komunikacija i podataka. Stoga se očekuje da će njezina primjena i dalje rasti u budućnosti kako se digitalno društvo sve više razvija.

U JavaScriptu postoje mnoge biblioteke koje olakšavaju generiranje ključeva, ali da biste ih koristili, potreban vam je Node.js i odgovarajući npm paket. Iako se dosta toga u JavaScriptu može napraviti u polju kriptografije s manjim brojevima, JavaScript nije pogodan za profesionalnu upotrebu jer koristi interpreter koji je dosta sporiji od jezika s pravim kompajlerima.

Moje mišljenje je to da je asimetrična enkripcija kao koncept jedan od najbitnijih izuma u zadnjih 100 godina, jer on omogućava koncept interneta, povezanosti svih ljudi svijeta, da bude bezazlen i siguran. Iako i dalje postoji uspješnih hakera koji svaki dan kradu podatke i klasificirane informacije, bez asimetrične enkripcije, svaka šifra, svaki e-mail i svaki podatak bio javni ili tajni bi bio dostupan prosječnoj osobi, sa prosječnim znanjem, sa prosječnom mašinom, što bi dovelo do krajnjeg kaosa i internet kakav ga danas poznajemo ne bi postojao.

# LITERATURA

[https://www.zdnet.com](https://www.zdnet.com/) (<https://www.zdnet.com/article/gchq-pioneers-on-birth-of-public-key-crypto/>), mart, 2023.

<https://www.britannica.com> (<https://www.britannica.com/topic/public-key-cryptography>, <https://www.britannica.com/topic/cipher>, <https://www.britannica.com/topic/transposition-cipher>, <https://www.britannica.com/topic/substitution-cipher>, <https://www.britannica.com/topic/RSA-encryption>), mart, 2023.

[www.wikipedia.org](http://www.wikipedia.org) (<https://en.wikipedia.org/wiki/Public-key_cryptography>; <https://en.wikipedia.org/wiki/One-way_function>; <https://en.wikipedia.org/wiki/P_versus_NP_problem>; <https://en.wikipedia.org/wiki/Caesar_cipher>; <https://en.wikipedia.org/wiki/Non-repudiation>, <https://en.wikipedia.org/wiki/Substitution_cipher>, <https://en.wikipedia.org/wiki/Transposition_cipher>, <https://en.wikipedia.org/wiki/Modular_arithmetic>, <https://en.wikipedia.org/wiki/One-time_pad>, <https://en.wikipedia.org/wiki/RSA_(cryptosystem)>, <https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange>, <https://en.wikipedia.org/wiki/Key_exchange>, <https://en.wikipedia.org/wiki/Modular_exponentiation>, <https://en.wikipedia.org/wiki/Merkle%27s_Puzzles>, <https://en.wikipedia.org/wiki/Hybrid_cryptosystem>, <https://en.wikipedia.org/wiki/Key_encapsulation_mechanism>, https://en.wikipedia.org/wiki/Encapsulation\_(networking)), februar, 2023.

**KOMENTAR**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Datum predaje rada: 10. mart 2023. godine

Datum odbrane rada: \_\_\_. \_\_\_\_\_ 2023. godine

Ocjena: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Članovi Komisije:

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, predsjednik

2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, član

3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, član

1. Clifford Cocks, 1973. [↑](#footnote-ref-1)