MATEMATIČKI FAKULTET,
UNIVERZITET U BEOGRADU

MASTER RAD

Primjer implementacije asimetričnog kriptosistema

Student:
Boris Milojković

Mentor:
Dr. Žarko Mijajlović

Komisija:
Dr. Miodrag Živković
Dr. Aleksandar Lipkoviski

Beograd, 2009 godine

Pitanja
1. Zašto JAVA?
2. Da li je za neki od asim. sistem dokazano da je problem dešifrovanja NP-težak?

Odbrana: 11.09.2009.
Komisija:
1. Aleksandar Lipkovski, predsjednik
2. Žarko Mijajlović, mentor
3. Miodrag Živković

Sažetak

U radu će biti predstavljene neke osnove kriptografskih sistema zasnovanih na osobinama problema diskretnog logaritma u konačnim cikličnim grupama. Posebno, biće opisane osnovne osobine grupa koje odgovaraju *eliptičkim krivama* nad konačnim poljima. Takođe, data je i implementacija osnovnih kriptografskih protokola zasnovanih na grupama definisanim *eliptičkim krivama* nad poljem $\mathbb{Z}_p$, gdje je p neparan prost broj.

Sadržaj

MATEMATIČKI FAKULTET,
UNIVERZITET U BEOGRADU

MASTER RAD

Primjer implementacije asimetričnog kriptosistema

Student:
Boris Milojković

Mentor:
Dr. Žarko Mijajlović

Komisija:
Dr. Miodrag Živković
Dr. Aleksandar Lipkoviski

Beograd, 2009 godine

Pitanja
1. Zašto JAVA?
2. Da li je za neki od asim. sistem dokazano da je problem dešifrovanja NP-težak?

Odbrana: 11.09.2009.
Komisija:
1. Aleksandar Lipkovski, predsj. kom.
2. Žarko Mijajlović, mentor
3. Miodrag Živković

Sažetak

U radu će biti predstavljene neke osnove kriptografskih sistema zasnovanih na osobinama problema diskretnog logaritma u konačnim cikličnim grupama. Posebno, biće opisane osnovne osobine grupa koje odgovaraju *eliptičkim krivama* nad konačnim poljima. Takođe, data je i implementacija osnovnih kriptografskih protokola zasnovanih na grupama definisanim *eliptičkim krivama* nad poljem $\mathbb{Z}_p$, gdje je p neparan prost broj.

Sadržaj

Glava 1

Osnovni pojmovi

Ljudi su oduvijek razmijenjivali informacije, međutim u ne rijetkim situacijama potreba da te informacije ne može "znati" niko drugi osim onaj kome je informacija namjenjena, nametnula je potrebu sakrivanja informacija. Sakrivena informacija bi imala vrijednost samo onome kome se uputila. Niko drugi ne bi trebao da zna za nju. *Kriptografija* predstavlja umijeće skrivanja informacija. Mogli bi reći da od kada postoji potreba za razmijenjivanje informacija postoji potreba za skrivanjem, tako da elemente sakrivanja informacija nalazimo u samim početcima novije civilizacije. U jednom velikom istorijskom periodu kriptografija je podrazumijevala samo sakrivanje informacija, međutim moderno doba nametnulo je i niz drugih problema koji su neophodni za tajnu, odnosno sigurnu komunikaciju.

Pitanje koje je najbliskije vezano sa pojmom tajne komunikacije je da li postoji mogućnost da neko kome informacija nije namjenjena dođe u posijed informacije. Naravno, čitava ideja tajnog pisanja je u tome da se takvi slučajevi onemoguće. *Kriptoanaliza* predstavlja umijeće otkrivanja informacija bez znanja svih parametara tajne komunikacije. Istorija je pokazala da su ove dvije discipline uvijek pratile jedna drugu, kao i da je ne rijetko kriptoanaliza dovodila do novih sistema u kriptografiji.

U današnje vrijeme i kriptografija i kriptologija gube značenje "umijeća"i sve više poprimaju oblike oblasti u pravim naukama, prije svega u matematici. Takođe, napredovanje savremene računarske tehnike, distribucija podataka i servisa dovodi do sve veće potrebe za korišćenjem kriptografije u svakodnevnom životu. Internet, mobilna i IP telefonija, bežične računarske mreže, digitalni potpis,... Možemo reći da skoro i da ne postoji ni jedna vrsta "moderne" komunikacije koja nema neke elemente kriptografije.

1.1 Uvod u kriptografiju

U početcima primjene kriptografije podrazumijevalo se da je jedini neophodan dio kriptografskog sistema dio koji omogućava šifriranje, odnosno dešifriranje.

frovanje poruke. Mogli bi reći da je "kriptografski sistem" za cilj imao jedino sakrivanje podataka. Tačnije nije bilo potrebe za implementacijom nekih drugih principa sigurne komunikacije.

Osnovne kriptoprimitive

U današnje vrijeme kompleksnost i raširenost komunikacije, zahtijeva mnogo više od samog skrivanja podataka. Osnovni principi koje zahtijevamo od jednog kriptografskog sistema su:

- tajnost,
- integeritet podataka,
- autentičnost pošiljaoca,
- autentičnost izvora podataka,

Da bi se što formalnije mogao zasnovati jedan kriptografski sistem, uvedimo sledeće pojmove [1]:

- $\mathcal{A}$ ozanačava konačni skup, alfabet. Na primjer $\mathcal{A} = \{0, 1\}$ označava binarni alfabet. Ovaj alfabet se dosta često koristi, sa obzirom da se mnogi drugi alfabeti jednostavno kodiraju u ovaj alfabet. Na primjer sa obzirom da je $2^5 = 32$ svako slovo naše azbuke se može na jedinstven način predstaviti jednom "petorkom" binarnog alfabeta.
- $\mathcal{M}$ označava skup poruka. $\mathcal{M}$ se sastoji od niski simbola iz alfabeta, $m \in \mathcal{M}$ često nazivamo i osnovnom (polaznom) porukom ili otvorenim tekstom.
- $\mathcal{C}$ označava skup šifrata, odnosno skup šifriranih poruka. $\mathcal{C}$ se sastoji od niski simbola alfabeta koji ne mora biti isti kao alfabet $\mathcal{A}$. Element $c \in \mathcal{C}$ nazivamo šifratom, odnosno šifriranim tekstom (kriptogramom).
- $\mathcal{K}$ označava skup ključeva. Bilo koji $e \in \mathcal{K}$ nazivamo ključem.
- Svaki element $e \in \mathcal{K}$ jedinstveno određuje bijekciju iz $\mathcal{M}$ u $\mathcal{C}$, tu bijekciju označavamo sa $E_e \colon \mathcal{M} \to \mathcal{C}$. E_e nazivamo i kriptujućom funkcijom ili transformacijom (Šifrovanje).
- Svaki element $d \in \mathcal{K}$ jedinstveno određuje bijekciju iz $\mathcal{C}$ u $\mathcal{M}$, tu bijekciju označavamo sa $D_d \colon \mathcal{C} \to \mathcal{M}$. D_d nazivamo i dekriptujućom funkcijom ili transformacijom (Dešifrovanje).
- Postupak primjene transformacije E_e na $m \in \mathcal{M}$ nazivamo kriptovanjem, odnosno šifriranjem.
- Postupak primjene transformacije D_d na $c \in \mathcal{C}$ nazivamo dekriptovanjem, odnosno dešifriranjem.

- Kripto shema se sadrži od skupova $\{E_e : e \in \mathcal{K}\}$ i odgovarajućeg $\{D_d : d \in \mathcal{K}\}$ takvih da za svako $e \in \mathcal{K}$ postoji jedinstveno $d \in \mathcal{K}$ takvo da je $D_d = E_e^{-1}$, tj.
$$D_d(E_e(m)) = m,\ m \in \mathcal{M}$$

- Ključeve e i d u kripto shemi nazivamo i parom ključeva. Par ključeva često označavamo (e, d), primjetimo da ključevi e i d mogu biti i jednaki.

Takođe, uvedimo i "osobe" Alisu, Boba i Evu. Osobe ustvari predstavljaju entitete[1] koji žele da razmijene poruke, ili entitete koje žele da kompromituju sam kriptosistem. Naime, praksa u opisivanju kriptografskih shema, je da umjesto recimo entitet A zamjenimo sa Alisom, entitet B zamjenimo sa Bobom, entitet E koji pokušava da dođe do sadržaja kriptograma, odnosno polaznog teksta nazivamo Evom.

Navodimo primjer kako kripto shema može da se koristi da bi se obezbijedila tajnost. Alis i Bob tajno[2] izaberu par ključeva (e, d). Kada Alis želi da pošalje poruku $m \in \mathcal{M}$ Bobu, ona "izračuna" $c = E_e(m)$ i pošalje dobijenu vrijednost Bobu. Kada Bob primi poruku c on "izračuna" $D_d(c) = m$ i tako dobije polaznu poruku. Pitanje koje proizilazi iz ove sheme je zašto su nam ključevi uopšte potrebni, tačnije, zašto se shema ne sastoji samo od para transformacija? Praksa je pokazala da je sistem ključeva mnogo fleksibilniji, odnosno da je mnogo jednostavnije mijenjati same ključeve nego same transformacije (Princip ključa i katanca, lakše je zamijeniti ključ nego mehanizam katanca, ovakav sistem pruža i veću fleksibilnost).

Jedna od osnovnih pretpostavki u kriptografiji je ta da su skupovi $\mathcal{M}$, $\mathcal{C}$, $\mathcal{K}$, $\{E_e : e \in \mathcal{K}\}$, $\{D_d : d \in \mathcal{K}\}$ javni. Jedina stvar koja je "tajna" je par ključeva (e, d) koji učesnici u tajnoj komunikaciji moraju da izaberu. Takođe, moguće je čuvati u tajnosti i same transformacije, ali praksa je pokazala da je čuvanje u tajnosti samih transformacija dosta težak zadatak[3].

Za kripto shemu kažemo da je moguće "razbiti", ako neka treća osoba, bez prethodnog znanja para ključeva (e, d), može na sistematičan način doći do otvorenog teksta polazeći od šifrovanog teksta u nekom "smislenom"[4] vremenskom intervalu, ili drastičnije, može "izračunati" ključ kojim je transformacija definisana. Konteksti u kojima se dešava "razbijanje" kriptosistema mogu biti različiti, a od kojih su najčešći:

1. Napad na osnovu šifrata,
2. Napad na osnovu parova (otvoreni tekst, šifrat),
3. Napad na osnovu izabranog osnovnog teksta

[1] U ovom kontekstu entitet može biti fizička osoba, računar, programska komponenta i sl.

[2] Pod ovim se misli da niko drugi nije u mogućnosti da dođe u posjed ovih ključeva.

[3] Primjer je čuveni RC4 sistem čiji je osnovni algoritam bio tajna, ali se jednog dana pojavio na *Cypherpunks* mejling listi

[4] Ovo se u osnovi odnosi na vrijeme (tačnije kompleksnost) računanja funkcije koja bi od šifrovanog teksta dala polazni tekst.

Kerkhofsov princip

1883 godine August Kerkofs (Auguste Kerckhoffs), holandski lingvista i kriptograf izdao je članak *"La Cryptographie Militare - Journal des sciences militaires, vol. IX, Jan. 1883"* (vojna kriptografija) u kojem je postavio šest zahtjeva koje treba da zadovoljava kriptografski sistem.

1. sistem bi trebao biti, ako ne "teoretski" moguće razbiti, onda nemoguće razbiti u praksi;
2. ako su detalji sistema kompromitovani, to ne bi smijelo stvoriti neprilike učesnicima u sigurnoj komunikaciji;
3. ključevi trebaju biti lako "pamtljivi" bez zapisivanja, kao i lako promijenjivi;
4. kriptogram bi trebao biti prenosiv putem telegrafa;
5. aparat kojim se vrši kriptovanje trebao bi biti prenosiv i trebao bi biti operabilan od strane jedne osobe;
6. sistem bi trebao biti jednostavan za korišćenje, bez korišćenja velikog mentalnog napora i bez korišćenja velikog skupa pravila;

Većinu pravila moramo uzeti u konteksu vremena kada je članak objavljen, ali pravilo broj 2 ostaje na snazi i danas, često to pravilo nazivamo i *Kerkofsov princip*. U osnovi Kerkofsov princip kaže da kriptosistem mora biti tako osmišljen da zavisi samo od ključeva. Ovaj princip je postavljen i od strane Kloda Šenona (*Claude Shannon*), tvorca teorije informacija i jednog od ključnih figura u modernoj kriptografiji. Kod Šenona princip glasi:

"Neprijatelj poznaje sistem"

i naziva se Šenonovom maksimom (*Shannon's maxim*).

Simetrična kripto shema

Posmatrajmo kripto shemu koja se sastoji od skupova transformacija $\{E_e : e \in \mathcal{K}\}$ i $\{D_d : d \in \mathcal{K}\}$ gdje je $\mathcal{K}$ skup ključeva. Za shemu kriptovanja kažemo da je sa simetričnim ključem ako je za svaki par ključeva (e, d) na "jednostavan" način možemo da odredimo d na osnovu e i e na osnovu d. Izraz simetrični dolazi otuda jer je u najvećem broju slučajeva $e = d$.

Pogledajmo primjer jednog poznatog sistema kriptovanja, tzv Cezarov sistem (Caesar chiper). Ovaj sistem se bazira na jednostavnoj *substituciji* alfabeta. Naime, ispod "standardnog" alfabeta napišemo alfabet pomjeren za n mjesta i tu permutaciju označimo sa e:

$$e = \begin{pmatrix} \text{A} & \text{B} & \text{C} & \dots & \text{U} & \text{V} & \text{W} & \text{X} & \text{Y} & \text{Z} \\ \text{D} & \text{E} & \text{F} & \dots & \text{X} & \text{Y} & \text{Z} & \text{A} & \text{B} & \text{C} \end{pmatrix}$$

Permutacija e nam služi kao kriptujuća funkcija, da bi dekriptovali jednostavno izaberemo $d = e^{-1}$. Formalno e možemo da zapišemo $E_n(x) = (x+n) \bmod 26$, a d kao $D_n(x) = (x-n) \bmod 26$ pri čemu je za naš primjer $n = 3$. Ovo je vrlo jednostavna kripto shema, ali se zadržala jako dugo u primjeni. Interesantno je kako je došlo do "razbijanja" ove šifre. Tačno se ne zna ko je prvi primjetio da pri ovoj shemi kriptovanja učestalost elementa alfabeta ostaje nepromjenjena, ali je arapski matematičar Al Kindi[5] koji je živio u 9 vijeku napisao dijelo *"Manuskript o dešifrovanju kriptografskih poruka"* u kojem je opisan algoritam dešifrovanja Cezarovog kriptosistema. Vjeruje se da je ovaj kriptoanalitičarski princip nasleđe bogate tradicije lingvističke analize *kurana*.

Jedan od osnovnih problema u simetričnim kriptosistemima je koncept same sheme. Naime ne postoji mogućnost tajne razmijene informacija ukoliko se ne "razmijene" ključevi, tačnije pored razmijene samih informacija učesnici u tajnoj komunikaciji moraju razmijeniti i ključeve. Znači da je neophodno postojanje tkz. *"sigurnog kanala"* preko kojeg se razmijenjuju tajne informacije. Sama komunikacija tačnije razmijena poruka ide preko *"nesigurnog"* kanala (poruke su kriptovane) samim tim i svakome dostupne. Možemo reći i da je problem *"distribucije ključa"* bio jedan od najvećih problema kriptografije.

Prije nego se upustimo u detaljniji opis rješenja ovog problema pogledajmo jednu kripto shemu [2].

Vernamov sisitem *(Gilbert Sanford Vernam)* je "stream" kriptosistem definisan na alfabetu $\mathcal{A} = \{0,1\}$. Binarna poruka $m = m_1m_2m_3...m_t$ se kriptuje sa binarnim ključem $k = k_1k_2k_3...k_t$ iste dužine. Transformaciju definišemo:

$$c = m \oplus k$$

tačnije:

$$c_i = m_i \oplus k_i, \quad 1 \leq i \leq t.$$

Dekriptujuća transformacija je ista kao i kripujuća, naime važi:

$$m = c \oplus k$$

U slučaju da je ključ k slučajno izabran i da se koristi samo jedanput, tačnije za kriptovanje samo jedne poruke, *Vernamov sistem* nazivamo i *One time padom* - (**OTP**).

Razmotrimo slučaj da *Alis* i *Bob* žele da razmijene poruku $m = m_1m_2...m_n$ pri čemu su se ranije sastali i definisali ključ $k = k_1k_2...k_n$ *bacanjem novčića* n puta (na slučajan način). Pod pretpostavkom da *Eva* može da vidi $n-1$ parova

$$(m_i, c_i), \quad 1 \leq i \leq n-1$$

postavlja se pitanje šta može da zaključi Eva o m_n ukoliko zna i poslednji element kriptograma c_n. Sa obzirom da je k_n slučajno izabrano verovatnoća da je $c_n = m_n$ je $\frac{1}{2}$, isto koliko i da je $c_n = m_n \oplus 1$. Ako na trenutak pretpostavimo da je i sama poruka koju Alis šalje *"slučajna"*, najbolje što Eva može je da sa

[5] Abu Jusuf ibn Ishak ibn as Sabah ibn Omran ibn Ismail al Kindi

verovatnoćom $\frac{1}{2}$ pogodi m_n. Kako u praksi poruke nisu slučajne, recimo da Eva zna da Alisa *"više voli"* jedinice umjesto nula, tada Eva može da pretpostavi da je $m_n = 1$ u najvećem broju puta. Bitna stvar je da je Eva ovo mogla da zaključi i ne znajući ništa o c_n. Eva nije dobila nikakve nove *informacije* gledajući kriptogram. Vidimo da u ovakvoj postavci kriptogram ne nosi nikakvu *informaciju* o samoj poruci. Ovo je osnovna ideja o savršenoj tajnosti u Šenonovom modelu kriptografije. Tačnije:

"Kriptogram ne otkriva nikakve dodatne informacije o otvorenoj poruci."

Glava 2

Kriptografija javnim ključem

Već ranije smo pomenuli da posjedujemo "matematička" sredstva kojim bi obezbijedili *apsolutnu sigurnost* sistema komunikacije. Međutim ovakav koncept, pored dosta teške implementacije, ima u svojoj osnovi još jednu bitnu pretpostavku, pretpostavku čije će "slabljenje" dovesti do potpuno novog pravca u razvoju kriptografije, kao i ogromnog upliva matematičkih teorija u samu kriptografiju. Naime, radi se o tome da razvoj sistema tipa **OTP**-a pretpostavlja da Eva ima "neograničenu" računarsku snagu na raspolaganju. Tačnije, čak i pri ovakvoj pretpostavci, sistem nam daje sigurnost pri komunikaciji. Dva su pitanja koja proizilaze iz kriptosistema kao što je **OTP**. Kao i kod svih simetričnih kriptosistema nameće se problem distribucije ključa (odnosno ključeva). Pored toga, nameće se i pitanje da li je neophodna pretpostavka da Eva posjeduje "neograničenu" računarsku snagu na raspolaganju? Ova dva pitanja su bila ključna za razvoj moderne kriptografije.

2.1 Uvod

I ranije smo pomenuli da je najveći problem kriptografije bio problem distribucije ključa (odnosno ključeva) u simetričnim kriptosistemima. Prije nego je kriptografija postala neophodan aspekt poslovne komunikacije problem se prevazilazio organizacijom dosta "robustnih" sigurnih kanala, ali sve većom primjenom računara u poslovnom svijetu i povećanim obimom komunikacije, problem distribucije ključeva postajao je sve veći i veći.

1976 godina predstavlja prekretnicu u razvoju kriptografije. Te godine Vitfild Difi(*Whitfield Diffie*) i Martin Helman (*Martin Hellman*) objavljuju članak "Novi pravci u kriptografiji" (*"New directions in cryptography"* [7]) u kojem je prvi put predstavljena shema komunikacije u kojoj Alisa i Bob razmjenjuju ključ, a da pri tome *javno* komuniciraju. Navodimo izvod iz [7] koji najbolje opisuje uticaj pomenutog rada na dalji razvoj kriptografije:

> Danas stojimo na ivici revolucije u kriptografiji. Razvoj jeftinih digitalnih uređaja nas je oslobodio ograničenja koja su nam nametala mehanički uređaji za računanje kao i što je spustio cijenu skupih kriptografskih uređaja koji mogu biti korišćeni u komercijalne svrhe u automatma za novac ili računarskim terminalima. Sa druge strane, takve aplikacije zahtjevaju novi tip kriptografije koja minimizuje potrebu za sigurnom distribucijom ključeva i daje nam ekvivalent ručnog potpisa. Istovremeno, razvoj teorije informacija i računarske nauke obećavaju nam dokazivo sigurne kriptosisteme, mijenjajući ovu starinsku umjetnost u nauku.

Naime, u radu je po prvi put predstavljeno rješenje problema distribucije ključa. Tačnije, Alis i Bob razmijenjuju ključ, tačnije "usaglase" njegovu vrijednost, (koji kasnije mogu koristiti u klasičnom smislu, pri simetričnom kriptovanju) bez da koriste bilo kakav tajni kanal. Uopšteno, shema se odvija tako što Alis i Bob svako za sebe izabere par ključeva (e_A, d_A) i (e_B, d_B), razmijene ključeve e_A i e_B (tkz. javne ključeve), pri čemu ključeve d_A i d_B zadržavaju za sebe (tkz. privatne ključeve). U shemi koja je prezentovana u radu, Alis i Bob dalje koriste svoje privatne ključeve da *izračunaju* zajednički ključ. Ova shema se često naziva i usaglašavanje ključa (*Key Agreement*), tačnije Difi-Helman usaglašavanje ključa (*Diffie-Hellman Key Agreement*) i predstavlja jedan vid razmijene ključa. Takođe data je i shema kriptovanja, odnosno dekriptovanja, koristeći par ključeva sa strane svakog učesnika u shemi. Neka je $\{E_e : e \in \mathcal{K}\}$ skup kriptujućih transformacija i neka je $\{D_d : d \in \mathcal{K}\}$ njemu odgovarajući skup dekriptujućih transformacija. Uočimo bilo koji par transformacija (E_e, D_d). Pretpostavimo da za bilo koji kriptogram $c \in \mathcal{C}$ nije moguće (u nekom smislenom vremenu) naći $m \in \mathcal{M}$ tako da je $E_e(m) = c$. E_e možemo gledati i kao prividno jednosmernu (*One way trap door*) funkciju. Kada Alis želi da pošalje Bobu poruku, poruku kriptuje Bobovim javnim ključem e_B, tačnije transformacijom E_{e_B}, tada je Bob "jedini" koji može da dekriptuje poruku jer posjeduje privatni ključ d_B, tačnije zna inverznu transformaciju D_{d_B}. Primjetimo da se sigurnost sistema u osnovi zasniva na *kompleksnosti* računanja funkcije D_d za poznato e. Sama shema usaglašavanje ključa originalno pripada Helmanu dok ideja asimertičnih ključeva originalno pripada Difiju. Međutim po navodima samog Helmana [10] veliki uticaj na samu ideju je dao Ralf Merkle(*Ralph Merkle*):

> Sistem koji sam ja nazvao ax1x2 sistem u ovom članku je postao poznat kao Difi-Helman usaglašavanje ključeva. Iako smo Difi i ja prvi put opisali sistem to je sistem koji je i Merkle razvio. Kao takav sistem bi trebalo zvati Difi-Helman-Merkle usaglašavanje ključeva.

Interesantno je da je Merkle sistem distribucije ključa opisao još kao student u svom seminarskom radu na smeru računarska sigurnost (naziv rada bio je *"Sigurna komunikacija preko nesigurnih kanala"*), ali sam koncept nije bio jasan profesoru (*Lance Hoffman*). Sama shema je tekla na sledeći način:

1. Bob generiše veliki broj (recimo 2^{20}) poruka (*zagonetki*) tipa : poruka x, ključ y, pri čemu je x slučajan broj, a y slučajan ključ. Koristeći

simetrični algoritam kriptuje svaku od ovih poruka sa različitim ključem. Tako kriptovane poruke šalje Alisi.

2. Alisa izabira bilo koju poruku i metodom *"grube sile"* otkriva polaznu (otvorenu) poruku.

3. Alisa kriptuje svoju poruku sa ključem koji je ona dobila kao sastavni dio otvorene poruke. Tako kriptovanu poruku šalje Bobu zajedno sa brojem poruke.

4. Bob na osnovu broja *zagonetke* zna koji je ključ Alisa koristila i jednostavno dekriptuje poruku.

Eva može da *"razbije"* sistem ali potrebno je mnogo više resursa, nego što je Alisi potrebno. Naime Eva bi morala da *"razbije"* svaku od zagonetki. Iako dosta nesiguran (i ne praktičan) Merkleov sistem nosi u sebi suštinu svakog kriptosistema sa javnim ključem, a to je da onaj entitet kome se poruka šalje učestvuje u procesu kriptovanja poruke. Navedimo samo osnovne aktere i jedne *"alternativne"* verzije otkrivanja iste sheme (Difi - Helman).

Engleska verzija - GCHQ

Naime, iako je "zvanična" verzija otkrivanja kriptografije javnim ključem uglavnom vezana za radove Helmana, Difija i Merklea, vrijedi napomenuti da postoji i "jedna" nezvanična verzija otkrivanja, istog protokola. Ova verzija je dugo vremena čuvana u tajnosti sa obzirom da čitava priča potiče iz Britanske vlade, tačnije iz vladinog štaba za komunikaciju, tzv. **GCHQ** - *Government Communication Headquarters*, instituciji koja je nastala na ostatcima čuvenog *Blečli Parka.*[1] Nekoliko godina prije Helmana i Difija, polazeći od problema distribucije ključeva, tačnije tragajući za rješenjem ovog problema, Džejms Elis (*James Ellice*) jedan od najuspješnijih kriptografa u GCHQ, je naišao na rad (ratni izvještaj) nepoznatog autora firme Bell Telephone (*"Final Report on Project C43"*). U tom izvještaju govori se o sigurnosti telefonskog razgovora. Ideja ometanja za "Evu" je bila da "Alisa" ukoliko prima signal stavlja određeni šum. Kada primi signal sa šumom jedino "Alisa" zna da ukloni šum, jer ga je ona postavila. Osnovni princip koji je prepoznat u ovoj shemi je da:

"Primalac učestvuje u postupku šifrovanja."

Iako je "ideja" bila jasna, do matematičke implementacije se čekalo nekoliko godina. Nakon tri godine od Elisovog otkrića, u GCHQ se zapošljava matematičar Kliford Koks (*Clifford Cocks*), novopečeni diplomac, koji je tokom studija specijalizirao teoriju brojeva. Prema Koksu, problem je rješio za nepunih pola sata polazeći od formulacije problema kao jednosmjerne funkcije i ubacujući u igru proste brojeve i faktorizaciju.[2] Rješenje problema, Koks je podjelio sa

[1] za vrijeme drugog svijetskog rata Blečli Park je bio dom Alana Tjuringa, i mjesto gdje je razbijena Njemačka šifarska mašina ENIGMA

[2] Trenutno "sva" slava za implementaciju pripada takođe amerikancima Rivestu, Šamiru i Adlmanu - RSA

kolegom i starim prijateljem Malkolmom Vilijamsonom (*Malcolm Williamson*). Vilijamson nije vjerovao da je nešto kao što je kriptografija javnim ključem uopšte moguće, pa je pokušao da dokaže da je Koks negdje pogriješio. Međutim nije uspjeo u tome, ali uspjeo je da pronađe shemu razmijene ključeva, istu onu koju je Helman našao (otprilike u isto vrijeme kada i Helman). Tako osnove kriptografije javnim ključem su pronađene nešto ranije nego je objavljen rad Helmana i Difija, ali zbog "zabrane" publikovanja ova otkrića ostaju u tajnosti.

2.2 Kompleksnost

Već smo pomenuli pojam *prividno jednosmerne funkcije*, odnosno transformacije. Naime, postupak kriptovanja E_e u shemi kriptovanja sa javnim ključem je upravo takva transformacija. Pod jednosmjernom (*one way*) funkcijom podrazumjevamo funkciju $f : X \to Y$ koja ima osobinu da je $f(x)$ *lako izračunati* za sve vrijednosti $x \in X$ i za *skoro sve* vrijednosti $y \in Im(f)$ nije *lako izračunati* $x \in X$, takvo da je $y = f(x)$. Ovo predstavlja dosta "intutivnu" definiciju jednosmjerne funkcije, sa obzirom na pojmove *lako izračunati* i *skoro sve*. Ukratko *skoro sve* ostavlja mogućnost da je za neke $y \in Y$ ipak moguće računati x ,tako da je $y = f(x)$, recimo da se za neke $x \in X$ uvijek može napraviti tabela $(x, f(x))$, i jednostavnim uvidom (pretragom) po tabeli naći odgovarajući x. Alternativni način da se iskaže ova osobina je da za slučajan izbor $y \in Im(f)$ x nije moguće *lako izračunati.* Takođe pojam *lako izračunati* je pojam koji u samu kriptografiju uvodi analizu algortama, tačnije teoriju kompleksnosti računanja. Možemo reći, da je u jednom velikom dijelu, moderna kriptografija upravo teorija kompleksnosti računanja.

U ovom kontekstu pojam *"računanja"* uzimamo u malo širem smislu, tačnije pod računanjem smatramo *Algoritam*, odnosno *dobro definisanu proceduru* koja za date ulazne podatke daje izlani rezultat (postoji mogućnost da se procedura nikada ne završi). Postoji mnogo sistema koji "formalizuju" pojam algoritma od kojih su najpoznatiji: Sistem rekurzivnih funkcija (K. Gödel), Tjuringova mašina (A. Turnig), λ-račun (A. Church). Postoji čitav dio matematike koji se bavi pojmom efektivne izračunljivosti (odnosno samim pojmom algoritma), ali iz aspekta same kriptografije pojedini pojmovi iz ove teorije su posebno interesantni. Takav je pojam kompleksnost algoritama. Naime za dati problem, iako znamo efektivan način kako ga rješiti (možda to znamo i da uradimo na više načina) postavlja se pitanje karakterizacije tog rješenja. Jedna od osnovnih karakteristika datog algoritma je "vrijeme" koje je neophodno za završavanje algoritma u zavisnosti od veličine ulaznih podataka, preciznije "vrijeme" je samo intuitivan pojam broja koraka koji moraju da se "izvrše" da bi došli do rješenja problema. U većini slučajeva teško je odrediti tačan broj koraka u pojedinom algoritmu, ali je moguće dati "aproksimaciju" broja koraka. Najčešće korišćena aproksimacija je *"asimptotsko gornje ograničenje"* u notaciji:

$$f(x) = O(g(x))$$

Prethodna relacija važi ukoliko postoji pozitivna konstanta c i pozitivan cijeli

broj n_0 takav da je $0 \leq f(n) \leq cg(n)$ za sve $n \geq n_0$. Intuitivno možemo reći da f ne raste asimptotski brže od g (osim za konstantu). Dalje, kažemo da se neki algoritam izvršava u *polinomijalnom vremenu* ako je jednak $O(n^k)$ gdje je n veličina ulaznog podatka i k je konstanta.[3] Ako je $n = O(\log N)$ i $p(n)$ je polinom, onda se za algoritam čije je vrijeme izvršavanja:

$$c^{p(n)}(c > 1)$$

kaže da ima eksponencijalnu vremensku složenost u odnosu na dužinu N. Intuitivno možemo shvatiti polinomijalne algoritme "dobrim" ili "efikasnim" dok eksponencijalne algoritme shvatamo "neefikasnim". Postavka kriptografskog sistema sa javnim ključem se u stvari može gledati i kao "traganje" za jednosmjernom transformacijom. U daljem dijelu prikazaćemo jednu funkciju za koju se "vjeruje" da je jednosmjerna, a koja je bila osnova kriptografskih razmatranja u [7]. Pokazaćemo i neke "napade" - algoritme koji za određene vrijednosti parametara kriptosistema mogu da budu efikasni.

Otvoreno je pitanje da li "prave" jednosmjerne funkcije zaista postoje.

2.3 Difi Helman problem i diskretni logaritam

Funkcije koje su bile opisane u radu [7] su stepenovanje i njemu inverzna operacija (logaritam) u multiplikativnoj grupi $\mathbb{Z}_p^*$ prstena $\mathbb{Z}_p$, gdje je p *veliki* prost broj. Kako je grupa $\mathbb{Z}_p^*$ ciklična, postoji $g \in \mathbb{Z}_p^*$ čiji je red jednak $p-1$ i koji generiše grupu $\mathbb{Z}_p^*$. Samim tim za proizvoljno $b \in \mathbb{Z}_p^*$ postoji $k \in \mathbb{Z}$ takvo da je:

$$b = g^k \pmod{p} \tag{2.1}$$

Računanje k na osnovu b i g je problem računanja diskretnog logaritma (**DLP**) u grupi $\mathbb{Z}_p^*$.

$$k = \log_g b \pmod{p} \tag{2.2}$$

Pri dobrom[4] izboru prostog broja p trenutno nije poznat algoritam koji u *podeksponencijalnom* vremenu računa diskretni logaritam u grupi $\mathbb{Z}_p^*$. Trenutno najbolji algoritam za rešavanje problema diskretnog logaritma zasniva se na "Number Field Sieve" algoritmu za faktorizaciju i pod određenim uslovima ima asimptotsko ponašanje:

$$O(e^{c(\ln p)^{1/3}(\ln(\ln p))^{2/3}})$$

Iako vjerujemo (postoji i matematički aparat kojim dokazujemo neke osnovne pretpostavke) da je problem diskretnog logaritma *težak* nije dokazano da je to tačno.

Pogledajmo sada shemu koja je izložena u radu [7].

[3] Definicija 2.59 u [1]: Algoritme koje ne možemo ograničiti na ovaj način kažemo da se izvršavaju u *eksponencijalnom vremenu.*

[4] kasnije će biti opisani neki algoritmi u kojima ključnu ulogu igra izbor prostog broja p

2.3.1 Difi Helman usaglašavanje ključa

Neka je p prost broj, $\mathbb{Z}_p^*$ multiplikativna grupa prstena $\mathbb{Z}_p$ i neka je $g \in \mathbb{Z}_p^*$ generator grupe $\mathbb{Z}_p^*$. Alis na slučajan način izabere $0 < a < p-1$, izračuna $r = g^a \bmod p$. Slično, Bob na slučajan način izabere $0 < b < p-1$, izračuna $s = g^b \bmod p$. Alis i Bob *"javno"* razmijene brojeve r i s. Za prethodno izabrano a i dobijeno s Alis računa $K = s^a \bmod p$ pa sa obzirom da je $s = g^b$ imamo da je $K = g^{ab} \bmod p$. Na sličan način i Bob dolazi do K.

primjer 2.3.1. Neka je $p = 23$, $g = 5$ je generator grupe $\mathbb{Z}_{23}^*$ (obzirom da $Ord(5)$ nije ni 2 ni 11). Alis izabere 6 (slučajan način), dok Bob izabere 15.

Alis		Bob	
Javno	Tajno	Javno	Tajno
5, 23	6	5, 23	15
$5^6 \equiv 8 \pmod{23}$	$19^6 \equiv 2 \pmod{23}$	$5^{15} \equiv 19 \pmod{23}$	$8^{15} \equiv 2 \pmod{23}$

Pitamo se šta Eva može da zaključi gledajući brojeve p, g, r i s ? Da bi došla do K morala bi da zna da sračuna $g^{ab} \bmod p$ znajući $s = g^a \bmod p$ i $r = g^b \bmod p$.

Ovaj problem naziva se **DHP** - *(Difi - Helman problem)*. Problem je u uskoj vezi je sa problemom diskretnog logaritma (**DLP** - *Problem diskretnog logaritma*). Tačnije, ako bi Eva uspijela da rješi dvije poslednje jednačine po a i b respektivno, onda bi uspijela da dođe i do g^{ab}. Naime, ako se u ovom slučaju rješi **DLP** rješava se i **DHP**. Znači **DHP** nije *"teži"* od **DLP**. Trenutno jedini način da se rješi **DHP** je da se rješi **DLP**. Još uvijek je otvoreno pitanje da li je **DHP** ekvivalentan **DLP**. Kasnije ćemo upoznati **ElGamal** kripto shemu i pokazaćemo da je **DHP** ekvivalentan sa **ElGamal** problemom koji proizilazi iz ove kripto sheme, kao i da je u ovoj shemi *otkrivanje tajnog ključa* na osnovu javnog ključa u stvari, malo drugačije iskazan **DLP**.

DHP problem možemo i da uopštimo na slučaj bilo koje konačne ciklične grupe $(G, *)$ pri čemu se u tom slučaju problem naziva i *generalisani Difi Helman problem* - **GDHP**. Tačnije za bilo koju grupu G, možemo uočiti podgrupu H generisanu nekim elementom grupe G. Ako je u grupi H problem diskretnog logaritma težak, ova grupa može da posluži za postavku kriptografskog sistema. Primjetimo da grupa G ne mora biti abelova. Takođe u nekim grupama problem ima *"povoljnije"* algoritme za postavku odgovorajućih kripto shema (manji ključevi, kompleksniji algoritmi, nemogućnost nekih napada koji su mogući u $\mathbb{Z}_p^*$).

Modifikacija koja je neophodna u Difi Helman shemi za usaglašavanje ključeva je da podaci koji Alis i Bob primaju kao javne ključeve, moraju biti autentični. Tačnije sama shema nije otporna na tip napada *"man in the middle"*. Ako bi recimo Eva, bila u mogućnosti da "zavara" i Alisu i Boba tako da će svakome od njih poslati svoj javni ključ (kao Bobov, odnosno Alisin). Kripto sistem mora obezbijediti i autentičnost ključeva koji se razmenjuju. Jedno od rješenja ovog problema daje protokol *"Station to Station"* u kojem se parametri

za usaglašavanje ključa potpisuju sertifikatima. Sama shema usaglašavanja ključeva, pri čemu se autentičnost ključeva "dokazuje" sertifikatima, je u osnovi *SSL - Secure Socket Layer* protokola. Naime tokom inicijalizacije sesije, tokom tkz. *handshake*-a, dolazi upravo do razmijene javnih ključeva i računanja ključa koji se kasnije koristi u nekom simetričnom algoritmu (DES/AES/RC4).

2.3.2 Računanje diskrenog logaritma

Neka je p prost broj i neka je α "primitivni" element (generator grupe $\mathbb{Z}_p^*$). Za dalje razmatranje uzećemo da su p i α fiksni. Problem diskretnog logaritma možemo iskazati na sledeći način: za dato $\beta \in \mathbb{Z}_p^*$ naći jedinstven eksponent a, $1 \leq a \leq p-2$, takav da je:

$$\alpha^a \equiv \beta \pmod{p} \tag{2.3}$$

Najjednostavniji način računanja diskretnog logaritma je linearna pretraga (*"metod grube sile"*). Tačnije za svako a, $1 \leq a \leq p-2$ računamo:

$$\alpha^a \equiv \beta \pmod{p}$$

Ovaj postupak možemo okarakterisati sa $O(p)$. Takođe, moguće je izvršiti i "predračun" računajući vrijednosti α^a i sortirati uređene parove $(a, \alpha^a \bmod p)$ prema drugoj koordinati. Ovaj postupak možemo okarakterisati sa $O(1)$ što se tiče vremena i sa $O(p)$ što se tiče memorije, takođe i "predračun" je $O(p)$.

Šanksov algoritam - (Giant Step - Baby step)

Danijel Šanks (*Daniel Shanks*[5]) dao je sledeći algoritam za računanje diskretnog logaritma.

Neka je data notacija kao u uvodu. Označimo $m = \lceil \sqrt{p-1} \rceil$. Algoritam teče na sledeći način:

1. Izračunavamo $\alpha^{mj} \bmod p$, $0 \leq j \leq m-1$,
2. Sortiramo m uređenih parova $(j, \alpha^{mj} \bmod p)$ prema drugoj koordinati i tako dobijemo listu L_1,
3. Izračunavamo $\beta\alpha^{-i} \bmod p$, $0 \leq i \leq m-1$,
4. Sortiramo m uređenih parova $(i, \beta\alpha^{-i} \bmod p)$ prema drugoj koordinati i tako dobijemo listu L_2,
5. Pronalazimo par $(j, y) \in L_1$ i par $(i, y) \in L_2$ koji imaju jednaku drugu koordinatu,
6. Definišemo $\log_\alpha \beta = mj + i \bmod p$.

[5] Američki matematičar čiji se rad odnosio uglavnom na numeričku analizu i teoriju brojeva

Pohlig - Helman

Primjer algoritma za računanje diskretnog logaritma u multiplikativnoj grupi čiji je red *smooth* celi broj. Algoritam se u osnovi zasniva na *kineskoj teoremi o ostacima.*

Originalno pripada Rolandu Silveru (*Roland Silver*), ali je prvi put publikovan od strane Stivena Pohliga (*Stephen Pohlig*) i Martina Helmana (*Martin Hellman*) nezavisno od Silvera.

Za pozitivan celi broj kažemo da je $B-gladak$ ($B-smooth$) ako ni jedan od njegovih prostih delilaca nije veći od B. Na primjer broj 1620 ima faktorizaciju $2^2 \cdot 3^4 \cdot 5$ pa za njega kažemo da je $5-gladak$.

Osnova algoritma je pronalaženje logaritma za manje module, tačnije ako je $\alpha^a \equiv \beta \pmod{p}$ odnosno $a \equiv \log_\alpha \beta \pmod{p}$ pokušamo naći rješenje $a \equiv \log_\alpha \beta \pmod{p-1}$. Sa obzirom da je

$$p-1 = \prod_{i=1}^{k} p_i^{c_i}$$

gdje su p_i različiti prosti brojevi, ako rješimo jednačine:

$$x_i \equiv a \pmod{p_i^{c_i}} \text{ za sve } 1 \leq i \leq k$$

tada na osnovu *kineske teoremo o ostacima* imamo rješenje diskretnog logaritma.

Neka je q prost broj i neka je:

$$p-1 \equiv 0 \pmod{q^c}$$

i

$$p-1 \not\equiv 0 \pmod{q^{c+1}}$$

pokazaćemo kako da izračunamo:

$$x \equiv a \pmod{q^c}$$

gdje je $0 \leq x \leq q^c - 1$. Iskažimo x u osnovi q:

$$x = \sum_{i=0}^{c-1} a_i q^i$$

pri čemu je $0 \leq a_i \leq q-1$. Takođe možemo da iskažemo a kao:

$$a = x + q^c \cdot s$$

za neki cijeli broj s. Prvi korak u algoritmu je da odredimo a_0. Da bi odredili a_0 dokažimo jednakost:

$$\beta^{(p-1)/q} \equiv \alpha^{(p-1)a_0/q} \pmod{p} \tag{2.11}$$

i	0
β_i	18
a_i	4

pa imamo da je $a = a_0 \equiv 4 \pmod 7$. Koristeći kinesku teoremu o ostacima imamo da je $a \equiv 11 \pmod{28}$ odnosno $\log_2 18 \equiv 11 \pmod{29}$.

Napomenimo da postoji još algoritama za računanje problema diskretnog logaritma, od kojih su najpoznatiji Polardov Ro metod (*John Pollard, ρ-Method*) kao i metod računanja indeksa. Takođe, računanja određenih bitova diskretnog logaritma je predmet izučavanja. Naime, pokazuje se da je najmanje značajan bit lako izračunati, kao i da bi računanje ostalih bitova moglo poslužiti za računanje diskretnog logaritma.

Dalje ćemo prikazati jednu asimetričnu kripto shemu koja se zasniva na problemu diskretnog logaritma.

2.4 ElGamal sistem

ElGamal (*Taher ElGamal, 1955*)[6] shema je asimtrična shema zasnovana na problemu diskretnog logaritma u cikličnoj multiplikativnoj grupi $\mathbb{Z}_p^*$ prstena $\mathbb{Z}_p$. Isti postupak se može lako generalisati i na druge konačne ciklične grupe, pri čemu se u tom slučaju naziva i *generalisani ElGamal* postupak. Godine 1985 ElGamal objavljuje rad *"A Public key Cryptosystem and A Signature Scheme based on discrete Logarithms"* u kojem je opisana shema asimetričnog kriptovanja, i shema potpisivanja[7]. Napomenimo samo da ElGamal sistem nije patentiran, ali prije nego se krene u neku implementaciju treba imati u vidu da firma PKP (Public Key Partners) smatra da je ovaj algoritam pokriven sa Difi - Helman patentom. Isticanjem važenja patenta (29. april 1997.) ElGamal je postao prvi algoritam za asimetrično kriptovanje i potpisivanje nepokriven patentima u SAD. Ovaj postupak se koristi u "GNU Privacy Guard" kao i u novijim verzijma "PGP"-ja, popularnih programa za kriptovanje. Dajemo opis osnovnih koraka u algoritmu. Postupak ElGamal se sastoji od tri dijela:

- generisanje ključeva,
- algoritma kriptovanja i
- algoritma dekriptovanja

Generisanje ključa u ElGamal sistemu

Alis, kada želi da primi poruku generiše "javni" ključ na sledeći način:

1. na slučajan način generiše veliki prost broj p i generator α ciklične multiplikativne grupe $\mathbb{Z}_p^*$ prstena $\mathbb{Z}_p$,
2. na slučajan način bira broj a, $1 \leq a \leq p-1$, sračunava $\alpha^a \pmod p$,
3. Alisin javni ključ je uređena trojka $(p, \alpha, \alpha^a \pmod p)$, a privatni ključ je broj a.

Algoritam kriptovanja u ElGamal sistemu

Bob kada želi da požalje poruku m Alisi, poruku kriptuje na sledeći način:

1. Bob dobija javni ključ Alise $(p, \alpha, \alpha^a \pmod p)$,
2. vrši "aritmetizaciju" otvorene poruke m u skupu $\mathbb{Z}_p = \{0, 1, 2, ..., p-1\}$,
3. na slučajan način bira broj k, gdje je $1 \leq k \leq p-2$,
4. izračunava $\gamma = \alpha^k \pmod p$ i $\delta = m(\alpha^a)^k \pmod p$,
5. šalje kriptogram $c = (\gamma, \delta)$ Alisi.

[6]Egipatski kriptograf, često se koristi i zapis Elgamal ili El Gamal, ali je uobičajen zapis ElGamal. Inače, vrijedi napomenuti da je ElGamal bio jedan od vodećih inžinjera protokola *SSL - Netscape*, kao da je radio kao glavni inžinjer u *RSA Security Inc.*

[7]Shema potpisivanja je postala osnova algoritma **DSA** (*Digital Signature Algorithm*) i usvojena je kao standard **DSS** (*Digital Signature Standard*)

Algoritam dekriptovanja u ElGamal sistemu

Alisa, da bi dobila polaznu poruku m (tačnije njen aritmetički kod) iz kriptograma c, postupa na sledeći način:

1. koristeći svoj privatni ključ a izračunava $\gamma^{p-1-a} \pmod{p}$, primjetimo da je:
$$\gamma^{p-1} \cdot \gamma^{-a} \equiv 1 \cdot \gamma^{-a} \equiv \alpha^{-ak} \pmod{p}$$
2. dekriptuje poruku m računajući $(\gamma^{-a}) \cdot \delta \pmod{p}$

Korektnost postupka slijedi iz sledećeg računa u $\mathbb{Z}_p^*$:

$$\gamma^{-a} \cdot \delta \equiv \alpha^{-ak} \cdot m \cdot \alpha^{ak} \equiv m \pmod{p}$$

primjer 2.4.1. Pogledajmo sada jedan jednostavan (male vrijednosti parametara) primjer ove sheme. Naime predpostavimo da Alisa želi da pošalje poruku Bobu.

1. Bob izabere $p = 29$, $\alpha = 2$ i $a = 5$, pa obzirom da je $2^5 = 3 \pmod{29}$ Bobov javni ključ je $(29, 2, 3)$, a privatni ključ je 5.
2. Kad bi Alisa htjela da enkriptuje poruku recimo $m = 6$, na slučajan način izabere k recimo $k = 14$ i računa:
$$\gamma \equiv \alpha^k \equiv 2^{14} \equiv 28 \pmod{29}$$
kao i:
$$\delta \equiv m \cdot (\alpha^a)^k \equiv 6 \cdot 3^{14} \equiv 23 \pmod{29}$$
Alisa šalje poruku $(28, 23)$ Bobu.
3. Da bi dekriptovao Bob računa:
$$\gamma^{p-1-a} \equiv 28^{29-1-5} \equiv 28^{23} \equiv 28 \pmod{29}$$
tada je polazna poruka:
$$m \equiv \gamma^{p-1-a} \cdot \delta \equiv 28 \cdot 23 \equiv 6 \pmod{29}$$

Odmah možemo primjetiti da je "kriptogram" dva puta duži od polaznog teksta, kao i da bitan korak u fazi kriptovanja predstavlja slučajan izbor broja k. Naime, slučajan izbor broja k utiče na to da (ako recimo na nivou bloka biramo k) tada isti znaci alfabeta mogu biti različiti kriptovani. Sam proces kriptovanja zahtijeva dvije operacije stepenovanja $\alpha^k \pmod{p}$ i $(\alpha^a)^k \pmod{p}$. Ova dva stepenovanja mogu biti "ubrzana" izborom k koji ima recimo malu *Hamingovu* težinu (samim tim i mali broj operacija u algoritmu stepenovanja kvadriranjem (*square and double*, *repeated squares*). Malo je "teže" opisati izbor, tačnije generisanje, ključa. Biranje ključa uključuje izbor prostog broja dužine k-bita, generator α grupe $\mathbb{Z}_p^*$ i broja a za koji važi $1 \leq a \leq p-1$. Naime, ako znamo faktorizaciju $p-1$ onda imamo efikasan algoritam za utvrđivanje da li je broj $\alpha \in \mathbb{Z}_p^*$ generator ili ne.

Sigurnost ElGamal sistema

Ako posmatramo problem da za dati javni ključ u ElGamal sistemu

$$(p, \alpha, \alpha^a \pmod p)$$

odredimo privatni ključ, problem je drugačije iskazan problem diskretnog logaritma. Ali možemo da uočimo "malo" drugačiji problem. Naime, za dati $(p, \alpha, \alpha^a \pmod p)$ javni ključ i dati kriptogram (γ, δ) odrediti polaznu poruku m. Dokazujemo da je poslednji problem ekvivalentan **DHP** problemu.

teorema 2.4.1. *Problemi* **ElGamal** *i* **DHP** *su tjuring ekvivalenti.*

Dokaz. Pretpostavimo da imamo algoritam koji rešava **DHP**. Za dati javni ključ u ElGamal sistemu $(p, \alpha, \alpha^a \pmod p)$ i kriptogram (γ, δ), polazeći od $\alpha^a \pmod p$ i $\gamma = \alpha^k \pmod p$, koristeći algoritam za **DHP** možemo da sračunamo $\alpha^{ak} \pmod p$. Dalje, koristeći *prošireni Euklidov* algoritam možemo da sračunamo $\alpha^{-ak} \pmod p$, pa samim tim i dobijamo i polaznu poruku $m = \alpha^{-ak} \cdot \delta$.

Obrnuto, pretpostavimo da imamo algoritam koji rešava **ElGamal** problem, i neka su nam dati $(p, \alpha, \alpha^a, \alpha^b)$. Koristeći algoritam, za parametre (p, α, α^a) i kriptogram $(\alpha^k, 1)$, možemo da dobijemo m pri čemu je $1 = m \cdot (\alpha^a)^k \pmod p$, znači m je inverz od $(\alpha^a)^k$. Dalje, koristeći *prošireni Euklidov* algoritam dolazimo do $(\alpha^a)^k$. □

Jako je bitno da koristimo različite brojeve k za kriptovanje različitih poruka m. Ako bi recimo sa istim k, kriptovali različite poruke m_1 i m_2, tada bi imali sledeću relaciju na paru kriptograma (γ_1, δ_1), (γ_2, δ_2):

$$\frac{\delta_1}{\delta_2} = \frac{m_1}{m_2}$$

pa polazni tekst m_2 lako možemo sračunati ako znamo m_1.

Moguće je da Alisa i Bob razmijene samo vrijednosti α^a i α^k pri čemu su vrijednosti p i α "opšte" prihvaćene. Metod opšte prihvaćenih vrijednosti za *ElGamal* kriptosistem omogućava da se algoritmi stepenovanja o ovim slučajevima mogu značajno ubrzati[8].

Preporuka za korišćenje u praksi su najmanje 768 bita za modul p, sa obzirom da 512 bita pruža samo osnovnu sigurnost. Za neke dugoročnije sisteme preporučuje se modul od 1024 bita ili veći.

Generalisani ElGamal sistem

Diskretni logaritam možemo posmatrati u bilo kojoj grupi $(G, *)$. Tačnije, za grupu $(G, *)$ uočimo $g \in G$, tada g generiše podgrupu H grupe G. U grupi H tada možemo posmatrati problem diskretnog logaritma. Takođe možemo je koristiti i kao postavku za ElGamal kriptografsku shemu. U osnovi svaki **DLP**

[8]Za poznate vrijednosti p i α moguće je izvršiti predračun nekih stepena osnove, i taj predračun koristiti u nekim algoritmima stepenovanja pri kriptovanju, odnosno dekriptovanju

problem možemo posmatrati u konačnoj cikličnoj grupi. Međutim problem nije svuda podjednako "težak". Tačnije, reprezentacija grupe utiče na to koliko je problem diskretnog logaritma težak. Uzmimo za primer aditivnu grupu $\mathbb{Z}_n$ i $\alpha \in \mathbb{Z}_n$ predpostavimo da je $(\alpha, n) = 1$ tada je α generator grupe $\mathbb{Z}_n$. U ovakvoj postavci problem diskretnog logaritma je rješiti po a jednačinu:

$$a \cdot \alpha \equiv \beta \pmod{n}$$

Kako je $(\alpha, n) = 1$ tada α ima multiplikativni inverz u $\mathbb{Z}_n$ pa možemo da izračunamo (koristeći prošireni Euklidov algoritam):

$$a = \log_\alpha \beta = \alpha^{-1}\beta \pmod{n}$$

Ranije smo razmatrali problem diskretnog logaritma u multiplikativnoj grupi $\mathbb{Z}_p^*$ gdje je p prost broj. $\mathbb{Z}_p^*$ je ciklična grupa reda $p-1$, pa je izomorfna aditivnoj grupi $\mathbb{Z}_{p-1}$. Ovaj izomorfizam nam "sugeriše" da bi problem diskretnog logaritma u $\mathbb{Z}_p^*$ mogli da svedemo na problem diskretnog logaritma u $\mathbb{Z}_{p-1}$. Naime postoji izomorfizam:

$$\phi : \mathbb{Z}_p^* \to \mathbb{Z}_{p-1}$$

to znači da je:

$$\phi(x \cdot y \pmod{p}) = (\phi(x) + \phi(y)) \pmod{p-1}$$

dalje:

$$\phi(\alpha^a \pmod{p}) = (a \cdot \phi(\alpha)) \pmod{p-1}$$

tako da važi:

$$\beta \equiv \alpha^a \pmod{p} \Leftrightarrow a \cdot \phi(\alpha) \equiv \phi(\beta) \pmod{p-1}$$

rješavajući po a gornju jednakost imamo:

$$\log_\alpha \beta = \phi(\beta)(\phi(\alpha))^{-1} \pmod{p-1}$$

Naime, iako znamo da postoji izomorfizam, i da nas taj izomorfizam vodi do "lakog" rješenja problema diskretnog logaritma u $\mathbb{Z}_p^*$ ostaje da "izračunamo" izomorfizam. U ovom slučaju problem računanja izomorfizma ($\phi : \mathbb{Z}_p^* \to \mathbb{Z}_{p-1}$) je "težak". Prethodno razmatranje nam kaže da problem diskretnog logaritma može biti i jednostavan za rješavanje zavisno od reprezentacije grupe u kojoj je postavljen. Trenutno, u kriptografiji predmet izučavanja su sledeće grupe:

- multiplikativna grupa polja ($GF(p^n)$) i
- grupa eliptičke krive nad konačnim poljem.

Glava 3

Eliptičke krive

Ranije smo pomenuli *generalisani ElGamal* postupak u kojem ključnu ulogu ima izbor grupe u kojoj izvodimo operacije kriptovanja, odnosno dekriptovanja. Grupa mora da obezbijedi nekoliko aspekata bitnih za kriptografski sistem. Prije svega, operacija u grupi mora biti takva da je problem računanja diskretnog logaritma jako težak, dok je stepenovanje relatino jednostavna operacija (sa stanovišta kompleksnosti računanja). Kao što smo vidjeli multiplikativna grupa $\mathbb{Z}_p^*$ prstena $\mathbb{Z}_p$, pri *"pažljivom"* izboru prostog broja p zadovoljava pomenute uslove. Nezavisno jedan od drugog, 1985 godine, Američki matematičari, Nil Koblic (*Neal Koblitz*) i Viktor Miler(*Victor Miller*) predložili su korišćenje grupa generisanih *eliptičkim krivima* nad konačnim poljima. Vjeruje se da je problem diskretnog logaritma u ovim grupama znatno teži, nego u grupi $\mathbb{Z}_p^*$.

Iako su *eliptičke krive* matematički objekti koji imaju primjene u mnogim oblastima matematike, zadržaćemo se samo na njihovim osobinama koje se tiču kriptografije. Tačnije pored opšte definicije eliptičkih krivih navećemo primjer eliptičkih krivih definisanih nad konačnim poljem $\mathbb{K}$ koje je karakteristike različite i od 2 i od 3.

3.1 Osnovni pojmovi

Eliptička kriva E nad konačnim poljem $\mathbb{K}$ definisana je sledećom jednakošću

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{3.1}$$

pri čemu $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ i $\Delta \neq 0$, Diskriminanta Δ je definisana:

$$\begin{aligned}
\Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\
d_2 &= a_1^2 + 4a_2 \\
d_4 &= 2a_4 + a_1a_3 \\
d_6 &= a_3^2 + 4a_6 \\
d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2
\end{aligned}$$

Jednačinu (3.1) nazivamo i Vajerštrasovom (*Weierstarass*)-ovom jednačinom.

Razlikujemo nekoliko *"pojednostavljenih"* oblika jednačine (3.1) u zavisnosti od

- karakteristike polja $\mathbb{K}$,
- vrijednosti koeficijenata $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$

Razmatranje osobina krivih, posebno njihove reprezentacije i osobina koje su bitne za implemetaciju u nekom kriptosistemu biće zasnovano na krivama nad poljem $\mathbb{K}$ čija je karakteristika različita i od 2 i od 3. U tom slučaju jednačina (3.1) se transaformacijom

$$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1 x}{216} - \frac{a_1^3 + 4a_1 a_2 - 12a_3}{24} \right) \tag{3.2}$$

svodi na oblik

$$E : y^2 = x^3 + ax + b \tag{3.3}$$

pri čemu $a, b \in \mathbb{K}$ i $\Delta = -16(4a^3 + 27b^2)$.

Krivoj E dodajemo i tačku u "beskonačnosti" $\mathcal{O} = \{\infty\}$. Ova tačka u stvari predstavlja jedinu tačku na *"beskonačnoj"* pravoj koja zadovoljava jednačinu (3.1) predstavljenu u projektivnoj ravni. Naime, u nekim situacijama[1] jednačinu (3.1) predstavljamo i u projektivnoj ravni, u tom slučaju (standardna projektivna ravan) projektivna tačka $(X : Y : Z)$, $Z \neq 0$ odgovara afinoj tački $(\frac{X}{Z}, \frac{Y}{Z})$, a projektivna jednačina eliptičke krive (3.3) je:

$$Y^2 Z = X^3 + aXZ^2 + bZ^3 \tag{3.4}$$

Tačka u beskonačnosti ∞ odgovara tački $(0 : 1 : 0)$.

Za implementaciju nekog kriptosistema posebno su bitne osobine eliptičkih krivih koje se odnose na njihove algebarske osobine. Naime, definišući operaciju "sabiranja" nad tačkama krive (3.3) dobijamo grupu koja nam može poslužiti kao osnov za primjenu *ElGamal* sistema. Kao što je već ranije napomenuto, implementaciju kriptosistema ćemo zasnovati nad poljem $\mathbb{Z}_p$ gdje je p neparan prost broj.

3.2 Grupa $E(\mathbb{Z}_p)$

Eliptička kriva E se može posmatrati i kao grupa, ako definičemo određene operacije na skupu tačaka (x, y) koje zadovoljavaju (3.3). Naime, možemo definisati strukturu grupe nad E ako definišemo sledeću operaciju *sabiranja*. Neka su $P = (x_1, y_1)$ i $Q = (x_2, y_2)$ tačke na krivoj E. Ako je $x_1 = x_2$ i $y_2 = -y_1$

[1] algoritmi za računanje inverza u nekim poljima mogu biti dosta "skupe" operacije, u tom slučaju pogodno je tačke predstaviti u projektivnoj ravni.

tada definišemo da je $P + Q = \mathcal{O}$ inače, definišemo da je $P + Q = (x_3, y_3)$ pri čemu je:

$$x_3 = \lambda^2 - x_1 - x_2 \tag{3.5}$$

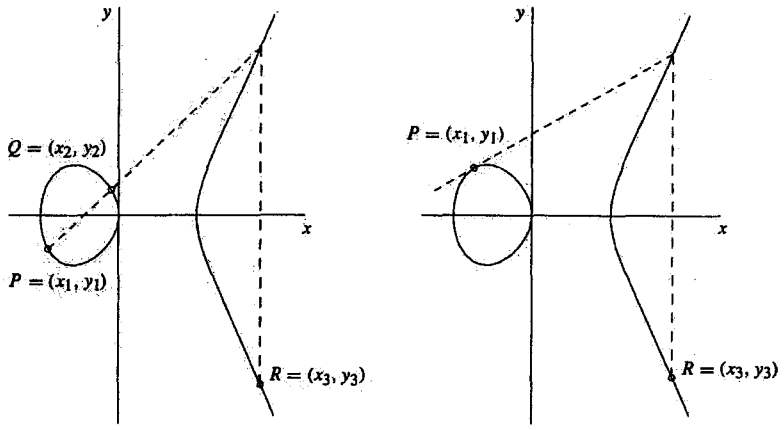
$$y_3 = \lambda(x_1 - x_3) - y_1 \tag{3.6}$$

pri čemu je:

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{ako je } P \neq Q \\ \frac{3x_1^2 + a}{2y_1}, & \text{ako je } P = Q \end{cases} \tag{3.7}$$

Definišemo još da je:

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



Ovo je u stvari metod *"sečice i tangente"* pri čemu je neutral tačka u beskonačnosti. Tačnije ako su P i Q tačke na krivoj E, sa PQ označimo tačku preseka prave koja prolazi kroz P i Q sa krivom E. Tačku $P+Q$ dobijamo u preseku krive E i vertikalne prave koja sadrži tačku PQ. U konačnim poljima ova intuitivna definicija "malo" gubi smisao, ali analogija ipak ostaje sa definicojom uvođenja grupe nad eliptičkim krivama nad poljem $\mathbb{R}$ realnih brojeva.

Većinu osobina operacije sabiranja je relativno jednostavno dokazati osim (asocijativnosti, malo teže), takođe sa obzirom da je operacija komutativna sve notacije označavamo aditivno.

Sledeći primjer pokazuje neka osnovna *"računanja"* u grupi $E(\mathbb{Z}_p)$. Posebno, isti *"račun"* ćemo koristiti u kasnijoj implementaciji.

primjer 3.2.1. Neka je E eliptička kriva definisana sa $E : y^2 = x^3 + x + 6$ nad poljem $\mathbb{Z}_{11}$. Tačke na krivoj E možemo prikazati tabelarno ($QR(11)$ je skup kvadratnih ostataka):

x	$x^3+x+6 \pmod{11}$	$\in QR(11)$	y
0	6	ne	-
1	8	ne	-
2	5	da	4,7
3	3	da	5,6
4	8	ne	-
5	4	da	2,9
6	8	ne	-
7	4	da	2,9
8	9	da	3,8
9	7	ne	-
10	4	da	2,9

Prema ovome kriva E ima 13 tačaka (uključujući i tačku $\mathcal{O}$). Samim tim, grupa $E(\mathbb{Z}_p)$ je ciklična i bilo koji element (tačka na krivoj) osim $\mathcal{O}$ je generator grupe E. Takođe, primjetimo da imamo efektivan način računanja, tačaka na krivoj E za $p = 11$. Tačnije, za $0 \leq x \leq 10$, računamo $f(x) = x^3+x+6 \bmod 11$ i pitamo se da li je $f(x)$ u tom slučaju kvadratni ostatak po modulu 11. Ukoliko $f(x)$ jeste kvadratni ostatak, tačnije važi:

$$f(x)^{\frac{p-1}{2}} \equiv 1 \pmod{p}, \tag{3.8}$$

tada jednačina:

$$y^2 \equiv f(x) \pmod{p}, \tag{3.9}$$

ima tačno dva rješenja y_0 i y_1 pri čemu je $y_1 \equiv -y_0 \pmod{p}$. Dalje, ukoliko je prost broj p posebnog oblika, recimo u slučaju da je $p = 3 \bmod 4$ onda su:

$$y \equiv \pm f(x)^{\frac{p+1}{4}} \pmod{p}$$

rješenja jednačine (3.9). U primjeru je $p = 11 = 3 \equiv \pmod 4$ pa za one elemente koji jesu kvadratni ostaci, rješenja jednačine (3.9) su data sa:

$$y \equiv \pm f(x)^3 \pmod{11}$$

Kriterijum (3.8) je ustvari Ojlerov (*Leonard Paul Euler*) kriterijum utvrđivanja da li je broj kvadratni ostatak. U vezi toga za brojeve $a \in \mathbb{N}$ i p neparan prost broj definišemo simbol[2]:

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{ako je a kvadratni ostatak mod p} \\ -1, & \text{ako je a nije kvadratni ostatak mod p} \end{cases}$$

Ojlerov kriterijum glasi:

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$$

[2]Simbol se naziva Ležandrovim simbolom - *Adrien-Marie Legendre*

Primjetimo dalje da su rješenja jednačine (3.9) uvijek jedno parno, a drugo neparno. Na osnovu ovoga, moguće je izvršiti "kompresiju" tačke, naime dovoljno je da imamo x koordinatu i jedan bit za parnost[3].

Jedno od osnovni pitanje u vezi sa grupom $E(\mathbb{Z}_p)$ je njena kardinalnost. Označimo sa $\#E$ broj elemenata grupe $E(\mathbb{Z}_p)$. U nekim jednostavnijim primjerima prostim nabrajanjem je moguće doći do broja $\#E$. Međutim za velike brojeve p to nije izvodljivo.

Navedimo Haseovu (*Helmut Hasse*) teoremu koja nam daje ocjenu kardinalnosti grupe $E(\mathbb{Z}_p)$.

teorema 3.2.1. *Neka je E eliptička kriva definisana nad $\mathbb{F}_q$ tada važi:*

$$q+1-2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q+1+2\sqrt{q}.$$

Pored ocjene kardinalnosti grupe $E(\mathbb{F}_q)$ napomenimo da postoji i "efikasan" algoritam (*René Schoof*) za računanje broja elemenata grupe $E(\mathbb{F}_q)$. U ovom slučaju vrijeme izvaršavanja je polinomijalno u odnosu $\log q$. Šofov algoritam je $O((\log q)^8)$ i možemo ga smatrati praktičnim za proste brojeve q reda veličine nekoliko stotina cifara.

Prije nego prikažemo primjere kriptosistema zasnovanim na eliptičkim krivama pogledajmo još jedan primjer:

primjer 3.2.2. Neka je E eliptička kriva definisana sa $E: y^2 = x^3+1$ nad poljem $\mathbb{Z}_7$. Tačke na krivoj E možemo prikazati tabelarno:

x	$x^3+1 \pmod 7$	$\in QR(7)$	y
0	1	ne	1,6
1	2	ne	3,4
2	2	da	3,4
3	0	da	0
4	2	ne	3,4
5	0	da	0
6	0	ne	0

prema ovome $E(\mathbb{Z}_7)$ ima 12 tačaka. Uočimo tačke $R=(5,0)$ i $Q=(1,3)$.

	$\cdot\ R=(5,0)$	$2R=\mathcal{O}$
$Q=(1,3)$	$Q+R=(2,3)$	
$2Q=(0,1)$	$2Q+R=(4,4)$	
$3Q=(3,0)$	$3Q+R=(6,0)$	
$4Q=(0,6)$	$4Q+R=(4,3)$	
$5Q=(1,4)$	$5Q+R=(2,4)$	
$6Q=\mathcal{O}$		

U ovom slučaju dvije tačke generiš u krivu.

Naredna teorema nam govori još o samoj strukturi grupe $E(\mathbb{Z}_p)$, kojom se može dalje okarakterisati grupa u zavisnosti od prostog broja p.

[3]Ovaj "detalj" ima dosta veliki praktičan značaj, sa obzirom da je u nekim kripto shemama veličinu ključeva moguće značajno smanjiti.

teorema 3.2.2. *Neka je E eliptička kriva definisana nad $\mathbb{Z}_p$, gdje je p prost broj, $p > 3$. Tada postoje brojevi $n_1, n_2 \in \mathbb{N}$ takvi da je E izomorfna sa $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$. Dalje $n_2 \mid n_1$ i $n_2 \mid (p-1)$.*

U slučaju da je $n_2 = 1$ tada je E ciklična grupa. Takođe ako je $\#E$ prost broj ili proizvod dva različita prosta broja, tada E mora biti ciklična grupa.

3.3 Kriptografija eliptičkim krivama

Sa obzirom na strukturu grupe $E(\mathbb{Z}_p)$ i njenu kardinalnost, za kriptografski sistem je bitno da nađemo cikličnu podgrupu grupe $E(\mathbb{Z}_p)$ u kojoj je problem diskretnog logaritma težak.

3.3.1 ElGamal

Shema ElGamal je ista kao u slučaju multiplikativne grupe $\mathbb{Z}_p^*$. Naime shema obezbijeđuje:

- generisanje ključeva,
- algoritma kriptovanja i
- algoritma dekriptovanja

Generisanje ključa u ElGamal sistemu

1. Za datu eliptičku krivu E odnosno grupu $E(\mathbb{Z}_p)$, izaberemo generator grupe G. U opštijem slučaju grupa generisana sa G može biti i podgrupa od $E(\mathbb{Z}_p)$.
2. Dalje, na slučajan način izaberemo "tajni" eksponent α, $1 \leq a \leq ord(G) - 1$, sračunamo vrijednost $\alpha \cdot G$.
3. Javni ključ je trojka $(E(\mathbb{Z}_p), G, \alpha \cdot G)$ dok je tajni ključ broj α. Napomenimo samo da prvi element u trojci koja predstavlja javni ključ možemo predstaviti i parametrima[4], naime prost broj p, zatim elementi $a, b \in \mathbb{Z}_p$ definišu grupu $E(\mathbb{Z}_p)$, samim tim javni ključ možemo predstaviti i kao $(p, a, b, G, \alpha \cdot G)$.

Algoritam kriptovanja u ElGamal sistemu

1. Za dati javni ključ (p, a, b, G, n) i za polazni "tekst" m, na slučajan način biramo broj k tako da je $1 \leq k \leq Ord(G) - 1$
2. Izračunavamo $\gamma = k \cdot G$ i $\delta = m + k \cdot (\alpha \cdot G)$
3. Kao i kod osnovnog ElGamal sistema kriptujuća transformacija je:

$$\begin{aligned} E(m, k) &= (\gamma, \delta) \\ &= (k \cdot G, m + k \cdot (\alpha \cdot G)) \end{aligned}$$

[4]U kasnijoj implementaciji ćemo koristiti parametre koji su već definisani, (p, a, b, G, n), pri čemu je n red elementa G.

Algoritam dekriptovanja u ElGamal sistemu

1. Dekriptujuća transformacija definisana je sa:

$$D(\gamma, \delta) = -\alpha \cdot \gamma + \delta$$

Pogledajmo sada prethodni primjer eliptičke krive

$$E : y^2 = x^3 + x + 6$$

nad poljem $\mathbb{Z}_{11}$. Već je pokazano da je grupa $E(\mathbb{Z}_{11})$ kardinalnosti 13 (prostim računanjem za svaki element polja). Samim tim je grupa $E(\mathbb{Z}_{11})$ je ciklična i za prizvoljan element grupe $\alpha \in E(\mathbb{Z}_{11})$ (koji nije jednak $\mathcal{O}$) znamo da je generator grupe. To nam je dovoljno za postavku parametara za *ElGamal* sistem. U nekim opštijim slučajevima izbor prostog broja p kao i parametara a i b i generatora G podgrupe same grupe eliptičke krive predstavlja predmet detaljnih ispitivanja. Naime, svi učesnici u kripto shemi moraju da se dogovore oko parametara sheme. U slučaju eliptičkih krivih to moraju biti prost broj p, parametri a i b koji definišu samu krivu, odnosno grupu $E(\mathbb{Z}_p)$. Posebno treba obratiti pažnju na izbor elementa $G \in E(\mathbb{Z}_p)$ generatora ciklične podgrupe H grupe $E(\mathbb{Z}_p)$, potrebno je da red elementa G bude prost broj. Takođe, kako red podgrupe deli red grupe imamo i celobrojni koeficijent (kofaktor):

$$h = \frac{|E(\mathbb{Z}_p)|}{|H|}$$

potrebno je da ovaj koeficijent bude što manji $h \leq 4$, a po mogućnosti $h = 1$.

Primjer generisanje ključa u ElGamal sistemu

Izaberimo na slučajan način generator grupe, recimo $\alpha = (2, 7)$. Takođe na slučajan način biramo tajni "eksponent" $a = 7$. Tada, javni ključ je trojka $(E, \alpha, a \cdot \alpha)$, dok je tajni ključ a. Tačnije, E možemo pretstaviti kao trojku (p, a, b), generator $\alpha = (x, y)$, kao i $(u, v) = a \cdot \alpha$ su tačke na krivoj E, pa je javni ključ u ovom slučaju $e = ((p, a, b), (x, y), (u, v))$, odnosno sa obzirom da je $(7, 2) = 7 \cdot (2, 7)$:

$$e = ((11, 1, 6), (2, 7), (7, 2))$$

tajni ključ u ovom slučaju je $d = a = 7$.

Primjer postupka kriptovanja u ElGamal sistemu

Kriptujuću transformaciju u ovom slučaju možemo iskazati na sledeći način:

$$\begin{aligned} E_e(m, k) &= (\gamma, \delta) \\ &= (k \cdot (2, 7), m + k \cdot (7, 2)) \end{aligned}$$

pri čemu je $m \in E(\mathbb{Z}_{11})$ i $0 \leq k \leq 12$.

Primjer postupka dekriptovanja u ElGamal sistemu

Dekriptujuću transformaciju u ovom slučaju možemo iskazati na sledeći način:

$$D_d(\gamma, \delta) = -a \cdot \gamma + \delta = -7 \cdot \gamma + \delta$$

Recimo da Alis želi da pošalje Bobu poruku $m = (10, 9)$ koristeći prethodnu shemu (sa već uvedenim ključem), takođe neka je $k = 3$. Tada je:

$$\begin{aligned} E_e((10,9),3) &= (3 \cdot (2,7), (10,9) + 3 \cdot (7,2)) \\ &= ((8,3), (10,9) + (3,5)) \\ &= ((8,3), (10,2)) \end{aligned}$$

Kada Bob primi kriptogram dekriptuje ga na sledeći način:

$$\begin{aligned} D_d((8,3),(10,2)) &= -7 \cdot (8,3) + (10,2) \\ &= -(3,5) + (10,2) \\ &= (3,6) + (10,2) \\ &= (10,9) \end{aligned}$$

Postoji nekoliko problema u samoj implementaciji *ElGamal* kripto sheme sa eliptičkim krivama. Prvo, kriptogram u klasičnoj shemi ima ekspanziju sa faktorom dva, dok u postavci sa eliptičkim krivama ima ekspanziju sa faktorom četiri. Naime, u osnovnom ElGamal postupku za polazni tekst (u ovom slučaju prirodan broj) dobijamo kao kriptogram uređen par prirodnih brojeva, u slučaju računanja na eliptičkoj krivoj za polazni tekst, ako je predstavljen nekim prirodnim brojem, kao rezultat kriptujuće transformacije dobijamo uređen par tačaka na eliptičkoj krivoj.

Problem predstavljanja otvorenog teksta nas dovodi do drugog i mnogo "težeg" problema tačnije do problema *kodiranja* otvorenog teksta (polaznog teksta). Naime, u prethodnom primjeru otvoreni tekst je u stvari tačka na krivoj E, a ne postoji odgovarajući metod koji na "deterministički"[5] način generiše tačke na E. Kasnije ćemo implementirati algoritam kojim na "probabilistički" način dolazimo do kodiranja (predstavljanje otvorenog teksta tačkama na krivoj) otvorenog teksta. Pod pojmom "probabilistički" podrazumjevamo da u postupku predstavljanja otvorenog teksta tačkom na krivoj koristimo slučajne veličine (u ovom primjeru bi to značilo da za isti polazni tekst nemoramo uvijek dobiti istu tačku na krivoj).

Za date parametre kripto sheme (p, a, b) i dati otvoreni tekst postupak "kodiranja", tačnije predstavljanje otvorenog teksta tačkom na krivoj, teče na sledeći način:

1. otvoreni tekst predstavimo nizom bajtova (npr. svakom karakteru azbuke dodjelimo neki broj 0 - 255).

[5]Ovdje pod pojmom "deterministički" podrazumjevamo da za date ulazne veličine uvijek dobijamo isti rezultat, što u slučaju primijenjenog algoritma za predstavljanje otvorenog teksta tačkom na krivoj nije slučaj

2. Na slučajan način izaberemo jedan bajt i stavimo ga na prvo mjesto u prethodnom nizu.

3. Novodobijeni niz bajtova pretvorimo u cijeli broj u, za dobijeni broj računamo vrijednost $z = u^3 + au + b \pmod{p}$

4. Ukoliko je $z \in QR(\mathbb{Z}_p)$ računamo koren $v = z^{\frac{1}{2}}$ i postupak završavamo, otvoreni tekst je predstavljen tačkom (u, v)

5. Ukoliko $z \notin QR(\mathbb{Z}_p)$ vratimo se na korak 2.

6. otvoreni tekst iz tačke dobijamo jednostavnim uklanjanjem provog bajta iz niza bajtova kojom je predstavljena prva koordinata tačke.

Sledeća kripto shema predstavlja pokušaj rješenje ovog problema. Naime u njoj se polazni tekst *"maskira"* eliptičkom krivom.

3.3.2 Menezes-Vanstone

Menezes i *Vanstone* su predložili kripto shemu koristeći eliptičke krive, koja na neki način predstavlja "varijaciju" *ElGamal* sistema. Razlika je, da se otvoreni tekst *"maskira"* eliptičkom krivom, nasuprot *"utapanja"* - predstavljanje otvorenog teksta tačkama na krivoj. Međutim pokazano je da se ovim *maskiranjem* gube neke od pretpostavki samog sistema. Naime, dokazano je da *Menezes-Vanstone* kriptosistem nije "probabilistički", nasuprot svom dizajnu.

Generisanje ključeva u Menezes-Vanstone sistemu

Neka je E eliptička kriva definisana nad $\mathbb{Z}_p$, gdje je $p > 3$ prost broj, koja sadrži cikličnu podgrupu H u kojoj je problem diskretnog logaritma težak. Neka je $\mathcal{M} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ i $\mathcal{C} = E \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*$, dalje definišemo:

$$\mathcal{K} = \{(E, \alpha, a, \beta) : \beta = a \cdot \alpha\}$$

pri čemu je $\alpha \in E$. Vrijednosti α i β su javni, a vrijednost a je tajna.

Algoritam kriptovanja u Menezes-Vanstone sistemu

Za $e = (E, \alpha, a, \beta)$, za (tajni) slučajan broj $k \in \mathbb{Z}_{|H|}$ i za $x = (x_1, x_2) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ definišemo:

$$E_e(x, k) = (y_0, y_1, y_2)$$

gdje je:

$$\begin{aligned} y_0 &= k \cdot \alpha, \\ (c_1, c_2) &= k \cdot \beta, \\ y_1 &= c_1 x_1 \bmod p, \\ y_2 &= c_2 x_2 \bmod p. \end{aligned}$$

U ovom postupku otvoreni tekst $x = (x_1, x_2)$ *"maskiramo"* tačkom $(c_1, c_2) = k \cdot \beta$ na krivoj E.

Algoritam dekriptovanja u Menezes-Vanstone sistemu

Za kriptogram $y = (y_0, y_1, y_2)$, definišemo:

$$D_e(y_0, y_1, y_2) = (y_1 c_1^{-1} \bmod p, y_2 c_2^{-1} \bmod p)$$

gdje je:

$$a \cdot y_0 = (c_1, c_2).$$

Korektnost ove sheme sledi iz računa:

$$a \cdot y_0 = a \cdot k \cdot \alpha = k \cdot a\alpha = k \cdot \beta = (c_1, c_2)$$

primjer 3.3.1. Vratimo se našem ranijem primjeru eliptičke krive:

$$E : y^2 = x^3 + x + 6$$

nad poljem $\mathbb{Z}_{11}$. Ranije smo vidjeli da grupa $E(\mathbb{Z}_{11})$ ima 13 elementa. Znači da smo u ElGamal postavci imali samo 13 mogućnosti za reprezentaciju polaznog teksta, dok u Menezes - Vanstone imamo $10 \times 10 = 100$. Neka je kao i prethodnom primjeru za E generator $\alpha = (2, 7)$ i neka je Bobov tajni eksponent $a = 7$. Tada imamo:

$$\beta = 7 \cdot \alpha = 7 \cdot (2, 7) = (7, 2)$$

Predpostavimo da Alis želi da pošalje "tekst":

$$x = (x_1, x_2) = (9, 1)$$

dalje Alisa bira slučajan broj $k = 6$. Prvo računa:

$$y_0 = k \cdot \alpha = 6 \cdot (2, 7) = (7, 9)$$

i

$$k \cdot \beta = 6 \cdot (7, 2) = (8, 3)$$

pa imamo da je $c_1 = 8$ i $c_2 = 3$. Dalje računa:

$$y_1 = c_1 \cdot x_1 \pmod p = 8 \cdot 9 \pmod{11} = 6$$

i

$$y_2 = c_2 \cdot x_2 \pmod p = 3 \cdot 1 \pmod{11} = 3$$

Kriptogram koji Alisa šalje Bobu je:

$$y = (y_0, y_1, y_2) = ((7, 9), 6, 3)$$

Kada Bob primi kriptogram prvo računa:

$$(c_1, c_2) = a \cdot y_0 = 7(7, 9) = (8, 3)$$

i nakon toga sračunava polazni "tekst" (račun u $\mathbb{Z}_{11}$):

$$\begin{aligned} x &= (y_1 c_1^{-1} \pmod p, y_2 c_2^{-1} \pmod p) \\ &= (6 \cdot 8^{-1} \pmod{11}, 3 \cdot 3^{-1} \pmod{11}) \\ &= (6 \cdot 7 \pmod{11}, 3 \cdot 4 \pmod{11}) \\ &= (9, 1) \end{aligned}$$

3.3.3 Zaključak

Kako smo već pomenuli, vjeruje se da je **ECDLP** problem (**DLP** u slučaju *eliptičkih krivih*) znatno teži, od "klasinčnog" **DLP**. Danas oblast *eliptičkih krivih* i njihove veze sa kriptografijom predstavljaju goruću temu savremenih istraživanja iz oblasti kako kriptografije, tako i kriptoanalize. Takođe, vjeruje se da će **ECDLP** zasnovani kriptografski sistemi (razmijena ključeva, potpisivanje) zamijeniti sisteme zasnovane na problemima faktorizacije cijelih brojeva (npr. RSA) kao i sisteme zasnovanim na aritmetici konačnih polja (npr. DSA). U prilog ovome je i podatak da je na RSA konferenciji 2005 godine Američka "Nacionalna Bezbjedonosna Agencija" (NSA) objavila uvođenje *Suite B* (skupa B) koji koristi kriptografiju eliptičkim krivama za potpisivanje i razmijenu ključeva. Skup B je namjenjen zaštiti kako tajnih, tako i javnih sistema u sistemima nacionalne bezbijednosti. U decembru 2006, NSA je podnijela "Internet Draft" o implementaciji *Suite B* kao dijela **IPsec** protokola. Ovaj draft je prihvaćen od strane IETF kao RFC4869. Napomenimo još da je veći dio sistema i algoritama koji koriste eliptičke krive zaštićen patentima. Kanadska firma **Certicom**[6] je vlasnik oko 350 patenata iz oblasti kriptografije eliptičkim krivama. Ovo je jedan od faktora koji dosta usporava primjenu ovih sistema u praksi. Na primjer, *OpenSSL* projekat (na kojem se zasniva sigurnost Apache servera, jednog od najzastupjenijih "http" servera) prihvatila je *ECC* "patch" tek 2005 godine iako je bio urađen 2002 godine. Dalje, napomenimo da je NSA platila oko 25 miliona američkih dolara za formiranje *Suite B* sistema kao i prava za implementaciju za vlastite potrebe.

[6]Certicom je kanadska firma čiji je jedan od osnivača i Skot Vanstone (*Scott Vanstone*) koautor čuvenog "Priručnika primjenjene kriptografije". Napomenimo da je Certicom kupljen od strane firme RIM, poznatije po proizvodnji "blackberry" uređaja

Glava 4

Implementacija - ECcrypto

Prikazaćemo implementaciju jednostavnog kriptosistema zasnovanog na **ElGamal** shemi koristeći grupe definisane eliptičkim krivama oblika:

$$y^2 = x^3 + ax + b \tag{4.1}$$

nad poljem $\mathbb{Z}_p$, gdje je p nepran prost broj. Pored jednostavnijih primjera (mali prost broj p), iskoristićemo i neke primjere eliptičkih krivih definisanih u [9]. Da bi koristili aritmetiku velikih brojeva, potreban nam je neki poseban sistem za rad sa velikim brojevima, a sa obzirom da većina programskih jezika ima u osnovi podržan rad sa 32-bitnim ili 64-bitnim cijelim brojevima (što se u osnovi mogu smatrati malim brojevima u kriptografiji) moramo koristiti ili neke matematičke pakete kao što su *Mathematica* ili *Mapple*, programski jezik UBASIC (koji izvorno podržava rad sa proizvoljnim velikim cijelim brojevima) ili neki od programskih jezika koji posjeduje biblioteke za rad sa velikim brojevima (tačnije cijelim brojevima proizvoljne tačnosti).

Jedan od programskih jezika koji, skoro od samog početka, ima klase za rad sa velikim brojevima je **JAVA**. Java posjeduje nekoliko osnovnih tipova koji reprezentuju cijele brojeve od kojih su sa *"najvećom tačnošću"*:

- tip long, odnosno `java.lang.Long` - tip koji predstavlja 64-bitne cijele brojeve, preciznije cijele brojeve u intervalu $[-2^{63}, 2^{63} - 1]$
- tip int, odnosno `java.lang.Integer` - tip koji predstavlja 32-bitne cijele brojeve, preciznije cijele brojeve u intervalu $[-2^{31}, 2^{31} - 1]$

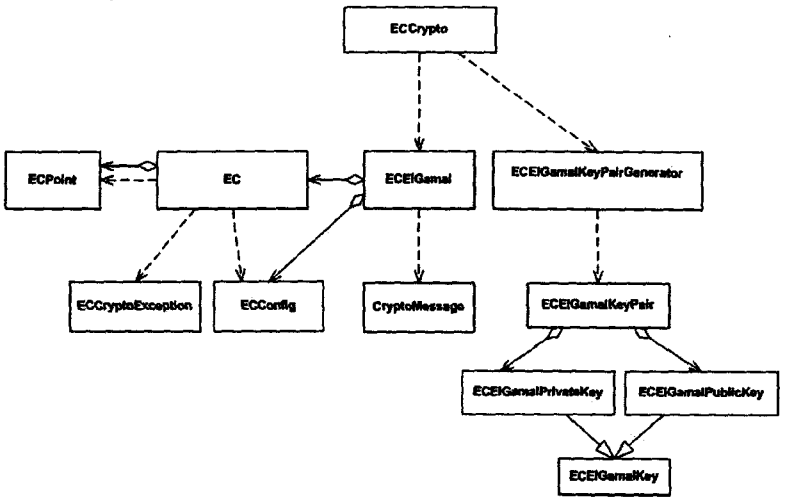
Kao što smo ranije napomenuli za *"praktičnu kriptografiju"* neophodni su brojevi veličine 256 bita i veći, neophodana je i klasa za rad sa velikim cijelim brojevima. Klasa koja je osnovna za rad sa velikim cijelim brojevima u Javi je `java.math.BigInteger`. Ova klasa pored osnovnih operacija za rad sa velikim brojevima podržava i osnovne operacije modularne aritmetike. Klasa je sastavni dio standardne java biblioteke (**JDK**) od verzije 1.1.

Implementacija ovog kriptosistema obuhvata sledeće mogućnosti :

- Generisanje ključeva,
- Kriptovanje i
- Dekriptovanje.

4.1 Osnovne klase i algoritmi

Dajemo opis nekih osnovnih klasa i algoritama koji su neophodni u implementaciji sistema. Sva aritmetika u polju $\mathbb{Z}_p$ se odvija pomoću metoda u klasi `java.math.BigInteger`. Osnovna klasa koja implementira aritmetiku na eliptičkoj krivoj je `EC`, `ECElGamal` je klasa u kojoj se izvode transformacije kriptovanja i dekriptovanja, dok se u klasi `ECElGamalKeyPairGenerator` implementira logika generisanja javnih i tajnih ključeva (parova ključeva). Slijedi detaljniji opis svih klasa u sistemu.



klasa `java.math.BigInteger`

Kao što je već napomenuto klasa `java.math.BigInteger` predstavlja osnovnu klasu za rad sa cijelim brojevima proizvoljne tačnosti. Pored osnovnih operacija modularne aritmetike sa cijelim brojevima proizvoljne tačnosti, klasa omogućava izračunavanje NZD-a, testiranje da li je neki broj prost, generisanje prostih brojeva, manipulacije brojeva na nivou bitova i niza "pomoćnih" metoda. Neke od pomoćnih metoda koje su korišćene pri aritmetizaciji otvorenog teksta su:

1. `byte[] toByteArray();`
   Ova metoda preslikava proizvoljan cijeli broj u niz bajtova. Preslikavanje se izvodi u osnovi 256, s tim što dobijamo bajtove koji su u rasponu

$[-2^7, 2^7-1]$. Tačnije dobijamo "dvo komplementnu" reprezentaciju cijelog broja.

2. konstruktor `BigInteger(byte[] array);`
   Za dati niz bajtova dobijamo cijeli broj.

klasa `eccrypto.EC`

Je "centralna" klasa sistema, sa obzirom da predstavlja eliptičku krivu i implementira "aritmetiku" na eliptičkoj krivoj. Osnovne metode koje implementiraju "aritmetiku" su:

1. `ECPoint addPoints(ECPoint a, ECPoint b);`
   Implementira operaciju sabiranja na eliptičkoj krivoj. U slučaju da su tačke `a` i `b` sa jednakom x koordinatom kao rezultat dobijamo tačku $\mathcal{O}$.

2. `ECPoint doublePoint(ECPoint a);`
   Implementira operaciju sabiranja tačke sa samom sobom na eliptičkoj krivoj (udvostručuje tačku). Koristimo je pri računanju stepena tačke na krivoj.

3. `ECPoint multiplyPoint(BigInteger n, ECPoint a);`
   Implementira "brute force" algoritam stepenovanja, naime za date parametre `n` i `a` računamo vrijednost $n \cdot a$, uzastopnim dodavanjem tačke `a` samoj sebi `n` puta. Napomenimo da ovaj algoritam nije efikasn i da postoji mnogo efikasniji algoritam za stepenovanje na eliptičkoj krivoj[1].

4. `ECPoint multiplyPointDP(BigInteger n, ECPoint a);`
   Implementira algoritam "stepenovanja kvadriranjem"[2], ovaj algoritam koristimo za "stepenovanje" na eliptičkoj krivoj. Algoritam se zasniva na ideji predstavljanja broja n u binarnom obliku. Za svaki bit u binarnoj predstavi broja n udvostručujemo vrijednost a, ali sabiranje ovih vrijednosti vršimo samo za one vrijednosti gdje je bit jednak 1. Ovim metodom značajno možemo smanjiti broj množenja.

5. `ECPoint negatePoint(ECPoint a);`
   Implementira operaciju u kojoj računamo inverz tačke a na eliptičkoj krivoj.

Klasa takođe implementira i metode potrebne za predstavljanje otvorenog teksta tačkama na krivoj.

1. `ECPoint codeDataToPoint(byte[] data);`
   Implementira algoritam "utapanja", odnosno predstavljanja otvorenog teksta tačkama na krivoj. Osnovna predpostavka je da je otvoreni tekst već "aritmetizovan", tačnije predstavljen prirodnim brojem. Takav tekst predstavimo nizom bajtova i ovom metodom "kodiramo" tačkom na krivoj.

[1] prikazan je samo zbog "ilustarcije"
[2] viditi [11], strana 25

Napomenimo da otvoreni tekst prije kodiranja tačkama eliptičke krive razbijemo na "blokove" određene dužine takve da je jedan blok moguće predstaviti tačkom na krivoj.

2. `byte[] codePointToData(ECPoint point);`
   Implementira "dekodiranje" tačke sa eliptičke krive u niz bajtova. Algoritam teč tako što za x koordinatu pretvorenu u niz bajtova, samo uklonimo bajt sa "prvog" mjesta, niz koji dobijemo kada pretvorimo u broj je polazni tekst.

klasa `eccrypto.ECPoint`

Predstavlja tačku na eliptičkoj krivoj.

klasa `eccrypto.ECcryptoException`

Predstavlja "exception" koji se "izbacuje" u slučaju da inicijalizacija krive parametrima nije u redu, tačnije ukoliko kriva nije regularna, odnosno ako je $\Delta = 0$.

klasa `eccrypto.ECConfig`

Predstavlja konfiguracione parametre za eliptičku krivu. Takođe ovo je parametar za konstruktor klase `eccrypto.EC`. Konfiguracioni parametri uključeni u ovu klasu su:

1. Prost broj p - karakteristika polja $\mathbb{Z}_p$,
2. Brojevi $a, b \in \mathbb{Z}_p$ - parametri eliptičke krive $y^2 = x^3 + ax + b$,
3. Tačka G na krivoj, bazna tačka - generator podgrupe od $E(\mathbb{Z}_p)$,
4. $ord(G)$ - red elementa G, $ord(G) \leq p$

Ova klasa sadrži i malu kolekciju, već definisanih parametara prema [9]. U sledećem poglavlju dati su primjeri korščenja ovih parametara i njihovo detaljnije objašnjenje.

klasa `eccrypto.elgamal.ECElGamal`

Predstavlja osnovnu klasu u "kriptosistemu" s obzirom da implementira kriptujuću i dekriptujuću transformaciju, takođe, klasa sadrži reference ka klasama koje su neophodne za izvođenje ovih transformacija.

Osnovne metode ove klase (postupak kriptovanja) su:

1. `void crypt(File file, ECElGamalPublicKey publicKey);`
2. `byte[][] crypt(String message, ECElGamalPublicKey publicKey);`

3. `byte[] crypt(byte[] message, ECElGamalPublicKey publicKey);`

Sam postupak *kriptovanja* teče na sledeći način:

1. "Pročitamo" tekstualnu datoteku - polazni tekst(otvoreni tekst), i smjestimo ga u promjenjivu tipa `String`,
2. Dobijeni `String` razbijemo na blokove takve dužine da je svaki blok aritmetizacijom uz pomoć klase `SimpleEncoder` moguće predstaviti tačkom na eliptičkoj krivoj - koristimo dužinu bita broja p kao parametar.
3. Svaki blok kriptujemo tako što prvo izvršimo "utapanje" bloka (predstavimo ga tačkom na krivoj) i zatim uradimo kriptujuću transformaciju.
4. Kriptogram je u osnovi niz parova tačaka na eliptičkoj krivoj (predstavljen "matricom" `byte[][]`),
5. Poslednji korak je zapisivanje takve "matrice" u datoteku.

Osnovne metode ove klase (postupak dekriptovanja) su:

1. `void decrypt(File file, ECElGamalPrivateKey privateKey);`
2. `byte[][] decrypt(byte[][] data, ECElGamalPrivateKey privateKey);`
3. `byte[] decrypt(byte[] message, ECElGamalPrivateKey privateKey);`

Postupak *dekriptovanja* teče na sledeći način:

1. "Pročitamo" tekstualnu datoteku - kriptogram i smjestimo ga u promjenjivu tipa `byte[][]`, format u kojem smo zapisali parove tačaka na eliptičkoj krivoj.
2. Svaki par "tačaka" kriptograma dekriptujemo.
3. "Matricu" `byte[][]` predstavimo nizom `byte[]` (operacija inverzna razbijanju teksta na blokove) i tako dobijeni niz, dekodiramo.

klasa `eccrypto.elgamal.CryptoMessage`

Predstavlja kriptovanu poruku. Tačnije u sistemu *ECElGamal* kriptovana poruka je par tačaka na eliptičkoj krivoj, tako da ova klasa u osnovi apstrahuje par tačaka (klasa `ECPoint`). Takođe, klasa implementira i dvije pomoćne metode:

1. `byte[] toByteArray(int type, boolean compressed),`
2. `CryptoMessage fromByteArray(byte[] data, int type, boolean compressed)`

Ove dvije metode služe kao mehanizam "serijalizacije" klase `CryptoMessage`, tj. iz niza bajtova moguće je rekonstruisati `CryptoMessage`, kao i "zapisati" `eccrypto.elgamal.CryptoMessage` kao niz bajtova. Ovu funkcionalnost koristimo ukoliko recimo želimo da u nekoj datoteci "sačuvamo" kriptogram.

klasa `eccrypto.elgamal.keys.ECElGamalKey`

Predstavlja nadklasu za `ECElGamalPublicKey` i `ECElGamalPrivateKey`. Osnovni elementi, koje klasa sadrži, su zajednički parametri ključeva: karakteristika polja $\mathbb{Z}_p$, parametri eliptičke krive a i b, generator (tačka na krivoj) G i red tačke $ord(G)$.

klasa `eccrypto.elgamal.keys.ECElGamalPublicKey`

Pored zajedničkih parametara, klasa sadrži tačku na krivoj - generator stepenovan tajnim eksponentom.

klasa `eccrypto.elgamal.keys.ECElGamalPrivateKey`

Pored zajedničkih parametara, klasa sadrži element tipa `BigInteger` - tajni eksponent.

klasa `eccrypto.elgamal.keys.ECElGamalKeyPair`

Predstavlja par od jednog javnog i jednog tajnog ključa.

klasa `eccrypto.elgamal.keys.ECElGamalKeyPairGenerator`

Predstavlja osnovnu klasu koja služi za generisanje objekata klase `ECElGamalKeyPair`. Sadrži i pomoćne metode za "čuvanje" ključeva. Naime, cjelokupna logika, generisanja, čuvanja, importovanja i eksportovanja ključeva je implementirana u ovoj klasi. Klasa ima konstruktor `ECElGamalKeyPairGenerator(String folderPath, ECConfig ecConfig)` kojim se postavljaju osnovni parametri za ovu klasu, naime pri generisanju ključeva koristimo parametre iz `ecConfig`, dok nam je `folderPath` putanja do direktorijuma u kojem će biti sačuvani ključevi.

Osnovne metode ove klase su:

1. `ECElGamalKeyPair generateKeyPair(String keyPairName)`
2. `ECElGamalPublicKey getPublicKey(String keyName)`
3. `ECElGamalPrivateKey getPrivateKey(String keyName)`
4. `void exportKey(String keyName)`
5. `importKey(String keyName)`

klasa `eccrypto.elgamal.ECCrypto`

Predstavlja (startnu) klasu iz koje se pozivaju sve ostale metode. Sam poziv programa se izvršava u ovoj klasi. Klasa obezbjeđuje mehanizam za parsiranje osnovnih komandi presleđenih sa komandne linije. Primjeri korišćenja programa dati su u sledećem poglavlju.

klasa `eccrypto.utils.SimpleEncoder`

Služi kao jednostavan "encoder". Naime, kako je osnovni tip `byte` u javi u rasponu $[-2^7, 2^7 - 1]$, to za neke znakove standardnog alfabeta koji su u nekom standardnom "encodingu" (recimo `WIN-1250` ili `WIN-1251`), kodirani brojevima preko 127, dobijamo negativne brojeve. To znači da bi za neki otvoreni tekst ako bi direktno koristili enkoding postavljen na operativnom sistemu (`WIN-1250` ili `WIN-1251`) za aritmetizaciju mogli dobiti i negativan broj. Da bi izbjegli ovu situaciju klasa `SimpleEncoder` primjenjuje svoj enkoding tako da se za aritmetizaciju otvorenog teksta uvijek dobije pozitivan broj. Također, klasa implementira i metodu za razbijanje niza bajtova na blokove zadate dužine:

1. `byte[] encode(String plainText);`
   Za dati tekst vraća niz bajtova, u osnovi ova metoda predstavlja aritmetizaciju zadatog teksta.

2. `String decode(byte[] encodedText);`
   Za dati niz bajtova, vraća tekst, u osnovi ova metoda je "inverz" prethodne metode.

3. `byte[][] getByBlock(byte[] data, int blockSize);`
   Razbija dati niz bajtova `byte[] data` na blokove dužine `blockSize`. Ukoliko zadata dužina bloka nije djelilac dužine zadatog niza bajtova, niz bajtova dopunimo bajtovima da dobijemo cijeli broj blokova. To uradimo tako što na "slučajan" način generišemo bajtove.

4. `byte[] fromBlock(byte[][] data, int blockSize);`
   Za datu matricu bajtova `data` vraća niz bajtova (vrstu po vrstu)

klasa `eccrypto.utils.BigIntegerUtils`

Implementira neke dodatne algoritme neophodne za računanje na eliptičkoj krivoj. Sledeći algoritmi su implementirani:

1. `boolean isQuadraticResidue(BigInteger a, BigInteger p);`
   Implementira *Ojlerov* kriterijum za utvrđivanje da li je broj kvadratni ostatak.

2. `BigInteger quadraticResidue(BigInteger a, BigInteger p);`
   Za dati p prost broj oblika $p \equiv 3 \bmod 4$ i cijeli broj $a \in QR(p)$ računa korijen od a po formuli:

$$u^2 \equiv a \pmod{p} \rightarrow u \equiv a^{\frac{p+1}{4}} \pmod{p}$$

4.2 Primjeri korišćenja

Tri su osnovna primjera korišćenja programa ECcrypto: upravljanje ključevima, kriptovanje i dekriptovanje. Kako je već pomenuto program koristi internu listu parametara prema *SEC2* - parametrima ([9]). Naime, u samom kodu predstavljena je lista eliptičkih krivih, tačnije osnovnih parametara neophodnih za implementaciju osnovnih operacija u grupi generisanoj eliptičkom krivom (klasa `eccrypto.ECConfig` je klasa koja sadrži listu parametara). Svaki konfiguracioni objekat sastoji se od sledećih parametara:

- p - prost broj, definiše polje $\mathbb{Z}_p$
- a - koeficijent a u krivoj $y^2 = x^3 + ax + b$
- b - koeficijent b u krivoj $y^2 = x^3 + ax + b$
- G - generator grupe (podgrupe), predstavljen koordinatama (G_x, G_y)
- n - red elementa G
- h - kofaktor, $\#E(\mathbb{Z}_p)/n$

Prema *SEC1* ([8]) parametri za polja oblika $\mathbb{Z}_p$ moraju imati:

$$\lceil log_2 p \rceil \in \{112, 128, 160, 192, 224, 256, 384, 521\}$$

Ovo ograničenje je sa jedne strane zbog implementatora (korišćenje istih parametara vodi ka kompatibilnosti implementacija), sa druge strane ako je polje $\mathbb{Z}_p$ takvo da je $\lceil log_2 p \rceil = 2t$ ono obezbijeđuje aproksimativno t bitova sigurnosti (vjerujemo da nam je za rješavanje problema diskretnog logaritma za ovu eliptičku krivu potrebno otprilike 2^t operacija). Navodimo primjer parametara za `SECP-112R`:

SECP-112R1		
p	DB7C 2ABF62E3 5E668076 BEAD208B	$(2^{128} - 3)/76439$
a	DB7C 2ABF62E3 5E668076 BEAD2088	
b	659E F8BA0439 16EEDE89 11702B22	
g	09487239 995A5EE7 6B55F9C2 F098 A89C E5AF8724 C0A23E0E	
n	DB7C 2ABF62E3 5E7628DF AC6561C5	
h	01	

Ovo je kriva koju koristimo i za generisanje ključa sa imenom `TEST` pri prvom pokretanju programa. Naime, po prvom pokretanju programa, u radnom direktorijumu (direktorijum u kojem je program pokrenut) kreiraju se dvije datoteke `pubring.txt` i `secring.txt` koje sadrže par ključeva imena `TEST` od 112 bita. Parametri krive koji su korišćeni za generisanje su `SECP-112R1` prema [9]. Program podržavo još dvije krive prema [9]. To su `SECP-128R1` i `SECP-160R1`, navodimo i parametre za ove krive:

SECP-128R1		
p	FFFFFFFD FFFFFFFF FFFFFFFF FFFFFFFF	$2^{128} - 2^{97} - 1$
a	FFFFFFFD FFFFFFFF FFFFFFFF FFFFFFFC	
b	E87579C1 1079F43D D824993C 2CEE5ED3	
g	161FF752 8B899B2D 0C28607C A52C5B86	
	CF5AC839 5BAFEB13 C02DA292 DDED7A83	
n	FFFFFFFE 00000000 75A30D1B 9038A115	
h	01	

SECP-160R1		
p	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7FFFFFFF	$2^{160} - 2^{31} - 1$
a	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7FFFFFFC	
b	1C97BEFC 54BD7A8B 65ACF89F 81D4D4AD C565FA45	
g	4A96B568 8EF57328 46646989 68C38BB9 13CBFC82	
	23A62855 3168947D 59DCC912 04235137 7AC5FB32	
n	00000000 00000000 0001F4C8 F927AED3 CA752257	
h	01	

Pokretanjem programa sa `java -jar ECCrypto.jar -?` dobijamo meni sa osnovnim uputama za korišćenje.

```
Dobrodošli u ECCrypto ver.0.0.1
============================================================
Pubring uspješno kreiran...
Secring uspješno kreiran...
naredba '-?'
-?  - pomoć (ova poruka)
-gk 112|128|160 ime - generisanje para ključeva sa imenom
-ek ime - eksport javnog ključa
-ik ime - import javnog ključa
-cr datoteka ime - kriptovanje datoteke javnim ključem
-dr datoteka ime - dekriptovanje datoteke tajnim ključem
```

Navodimo neke primjere poziva programa u ovim slučajevima.

- `java -jar ECCrypto.jar -gk 112 Alice`
  generiše ključ sa imenom 'Alice', koriste se parametri `SECP-112R1`. U primjeru parametra 128 koristimo krivu `SECP-128R1`, dok u primjeru parametra 160 koristimo krivu `SECP-160R1`. Generisani ključevi se sačuvaju u datotekama `pubring.txt` i `secring.txt` respektivno.

- `java -jar ECCrypto.jar -ek Alice`
  eksportujemo ključ sa imenom 'Alice'. U radnom direktorijumu kreira se datoteka Alice.egk

- `java -jar ECCrypto.jar -ik Bob`
  importujemo ključ sa imenom 'Bob'. Iz datoteke Bob.egk koja se nalazi u radnom direktorijumu importujemo javni ključ imana Bob.

- `java -jar ECCrypto.jar -cr c:\temp\test.txt Bob`
  kriptujemo ključem (javnim) sa imenom 'Bob' sadržaj datoteke `c:\temp\test.txt`. Kreira se datoteka `c:\temp\test.txt.ecg`
- `java -jar ECCrypto.jar -dr c:\temp\test.txt.ecg Bob`
  dekriptujemo ključem (tajnim) sa imenom 'Bob' sadržaj datoteke `c:\temp\test.txt.ecg`. Kreira se datoteka `c:\temp\test.txt.ecg.ecg`

Bibliografija

[1] A.J. Menezes, P.C.Van Oorschot, S.A. Vanstone. *Handbook of Applied Cryptography.* CRC press, 1997.

[2] John Talbot, Dominic Welsh. *Complexity and Cryptography.* Cambridge university press 2006.

[3] Douglas Stinson. *Cryptography: Theory and Practice.* CRC Press, 1995.

[4] Neal Koblitz. *A Course in Number Theory and Cryptography.* Springer, 1994.

[5] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer, 2003.

[6] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, Inc., second edition, 1996.

[7] Whitfield Diffie, Martin Helmman. *New Directions In Cryptography.* IEEE Transactions on Information Theory, Volume 22, Issue 6, Nov 1976.

[8] *SEC1: Elliptic Curve Cryptography.* Certicom Corp., May 21, 2009 version 2.0.

[9] *SEC2: Recommended Elliptic Curve Domain Parameters.* Certicom Corp., September 20, 2000 version 1.0.

[10] Martin Helmman. *An Overview of Public Key Cryptography.* IEEE Communications magazine, November 1978 - Volume 16, Number 16.

[11] Ed Schaefer. *An introduction to cryptography.* http://math.scu.edu/ eschaefe/crylec.pdf.

[12] Sajmon Sing. *Knjiga o Šiframa.* DN Centar, PLATO, Beograd 2003.