Kriptografski postupci zasnovani na eliptičnim krivuljama

Turković, Deni

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

KRIPTOGRAFSKI POSTUPCI ZASNOVANI NA ELIPTIČNIM KRIVULJAMA

Rijeka, rujan 2023.

Deni Turković

0069058540

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

KRIPTOGRAFSKI POSTUPCI ZASNOVANI NA ELIPTIČNIM KRIVULJAMA

Mentor: izv. prof. dr. sc. Jonatan Lerga

Rijeka, rujan 2023.

Deni Turković

0069058540

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 21. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Kodiranje i kriptografija**
Grana: **2.09.03 obradba informacija**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Deni Turković (0069058540)**
Studij: Diplomski sveučilišni studij računarstva
Modul: Računalni sustavi

Zadatak: **Kriptografski postupci zasnovani na eliptičnim krivuljama / Elliptic Curve Based Cryptography Methods**

Opis zadatka:

U radu je potrebno opisati teoretske osnove kriptografskih postupaka koji koriste eliptične krivulje. Nadalje, potrebno je objasniti zašto i u kojim kriptografskim sustavima i procesima se koriste eliptične krivulje te koje su njihove prednosti i nedostaci. Također potrebno je izraditi programsko rješenje kriptografskog sustava zasnovanog na eliptičnim krivuljama.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Deni Turković

Zadatak uručen pristupniku: 21. ožujka 2022.

Mentor:

Izv. prof. dr. sc. Jonatan Lerga

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam ovaj rad izradio samostalno uz primjenu stečenih znanja na diplomskom sveučilišnom studiju računarstva pod mentorstvom izv. prof. dr. sc. Jonatana Lerge.

Rijeka, rujan 2023.

_____

Deni Turković

ZAHVALA

Zahvaljujem se mentoru izv. prof. dr. sc. Jonatanu Lergi na povjerenju, podršci, prijedlozima i uputama koje mi je pružao tijekom studiranja i izrade ovog rada. Također se zahvaljujem roditeljima na podršci i strpljenju.

SADRŽAJ

1 UVOD

Potreba da se informacije zaštite od trećih strana ili konkurenata, potaknula je korištenje brojnih strategija, od kojih je jedna kodiranje razmijenjenih poruka u nečitljiv format koji mogu obraditi i razumjeti samo oni kojima je ta poruka namijenjena. Iako je ovo učinkovito funkcioniralo za špijune tijekom Drugog svjetskog rata, pitanje je što se događa kada se, nazovimo ih naši tajni glasnici, ne mogu unaprijed dogovoriti oko ključa? Kriptografija temeljena na javnom ključu poznata je kao asimetrična kriptografija, stvorena kako bi dvije strane mogle sigurno komunicirati bez prethodnog susreta radi dogovora o zajedničkom ključu za šifriranje ili rizika od presretanja ključa preko nezaštićenih komunikacijskih kanala.

Razvojem interneta sve više surađujemo s entitetima koje ne poznajemo i kojima ne vjerujemo. Stoga je porast komunikacije putem računalne mreže promijenio običnog korisnika kriptografije, a zahtjev za izvršavanjem čestih transakcija s različitim stranama učinio je kriptografiju temeljenu na privatnom ključu, poznatu kao simetrična kriptografija, zastarjelom, ranjivom i manje pouzdanom. Kriptografija se razvila tako da uključuje složene algoritme i protokole te kao rezultat toga, kriptografija s javnim ključem sada se koristi za osiguranje velike većine naših sustava.

U ovom radu pobliže ćemo upoznati kriptografiju s javnim ključem zasnovanu na eliptičnim krivuljama, matematičkom analizom samih eliptičnih krivulja na kojoj se temelje. Također proučit ćemo procese i algoritme koji se koriste te ćemo objasniti pojedine koje koriste eliptične krivulje za generiranje i razmjenu ključeva, šifriranje i dešifriranje poruka.

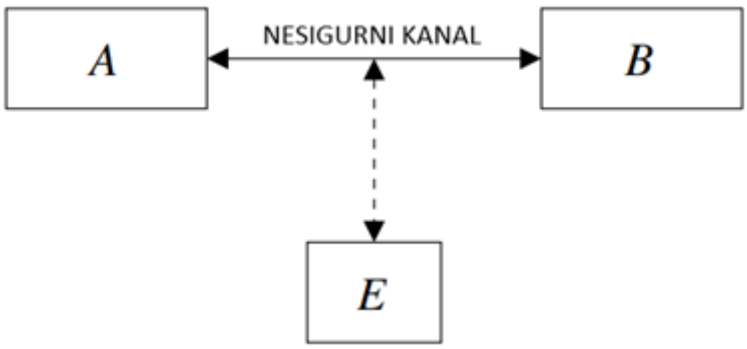
2 KRIPTOGRAFSKI SUSTAVI

Kriptografija se bavi dizajnom i analizom matematičkih tehnika koje omogućuju sigurnu komunikaciju u prisutnosti zlonamjernih sudionika dok je kriptografski sustav računalni sustav koji koristi kriptografiju. Kako bi podaci bili sigurni, kriptosustavi uključuju tehnike šifriranja i dešifriranja, metode za digitalne potpise, kriptografske hash funkcije i tehnike upravljanja ključevima.

U središtu kriptografskih operacija je kriptografski ključ, niz bitova koje koristi kriptografski algoritam za transformaciju otvorenog teksta u šifrirani tekst ili obrnuto. Ključ je dio varijabilnih podataka danih kao ulaz u kriptografski algoritam za izvršavanje ove vrste operacije. Sigurnost kriptografske sheme ovisi o sigurnosti korištenih ključeva.

2.1 Osnovni komunikacijski model

Entiteti A i B komuniciraju preko nezaštićenog kanala. Pretpostavljamo da se sve komunikacije odvijaju u prisutnosti sudionika E čiji je cilj probiti sve sigurnosne usluge koje se pružaju A i B (Slika 2.1).



Slika 2.1 Osnovni komunikacijski model

Na primjer, A i B mogu biti dvoje ljudi koji komuniciraju putem mobilne telefonske mreže, a E pokušava prisluškivati njihov razgovor. Isto tako A može biti web preglednik pojedinca A koji je u procesu kupnje proizvoda iz internetske trgovine B koju predstavlja njegova web stranica B. U ovom scenariju, komunikacijski kanal je internet. Sudionik E bi

mogao pokušati pročitati promet od A do B i tako saznati podatke o kreditnoj kartici A ili bi mogao pokušati oponašati A ili B u transakciji. Kao treći primjer, zamislimo situaciju u kojoj A šalje poruku e-pošte prema B putem Interneta. Sudionik E mogao bi pokušati pročitati poruku, modificirati odabrane dijelove ili oponašati A slanjem vlastitih poruka B. Razmotrimo još scenarij u kojem je A pametna kartica koja je u procesu provjere autentičnosti svog vlasnika A glavnom računalu B u sjedištu banke. Ovdje E može pokušati nadzirati komunikaciju kako bi dobio informacije o A-ovom računu ili bi mogao pokušati oponašati A kako bi povukao sredstva s njegovog računa. Iz ovih primjera je vidljivo da entitet koji komunicira nije nužno čovjek, već može biti računalo, pametna kartica ili softverski modul koji djeluje u ime pojedinca ili organizacije kao što je trgovina ili banka.

2.2 Sigurnosni ciljevi

Pregled gore navedenih scenarija otkriva sljedeće temeljene ciljeve sigurne komunikacije:

- povjerljivost – čuvanje podataka u tajnosti od svih osim onih koji su ovlašteni da ih vide. Poruke koje šalje A do B, E ne bi smio moći čitati.
- integritet – osiguranje da podaci nisu izmijenjeni neovlaštenim sredstvima. B bi trebao moći otkriti kada je E izmijenio podatke koje je poslao A.
- provjera autentičnosti izvora podataka – potvrđivanje izvora podataka. B bi trebao moći provjeriti da podaci koje je navodno poslao A doista potječu od A.
- autentifikacija – potvrđivanje identiteta entiteta. B bi trebao biti uvjeren u identitet drugog entiteta s kojim komunicira.
- neporecivost – sprječavanje entiteta da porekne prethodne radnje. Kada B primi poruku navodno od A, ne samo da je B uvjeren da je poruka potekla od A, već B može u to uvjeriti neutralnu treću stranu; stoga A ne može poreći da je poslao poruku B-u.

Neke aplikacije mogu imati druge sigurnosne ciljeve kao što je anonimnost subjekata koji komuniciraju ili kontrola pristupa [1].

Kako bismo modelirali realne prijetnje s kojima se suočavaju A i B, pretpostavimo da protivnik E ima značajne sposobnosti. Osim što može čitati sve podatke koji se prenose putem kanala, E može modificirati prenesene podatke i ubaciti svoje podatke. Štoviše, E ima na raspolaganju značajne računalne resurse, potpuni opis komunikacijskih protokola i svih

primijenjenih kriptografskih mehanizama, osim informacija o tajnom ključu. Izazov za kriptografe je dizajnirati mehanizme za sigurnu komunikaciju u slučaju tako moćnih neprijatelja.

2.3 Simetrični kriptosustavi

Kriptografski sustavi mogu se općenito podijeliti u dvije vrste. U shemama sa simetričnim ključevima (Slika 2.2.), entiteti koji komuniciraju najprije se dogovore o ključu koji je i tajan i autentičan. Naknadno mogu koristiti shemu šifriranja sa simetričnim ključem kao što je DES (engl. *Data Encryption Standard*), RC4 (engl. *Rivest Cipher 4*) ili AES (engl. *Advanced Encryption Standard*) za postizanje povjerljivosti. Također mogu koristiti algoritam koda za provjeru autentičnosti poruke kao što je HMAC (engl. *Hash-based Message Authentication Cod*e) za postizanje integriteta podataka i provjeru autentičnosti podataka.

Slika 2.2 Primjer simetrične enkripcije [2]

Ako je k, ključ koji dijele A i B i želimo povjerljivost poruke m, tada bi A šifrirao poruku m koristeći funkciju šifriranja ENC (*engl. Encription Scheme*) i ključ k, te poslao rezultirajući šifrirani tekst $c = ENC_k(m)$ k B-u. Kod primitku šifriranog teksta c, B bi koristio funkciju dešifriranja DEC (*engl. Decryption scheme*) i isti ključ k za oporavak $m = DEC_k(c)$.

Ukoliko se želi postići integritet podataka i provjera autentičnosti podataka, tada bi se A i B prvo dogovorili oko tajnog ključa k, nakon čega bi A izračunao oznaku provjere autentičnosti (*engl. tag*) $t = MAC_k(m)$ poruke m koristeći MAC (engl. *Message Authentication Code*), algoritam i ključ k. A bi tada poslao B-u poruku m i oznaku t. Po primitku m i t, B bi upotrijebio MAC algoritam i isti ključ k za ponovno izračunavanje oznake $t = MAC_k(m)$ i prihvatio poruku kao da potječe od A ako je $t = t$ [1].

2.3.1 Distribucija i upravljanje ključevima

Glavna prednost kriptografije sa simetričnim ključem je visoka učinkovitost, međutim, postoje veliki nedostatci ovakvih sustava. Primarni nedostatak je takozvani problem distribucije ključeva, odnosno zahtjev za kanalom koji je i tajan i autentificiran za razmjenu ključeva. Za neke se aplikacije ova distribucija može praktično obaviti korištenjem fizički sigurnog kanala kao što je pouzdana dostava. Drugi način je korištenje „on-line“ usluga treće strane od povjerenja koja inicijalno uspostavlja tajne ključeve sa svim entitetima u mreži i zatim koristi te ključeve za sigurnu distribuciju ključeva entitetima koji po potrebi međusobno komuniciraju. Rješenja poput ovih mogu biti dobro prilagođeni okruženjima u kojima postoji prihvaćeno i pouzdano središnje tijelo, ali su očito nepraktični u aplikacijama kao što je e-pošta putem Interneta.

Drugi nedostatak je problem upravljanja ključem. U mreži od N entiteta, svaki entitet ima različit ključ sa svakim od ostalih $N - 1$ entiteta. Ovaj se problem može ublažiti upotrebom „on-line“ usluga treće strane od povjerenja koja distribuira materijal za ključeve prema potrebi, čime se smanjuje potreba entiteta da sigurno pohranjuju više ključeva. Međutim, takva rješenja nisu praktična u nekim scenarijima. Budući da se ključ dijeli između dva ili više entiteta, tehnike simetričnog ključa ne mogu se koristiti za izvedbu shema digitalnog potpisa koje pružaju neporecivost. To je zato što je nemoguće razlikovati akcije koje poduzimaju različiti vlasnici tajnog ključa.

2.4 Asimetrični kriptosustavi

Pojam kriptografije s javnim ključem (Slika 2.3.) uveli su Diffie, Hellman i Merkle kako bi riješili gore navedene nedostatke kriptografije sa simetričnim ključem [1]. Za razliku od shema simetričnog ključa, sheme javnog ključa zahtijevaju samo da entiteti koji komuniciraju razmijene javne ključeve koji su autentični. Svaki entitet odabire jedan par ključeva (e, d) koji se sastoji od javnog ključa e i povezanog privatnog ključa d koji entitet drži u tajnosti. Ključevi imaju svojstvo da je računalno neizvedivo odrediti privatni ključ samo na temelju znanja javnog ključa.

Slika 2.3 Primjer enkripcije javnim ključem [2]

2.4.1 Povjerljivost

Ako entitet A želi poslati entitetu B povjerljivu poruku m, on dobiva kopiju B-ovog javnog ključa e_B i koristi shemu šifriranja s javnim ključem ENC za izračunavanje šifriranog teksta $c = ENC_{e_B}(m)$. A zatim šalje šifrirani tekst c k B-u, koji koristi funkciju dešifriranja DEC i svoj privatni ključ d_B za vraćanje u izvorni tekst: $m = DEC_{e_B}(c)$. Pretpostavka je da sudionik kojemu je poznat samo e_B, ne može dešifrirati c. Imajmo na umu da na e_B nema zahtjeva za tajnošću. Bitno je samo da A dobije autentičnu kopiju e_B, inače bi A šifrirao m

korištenjem javnog ključa e_E nekog entiteta E koji se predstavlja kao B, a m bi mogao dešifrirati E [1].

2.4.2 Neporecivost

Sheme digitalnog potpisa mogu se koristiti za autentifikaciju izvora podataka i integritet podataka, te za olakšavanje pružanja usluga neporicanja. Entitet A koristio bi shemu za generiranje potpisa i koristio svoj privatni ključ d_A za izračunavanje potpisa poruke: $s = SIGN_{d_A}(m)$. Nakon primanja poruke m i potpisa s, entitet B koji ima autentičnu kopiju javnog ključa A, e_A, koristi algoritam za provjeru potpisa kako bi potvrdio da je s doista generiran iz m i d_A. Budući da d_A vjerojatno zna samo A, B je siguran da je poruka doista potekla od A. Štoviše, budući da verifikacija zahtijeva samo poruku m i javni ključ e_A, potpis s za poruku m također može verificirati treća strana koja može riješiti sporove ako A porekne da je potpisao poruku m. Za razliku od rukom pisanih potpisa, A-ov potpis s ovisi o poruci m koju potpisuje, sprječavajući krivotvoritelja da jednostavno doda s drugoj poruci m i tvrdi da je A potpisao m [1].

3 MATEMATIČKA ANALIZA

3.1 Konačna polja

Konačno polje sastoji se od konačnog skupa objekata koji se nazivaju elementi polja. Postoje dvije operacije koje se mogu izvesti na parovima elemenata, zbrajanje i množenje. Ove operacije moraju imati određena svojstva [3]. Konačno polje koje sadrži q elemenata polja, postoji ako i samo ako je q potencija prostog broja. Zatim, za svaki takav q postoji točno jedno konačno polje. Ono konačno polje koje sadrži q elemenata označava se s F_q. Koristiti ćemo dvije vrste konačnih polja. F_p s prostim brojem q = p, koje nazivamo primarna konačna polja, a konačna polja F_{2^m} s $q = 2^m$ za $m \geq 1$ koja se nazivaju binarna konačna polja. Kako bi se kriptografske sheme na temelju ECC-a (engl. *Elliptic Curve Cryptography)* precizno specificirale potrebno je pobliže opisati navedena konačna polja.

3.1.1 Primarna konačna polja F_q

Konačno polja F_p je primarno konačno polje koje sadrži p elemenata. Iako postoji samo jedno primarno konačno polje F_p za svaki prost broj p, postoji mnogo načina za predstavljanje elemenata F_p.

Konačno polje F_p izraženo je skupom cijelih brojeva $\{0, 1, \ldots, \mathrm{p} - 1\}$ čije je zbrajanje i množenje definirano na sljedeći način :

- zbrajanje: ako su $a, b \in F_p$ tada je $a + b = r \in F_p$ gdje je $r \in [0, p-1]$ ostatak kada je cijeli broj $a + b$ podijeljen s p. Ovo je poznato kao zbrajanje po modulu p što možemo zapisati kao $a + b = r \ (mod\ p)$.
- množenje: ako su $a, b \in F_p$ tada je $ab = s \in F_p$ gdje je $s \in [0, p-1]$ ostatak kada je cijeli broj ab podijeljen s p. Ovo je poznato kao množenje po modulu p što možemo zapisati kao $ab = s \ (mod\ p)$.

Zbrajanje i množenje u F_p može se učinkovito izračunati korištenjem standardnih aritmetičkih funkcija. U ovom obliku F_p, aditivni identitet ili nulti element je cijeli broj 0, a multiplikativni identitet je cijeli broj 1 [3].

Oduzimanje i dijeljene elemenata polja možemo definirati kao oduzimanje i dijeljenje cijelih brojeva. Da bi to učinili, moramo opisati inverz zbrajanja i inverz množenja:

- inverz zbrajanja: ako je a ∈ F_p, tada je inverz zbrajanja (−a) ∈ F_p jedinstveno rješenje jednadžbe a + x = 0 (mod p).
- inverz množenja: ako je a ∈ Fp, $a \neq 0$, tada je inverz množenja $a^{-1} \in F_p$ jedinstveno rješenje jednadžbe ax = 1 (mod p).

Inverz zbrajanja i inverz množenja u F_p mogu se učinkovito izračunati. Inverz množenja se može izračunati pomoću proširenog Euklidovog algoritma. Dijeljenje i oduzimanje su definirani kao inverz zbrajanja i množenja: $a - b \, mod \, p$ je $a + (-b)$, a $a \div b \, mod \, p$ je $a(b^{-1})$ [3].

3.1.2 Binarna konačna polja F_{2^m}

Konačno polje F_{2^m} je binarno konačno polje koje sadrži 2^m elemenata. Iako postoji samo jedno karakteristično konačno polje F_{2^m} za svaku potenciju 2^m gdje je $m \geq 1$, postoji više različitih načina predstavljanja elemenata F_{2^m}. Elementi F_{2^m} izraženi su skupom binarnih polinoma stupnja $m - 1$ ili manje:

$$\{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0;\ a_0 : a_i \in \{0,1 \ldots i\}\}$$

Zbrajanje i množenje definirano u terminima prostog binarnog polinoma f(x) stupnja m, poznat kao redukcijski polinom:

- zbrajanje: ako je $a = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0, b = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1x + b_0 \in F_{2^m}$, tada slijedi da je $a + b = r \in F_{2^m}$ gdje je $r = r_{m-1}x^{m-1} + r_{m-2}x^{m-2} + \cdots + r_1x + r_0$ pri čemu je $r_i = a_i + b_i (mod\ 2)$.
- množenje: ako je $a = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0, b = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1x + b_0 \in F_{2^m}$, tada slijedi da je $ab = s \in F_{2^m}$ gdje je $s = s_{m-1}x^{m-1} + s_{m-2}x^{m-2} + \cdots + s_1x + s_0$ ostatak kada se polinom ab podijeli s $f(x)$ sa svim aritmetičkim koeficijentima izvedenim po modulu 2.

Zbrajanje i množenje u F_{2^m} može se učinkovito izračunati korištenjem standardnih aritmetičkih funkcija. U ovom obliku F_{2^m}, aditivni identitet ili nulti element je cijeli broj 0, a multiplikativni identitet je cijeli broj 1 [3].

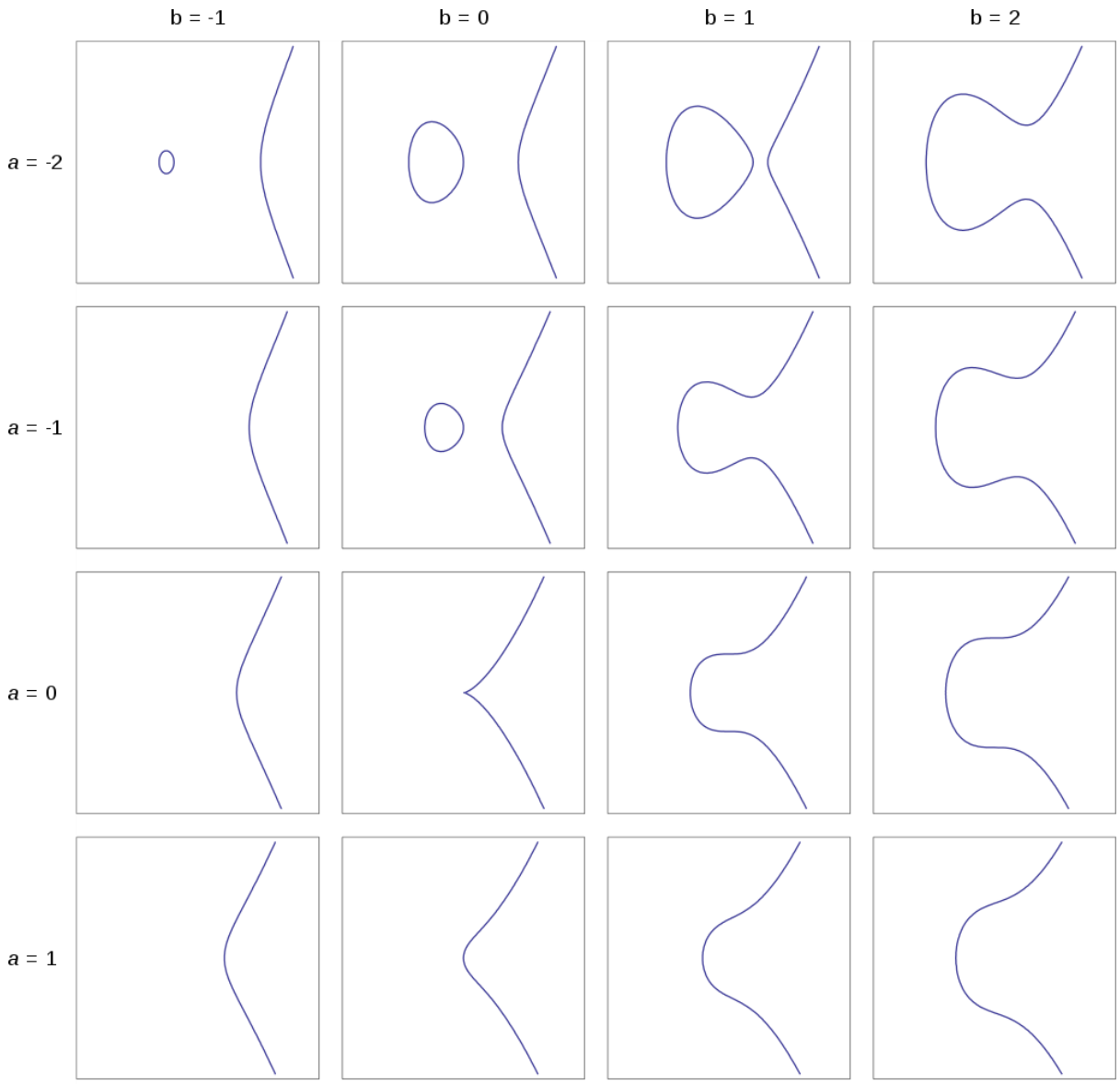
Oduzimanje i dijeljenje elemenata definirati ćemo kao inverz zbrajanja i inverz množenja elementa polja:

- inverz zbrajanja: ako je a $\in F_{2^m}$, tada je inverz zbrajanja $(-\mathrm{a}) \in F_{2^m}$ jedinstveno rješenje jednadžbe $\mathrm{a} + \mathrm{x} = 0$. Primjetimo da je je $-a = a$ za sve $a \in F_{2^m}$.
- inverz množenja: ako je a $\in F_{2^m}, a \neq 0$, tada je inverz množenja a^{-1} od $a \in F_{2^m}$ jedinstveno rješenje jednadžbe $\mathrm{ax} = 1 \in F_{2^m}$.

Inverz zbrajanja i množenja u polju F_{2^m} mogu se učinkovito izračunati. Inverz množenja se može izračunati pomoću proširenog Euklidovog algoritma. Dijeljenje i oduzimanje su definirani kao inverz zbrajanja i množenja: $a - b \in F_{2^m}$ je $a + (-b)$, a $a \div b \in F_{2^m}$ je $a(b^{-1}) \in F_{2^m}$.

3.2 Eliptične krivulje

Eliptične krivulje nisu krivulje u obliku elipse kao što bi to pretpostavili po nazivu, već su skup točaka koje zadovoljavaju određenu jednadžbu kao na primjer $y^2 = x^3 + ax + b$. Slika 3.1 prikazuje nekoliko krivulja sa zadanim koeficijentima a i b na području $x, y \in [-3, 3]$. Jedino za koeficijente $a = 0$ i $b = 0$ ne možemo reći da je krivulja eliptična jer nije glatka, odnosno ima šiljak.



Slika 3.1 Primjer potencijalnih eliptičnih krivulja [4]

3.2.1 Eliptične krivulje u polju F_p

Neka je F_p konačno polje tako da je p prost broj i neka $a, b \in F_p$ zadovoljava $4a^3 + 27b^2 \neq 0 \ (mod\ p)$. Tada se eliptična krivulja $E(F_p)$ u polju F_p definirana parametrima $a, b \in F_p$ sastoji od skupa rješenja ili točaka $P = (x, y)$ za $x, y \in F_p$ po jednadžbi:

$$y^2 = x^3 + ax + b \ (mod\ p)$$

zajedno s dodatnom točkom $\mathcal{O}$ koju nazivamo točkom beskonačnosti. Točka beskonačnosti je zamišljena točka koja leži na pravcu okomitom na točku P, te zapravo ne postoji u xy ravnini.

Jednadžba $y^2 = x^3 + ax + b\ (mod\ p)$ je definicijska jednadžba $E(F_p)$. Za danu točku $P = (x_p, y_p)$, x_p je x-kordinata, a y_p y-koordinata točke P.

Broj točaka na $E(F_p)$ označavamo s $\#E(F_p)$. Hasseov teorem daje nam točnu informaciju o redu grupe:

$$p + 1 - 2\sqrt{p} \leq \#E(F_p) \leq p + 1 + 2\sqrt{p}$$

Definirana su pravila zbrajanja za točke na E (Slika 3.2.) [5] . Pravila su sljedeća:

1. zbrajanje dvije točke u beskonačnosti:

$$\mathcal{O} + \mathcal{O} = \mathcal{O}$$

2. zbrajanje točke u beskonačnosti s bilo kojim drugom točkom:

$$(x, y) + \mathcal{O} = \mathcal{O} + (x, y) = (x, y) \text{ za sve } (x, y) \in E(F_p)$$

3. zbrajanje točke s točkom koja ima istu x-koordinatu, a suprotnu y-koordinatu (njezinim negativom):

$$(x, y) + (x, -y) = \mathcal{O} \text{ za sve } (x, y) \in E(F_p)$$

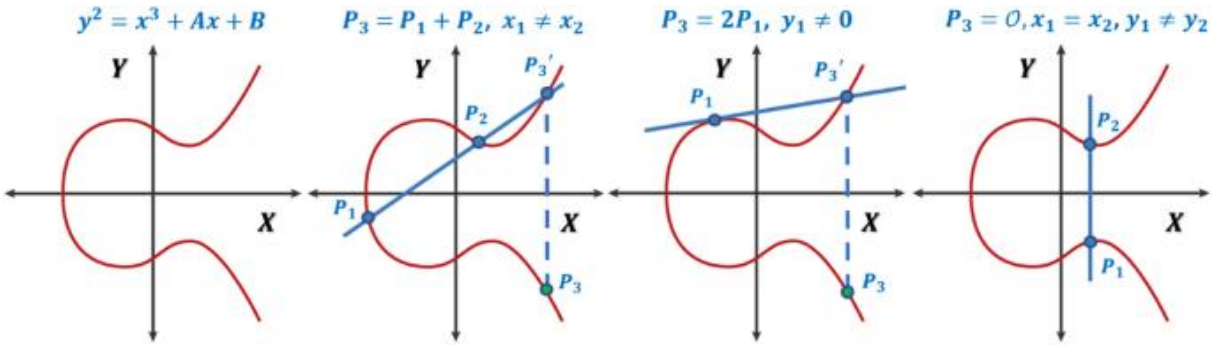
   Napomena: negativ točke (x, y) je $-(x, y) = (x, -y)$

4. zbrajanje dvije točke s različitim x-koordinatama: neka su $(x_1, y_1) \in E(F_p)$ i $(x_2, y_2) \in E(F_p)$ dvije točke tako da vrijedi $x_1 \neq x_2$. Tada je $(x_1, y_1) + (x_2, y_2) = (x_3, y_3) \in E(F_p)$ gdje je:

$$x_3 = \lambda^2 - x_1 - x_2, y_3 = \lambda(x_1 - x_2) - y_1, \lambda = \frac{y_2 - y_1}{x_2 - x_1}.$$

5. zbrajanje točke sa samom sobom(dupliciranje): Neka $(x_1, y_1) \in E(F_p)$ bude točka s $y_1 \neq 0$. Tada vrijedi $(x_1, y_1) + (x_1, y_1) = (x_3, y_3) \in E(F_p)$ gdje je:

$$x_3 = \lambda^2 - 2x_1, y_3 = \lambda(x_1 - x_3) - y_1, \lambda = \frac{3{x_1}^2 + a}{2y_1}$$



Slika 3.2 Primjer zbrajanja točaka eliptične krivulje [5]

Skup točaka na $E(F_p)$ prema pravilu zbrajanja tvori grupu. Stvorena grupa je Abelova grupa, što znači vrijedi $P_1 + P_2 = P_2 + P_1$ za sve točke $P_1, P_2 \in E(F_p)$. Pravilo zbrajanja se uvijek može učinkovito izračunati pomoću jednostavne aritmetike polja [3, 6].

Kriptografske sheme bazirane na ECC oslanjaju se na skalarno množenje točaka eliptične krivulje. S obzirom na cijeli broj k i točku $P \in E(F_p)$ skalarno množenje je proces dodavanja (zbrajanja) točke P sa samom sobom k puta.

3.2.2 Eliptične krivulje u polju F_{2^m}

Neka je F_{2^m} binarno konačno polje i neka $a, b \in F_{2^m}$ zadovoljava $b \neq 0$. Tada se eliptična krivulja $E(F_{2^m})$ u polju definirana parametrima $\mathrm{a}, \mathrm{b} \in F_{2^m}$ sastoji od skupa rješenja ili točaka $P = (x, y)$ za $\mathrm{x}, \mathrm{y} \in F_{2^m}$ po jednadžbi:

$$y^2 + xy = ax^2 + b \text{ u } F_{2^m}$$

zajedno s dodatnom točkom $\mathcal{O}$. Ovdje nas zanimaju samo nesingularne eliptične krivulje u polju F_{2^m}. Nesingularnost označava da krivulja nema šiljaka, ne siječe se sama sa sobom i ne postoje izolirane točke.

Broj točaka na $E(F_{2^m})$ označavamo s $\#E(F_{2^m})$. Hasseovim teoremom dolazimo do:

$$2^m + 1 - 2\sqrt{2^m} \leq \#E(F_{2^m}) \leq 2^m + 1 + 2\sqrt{2^m}.$$

Kao za polje F_p, imamo pravila zbrajanja točaka na E, koja su:

- zbrajanje dvije točke u beskonačnosti:

$$\mathcal{O} + \mathcal{O} = \mathcal{O}$$

- zbrajanja točke u beskonačnosti s bilo kojim drugom točkom:

$$(x, y) + \mathcal{O} = \mathcal{O} + (x, y) = (x, y) \text{ za sve } (x, y)$$

- zbrajanje točke s točkom koja ima istu x-koordinatu, a suprotnu y-koordinatu (njenim negativom):

$$(x, y) + (x, x + y) = \mathcal{O} \text{ za sve } (x, y)$$

  Napomena: negativ točke (x, y) je $-(x, y) = (x, x + y)$

- zbrajanje dvije točke s različitim x-koordinatama: neka su $(x_1, y_1) \in E(F_{2^m})$ i $(x_2, y_2) \in E(F_{2^m})$ dvije točke tako da vrijedi $x_1 \neq x_2$. Tada je $(x_1, y_1) + (x_2, y_2) = (x_3, y_3) \in E(F_{2^m})$ gdje je:

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a, y_3 = \lambda(x_1 + x_3) + x_1 + y_1, \lambda = \frac{y_1 + y_2}{x_1 + x_2}$$

- zbrajanje točke sa samom sobom (dupliciranje): Neka $(x_1, y_1) \in E(F_p)$ bude točka s $y_1 \neq 0$. Tada vrijedi $(x_1, y_1) + (x_1, y_1) = (x_3, y_3)$ gdje je:

$$x_3 = \lambda^2 + \lambda + a,\ y_3 = {x_1}^2 + (\lambda + 1)x_3,\ \lambda = x_1 + \frac{y_1}{x_1}.$$

Skup točaka na $E(F_p)$ čini Abelovu grupu pod pravilom zbrajanja. Kao u prethodnom primjeru, ako imamo cijeli broj k i točku $P \in E(F_p)$, skalarnim množenjem možemo izračunati kP [3].

4 MAC

Mogućnost provjere integriteta informacija koje se prenose preko ili pohranjuju u nepouzdanom mediju glavna je potreba u svijetu otvorenog računalstva i komunikacija. Mehanizmi koji omogućuju takvu provjeru integriteta na temelju tajnog ključa obično se nazivaju kodovi za autentifikaciju poruka ili skraćeno MAC. Obično se kodovi za provjeru autentičnosti poruke koriste između dviju strana koje dijele tajni ključ kako bi se potvrdile informacije koje se prenose između njih.

4.1 HMAC

HMAC je MAC mehanizam baziran na *hash* funkciji te se može koristiti u kombinaciji s bilo kojom iteritativnom kriptografskom *hash* funkcijom. MD5, SHA-2 i SHA-3 primjeri su takvih *hash* funkcija. HMAC također koristi tajni ključ za izračun i provjeru vrijednosti autentifikacije poruke. Glavni ciljevi HMAC-a su [7]:

- korištenje dostupnih *hash* funkcija bez izmjena. Konkretno, *hash* funkcije koje dobro funkcioniraju u softveru i za koje je kod besplatno i široko dostupan
- sačuvati izvorne performanse *hash* funkcije bez značajnog pogoršanja
- koristiti i rukovati ključevima na jednostavan način
- dobro razumjeti kriptografsku analizu snage autentifikacijskog mehanizma na temelju razumnih pretpostavki o temeljnoj *hash* funkciji
- omogućiti laku zamjenu temeljne *hash* funkcije u slučaju da se pronađu ili su potrebne brže ili sigurnije *hash* funkcije.

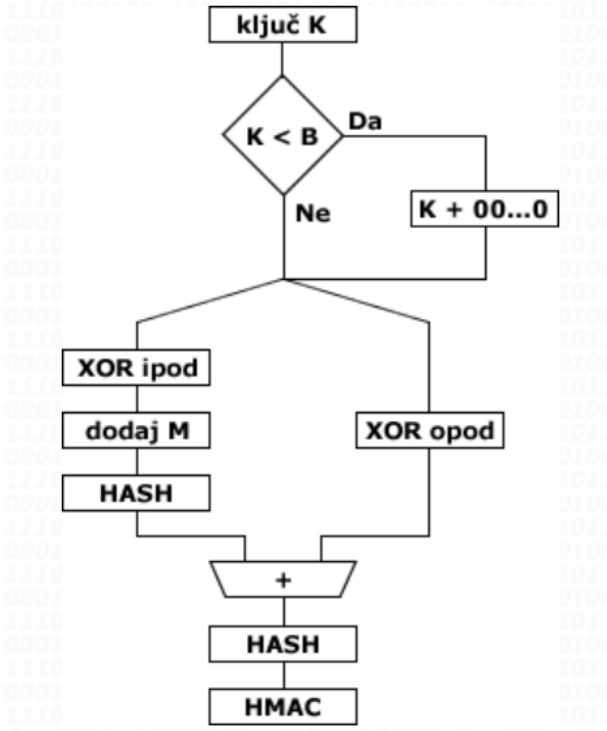
Da bismo definirali HMAC funkciju, potrebno je odabrati *hash* funkciju H i tajni ključ k. Pretpostavimo da je H *hash* funkcija gdje se podaci sažimaju iteriranjem osnovne funkcije za sažimanje bloka podataka. Dužinu tih blokova označimo s B, a duljinu izlaza iz *hash* funkcije s L. Tajni ključ K može biti manji ili jednak duljini bloka ulaznih podataka B *hash* funkcije H. Ako se koriste ključevi duži od B okteta potrebno je prvo sažeti

ključ *hash* funkcijom H te koristiti izlaz iz funkcije kao stvarni HMAC ključ. Svakako, L je minimalna preporučena dužina ključa K [7].

Definirajmo dva nepromjenjiva i različita niza znakova „ipod“ i „opod“. HMAC nad porukom M izračunavamo na sljedeći način (Slika 4.1):

$$\mathrm{H}(\mathrm{K} \oplus opad, H(K \oplus ipad, M),$$

1. ako je potrebno, dodati nule na kraj ključa K da se dobije niz znakova od ukupno B okteta
2. izračunati isključivo ILI funkciju ($\oplus$) između niza znakova dobivenog u prethodnom koraku i „ipod“ niza znakova
3. poruka M se dodaje nizu iz predhodnog koraka
4. na dobiveni niz podataka iz prethodnog koraka primjeni se *hash* funkcija H
5. izračunati isključivo ILI funkciju između niza znakova dobivenog u prvom koraku i „opod“ niza znakova
6. izlazu iz koraka 4. dodaje se niz iz prethodnog koraka dužine B okteta
7. na dobiveni niz podataka iz prethodnog koraka primjeni se *hash* funkcja H



Slika 4.1 Struktura HMAC algoritma [7]

4.1.1 Ključevi

Duljine ključeva za HMAC mogu biti proizvoljne dužine, s tim da se ključevi duži od *B* okteta prvo sažimaju *hash* funkcijom H. Međutim, ne preporučuju se ključevi kraći od L okteta jer bi smanjili sigurnost funkcije. Ključevi duljine veće od L okteta se prihvaćaju, ali se dodatnim produljenjem ne povećava sigurnost funkcije. Preporuča se duži ključ ako se slučajnost ključa smatra slabošću. Ključevi trebaju biti generirani slučajnim odabirom i moraju se periodički mijenjati. Trenutni napadi nisu pokazali koliko često bi se ključevi trebali mijenjati, ali je poznato da je periodičko osvježavanje ključa temelj sigurnosti, koji pomaže protiv potencijalnih slabosti funkcije i ključeva te smanjuje štetu nastalu izloženošću ključa [7].

4.1.2 Implementacija

HMAC je definiran na način da se temeljna *hash* funkcija *H* može koristiti bez modifikacije njezinog koda. Konkretno, koristi se funkcija *H* s unaprijed definiranom početnom vrijednošću *i* (fiksnom vrijednošću koju određuje svaka iterativna *hash* funkcija za

inicijalizaciju svoje funkcije kompresije). Ako želimo poboljšanje performansi, ono se može postići po cijenu moguće modifikacije koda H za podršku varijabli i [7].

4.1.3 Sigurnost

Sigurnost predstavljenog HMAC-a ovisi o kriptografskim svojstvima *hash* funkcije H, otpornosti na pronalaženje kolizije (ograničeno na slučaj kada je početna vrijednost tajna i slučajna i gdje izlaz funkcije nije eksplicitno dostupan napadaču) i svojstvo provjere autentičnosti poruke funkcije k. Ova svojstva, pa i bolja, pretpostavljaju se za *hash* funkcije koje se koriste s HMAC-om. Ako gornja svojstva ne vrijede za *hash* funkciju, takva funkcija postala bi neprikladna za većinu ili čak sve kriptografske aplikacije, uključujući alternativne sheme provjere autentičnosti poruka temeljene na takvim funkcijama kompresije H kada se primijeni na pojedinačne blokove [7].

Obzirom na dosad stečenu pouzdanost u pogledu kriptografske snage *hash* funkcija, važna su sljedeća dva svojstva kod konstrukcije HMAC-a i njegovu sigurnu upotrebu za provjeru autentičnosti poruke:

- konstrukcija HMAC-a je neovisna o pojedinostima određene *hash* funkcije H koja se koristi, a zatim se potonja može zamijeniti bilo kojom drugom sigurnom iterativnom kriptografskom *hash* funkcijom.
- autentifikacija poruke, za razliku od enkripcije, ima "prolazni" učinak. Objavljivanje kršenja sheme provjere autentičnosti poruke dovelo bi do zamjene te sheme, ali ne bi imalo kontradiktorni učinak na podatke provjerene u prošlosti, što je u suprotnosti s enkripcijom, gdje informacije šifrirane danas, mogu biti izložene u budućnosti ako se algoritam enkripcije probije.

5 KDF

KDF (engl. *Key derivation function*) je funkcija koja iz lozinke ili drugih izvora podataka izvodi ključeve prikladne za kriptografske operacije pomoću pseudoslučajne funkcije PRF (engl. *Pseudo-random function*). Različiti KDF-ovi prikladni su za različite zadatke kao što su:

- izvođenje ključa prikladnog za korištenje kao ulaz u algoritam šifriranja. To obično znači uzeti lozinku ili neki slabi ključ i provući ih kroz algoritam kao što je PBKDF2 (engl. *Password-based Key Derivation Function*) ili HKDF (engl. *Hash-based Key Derivation Function*) . Ovaj proces obično je poznat kao istezanje ključa. (engl. *key stretching*).
- generiranje više ključeva iz jednog izvornog ključa što nam omogućuje korištenje zasebnih ključeva za svaki aspekt kriptosustava.
- sigurno pohranjivanje lozinki

  Prilikom pohranjivanja lozinki želimo koristiti algoritam koji je računski zahtjevan. Korisnik proces računanja izvodi samo jednom, na primjer uzeti korisničku lozinku, provući ju kroz KDF, zatim je usporediti s pohranjenom vrijednošću, dok će napadači to morati učiniti milijarde puta. Idealni KDF-ovi za pohranu lozinki bit će zahtjevni i za računalne i za memorijske resurse [9, 10].

5.1 PBKDF2

PBKDF2 (engl. *Password-based Key Derivation Function*) je KDF na temelju lozinke objavljen 2000. godine kao dio RSA Laboratories standarda o javnim ključevima poznat kao PKCS (engl. *Public-Key Cryptography Standards*) [9]. PKCS postavlja osnove za implementaciju sigurnog kriptosustava s javnim ključem. PBKDF2 dolazi kao zamjena za PBKDF1, koji proizvodi ključeve koji više nisu dovoljno veliki da bi bili sigurni za današnje standarde. Jedan od ciljeva PBKDF2 je pretvoriti lozinku, tajnu s niskom entropijom, u ključ odgovarajuće duljine i slučajnosti dostatne uniformnoj distribuciji. PBKDF2 to čini koristeći četiri ulaza:

- lozinka – ovo je korisnička lozinka koju treba pretvoriti u jak ključ

- sol (engl. *salt*) – dodatni nasumični unos podataka koji KDF obrađuje uz lozinku. Sol ne mora biti tajna i pohranjuju se zajedno s hashovima zaporki.
- broj iteracija – određuje koliko će puta vrijednosti proći kroz funkciju.
- duljina ključa – duljina ključa koju želimo na izlazu

PBKDF2 provlači lozinku i sol kroz PRF određeni broj puta, ovisno o vrijednosti broja ponavljanja. Sol dodajemo kako bismo smanjili mogućnost korištenja unaprijed izračunatih hasheva. Standard preporuča duljinu soli od najmanje 64 bita dok je NIST (engl. *National Institute of Standards and Technology*) preporuka iz 2021. godine duljina soli od 128 bita. Konačni izlaz je jaki ključ duljine koja je navedena u ulazu izvedenenog ključa. Ovaj se ključ zatim može koristiti za sigurnost različitih aspekata kriptosustava. PBKDF2 može koristiti niz različitih pseudoslučajnih funkcija kao što su HMAC-SHA-256, HMAC-SHA-512 čak i HMAC-SHA-1, iako mu je potreban mnogo veći broj ponavljanja kako bi bio siguran. SHA-1 obično se ne smatra sigurnim u većini drugih aplikacija.

Funkcija PBKDF2 može se napisati ovako:

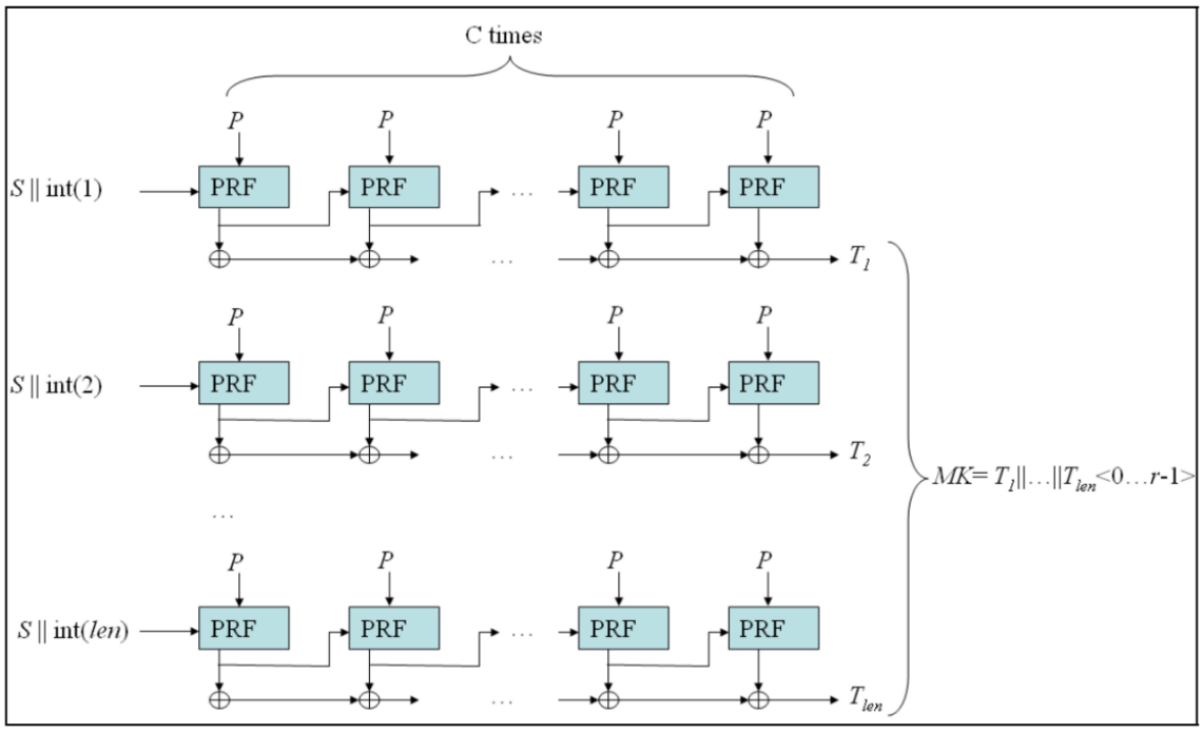
$F = (P, S, c, i) = U_1 \, XOR \, U_2 \, XOR \, \dots XOR \, U_C$ gdje je: $U_1 = PRF\big(P, S \,||INT(i)\big)$,

$U_2 = PRF(P, U_1)$, $U_C = PRF(P, U_{C-1})$.

Gdje je:

- P – lozinka
- S – salt
- c – broj iteracija
- i – indeks bloka

Budući da u ovom jednostavnom primjeru postoji samo jedan blok za obradu, naš izlaz, odnosno konačni izvedeni ključ isti je kao U_C. Ako postoji više blokova, krajnji rezultati svakog bloka morali bi biti spojeni zajedno da bi se dobio konačni ključ (slika 5.1.) [9, 10].



Slika 5.1 Princip rada PBKDF2 [11]

6 DIGITALNI POTPIS

Sheme digitalnog potpisa dizajnirane su za korištenje od strane dva entiteta, entitet A koji želi poslati autentičnu poruku m i entitet B koji želi provjeriti autentičnost poruke m. Kada je poruka potpisana, svaki entitet B koji ima kopiju javnog ključa entiteta A, može verificirati potpis, što ne mora nužno biti entitet kojemu je A izvorno poslao poruku. Sheme digitalnog potpisa ćemo opisati kao operaciju potpisivanja, operaciju provjere te procedure generiranja i razmjene ključeva.

Kada entiteti A i B žele komunicirati, trebali bi koristiti sheme na sljedeći način. Prvo se A i B moraju dogovoriti koju shemu i s kojim parametrima će ju koristiti, zatim A treba izvršiti proceduru za generiranje para ključa od kojih će B dobiti kopiju A-ovog javnog ključa. A će koristiti svoj privatni ključ kako bi proveo operaciju potpisa poruke dok će B korsititi dobiveni javni ključ za operaciju provjere potpisa. Svaki put kada želi poslati poruku m, entitet A mora primijeniti operaciju potpisa na m koristeći svoj privatni ključ kako bi dobio potpis s na poruci m, zatim formirati potpisanu poruku od m i s te proslijediti potpisanu poruku do B. Kada konačno primi potpisanu poruku, B treba primijeniti operaciju provjere na potpisanoj poruci. Ako je provjera „valjana“ B zaključuje da je potpisana poruka doista autentična.

Postoje dvije vrste shema digitalnog potpisa, ovisno o obliku potpisa koja se proslijeđuje entitetu B:

- sheme poruka s dodatkom u kojem entitet A mora poslati entitetu B poruku m i potpis s
- sheme poruka s oporavkom poruke u kojima se m može oporaviti iz s, tako da je potrebno poslati samo potpis s

Sheme potpisa dizajnirane su tako da će sudionik E koji ne zna A-ov tajni ključ, teško krivotvoriti važeće potpisane poruke. Stoga sheme potpisa osiguravaju autentičnost, integritet i neporecivost poruke.

6.1 ECDSA

ECDSA (*engl. Elliptic Curve Digital Signature Algorithm)* je shema potpisa koja koristi ključeve izvedene iz ECC. Koristi se u mnogim sigurnosnim sustavima, a popularan je za

upotrebu u aplikacijama za sigurnu razmjenu podataka te je osnova sigurnosti pojedinih kriptovaluta poput Bitcoina. Glavna značajka ECDSA u odnosu na drugu popularnu shemu, RSA, je da ECDSA omogućuje viši stupanj sigurnosti s kraćim duljinama ključeva. Ovo dodatno povećava njegovu isplativost (ROI – engl. *Return On Investment*) jer ECDSA koristi manje računalne snage od RSA.

6.1.1 Postavljanje sheme

Entiteti A i B trebaju pratiti slijedeću proceduru kako bi se pripremili za korištenje ECDSA [1, 10]:

1. entitet A treba odrediti koja parametre eliptične krivulje te *hash* funkciju i duljinu *hash* zapisa će se koristiti pri generiranju potpisa.
2. entitet A treba odrediti eliptičnu krivulju D = (q, a, b, P, n, h) za $D \in F_p$ ili $D = (m, f(x), a, b, G, n, h)$ za $D \in F_{2^m}$ i njene parametre koji će biti korišteni pri generiranju potpisa. Fokusirajmo se na konačno polje F_p čiji su parametri:
    - $q = p$ ili $q = 2^m \rightarrow$ p je prost broj
    - a i $b \rightarrow$ određuju jednadžbu krivulje
    - $P \rightarrow$ bazna točka krivulje $E(F_q)$
    - $n \rightarrow$ red točke P
    - $h = \#E(F_q) \div n$
3. entitet B treba na autentičan način dobiti odabaranu *hash* funkciju i krivulju iz koraka 1. i 2.

6.1.2 Generiranje i raspodjela ključeva

Entiteti A i B trebaju izvršiti sljedeće korake kako bi generirali valjane ključeve za korištenje ECDSA [1, 10]:

1. entitet A treba generirati par ključeva eliptičke krivulje (d, Q) na sljedeći način [1]:
    1. odrediti slučajni broj d iz intervala $[1, n-1] \rightarrow$ privatni ključ

2. izračunati $Q = dP \rightarrow$ javni ključ

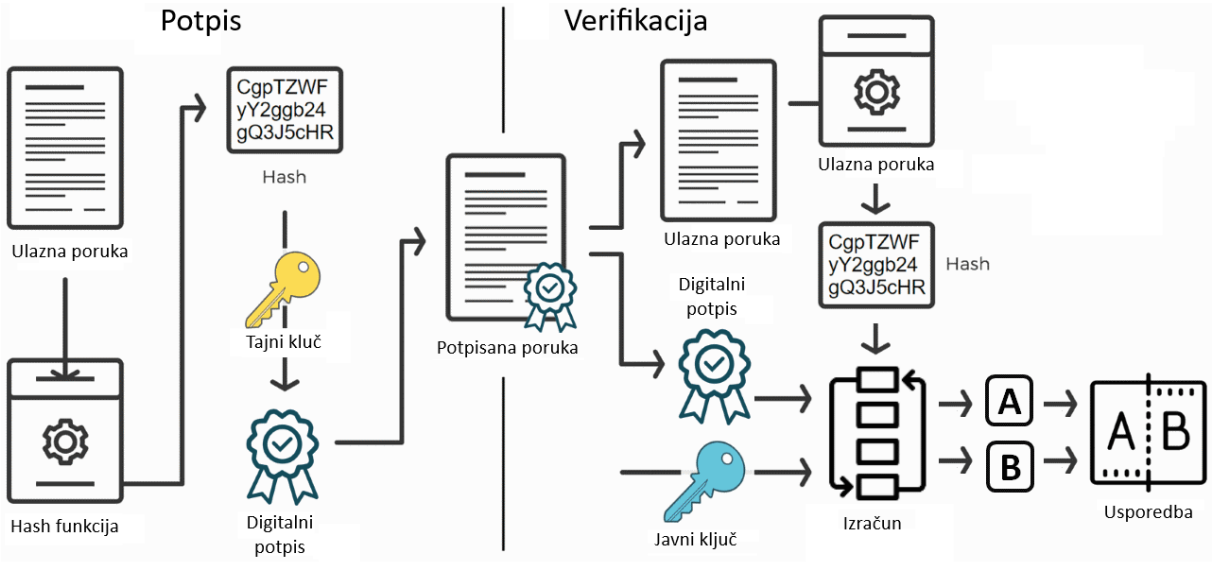
2. entitet A treba ispitati uspravnost javnog ključa $Q = (x_1, x_2)$ na sljedeći način:
    1. je li $Q = \mathcal{O}$
    2. je li $nQ = \mathcal{O}$
    3. jesu li x_1, x_2 ispravni elementi polja F_q
    4. je li Q točka na krivulji određenoj koeficijentima a i b
3. entitet B treba na autentičan način dobiti javni ključ eliptičke krivulje Q generiran od strane A

6.1.3 Postupak potpisivanja

Entitet A mora potpisati poruku koristeći generirani privatni ključ i parametre utvrđene tijekom postupka postavljanja.

Postupak potpisa (Slika 6.1.) na ulazu prima poruku m koju treba potpisati, a na izlazu daje potpis $S = (r, s)$ na m koji se satoji od para cijelih brojeva r i s. Da bi entitet A potpisao poruku potrebno je sljedeće [12]:

1. odrediti slučajni broj k iz intervala $[1, n-1]$
2. odrediti $kP = (x, y)$ i $r = x \; mod \; n$, ako je $r = 0 \rightarrow$ povratak na korak 1.
3. izračunati $t = k^{-1} \; mod \; n$
4. izračunati e $= hash$(m)
5. izračunati $s = k^{-1} \; (e + rd) mod \; n$, ako je $s = 0 \rightarrow$ povratak na korak 1.



Slika 6.1 Postupak digitalnog potpisa te njegove verifikacije [12]

6.1.4 Postupak verifikacije

Entitet B treba verificirati potpisanu poruku dobivenu od entiteta A (Slika 6.1.) koristeći poznate parametre kriptosustava D te pošiljateljev javni ključ Q [10]:

1. provjeri jesu li r i s iz intervala $[1, n-1]$
2. izračunati $e = hash(m)$
3. izračunati $w = s^{-1}\ mod\ n$
4. izračunati $u_1 = ew$ i $u_2 = rw$
5. izračunati $X = (x_1, x_2) = u_1 P + u_2 Q$
6. ako je $X = 0$, odbiti potpis, inače izračunati $v = x_1\ mod\ n$
7. ako je $v = r$, prihvatiti potpis za poruku m, inače potpis nije validan

7 SHEME ZA RAZMJENU KLJUČEVA

Sheme za razmjenu ključa (KA – engl. *Key agreement*) su posebne sheme za enkripciju temeljenu na javnom ključu kojima je cilj da šifrirana poruka m sadrži kriptografski ključ, najčešće simetrični.

7.1 ECDH

Shema za razmjenu ključeva ECDH (*engl. Elliptic Curve Diffie-Hellman*) omogućuje dvjema stranama, od kojih svaka ima svoj par ključeva, javni i privatni, da uspostave zajedničku tajnu preko nesigurnog kanala. Dijeljena tajna može se izravno koristiti kao ključ ili za izvođenje drugog ključa. Ključ, ili izvedeni ključ, tada se može u kasnijoj komunikaciji koristiti za šifriranje poruka pomoću sheme sa simetričnim ključem. ECDH je varijanta Diffie-Hellman sheme koja koristi ECC.

7.1.1 Postavljanje sheme

Pretpostavimo da entitet A želi razmijeniti ključ s entitetom B, ali njihov jedini dostupni kanal može biti prisluškivan od treće strane. Prije generiranja ključeva, entiteti moraju odrediti eliptičnu krivulju $D = (q, a, b, P, n, h)$ za $D \in F_p$ ili $D = (m, f(x), a, b, G, n, h)$ za $D \in F_{2^m}$ i njene parametre koji će biti korišteni. Fokusirajmo se na konačno polje F_p čiji su parametri:

- $q = p$ ili $q = 2^m \rightarrow$ p je prost broj
- a i $b \rightarrow$ određuju jednadžbu krivulje
- $P \rightarrow$ bazna točka krivulje $E(F_q)$
- $n \rightarrow$ red točke P
- $h = \#E(F_q) \div n$

7.1.2 Generiranje i raspodjela ključeva

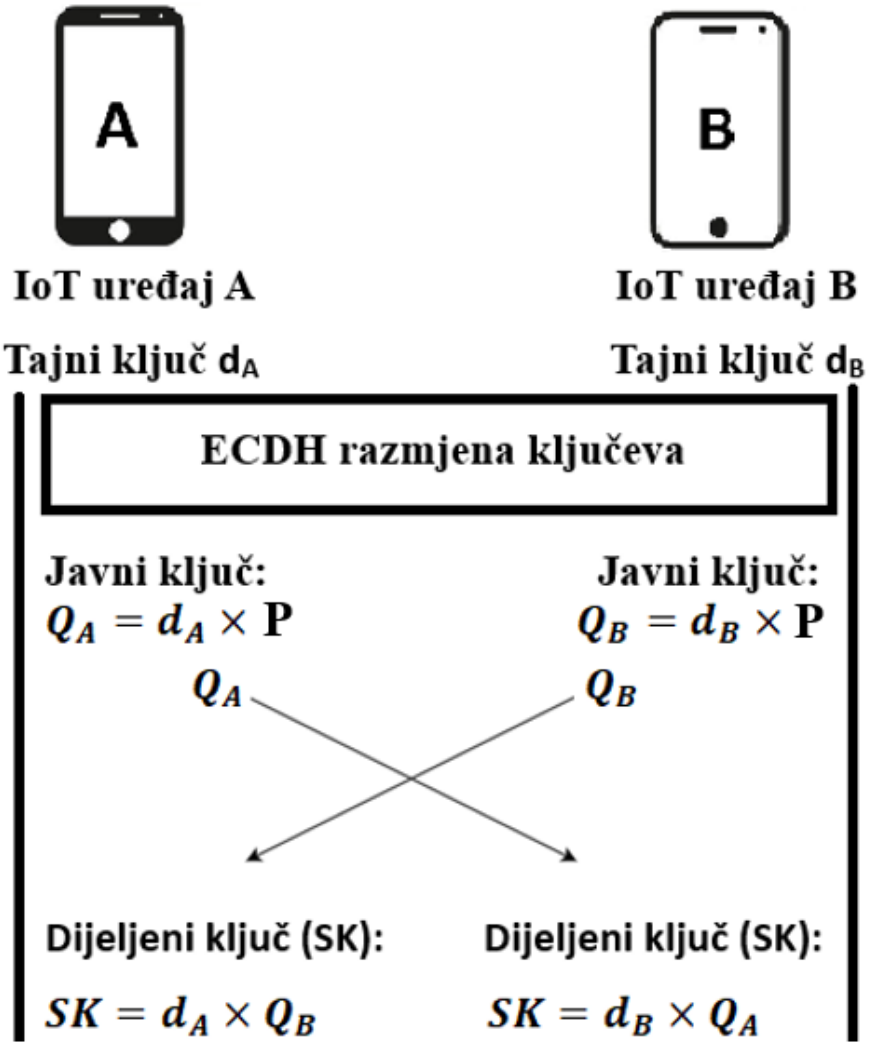
Kako bi pravilno generirali i raspodijelili ključeve, entiteti A i B trebaju pratiti sljedeći postupak [1, 15]:

1. entitet A treba generirati par ključeva eliptičke krivulje (d_A, Q_A) na sljedeći način:
    1. odrediti slučajni broj d_A iz intervala $[1, n-1]$ → privatni ključ
    2. izračunati $Q_A = d_A P$ → javni ključ
    3. poslati Q_A entitetu B
2. entitet B treba generirati svoj par ključeva eliptičke krivulje (d_B, Q_B) na sljedeći način:
    1. odrediti slučajni broj d_B iz intervala $[1, n-1]$ → privatni ključ
    2. izračunati $Q_B = d_B P$ → javni ključ
    3. poslati Q_B entitetu A
3. oba entiteta A i B trebaju ispitati ispravnost dobivenog javnog ključa Q= (x_1, x_2) na sljedeći način:
    1. je li $Q = \mathcal{O}$
    2. je li $nQ = \mathcal{O}$
    3. jesu li x_1, x_2 ispravni elementi polja F_q
    4. je li Q točka na krivulji određenoj koeficijentima a i b
4. A izračuna $(x_k, y_k) = d_A Q_B$, dok B izračuna $(x_k, y_k) = d_B Q_A$,
5. x_k je dijeljena tajna

Dijeljena tajna koju izračunaju obje strane je identična jer je:

$$d_A Q_B = d_A d_B P = d_B d_A P = d_B Q_A$$

Većina standardiziranih protokola temeljenih na ECDH izvodi simetrični ključ iz x_k koristeći KDF.



Slika 7.1 ECDH razmjena ključeva [14]

8 SHEME ZA KRIPTIRANJE

Sheme šifriranja s javnim ključem opisane su u smislu operacije šifriranja, operacije dešifriranja te povezanih postupaka postavljanja i raspodjele ključa. Kada entiteti A i B žele komunicirati trebali bi koristiti shemu na sljedeći način. Recimo da je entitet A pošiljatelj (engl. *sender*), a entitet B primatelj (engl. *receiver*) poruke. Entiteti se prvo moraju dogovoriti koju shemu i s kojim parametrima će koristiti, zatim B treba koristiti proceduru za generiranje para ključa od kojih će A dobiti kopiju B-ova javnog ključa. A će koristiti B-ov javni ključ za postupak šifriranja, a B će koristiti njegov par, privatni ključ za postupak dešifriranja. Svaki put kada entitet A želi poslati poruku m entitetu B, treba primijeniti operaciju šifriranja m koristeći B-ov javni ključ za izračunavanje enkripcije ili šifriranog teksta c od m. Tada šifrirani tekst c šalje entitetu B. Kada primi šifrirani tekst c, B nad njim mora koristiti operaciju dešifriranja svojim privatnim ključem kako bi dobio poruku m.

Jednostavno rečeno, sheme šifriranja s javnim ključem dizajnirane su tako da je entitetu E koji ne posjeduje B-ov tajni ključ teško pročitati poruke iz šifriranog teksta. Iz toga možemo zaključiti da sheme šifriranja javnim ključem osiguravaju povjerljivost podataka.

Sheme šifriranja javim ključem mogu se koristiti za šifriranje poruka bilo koje vrste. Mogu se koristiti za prijenos ključa ili poruke od A do B.

8.1 ECIES

Elliptic Curve Integrated Encryption Scheme (ECIES) je hibridna shema za šifriranje javnim ključem bazirana na ECC-u. ECIES kombinira asimetričnu kriptografiju temeljenu na ECC-u sa simetričnim shemama kako bi omogućio šifriranje podataka ECC privatnim ključem i dešifriranje odgovarajućim ECC javnim ključem.

Shema šifriranja ECIES koristi kombinaciju sljedećih funkcionalnosti [16]:

- protokol za dogovor oko ključeva (KA)
- ECC kriptografiju
- funkciju izvođenja ključa (KDF)
- algoritam simetrične enkripcije

- MAC algoritam

ECIES je dizajnirana da bude semantički sigurna u prisutnosti aktera spremnog na napad odabranim otvorenim ili šifriranim tekstom.

8.1.1 Postavljanje sheme

Ukoliko entiteti A i B žele uspostaviti komunikaciju pomoću ECIES-a, postupak je sljedeći [17]:

1. Odrediti eliptičnu krivulju D = (q, a, b, G, n, h) za $D \in F_p$ ili $D = (m, f(x), a, b, G, n, h)$ za $D \in F_{2^m}$ te njene parametre koji će biti korišteni pri generiranju potpisa. Fokusirajmo se na konačno polje F_p čiji su parametri:
    - $q = p$ ili $q = 2^m \rightarrow$ p je prost broj
    - a i $b \rightarrow$ određuju jednadžbu krivulje
    - $P \rightarrow$ bazna točka krivulje $E(F_q)$
    - $n \rightarrow$ red točke P
    - $h = \#E(F_q) \div n$
2. odrediti *hash* funkciju i duljinu *hash* zapisa koji će se koristiti pri generiranju potpisa.
3. entitet B treba na autentičan način dobiti informacije o odabaranoj *hash* funkciji i krivulju iz koraka 1. i 2.
4. odrediti KDF
5. odrediti simetrični shemu za šifriranje poruke
6. osigurati da entitet B dobije sve potrebne informacije iz gornjih koraka kako bi mogao uspješno izvršiti dešifriranje poruke.

8.1.2 Generiranje i raspodjela ključeva

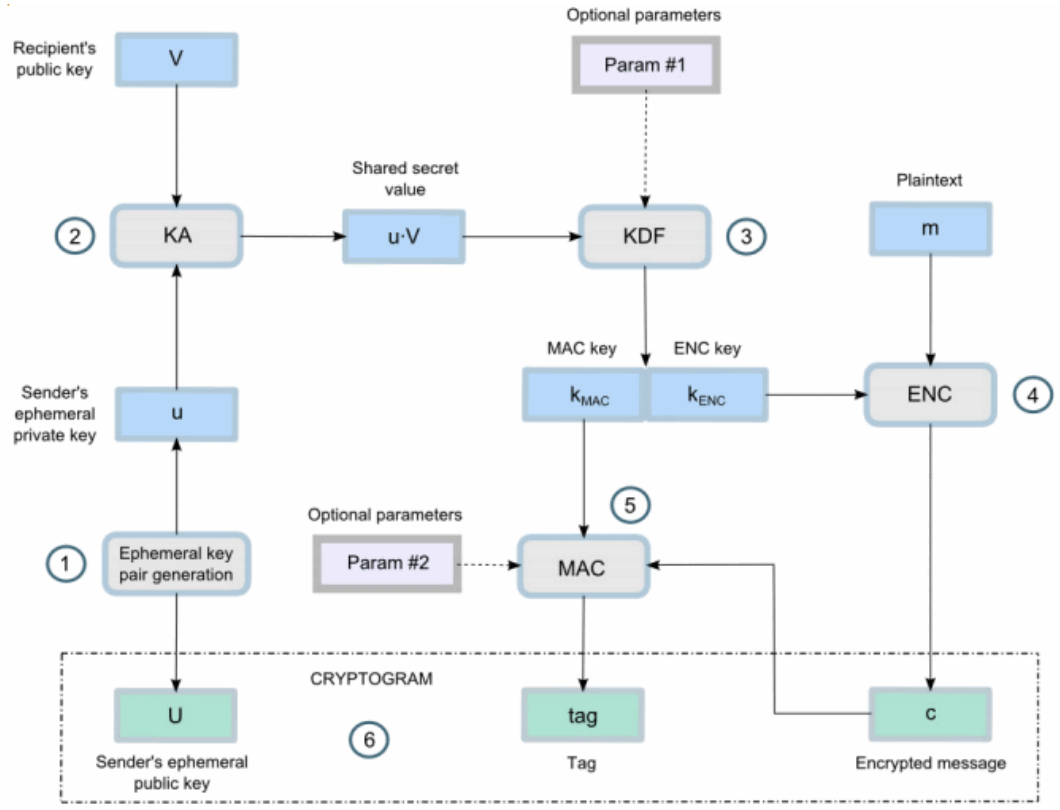
Postupak generiranja ključeva koji će biti korišteni za ECIES [1, 17]:

1. entitet B treba generirati par ključeva eliptične krivulje (d, Q) na bazi parametara domene eliptične krivulje D koji su uspostavljeni tijekom postupka postavljanja sheme
2. entitet B treba ispitati ispravnost javnog ključa Q= (x_1, x_2) na sljedeći način:
    1. je li $Q = \mathcal{O}$
    2. je li $nQ = \mathcal{O}$
    3. jesu li x_1, x_2 ispravni elementi polja F_q
    4. je li Q točka na krivulji određenoj koeficijentima a i b
3. entitet A treba dobiti javni ključ Q kojeg je generirao B

8.1.3 Postupak šifriranja

Za šifriranje poruke m (Slika 8.1.), entitet A treba napraviti sljedeće [17]:

1. generirati novi privremeni (engl. *ephemeral*) par ključeva (r, R) tako da odaberemo nasumični $r \in [1, n-1]$ te izračunamo $R = rG$
2. izračunati dijeljenu informaciju (*engl. shared secret*) $S = P_x$ gdje je $P = (P_x, P_y) = rQ$ i $P \neq 0$
3. koristiti KDF za derivaciju ključa simetrične enkripcije i MAC ključa: $k_E||k_M = KDF(S||S_1)$
4. šifrirati poruku shemom simetrične enkripcije $c = ENC(k_E, m)$
5. izračunati oznaku provjere (engl. *tag*) kriptirane poruke S_2: $d = MAC(k_M, c||S_2)$
6. rezultat je šifrat (engl. *cryptogram*) $R||c||d$

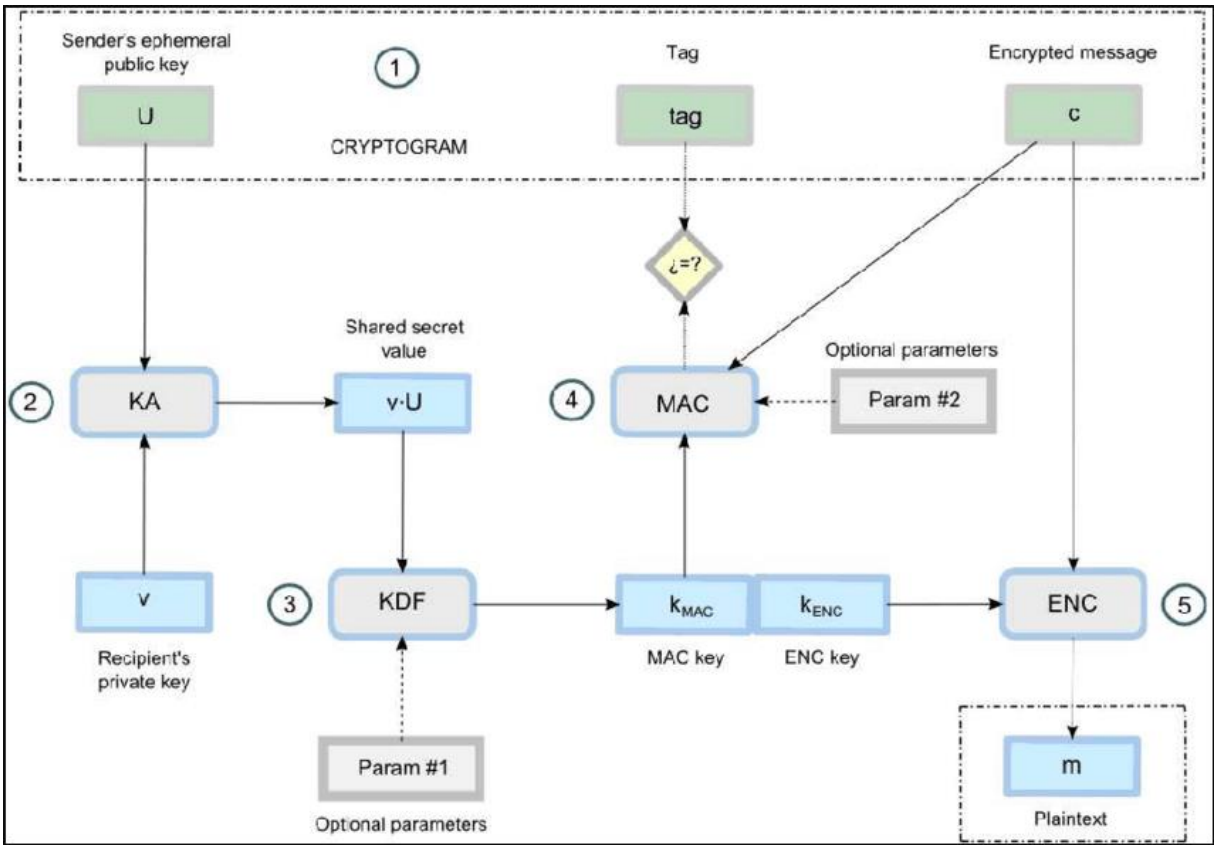


Slika 8.1 ECIES postupak šifriranja [16]

8.1.4 Postupak dešifriranja

Za dešifriranje šifriranog teksta $R||c||d$ (Slika 8.2), entitet B radi slijedeće [17]:

1. izračunati dijeljenu informaciju $S = P_x$, gdje je $P = (P_x, P_y) = qR$, što je identično kao 2. korak enkripicije jer $P = qR = qrG = rqG = rQ$, ako je $P = 0$ rezultat nije valjan
2. koristiti KDF za izvođenje ključa simetrične enkripcije i MAC ključa kao u postupku enkripcije $k_E||k_M = KDF(S||S_1)$
3. koristiti MAC funkciju da bi se provjerila oznaka provjere: izlaz nije valjan ako je $d \neq MAC(k_M; c||S_2)$
4. koristiti shemu simetrične enkripcije kojom je poruka šifrirana za dešifriranje poruke $m = ENC(K_E; c)$



Slika 8.2 ECIES postupak dešifriranja [16]

9 SPECIFIČNOSTI SHEMA ELIPTIČNIH KRIVULJA

9.1 Performanse i sigurnost

Tablica 9.1. prikazuje veličine ključeva u bitovima za različitie sheme koje pružaju jednaku razinu sigurnosti. Npr. za zaštitu 128 bitnog AES simetričnog ključa potrebno je koristiti 3072 bitni RSA, dok je za isti ključ potrebno 256 bitni ključ shema eliptičnih krivulja.

Tablica 9.1.Duljine ključeva u bitovima prema preporuci NIST organizacije [12]

Sigurnosna razina	DSA/RSA/Diffie-Hellman	Eliptične krivulje
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Sigurnost nije jedina prednost shema eliptičnih krivulja u odnosu na RSA i Diffie-Hellman sheme. Iako imaju složenije operacije po bitu ključa, sheme eliptičnih krivulja su manje računski zahtjevni od spomenutih shema. Tablica 9.2. prikazuje odnos broja računskih operacija za Diffie-Hellman shemu i ECDH shemu baziranu na eliptičnim krivuljama prema različitim razinama sigurnosti [12].

Tablica 9.2. Odnos složenosti operacija Diffie-Hellman algoritma i ECDH prema različitim razinama sigurnosti [12]

Broj bitova	Omjer složenosti operacija Diffie-Hellman i ECDH algoritma
80	3:1
112	6:1
128	10:1
192	32:1
256	64:1

9.2 Kompleksnost diskretnog logaritamskog problema

Stvarna kompleksnost diskretnog logaritamskog problema za eliptične krivulje nije u potpunosti jasna, što je veliki problem. Neke su se krivulje pokazale neprikladnima za ECC, iako se prije smatralo da jesu. Kada je bazna točka P jednaka prostom broju p, ECDLP (*engl. Elliptic Curve Discrete Logarithm Problem*) čiji je temelj nemogućnost pronalaženja faktora skalarnog umnoška, može se jednostavno riješiti [12].

9.3 Generiranje krivulja

Prilikom definiranja sustava eliptične krivulje, potrebno je definirati krivulju i baznu točku P. Kada imamo krivulju i baznu točku, lako je generirati privatni ključ, što je nasumični cijeli broj k i javni ključ, što je točka kP na krivulji. Najveći problem je pronalazak ukupnog broja točaka na krivulji. Kada to i napravimo, trebamo odrediti baznu točku P koja mora imati slijed kojim će osigurati kompleksnost ECDLP-a te mora podijeliti broj točaka na krivulji. Sve ovo nam govori da je pronalaženje bazne točke P vrlo složen proces [12].

9.4 Nekompatibilni sustavi

Implementacija parnih krivulja je slična implementaciji neparnih krivulja, ali opet dovoljno različita da one nisu kompatibilne. Kod neparnih krivulja javlja se problem uzrokovan razlikama u prezentaciji krivulje i baznog P, što rezultira pogreškom u komunikaciji između korisnika ako koriste različite prezentacije. To je razlika u odnosu na RSA shemu kod koje su sve korisničke implementacije kompatibilne [12].

9.5 Procesiranje

Sustav kriptiranja eliptičnih krivulja koristi manje ključeve od ostalih sustava. U tablici 9.3. se uspoređuje brzina generiranja i provjere potpisa između RSA i ECDSA. Test je napravljen na procesorima brzine takta 66Mhz.

Tablica 9.3. Trajanje generiranja i provjere potpisa RSA i ECDSA sheme [12]

Algoritam	Generiranje potpisa	Provjera potpisa
RSA (1024 bit)	25 ms	< 2 ms
ECDSA (160 bit)	32 ms	33 ms
RSA (2048 bit)	120 ms	5 ms
ECDSA (216 bit)	68 ms	70 ms

S povećanjem veličine ključa, generiranje potpisa postaje sve brže kod ECDSA u usporedbi s RSA, provjera potpisa je sporija kod ECDSA u odnosu na RSA. Razlog zašto je ECDSA brži u generiranju ključeva je u tome što ECDSA za korisnikov privatni ključ treba samo stvoriti slučajni broj te iz njega izračunati javni ključ, dok RSA računa velike primarne brojeve. Vrijeme koje je potrebno za provjeru potpisa korištenjem ECDSA sheme, ponekad ima negativne učinke na performanse sustava, pa su stoga RSA sustavi ponekad prikladniji od shema baziranih na eliptičnim krivuljama [12].

9.6 Intelektualno vlasništvo

Iz prikazanog se može vidjeti da eliptične krivulje imaju mnoge prednosti i prihvaćene su od brojnih korisnika, ali međutim u praksi nisu dovoljno implementirane. Smatra se da je razlog veliki broj prijavljenih patenata nad eliptičnim krivuljama. Pa tako primjerice, kanadska tvrtka Certicom Inc posjeduje više od 300 patenata koje su povezane s eliptičkim krivuljama i kriptografijom. NSA (engl. *National Security Agency*) je kupila licencu od organizacije Certicom koja obuhvaća sve njihovo intelektualno vlasništvo s ograničenom mogućnošću korištenja. Razlog kupnje je da bi se eliptične krivulje mogle koristiti kod zaštite tajnih vladinih informacija. Licenca je ograničena samo na polja prostih brojeva F_p gdje je prosti broj veći od 2255 [12].

10 PRAKTIČNI RAD

Za potrebe ovoga rada, implementiran je kriptosustav koji koristi sljedeće sheme s javnim ključem bazirane na eliptičkim krivuljama:

- ECDSA
- ECIES
    - kombinaciju ECDH i AES-256 kao dio ECIES sheme

Eliptične krivulje podržane u aplikaciji:

- SECP224R1
- SECP256R1
- SECP384R1
- SECP521R1
- SECP521R1
- SECT571K1
- SECT409K1
- SECT283K1
- SECT163R2

Hash funkcije podržane u aplikaciji:

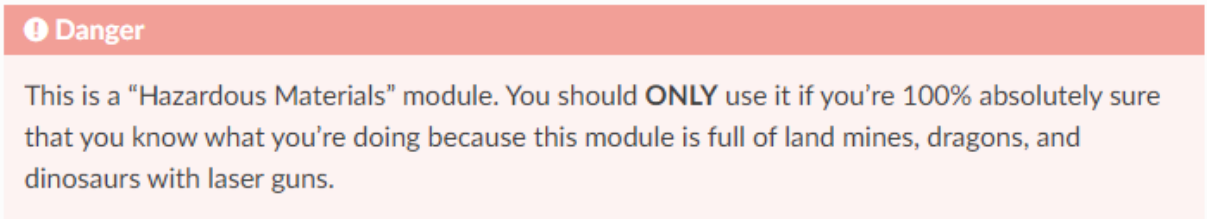
- SHA224
- SHA256
- SHA384
- SHA512
- SHA3_224
- SHA3_256
- SHA3_384
- SHA3_512

Kriptosustav je kompletno realiziran u Python-u uz pomoć knjižnice funkcija cryptography koja se koristi za postupke kriptografije dok je knjižnica funkcija tkinter korištena za grafičko sučelje.

10.1 Knjižnica funkcija cryptography

Knjižnica funkcija cryptography uključuje korištenje sučelja na visokoj i niskoj razini apstrakcije za standardne kriptografske sheme kao što su simetrične i asimetrične sheme, sažetci poruka i sheme za izvedbu ključeva [18].

Cryptography se općenito dijeli na dvije razine. Jedna sa sigurnim kriptografskim metodama koji zahtijevaju malo ili nimalo izbora konfiguracijskih parametara, koje su sigurne i jednostavne za korištenje i ne zahtijevaju od programera da donose mnogo odluka. Dok druga razina uključuje metode niske razine. One su često opasne i mogu se koristiti neispravno. Ova razina zahtijeva donošenje odluka i dubinsko poznavanje kritografskih koncepata u praksi. Zbog potencijalne opasnosti pri radu na ovoj razini, ona se naziva sloj "opasnih materijala" ili "hazmat". Smještene su u paketu cryptography.hazmat te u njihovoj dokumentaciji uvijek nalazimo upozorenje na vrhu (Slika 10.1). Preporuča se korištenje metoda na visokoj razini gdje god je to moguće, a na niskoj razini samo kada je to potrebno [18].



Slika 10.1 Primjer upozorenja za "hazmat" funkcionalnosti

10.2 Korištenje implementiranog kriptosustava

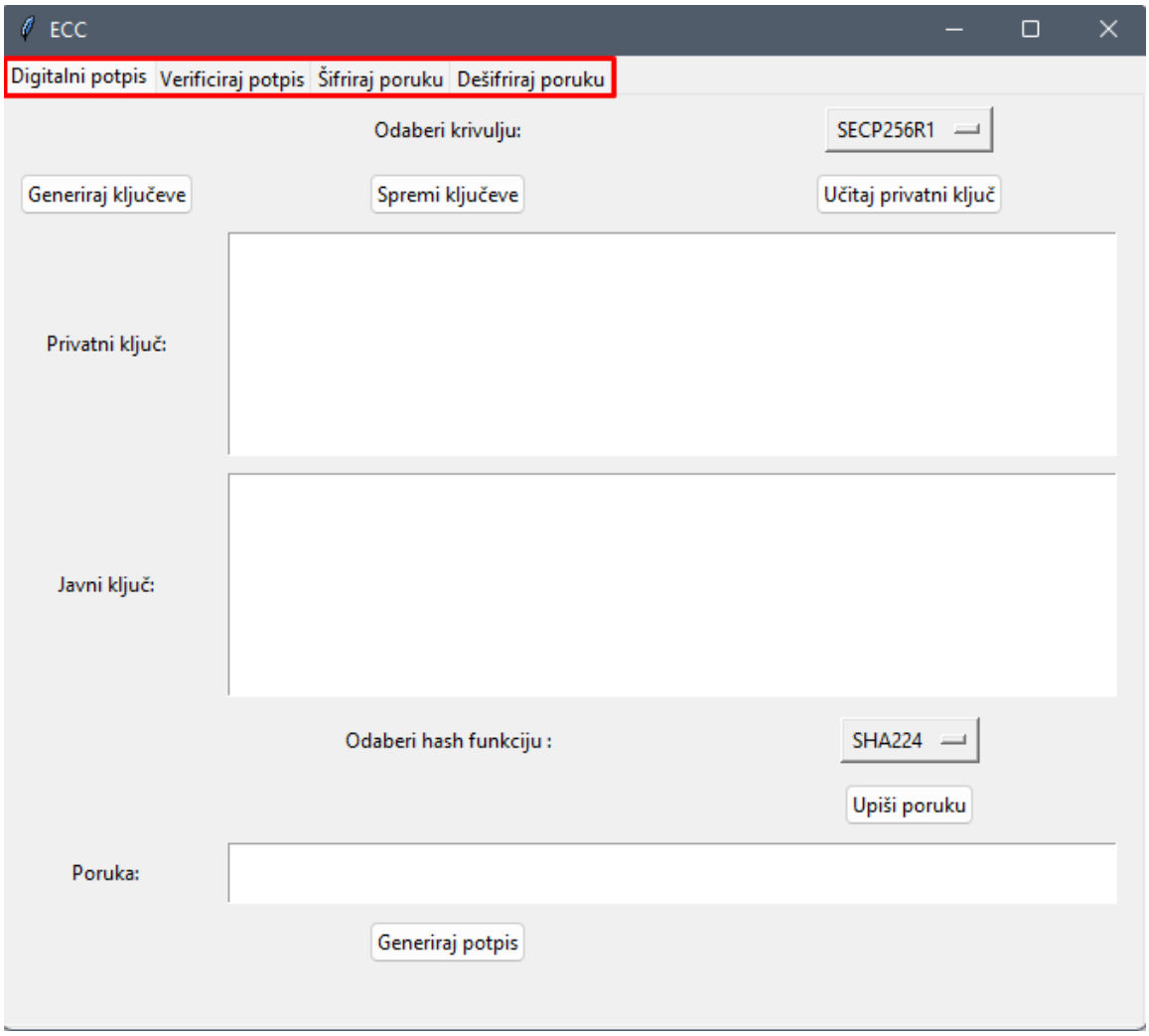
U sklopu rada, implementirana je aplikacija sa sljedećim funkcionalnostima:

- ECDSA digitalni potpis
- ECDSA verifikacija potpisa

- ECIES šifriranje
- ECIES dešifriranje

Sve funkcionalnosti se primjenjuju na upisani tekst ili učitanu tekstualnu datoteku.

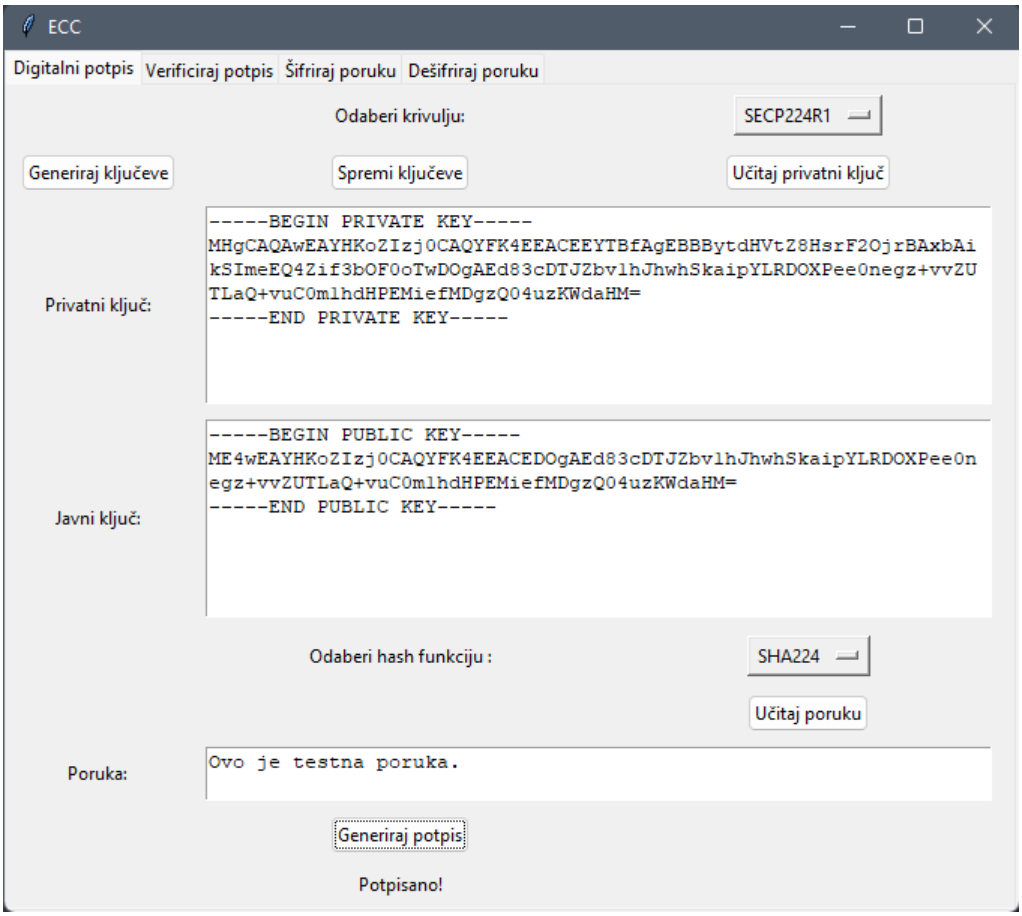
Pokretanjem aplikacije otvara se glavni prozor ECC koji sadrži četiri kartice (Slika 10.2.), po jedna za svaku funkcionalnost:



Slika 10.2 Početni prozor aplikacije

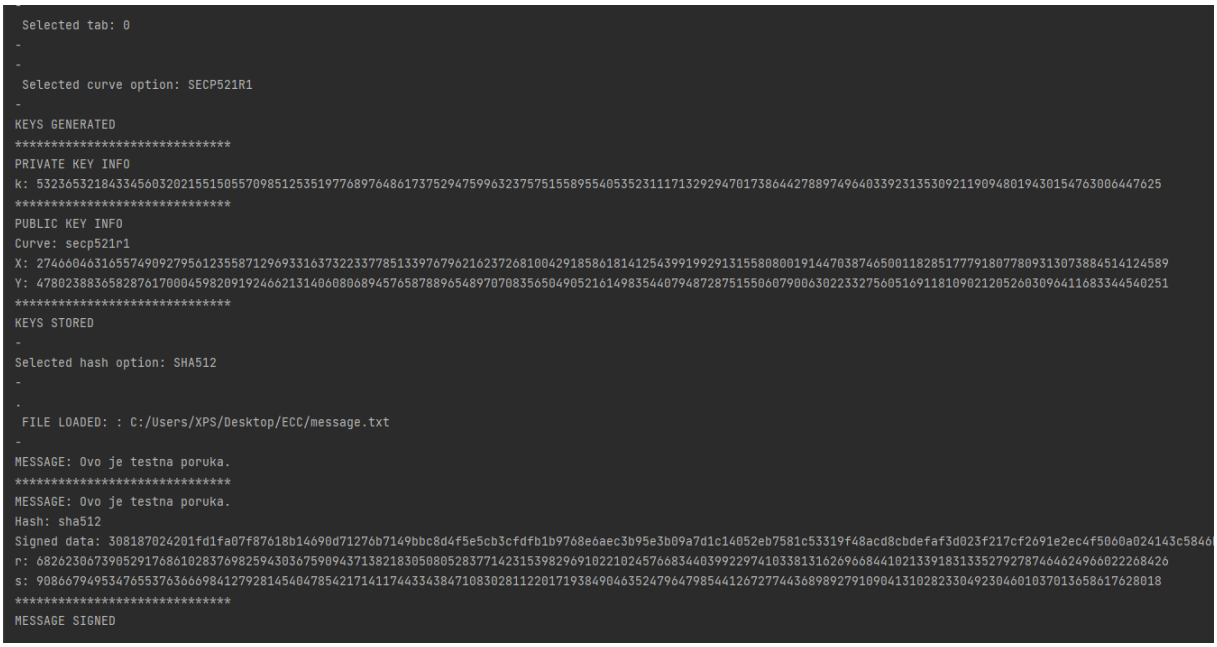
10.2.1 Digitalni potpis

Postupak digitalnog potpisivanja je sljedeći. U padajućem izborniku „Odaberi krivulju“ odaberemo krivulju, odnosno domenske parametre potrebne za generiranje ključeva. Imamo dvije opcije odabira ključeva, generiranje novog para ključeva ili učitavanje već postojećeg privatnog ključa. Na pritisak gumba „Generiraj ključeve“ generiraju se privatni i javni ključ nad odabaranom krivuljom te se ključevi prikazuju u PKCS formatu u tekstualnim okvirima ispod. Gumb „Spremi ključeve“ uzima ključeve iz tekstualnih okvira te ih sprema u datoteke private_key.pem odnosno public_key.pem. Gumb „Učitaj privatni ključ“ učitava postojeći privatni ključ kojim ćemo kasnije potpisati poruku. Zatim u padajućem izborniku „Hash“ odabiremo *hash* funkciju. Poruku možemo ručno upisati ili učitati iz postojeće datoteke pritiskom na gumb „Učitaj poruku“. Pritiskom na gumb „Generiraj potpis“ potpis se generira na temelju učitanih datoteka i odabranih parametara te se poruka i potpis spremaju u datoteke data.txt odnosno signature.txt (slika 10.3.) .



Slika 10.3 Postupak ECDSA digitalnog potpisa

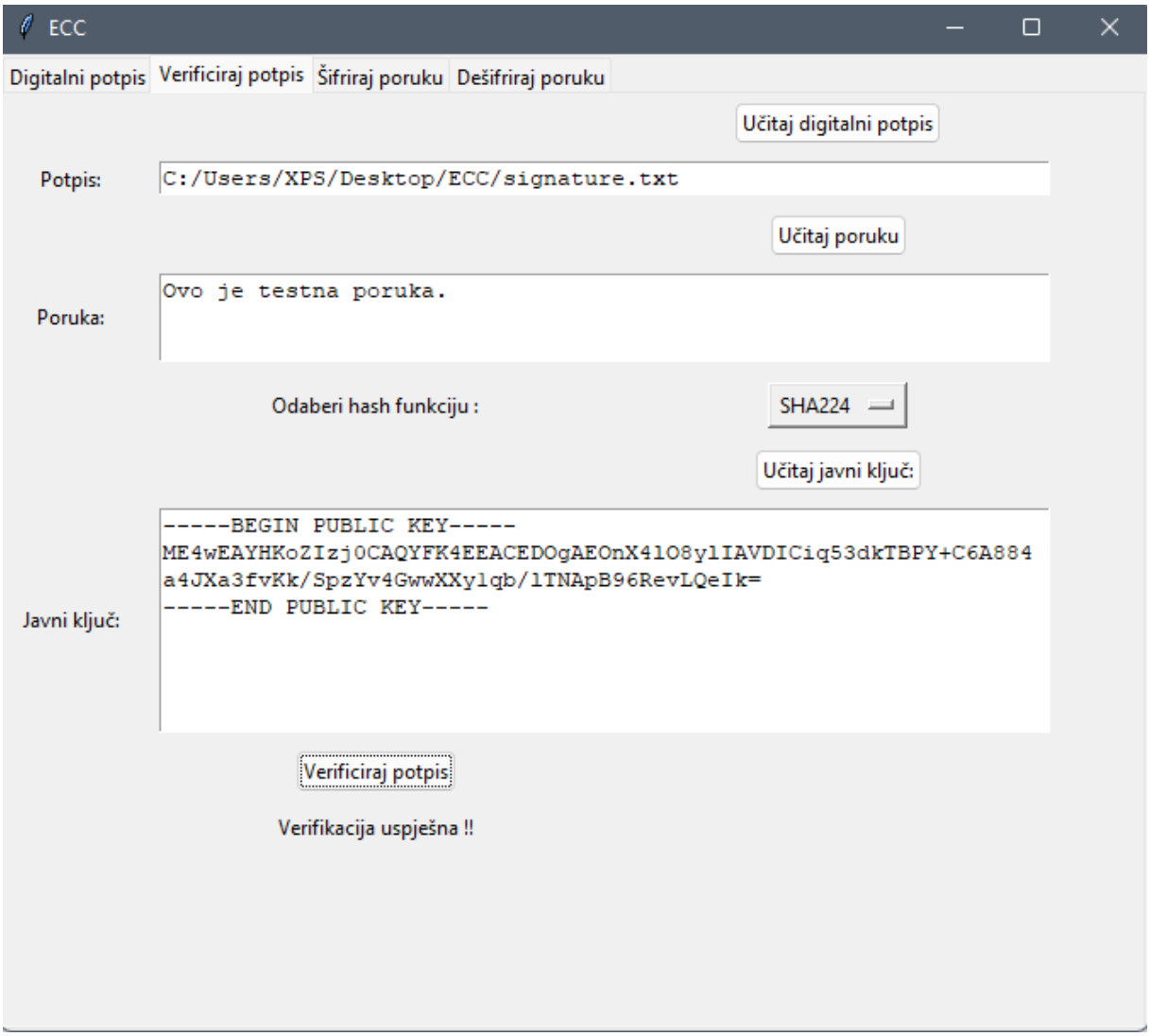
Također, za svaki korak učitavanja datoteke ili odabira parametara, u teminalu se ispisuju logovi napravljene akcije s vrijednostima parametara (slika 10.4.)



Slika 10.4 Primjer ispisa terminala za generiranje potpisa

10.2.2 Verifikacija digitalnog potpisa

Postupak verifikacije digitalnog potpisa je sljedeći. Odaberemo karticu „Verificiraj potpis“. Zatim klikom na gumbe „Učitaj digitalni potpis“ i „Učitaj poruku“ učitavamo datoteke generirane u procesu potpisivanja. Odabiremo *hash* funkciju kojom želimo verificirati potpis. Da bi verifikacija bila uspješna, moramo odabrati istu *hash* funkciju kojom je potpis generiran. Isto vrijedi i za javni ključ koji mora biti par privatnog ključa kojim je potpis generiran. Javni ključ učitavamo gumbom „Učitaj javni ključ“. Na kraju, pritiskom na „Verificiraj potpis“ vrši se verifikacija korištenjem učitanih i odabranih parametara. Ako je verifikacija uspješna aplikacija će ispisati „Verifikacija uspješna“ (Slika 10.5.), u suprotnom ispisat će „Verifikacija neuspješna“.



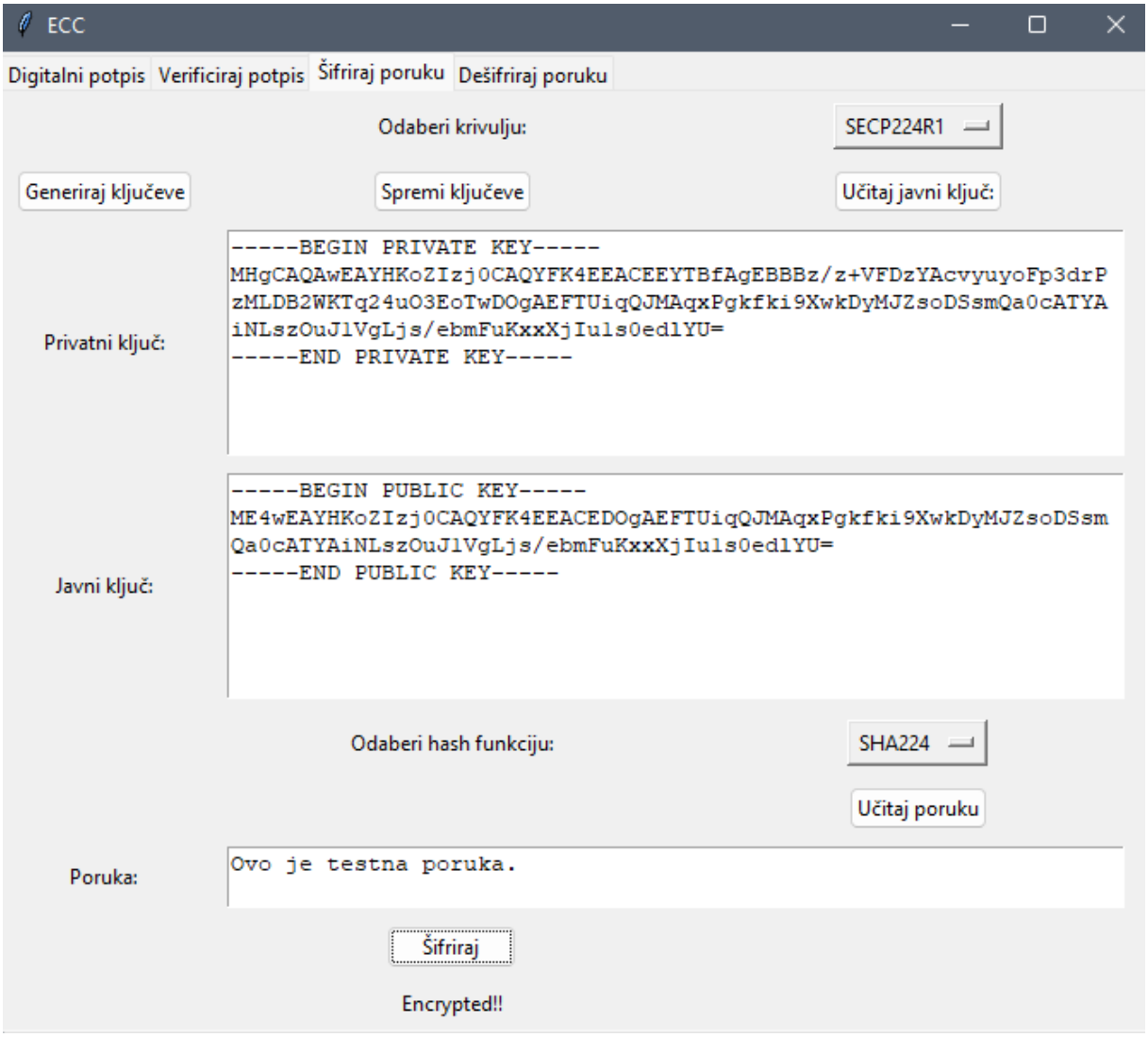
Slika 10.5 Postupak provjere ECDSA potpisa

Kao u prethodnom koraku, imamo ispis koraka i vrijednosti parametara u terminalu (Slika 10.6)

10.2.3 Šifriranje poruke

Postupak šifriranja poruke je sljedeći. U padajućem izborniku odaberemo krivulju, odnosno domenske parametre koji su nam potrebni za generiranje ključeva. Imamo dvije opcije odabira javnog ključa , možemo geneirati novi par ključeva ili učitati već postojeći javni ključ. Na pritisak gumba „Generiraj ključeve“ generiraju se privatni i javni ključ nad odabranom krivuljom te se ključevi prikazuju u PKCS formatu u tekstualnim okvirima ispod. Gumb „Spremi ključeve“ uzima ključeve iz tekstualnih okvira te ih sprema u datoteke

private_key.pem odnosno public_key.pem. Gumb „Učitaj javni ključ“ učitava već postojeći javni ključ kojim će se šifrirati poruka. Zatim u padajućem izborniku odabiremo *hash* funkciju. Poruku možemo ručno upisati ili učitati iz postojeće datoteke pritiskom na gumb „Učitaj poruku“. Pritiskom na gumb „Šifriraj“ poruka se šifrira na temelju učitanih datoteka i odabranih parametara te se šifrirana poruka sprema u datoteku encrypted_message.txt (Slika 10.6).

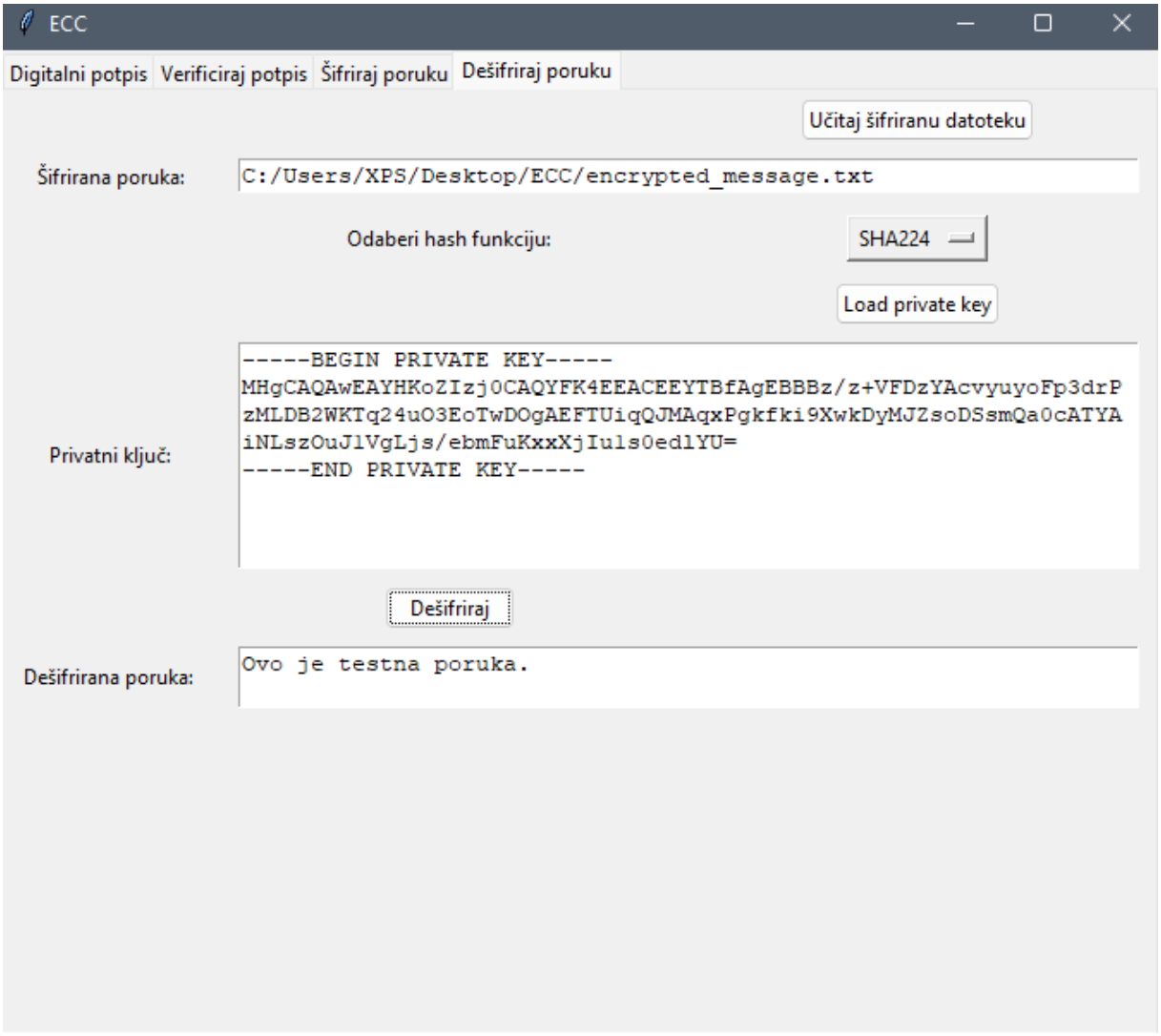


Slika 10.6 ECIES postupak šifriranja poruke

10.2.4 Dešifriranje poruke

Postupak šifriranja poruke je sljedeći. Šifriranu datoteku učitavamo tako da stisnemo na gumb „Učitaj šifriranu datoteku“ ili ručno unesemo putanju. Odaberemo *hash* funkciju koja

mora biti ista koja je korištena za šifriranje poruke. Zatim učitavamo privatni ključ koji je par s javnim ključem korištenim za šifriranje poruke. Pritiskom na gumb „Dešifriraj“ ispisuje se dešifrirana poruka (Slika 10.7.). U slučaju da se izračunata oznaka provjere šifrirane poruke ne podudara s oznakom provjere izračunatom kod šifriranja, program vraća Error.



Slika 10.7 ECIES postupak dešifriranja poruke

11 ZAKLJUČAK

Kriptografija temeljena na eliptičnim krivuljama moćna je tehnika šifriranja koja se široko koristi u modernim digitalnim komunikacijama. ECC se temelji na matematici eliptičkih krivulja te pruža višu razinu sigurnosti od tradicionalnih metoda šifriranja poput RSA. Osim toga, ECC zahtijeva kraće duljine ključeva od RSA kako bi se postigla ista razina sigurnosti, što ga čini učinkovitijim i bržim za korištenje. ECC se danas koristi u širokom rasponu aplikacija, od osiguravanja online transakcija do zaštite osjetljivih podataka pohranjenih na mobilnim uređajima. Također se široko koristi u IoT industriji, gdje je bitno imati jednostavnu i učinkovitu metodu enkripcije. Kako se sve više i više podataka prenosi putem interneta, neophodno je imati učinkovite metode šifriranja kao što je ECC kako bismo osigurali sigurnost i privatnost nad podacima. To nam omogućuje svojom višom razinom sigurnosti, kraćim duljinama ključeva i otpornošću na kvantno računalstvo, barem za sada.

12 LITERATURA

[1] D.Hankerson, A. Menezes, S.Vanstone: „Guide to Elliptic Curve Cryptography“, Springer-VL-rlag New York, New York, srpanj 2003.

[2] Sectigo store: „Types of Encryption: What to Know About Symmetric vs Asymmetric Encryption“, s Interneta: https://sectigostore.com/blog/types-of-encryption-what-to-know-about-symmetric-vs-asymmetric-encryption/, travanj 2020.

[3] Daniel R. L. Brown: “ Standards for Efficient Cryptography 1 (SEC 1)“, Certicom Corp., 2009.

[4] Wikipedia; „Elliptic curve“, s Interneta: https://en.wikipedia.org/wiki/Elliptic_curve, rujan 2023.

[5] Sherif H. AbdElHaleem, Salwa K. Abd-El-Hafiz, Ahmed G. Radwan; „A generalized framework for elliptic curves based PRNG and its utilization in image encryption“, s interneta: https://www.nature.com/articles/s41598-022-17045-x, kolovoz 2022.

[6] Andrej Dujella: „Algoritmi za eliptičke krivulje“, s interneta: https://web.math.pmf.unizg.hr/~duje/elipticke/algelip.pdf, 2008.

[7] H. Krawczyk, M. Bellare, R. Canetti: „RFC 2104 - HMAC: Keyed-Hashing for Message Authentication“, s Interneta: http://www.faqs.org/rfcs/rfc2104.html, veljača 1997.

[8] Martin Grmek: „HMAC“, s interneta: http://sigurnost.zemris.fer.hr/algoritmi/hash/2005_grmek/index.html, 2004.

[9] Josh Lake: „What is a key derivation function (KDF) and how do they work?“, s interneta: https://www.comparitech.com/blog/information-security/key-derivation-function-kdf/, kolovoz 2022.

[10] B. Kaliski: „Password-Based Cryptography Specification Version 2.0“, s Interneta: https://datatracker.ietf.org/doc/html/rfc2898#section-5.2, rujan 2000.

[11] Srikanthreddy Baddam: „Location Based Encryption-Decryption System For Android“ s Interneta: https://www.researchgate.net/publication/337223732_Location_Based_Encryption-Decryption_System_For_Android, studeni 2019.

[12] CARnet: „Korištenje eliptičnih krivulja u kriptografiji“, s interneta: https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2006-11-169.pdf, rujan 2009.

[13] Svetlin Nakov: „practical Cryptography for Developers“, s Interneta: https://cryptobook.nakov.com/digital-signatures, rujan 2023.

[14] Adel Ali Ahmed, Omar M. Barukab: „Unforgeable Digital Signature Integrated into Lightweight Encryption Based on Effective ECDH for Cybersecurity Mechanism in Internet of Things“, s Interneta: https://www.mdpi.com/2227-9717/10/12/2631?type=check_update&version=3, prosinac 2022.

[15] Wikipedia: „Elliptic-curve Diffie–Hellman“, s interneta; https://en.wikipedia.org/wiki/Elliptic-curve_Diffie%E2%80%93Hellman, rujan 2023.

[16] V. Gayoso Martinez, L. Hernandez EncinasAND A. Queiruga Dios: „Security and Practical Considerations when Implementing the Elliptic Curve Integrated Encryption Scheme“, s Interneta: https://www.tic.itefi.csic.es/CIBERDINE/Documetos/Cryptologia%20-%20Security%20and%20practical%20considerations%20when%20implementing%20ECIES%20-%20v1.0.pdf, rujan 20023.

[17] Wikipedia: „Integrated Encryption Scheme“ https://en.wikipedia.org/wiki/Integrated_Encryption_Scheme, s interneta, rujan 2023.

[18] „Cryptography documentation“, s Interneta:https://cryptography.io/en/latest/, rujan 2023.

13 POPIS SLIKA

14 POPIS TABLICA

15 POPIS OZNAKA I KRATICA

Kratica	Značenje
AES	Advanced Encryption Standard
DEC	Decryption scheme
DES	Data Encryption Standard
ECC	Elliptic curve cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ENC	Encryption scheme
HAMC	Hash-based Message Authentication Code
HKDF	Hash-based Message Authentication Code
KA	Key-agreement protocol
KDF	Key Derivation Function
MAC	Message Authentication Code
MD5	Message-Digest Algorithm
NIST	National Institute of Standards and Technology
NSA	National Security Agency
PBKDF2	Password-based Key Derivation Function
PKCS	Public-Key Cryptography Standards
PRF	Pseudo-Random Functions
RC4	Rivest Cipher 4
ROI	Return on investment
RSA	Rivest–Shamir–Adleman

SHA-2	Secure Hash Algorithm 2
SHA-3	Secure Hash Algorithm 3
SIGN	Signature Scheme

16 SAŽETAK I KLJUČNE RIJEČI

Ovaj diplomski rad je fokusiran na kriptografiju zasnovanu na eliptičnim krivuljama. U uvodnom dijelu opisano je kako je došlo do potrebe za kriptografijom i kako se razvijala kroz povijest. Zatim drugo objašnava osnove kriptografije te njenu podjelu i način na koji funkcionira. U trećem poglavlju se fokusiramo na matematičke principe na kojima je ECC zasnovana. U četvrtom i petom poglavlju, upoznajemo se sa MAC-om i KDF-om, pomoćnim funkcionalnostima koje se koriste u kriptografiji općenito pa i u ECC-u. U poglavljima šest, sedam i osam opisujemo neke od glavnih kriptografskih shema zasnovanih na eliptičnim krivuljama te je u devetom poglavlju predstavljen u Pythonu implemetirani ECC kriptosustav.

Ključne riječi: kriptografija, eliptične krivulje, ECC, ECDSA, ECDHA, ECIES, Python

17 ABSTRACT AND KEYWORDS

This thesis is focused on cryptography based on elliptic curves. The introductory part describes how the need for cryptography came about and how it developed throughout history. Then the second explains the basics of cryptography and its division and the way it works. In the third chapter, we focus on the mathematical principles on which ECC is based. In the fourth and fifth chapters, we get acquainted with MAC and KDF, auxiliary functionalities that are used in cryptography in general and in ECC as well. In chapters six, seven and eight we describe how some of the main cryptographic schemes based on elliptic curves work, and in chapter nine an ECC cryptosystem implemented in Python is presented.

Keywords: cryptography, elliptic curves, ECC, ECDSA, ECDHA, ECIES, Python