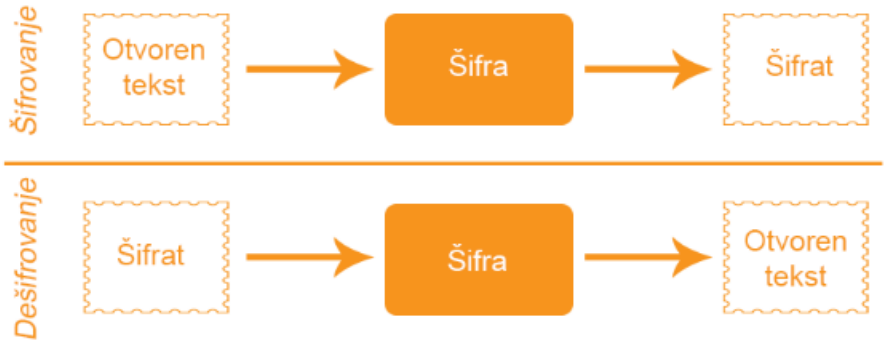
2 Kriptografske primitive

Elektronsko poslovanje i digitalni svet današnjice je zasnovano na upotrebi moderne kriptografije. Zahvaljujući kriptografiji, Alisa može da pošalje poruku Bobu preko interneta, i da se pri tom može matematički garantovati da poruka nije bila oštećena, da je zaista Alisa poslala tu poruku, kao i da je Bob jedina osoba koja može da pročita njen sadržaj. Kriptografija je omogućila građaninu da kupuje proizvode putem interneta i plaća svoje račune bez brige da će mu novac biti ukraden. Svo poslovanje preduzeća, državnih institucija i pojedinaca, koje se vrši putem interneta, je omogućeno zahvaljujući kriptografiji.

U ovom poglavlju će biti objašnjene kriptografske primitive koje čine modernu kriptografiju, kao i načine primene ovih primitiva u modernim informacionim sistemima[1].

2.1 Osnovna terminologija

U najprostijem obliku, kriptografski algoritam predstavlja šifru (engl. *Cipher*). Šifra je algoritam koji vrši šifrovanje (engl. *Encryption*) i dešifrovanje (engl. *Decryption*). Šifrovanje predstavlja funkciju koja pretvara otvoreni tekst (engl. *Plaintext*) u šifrat (engl. *Ciphertext*), dok dešifrovanje predstavlja funkciju koja pretvara šifrat u originalni otvoreni tekst (Slika 2.1).



Slika 2.1 Proces šifrovanja i dešifrovanja

Kako kriptografija postoji već hiljadama godina, moguće je naći primere za proste šifre koje su bile u upotrebi tokom vekova. Cezarova šifra je jedna od najstarijih i najprostijih šifri, i pripada porodici šifre zamenjivanja. Otvoreni tekst se šifruje tako što se svaki karakter u otvorenom tekstu zameni sa karakterom koji je određen broj karaktera udaljen u alfabetu. U slučaju alfabeta, koji se sastoji od 26 karaktera, funkcija za šifrovanje jednog karaktera se matematički može opisati kao:

$$E_n(p) = (p + n)\ mod\ 26$$

Gde *p* predstavlja karakter otvorenog teksta, a *n* je ceo broj koji predstavlja pomeraj u alfabetu putem kog se nalazi karakter šifrata. Za n = -3, karakter „G“ u otvorenom tekstu postaje karakter „D“ u šifratu. Karakter „C“ postaje karakter „Z“. Analogno se može opisati funkcija za dešifrovanje:

$$D_n(c) = (c - n)\ mod\ 26$$

Gde je *c* karakter šifrata, a *-n* je ceo broj koji predstavlja pomeraj u alfabetu putem kog se nalazi karakter otvorenog teksta.

[1] Sami koraci kriptografskih algoritama, kao i kompleksna matematika koja garantuje sigurnost algoritma i otpornost na napade izlaze van opsega ovog udžbenika.

Posmatrajući Cezarovu šifru, moguće je uočiti sve pojmove koji igraju ulogu kako u zastarelim, tako i u modernim algoritmima za šifrovanje i dešifrovanje. Pored otvorenog teksta, šifrata i same šifre, pojavljuje se ključ, čija vrednost mora ostati tajna da bi šifrat ostao bezbedan, a to je broj *n*. U vreme pre računara, razvijene računice, i pismenosti, Cezarova šifra je mogla da zaštiti poruke o ratnim planovima i intrigama senata, dok god je sam broj *n*, i pogotovo algoritam, ostao tajan.

Danas je jednostavno poraziti Cezarovu šifru bez poznavanja broja *n*, pa i bez poznavanja samog algoritma. Posmatrajući šifrat:

```
sryhuomlyrvw srgudcxphyd cdvwlwx srgdwdnd rg djhqdwd nrml qhpdmx sudyr
gd sulvwxsh gdwlp srgdflpd lqwhjulwhw vh rgqrvl qd vsrvreqrvw
vsuhfdydqmd qhgrcyromhqh l qhsuhgylgmhqh surphqh srgdwdnd grvwxsqrvw
srgudcxphyd vsrvreqrvw sulvwxsd srgdflpd, vhuylvlpd, xuhđdmlpd,
kdugyhux lol elor nrp uhvxuvx x wuhqxwnx ndgd mh wr srwuheqr
```

Može se analizirati frekvencija pojavljivanja karaktera u rečima. Karakter koji se najviše pojavljuje je „d“ sa 33 pojavljivanja, zatim „r“ sa 29 pojavljivanja, i onda „h“ sa 26 pojavljivanja. Ukoliko je poznato da je tekst koji se nalazi iza šifrata napisan na srpskom jeziku, moguće je izvršiti mapiranje na najčešća slova u srpskom jeziku. To su, redom, „a“, „o“ i „i“, gde je i „e“ relevantno, jer je četvrto najučestalije slovo koje se pojavljuje mnogo češće od preostalih slova [1]. Uz malo isprobavanja dolazi se do zaključka da je „d“ zapravo „a“, „r“ je „o“ i „h“ je „e“, i indukcijom se primećuje da je korištena Cezarova šifra nad alfabetom, gde je $n = 3$.

Proces koji je opisan kroz prethodni primer spada u oblast koja se naziva kriptoanaliza. Kriptoanaliza je oblast koja se bavi izučavanjem metoda za saznavanje informacija iz šifrata, bez posedovanja tajnih podataka koji su potrebni da bi se pristupilo datim informacijama. Analiza frekvencije pojavljivanja karaktera je jedna od najprostijih metoda kriptoanalize. Kriptoanaliza i kriptografija su dve celine oblasti koja je poznata kao kriptologija.

Cezarova šifra se, pre svega, uzdala u tajnost samog algoritma. Moderni kriptografski algoritmi se ne mogu uzdati u tajnost algoritma, niti je to dobra strategija. Šifra dizajnirana od strane par pojedinaca se ne može takmičiti sa šifrom koja je javno dostupna, gde ceo svet može da pronađe ranjivosti u algoritmu, i doprinese razvoju sistema koji se matematički može dokazati da je siguran.

Pored tajnosti algoritma, Cezarova šifra se uzdala u broj *n* koji je predstavljao jednostavan ključ po kom se šifrovanje i dešifrovanje vršilo. Današnji šifrati, šifrovani od strane pouzdanih, matematički proverenih šifri su zaštićeni dok god je ključ koji se koristi za dešifrovanje šifrata zaštićen. Drugim rečima, moderna šifra treba da garantuje bezbednost i u slučaju kada je napadač detaljno poznaje, dok god ključ za dešifrovanje ostane tajan. Sa druge strane, ključ koji se koristi za Cezarovu šifru ima nekoliko desetina vrednosti, što znači da se brzo može pogoditi. Moderni kriptografski algoritmi koriste ključeve dužine od više stotine bitova, što ih čini otpornim na napad pogađanja (engl. *Brute force*).

2.2 Simetrične šifre

Šifra koja sa proizvoljnim ključem šifruje otvoreni tekst i, nakon toga, koristi isti ključ da dešifruje nastali šifrat kako bi se dobio početni otvoreni tekst, spada u grupu simetričnih šifri. Alisa i Bob žele da razmene poruku preko interneta, koristeći simetrični šifru:

1. Alisa i Bob dogovaraju konkretnu simetričnu šifru koju će koristiti, kao i ključ K;
2. Alisa šifruje poruku sa ključem K koristeći dogovorenu šifru;
3. Alisa šalje rezultujući šifrat, preko mreže, Bobu;
4. Bob, koristeći istu šifru i ključ K, dešifruje šifrat i dobija originalnu poruku.

Formalno, simetrična šifra se može definisati kao skup dve funkcije, funkcije za šifrovanje *E* i funkcije za dešifrovanje *D*. Za svaku poruku *M* i ključ *K* se može dobiti šifrat *C*, tako da važi:

$$C = E(M, K); M = D(C, K)$$

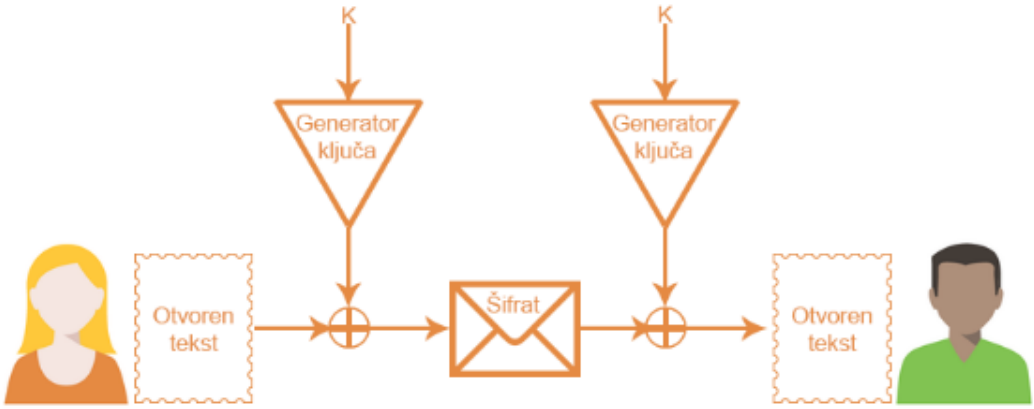
Postoje dve osnovne grupe simetričnih šifri, i to su:

- Šifra niza (engl. *Stream*), gde se redom šifruje jedan po jedan bit ili bajt;
- Blok (engl. *Block*) šifre, gde se u svakoj rundi algoritma šifruje blok podataka, koji može biti različite veličine, na primer 64 bita ili 128 bita.

Šifre niza

Šifre niza svoje ime dobijaju po ključu koji se generiše kao beskonačan niz bitova upotrebom generatora pseudo slučajnih brojeva, koji treba da simulira jednokratnu svesku (engl. *One time pad*). Alisa želi da pošalje šifrovanu poruku Bobu upotrebom šifre niza:

1. Alisa i Bob dogovaraju algoritam koji će da koriste, uključujući generator pseudo slučajnih brojeva koji će da generiše ključ, i početni ključ K koji će se koristiti kao nasumičan *seed* za generator;
2. Alisa prosleđuje početni ključ generatoru pseudo slučajnih brojeva, koji potom generiše bajt za svaki bajt otvorenog teksta Alisine poruke. Putem operacije XOR se spaja bajt generisanog ključa i otvorenog teksta, čime se dobija bajt šifrata;
3. Alisa prosleđuje šifrat Bobu;
4. Bob upotrebom dogovorenog ključa aktivira svoj generator pseudo slučajnih brojeva i generiše identičan ključ koji je Alisa generisala. Upotrebom XOR operacije između bajtova generisanog ključa i šifrata dobija se otvoreni tekst, odnosno Alisina poruka (Slika 2.2).



Slika 2.2 Upotreba šifre niza da se prenese poruka

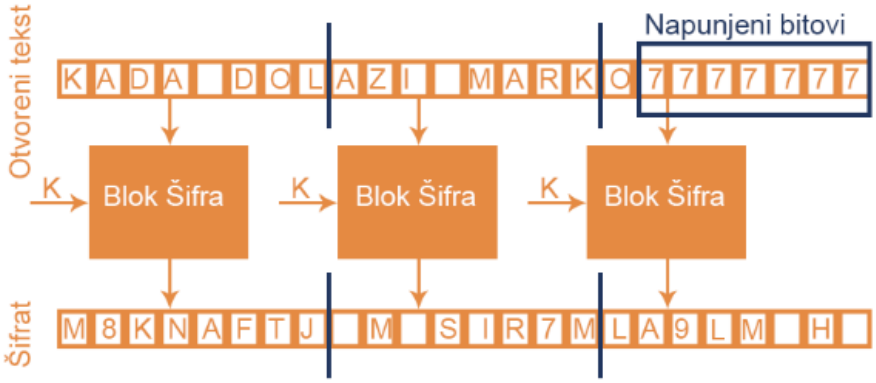
Šifre niza nisu dostigle nivo upotrebe kao blok šifre, ali pronalaze primenu zbog svoje efikasnosti. Ovi algoritmi zbog svoje jednostavnosti rade brzo i zahtevaju malo hardverskih resursa. Ključan problem predstavlja ispravna implementacija generatora slučajnih brojeva koji formiraju ključ, i mnogo algoritama iz ove grupe pati od dizajna koji nije otporan na metode kriptoanalize.

Blok šifre

Blok šifre dele otvoren tekst u niz blokova određene dužine, gde se jedan blok otvorenog teksta prosleđuje algoritmu za šifrovanje, i pretvara u jedan blok šifrata. Ukoliko dužina otvorenog teksta nije jednaka umnošku dužine bloka, poslednji blok se dopuni sa određenim bitovima (engl. *Padding*). Ukoliko dužina otvorenog teksta jeste jednaka umnošku dužine bloka, kreira se još jedan blok koji će biti u potpunosti ispunjen određenim bitovima.

Bitno je ispravno izabrati strategiju dopunjavanja poslednjeg bloka, tako da ne dođe do otkrivanja informacija o ključu koji je korišten za formiranje šifrata. Najosnovnija strategija jeste da se preostali prostor popuni nulama. U tom slučaju, napadač koji zna za upotrebu te strategije čita poslednji blok šifrata i kriptoanalizom dolazi do informacije o pojedinim bitovima ključa. Strategija za punjenje bita bazirana na PKCS#7 standardu [2] diriguje da se preostali prostor dopuni sa brojevima koji označavaju kolika je dužina dopunjenih bajta.

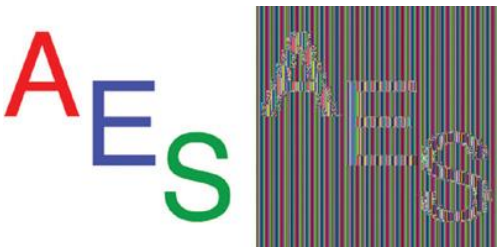
Postoji više režima (engl. *Mode*) u kojim blok šifra drugačije orkestrira blokove, kombinuje ih i šifruje, kako bi se povećala bezbednost same šifre. ECB (engl. *Electronic Code Book*) režim, je najprostiji oblik orkestracije ovih blokova, gde se jedan blok otvorenog teksta direktno šifruje u jedan blok šifrovanog teksta (Slika 2.3).



Slika 2.3 ECB režim rada blok šifre

Problem kod ECB režima rada se ogleda u činjenici da isti ključ za isti otvoreni tekst proizvodi isti šifrat. Ako napadač osluškuje razmenu imejlova između Alise i Boba, uočiće iste blokove u šifratima koji putuju sa jednog računara na drugi. Ovo bi mogla biti *To*, *From* i *Subject* polja u zaglavlju imejla, ili potpis pri dnu. Kada poseduje šifrat i otvoreni tekst, napadač može potencijalno da otkrije deo ključa, čime smanjuje potreban posao da se pogodi ceo ključ. Na primer, ako napadač zna da će prva dva bloka nekog šifrata biti standardna polja zaglavlja imejla, onda za ta dva bloka on zna sadržaj šifrata i otvorenog teksta, čime može da otkrije, na primer, 10 od 100 bita ključa, što je umanjilo potreban maksimalan broj pokušaja da se pogodi ključ sa 2^{100} na 2^{90}. Ako je ista priča za potpis pri dnu imejla, možda napadač može da otkrije još 10 bita, čime dodatno smanjuje sebi posao.

ECB režim rada blok šifre očuvava u šifratu šablone koji se nalaze u otvorenom tekstu (Slika 2.4).



Slika 2.4 ECB režim rada blok šifre preslikava šablone iz otvorenog teksta u šifrat

Prevaziđen, ali istorijski značajna blok šifra predstavlja DES (engl. *Data Encryption Standard*) [3], standardizovana 1977. Algoritam opisan standardom deli otvoreni tekst na blokove i koristi ključ od 56 bita za šifrovanje. Glavni razlog zašto je algoritam prevaziđen se pronalazi u dužini ključa. Iako je DES godinama važio za najbolje rešenje koje pruža sigurnu komunikaciju, upotrebom specijalnog hardvera je 1999. razbijen šifrat pogađanjem ključa za manje od jednog dana [4].

Iako tajne šifre nisu dobra ideja, posmatrajući prethodni tekst jasno je da se i kod proverenih i javno prihvaćenih šifri vremenom mogu pronaći ranjivosti. DES je primer jednog takvog algoritma, koji je koristan bio u prethodnom veku, ali je sada već uveliko prevaziđen. Dalje, prikazano je kako ECB režim rada blok šifre ima ranjivost (Slika 2.4), a ranjivosti se pronalaze i u načinu dopunjavanja poslednjeg bloka.

Trenutno najsigurniju simetričnu šifru predstavlja AES (engl. *Advanced Encryption Standard*) [5], koristeći ključ od 256 bita, CBC ili CTR režim rada [6], i oslanjajući se na PKCS#7 strategiju za dopunu bitova. Međutim, ovo ne znači da vremenom neće biti otkrivene ranjivosti u algoritmu, režimu rada ili strategiji za dopunu bitova. Kada se pojavi potreba za upotrebu simetrične šifre u projektu, neophodno je istražiti šta se u tom trenutku smatra za sigurnu šifru, ali i koje su najbolje prakse za konfiguraciju date šifre.

2.3 Asimetrične šifre

Još od Cezarove šifre, pa sve do 1976. je kriptografija patila od problema razmene ključeva. Ukoliko su dve strane želele da razmenjuju šifrate, prvo su morali da razmene ključeve, pre nego što su mogli da bezbedno komuniciraju. Ovo se promenilo sa otkrivanjem asimetrične šifre.

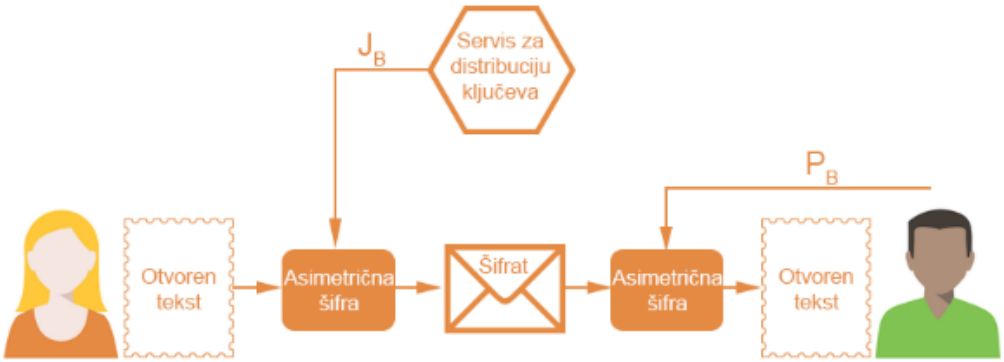
Asimetrične šifre koriste dva ključa, gde šifrovanje otvorenog teksta sa jednim ključem proizvodi šifrat koji se može dešifrovati sa drugim. Jedan ključ je javno dostupan (engl. *Public key*), dok je drugi privatan (engl. *Private key*) i treba se čuvati njegova tajnost. Otvoreni tekst je, dakle, moguće šifrovati sa javnim ili privatnim ključem, a rezultujući šifrat se dešifruje sa drugim.

Asimetrična šifra se definiše kao skup dve funkcije, funkcije za šifrovanje E i funkcije za dešifrovanje D. Za svaku poruku M i par javnog i privatnog ključa J i P se mogu se dobiti šifrati C i X, tako da važi:

$$C = E(M, J);\ M = D(C, P); X = E(M, P); M = D(X, J)$$

Alisa i Bob žele da komuniciraju upotrebom asimetrične šifre:

1. Alisa i Bob su dogovorili konkretnu šifru putem koje će komunicirati;
2. Alisa i Bob generišu sebi par ključeva, odnosno svoj javni i privatni ključ;
3. Alisa i Bob razmenjuju javne ključ, putem direktne komunikacije, ili upotrebom servisa za objavu javnih ključeva, poput MIT PGP Public Key Server [7];
4. Alisa koristi Bobov javni ključ, J_B, da šifruje poruku;
5. Alisa šalje rezultujući šifrat Bobu;
6. Bob koristi svoj privatni ključ, P_B, da dešifruje poruku i pročita njen sadržaj (Slika 2.5).



Slika 2.5 Razmena poruke putem asimetrične šifre

Putem asimetričnih šifri moguće je otvoreni tekst šifrovati kako sa javnim, tako i sa privatnim ključem. Šifrovanje poruke sa javnim ključem znači da rezultujući šifrat može dešifrovati samo subjekt koji je u

posedstvu odgovarajućeg privatnog ključa, čime se može garantovati poverljivost poruke. Sa druge strane, ako se poruka šifruje sa privatnim ključem, svako ko ima pristup javnom ključu može da dešifruje rezultujući šifrat, što je u opštem slučaju velik broj, potencijalno nepoznatih, subjekata. Ova strategija ne doprinosi poverljivosti poruke, ali dati mehanizam ima svoju upotrebu. Ako se poruka može dešifrovati nečijim javnim ključem, sledi da je ta poruka bila formirana od strane tog nekog, što je koristan podatak.

Problem kod asimetričnih šifri jeste provera garancije da neki javni ključ stvarno pripada datoj osobi. Na servisu za distribuciju ključeva se nalazi javni ključ uz koji stoje ime Alisa, ali to samo po sebi ne garantuje da neki maliciozni subjekt nije postavio svoj javni ključ uz tuđe ime. Upravljanje i organizacija ključeva je ozbiljan problem koji će biti razmotren u sledećem poglavlju.

Bezbednost asimetričnih šifri je garantovana matematičkim tehnikama koje se nazivaju jednosmerne funkcije sa tajnom (engl. *Trapdoor function*). Ideja kod ovih tehnika jeste da je računanje funkcije za neki ulaz računski jednostavno, dok je računanje inverzne funkcije izuzetno zahtevno, ukoliko se ne zna tajna informacija (engl. *Trapdoor*).

Jedan primer ovakve funkcije predstavlja množenje dva velika prosta broja. Dobijanje rezultata množenja prostog broja 16 061 sa prostim brojem 99 607 se računski izvršava za manje od milisekunde. Sa druge strane, određivanje koji prost broj puta koji prost broj proizvodi 1 599 788 027 je računski daleko složeniji problem i do danas nije otkriven efikasan algoritam koji ga rešava. Treba napomenuti da samo zato što takav algoritam nije otkriven ne znači da ne postoji. Bezbednost većine jednosmernih funkcija sa tajnom ne leži u matematičkom dokazu, već empirijskom, gde se nakon značajnog broja pokušaja da se slomi šifa smatra da je ona dokazano bezbedna.

RSA (engl. *Rivest-Sharmir-Adleman*) [8] je aktuelna asimetrična šifra, čija bezbednost se zasniva na prethodno navedenom matematičkom problemu. Koraci algoritma za kreiranje ključeva su:

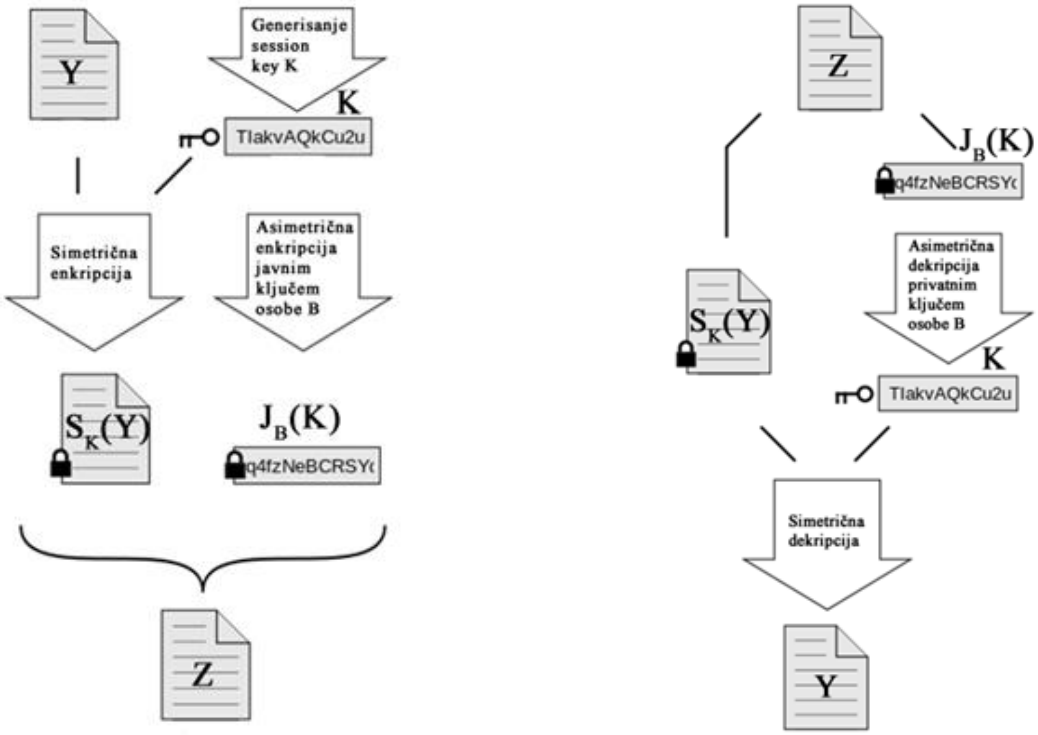
1. Alisa bira dva velika prosta broja p i q, i računa proizvod $n = pq$ i $\varphi(n) = (p - 1)(q - 1)$;
2. Alisa bira broj e čiji najveći zajednički delilac sa $\varphi(n)$ je 1, i izračunava $d = e^{-1} \bmod \varphi(n)$;
3. Alisa objavljuje (e, n) kao svoj javan ključ, dok d predstavlja privatni ključ;
4. Kada želi da joj pošalje poruku m, Bob izračunava $c = m^e \bmod n$, gde je dužina od m manja od n;
5. Alisa prihvata šifrat c i dešifruje ga po formuli $c^d \bmod n$.

Detaljnija analiza RSA, kao i interaktivan kalkulator za računanje ključeva i šifrovanje poruka se može naći u [9]. Pored ove šifre, aktuelna vodeća asimetrična šifra je i ECC (engl. *Elliptic Curve Cryptography*) [10]. ECC šifre su zasnovane na problemu diskretnog logaritma primenjenog nad eliptičnim krivama. Matematika iza algoritma je složenija od RSA, i sam algoritam je teže implementirati. Kvalitetno objašnjenje o kriptografiji nad eliptičnim krivama se može naći u [10]. RSA šifre se pokazala jednostavnijom za razumeti i ispravno implementirati, ali ECC šifre nude veću sigurnost sa kraćim ključevima (Tabela 1). Ovo smanjuje prostor potreban da se skladišti ključ i povećava brzinu operacije generisanja ključeva i digitalnog potpisivanja. ECC šifre se trenutno smatraju kao najkvalitetnijim rešenjem u domenu asimetričnih šifri. Međutim, kontroverzija prati ovu grupu šifri, uključujući i odluku NSA organizacije da izbaci ECC šifre iz upotrebe pred kraj 2015. godine [11].

Tabela 1 Dužina ključeva u bitovima za različite grupe šifri

ECC šifre	RSA šifre	Simetrične šifre
224	2048	112
256	3072	128
384	7680	192
521	15360	256

U poređenju sa asimetričnim šiframa, simetrične šifre nude veću bezbednost po bitu ključa. Simetrični šifre koriste kraće ključeve i, u opštem slučaju, rade brže i zahtevaju manje energije. Sa druge strane, asimetrične šifre ne pate od problema razmene ključeva. Upravo zbog toga se asimetrične i simetrične šifre zajedno koriste (Slika 2.6).



Slika 2.6 Proces šifrovanja i dešifrovanja upotrebom kombinacije simetrične i asimetrične šifre

Alisa želi da pošalje poruku *Y* Bobu:

1. Alisa i Bob dogovaraju koji skup simetričnih i asimetričnih šifri će se koristiti;
2. Alisa generiše ključ *K* za simetričnu šifru koji će se koristiti za ovu poruku (engl. *Session key*);
3. Alisa šifruje poruku *Y* putem simetrične šifre i generisanog ključa i dobija šifrat $S_K(Y)$;
4. Koristeći asimetričnu šifru i Bobov javni ključ J_B, Alisa šifruje ključ *K* i dobija šifrat $J_B(K)$;
5. Šifrovana poruka i šifrovan ključ se spajaju u poruku *Z*, koja se šalje preko mreže;
6. Sa druge strane Bob rastavlja poruku *Z* na šifrovanu poruku i šifrovan simetrični ključ;
7. Bob koristi svoj privatni ključ, P_B, da dobije ključ *K* za simetričnu šifru;
8. Bob koristi ključ *K* da dešifruje šifrovanu poruku i dobija poruku *Y*.

Šifrovanje poruka asimetričnim šiframa je neefikasno, pogotovo u slučaju veoma dugačkih poruka [12].

Upotrebom asimetrične šifre, pogotovo u kombinaciji sa simetričnom šifrom, moguće je na efikasan način omogućiti očuvanje svojstva poverljivosti poruke, i ovaj mehanizam se koristi u HTTPS protokolu da zaštiti poverljivost poruke koja se šalje sa klijenta na server.

2.4 Heš funkcije

Heš funkcije su jednosmerne funkcije koje preslikavaju otvoreni tekst proizvoljne dužine na heš fiksne dužine. Ukoliko je heš funkcija bezbedna, subjekat koji poseduje heš ne može da izračuna otvoren tekst od kog je nastao heš.

Jedna od primena heš funkcije jeste garancija integriteta podataka, gde igra ulogu *checksum* vrednosti. Alisa šalje poruku X Bobu i želi da se osigura da greška u protokolu komunikacije ne izmeni njenu poruku:

1. Alisa računa heš poruke X, i dobija H(X);
2. Alisa spaja poruku X i heš H(X) i dobija poruku Y, koju prosleđuje Bobu;
3. Bob rastavlja poruku na X i H(X) i izračunava heš od poruke X, čime dobija H'(X);
4. Ukoliko je H(X) = H'(X) znači da poruka X nije bila izmenjena, bar ne od strane greške u komunikaciji.

Kako otvoreni tekst može biti proizvoljne dužine, a heš funkcije generišu heševe fiksne dužine od, na primer, 256 bita, heševi nisu jedinstveni. Do kolizije heševa dolazi kada dve različite poruke proizvedu isti heš. Ukoliko je heš funkcija bezbedna, napadač koji poseduje heš neće u razumnom vremenu pronaći poruku koja se sračunava u dati heš, odnosno neće lako pronaći originalnu poruku niti izazvati koliziju heševa.

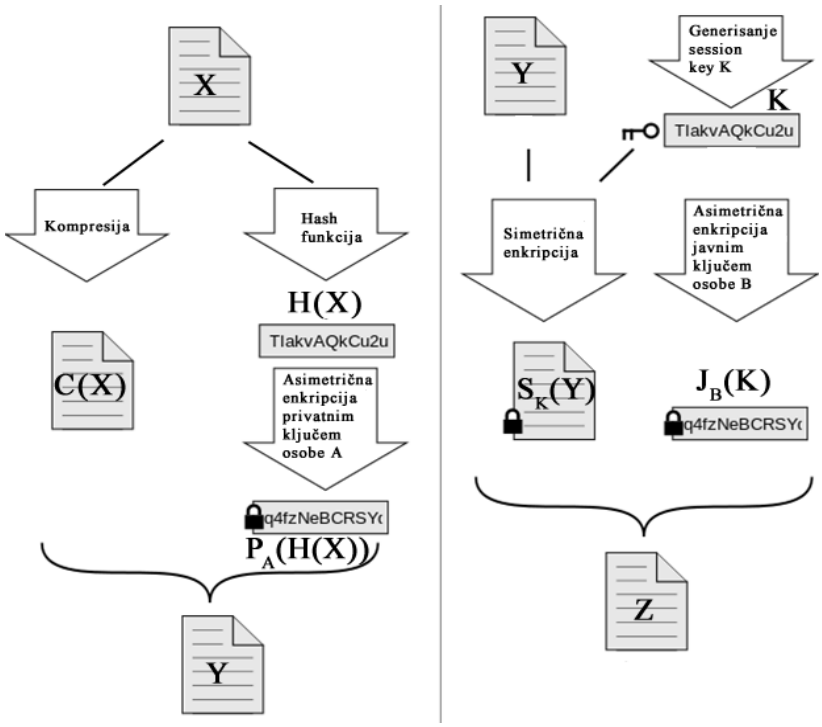
Idealna heš funkcija će ispunjavati sledeće zahteve:

- Funkcija je deterministička, gde isti tekst uvek proizvodi isti heš;
- Izračunavanje heša treba da bude brzo, ali ne previše brzo kako bi se otežao napad pogađanja;
- Nije moguće generisati originalan tekst iz heša;
- Izmena proizvoljnog bita u tekstu menja otprilike pola bitova generisanog heša (efekat lavine);
- Nije moguće, u razumnom vremenu, pronaći dve različite poruke koje proizvode isti heš.

Algoritama za heširanje je bilo više tokom istorije, i mnogi, poput MD5, su napušteni zbog svojih ranjivosti. Aktuelna bezbedna rešenja predstavljaju algoritmi iz SHA-2 i SHA-3 grupe [13], što ne znači da će u budućnosti to biti slučaj. Kao i uvek kod informacione bezbednosti, kada se pojavi zahtev za upotrebu nekog bezbednosnog mehanizma poput funkcije za heširanje, potrebno je sprovesti istraživanje da se proveri koji algoritam se smatra za zlatni standard.

2.5 Bezbedna komunikacija

Upotrebom simetrične šifre, asimetrične šifre i heš funkcije zajedno, moguće je garantovati poverljivost, integritet i neporecivost komunikacije preko interneta ili neke slične mreže (Slika 2.7).



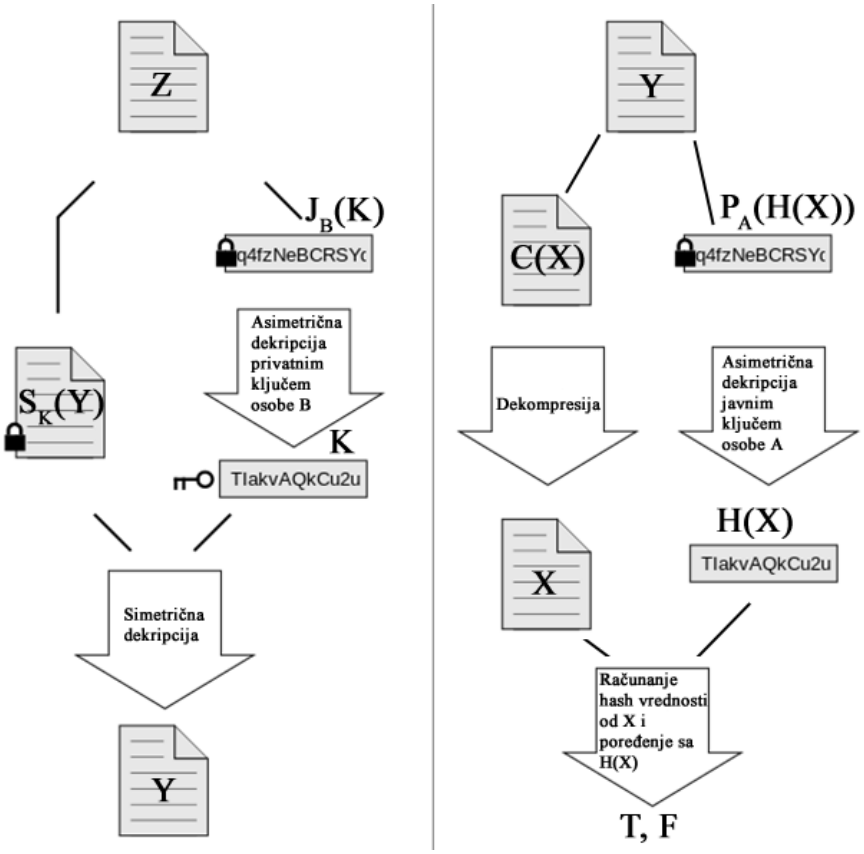
Slika 2.7 Šifrovanje poruke tako da se garantuje poverljivost, integritet i neporecivost

Alisa želi da pošalje poruku X Bobu na bezbedan način. Kada poruka stigne do Boba, treba da bude siguran da je poruka stvarno stigla od Alise, da se sadržaj poruke nije menjao u tranzitu i da niko drugi nije mogao da presretne poruku i pročita njen sadržaj. Ovo se može postići na sledeći način:

1. Alisa formira heš poruke *X* i dobija *H(X)*;
2. Upotrebom asimetrične šifre i svog privatnog ključa, Alisa šifruje heš i dobija šifrat $P_A(H(X))$;
3. Nakon što izvrši kompresiju poruke *X* i dobije *C(X)*, Alisa spaja poruku i $P_A(H(X))$ u *Y*;
4. Koristeći ranije opisan postupak (Slika 2.6), Alisa šifruje poruku i šalje šifrat Z Bobu.

Ovde su interesantna prva tri koraka, pre svega $P_A(H(X))$. Svako ko ima pristup Alisinom javnom ključu (J_A) može da dešifruje ovaj šifrat, te poverljivost nije garantovana. Međutim, šifrat se može dešifrovati upotrebom Alisinog javnog ključa samo u slučaju da je ona upotrebila svoj privatni ključ da šifruje heš, čime se postiže svojstvo autentifikacije, a kada postoji integritet i autentifikacija moguće je garantovati neporecivost (ako je poruka šifrovana Alisinim privatnim ključem, a nije modifikovana sledi da je poruka došla u tom obliku baš od Alise). Heš vrednost poruke, i bilo kog dokumenta, šifrovan privatnim ključem nekog subjekta predstavlja **digitalni potpis** subjekta nad tim dokumentom.

Preduslov da Bob može da dešifruje poruku, proveri da li je ona došla od Alise i da li se menjala u toku tranzita jeste da poznaje i koristi sve algoritme koje je Alisa koristila, kao i da ima pristup Alisinom javnom ključu. Slika 2.8 ilustruje sam proces.



Slika 2.8 Dešifrovanje poruke tako da se garantuje poverljivost, integritet i neporecivost

1. Bob dešifruje poruku *Z* i dobija poruku *Y* po prethodno opisanom postupku (Slika 2.6);
2. Koristeći javni ključ od Alise, dešifruje $P_A(H(X))$ i dobija heš originalne poruke, odnosno *H'(X)*;
3. Bob vrši dekompresiju originalne poruke i izračunava njen heš, koji poredi sa *H(X)*;
4. Ukoliko je *H'(X) = H(X)* znači da poruka nije menjana u toku tranzita, kako od strane greške u transportu tako i od strane malicioznog napadača, te je Alisin digitalan potpis validan.

Komunikacija je istinski sigurna samo ako javni ključ J_A stvarno pripada Alisi, i samo ako javni ključ J_B stvarno pripada Bobu.

2.6 Rezime

U ovom poglavlju su pokriveni osnovni kriptografski koncepti. Nakon prolaska kroz osnovnu terminologiju vezanu za kriptografiju razmotrene su tri ključne funkcije koje predstavljaju temelje kriptografije – simetrične šifre, asimetrične šifre i heš funkcije. Na kraju su sve tri funkcije spojene u jedan proces putem kog se ostvaruje bezbedna komunikacija između dva subjekta, gde je poverljivost, integritet i neporecivost svake poruke garantovana.

2.7 Zadaci

1. Razmotriti zbog čega su šifre zamene poput Cezarove šifre, morale da se uzdaju u tajnovitost algoritma. Koncipirati što jednostavniju šifru zamene koja se isključivo uzda u tajnovitost ključa.
2. Kod simetrične šifre niza, generiše se ključ K upotrebom generatora pseudo slučajnih brojeva i zatim se vrši operacija XOR između K i otvorenog teksta da bi se dobio šifrat. Razmotriti zašto je operacija XOR tu pogodna i na koji način upravo ona doprinosi bezbednosti šifre.
3. Alisa i Bob žele da bezbedno komuniciraju upotrebom AES šifre, ali treba da razmene ključ K. Razmotriti načine razmene ključa, ranjivosti u postupku i kontrole koje bi ih regulisale.
4. Razmotriti pod kojim okolnostima bi neki subjekat menjao svoj javni i privatni ključ.
5. Heš funkcija ima primenu u garanciji integriteta poruke. U primeru gde Alisa šalje poruku i njen heš Bobu, maliciozan subjekat može da presretne poruku, izmeni njen sadržaj, sračuna heš nove poruke i to prosledi Bobu, koji će dobiti jednakost da je H(X) = H'(X). Navesti najjednostavniji proširenje ove razmene, kako bi se istinski garantovao integritet poruke, čak i u slučaju malicioznog napadača koji presreće komunikaciju.
6. Razmotriti problem kolizije heša i kako se može iskoristiti da se eksploatiše digitalno potpisivanje.
7. Ispitati redosled operacija digitalnog potpisivanja, šifrovanja poruke i kompresije. Razmotriti koji redosledi te tri operacije imaju smisla, i šta su mane preostalih redosleda.
8. Uzimajući u obzir da ključevi ističu ili mogu biti povučeni, formirati proširenje do sada navedenog algoritma za digitalno potpisivanje, takvo da je moguće garantovati ispravnost potpisa ukoliko je on nastao dok je ključ bio u upotrebi. Takođe osmisliti rešenje za zaštitu od *reply* napada.
9. Razmatrajući FTN sistem, iskoristiti kriptografske primitive navedene u ovom poglavlju da se zaštiti poverljivost i integritet svih digitalnih resursa koji su prethodno identifikovani.
10. Analizirati kriptografski ključ kao resurs u sistemu koji treba zaštititi.
11. Diskutovati problem ranjivih bezbednosnih kontrola i definisati dobre prakse za upotrebu bezbednosnih kontrola u nekom softveru, tako da se smanji verovatnoća upotrebe ranjive implementacije.

Reference

[1] Singh, S., 1999. *The code book: the evolution of secrecy from Mary, Queen of Scots, to quantum cryptography*. Doubleday.

[2] PKCS#7 standard, https://www.ietf.org/rfc/rfc2315.txt, pristupljeno: 2.2.2018.

[3] Coppersmith, D., 1994. The Data Encryption Standard (DES) and its strength against attacks. *IBM journal of research and development*, *38*(3), pp.243-250.

[4] Virtual Exhibition in Informatics, DES Challenge, http://cs-exhibitions.uni-klu.ac.at/index.php?id=263, pristupljeno: 2.2.2018.

[5] Daemen, J. and Rijmen, V., 2001. Specification for the advanced encryption standard (AES). *Federal Information Processing Standards Publication*, *197*.

[6] Lipmaa, H., Wagner, D. and Rogaway, P., 2000. Comments to NIST concerning AES modes of operation: CTR-mode encryption.

[7] MIT PGP Public Key Server, http://pgp.mit.edu/, pristupljeno: 2.2.2018.

[8] Rivest, R.L., Shamir, A. and Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), pp.120-126.

[9] Popyack, J.L., RSA Calculator, https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html, pristupljeno: 2.2.2018.

[10] Corbellini, A., Elliptic Curve Cryptography: a gentle introduction, http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/, pristupljeno: 2.2.2018.

[11] Green, M., A riddle wrapped in a curve, https://blog.cryptographyengineering.com/2015/10/22/a-riddle-wrapped-in-curve/, pristupljeno: 2.2.2018.

[12] Why is asymmetric cryptography bad for huge data?, http://crypto.stackexchange.com/questions/5782/why-is-asymmetric-cryptography-bad-for-huge-data, pristupljeno: 2.2.2018.

[13] What is SHA-3 and why did we change it?, http://security.stackexchange.com/questions/21112/what-is-sha-3-and-why-did-we-change-it, pristupljeno: 2.2.2018.