SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Josip Iveković

HIBRIDNI KRIPTOSUSTAVI

Diplomski rad

Voditelj rada:
prof. dr. sc. Andrej Dujella

Zagreb, rujan 2014

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Nesigurna okolina poticala je ljude kroz povijest da zaštite svoje podatke. Neki podaci mogli su se sakriti jednostavno izolirajući ih iz nesigurne okoline, no neke podatke nije bilo moguće, na taj način, zaštiti jer izolirajući ih izgubili bi svoju svrhu. To se prvenstveno odnosi na komunikaciju, jer poruka čija se tajnost čuva izoliranjem gubi svoju svrhu. Radi toga ljudima je bila potrebna metoda kojom će moći komunicirati nesigurnim kanalima, na način da sadržaj poruke neće biti kompromitiran čitanjem od strane kojoj ta poruka nije namijena. Postoje razni pokušaji kroz povijest kojim se pokušavalo zaštiti poruke, no do otprilike 16. stoljeća glavna metoda je bila supstitucija koja nije bila teška za "probiti". U 16. stoljeću kriptografija, kao znanost koja se bavi proučavanjem "sigurnih" metoda za izmjenu poruka, se počinje razvijati. Pod terminom "sigurne" metode misli se na one metode koje osiguravaju da jedino osoba kojoj je poruka namijenjena može istu pročitati u izvornom obliku. Do i tokom 20. stoljeća kriptografija se razvijala do zavidnog nivoa, no uvijek je postojala jedna manjkavost cijelog sistema, a taj je bio da ukoliko bi 2 osobe željele izmjenjivati poruke na siguran način, nesigurnim kanalom, one su se prvo trebale osobno naći i izmijeniti potrebne informacije za uporabu metoda kojima bi osigurali poruke. Ponekad bi tada kriptografija tu gubila smisao jer su te dvije osobe mogle razmijeniti informacije koje su željele poslati porukama u osobnom kontaktu. U drugoj polovici 20. stoljeća javlja se nova ideja koja je omogućavala izmjenu poruka na siguran način bez da se te dvije osobe moraju osobno naći. Krajem 20. stoljeća razvija se internet, a s njime i popratne usluge koje zahtjevaju visoki stupanj sigurnosti, a kako je internet nesiguran kanal komunikacije, kriptografija i njezina primjena se sve više i više razvijaju. Sustav u kojem su osobe trebale prvo osobno izmijeniti podatke prije slanja poruka naziva se simetrični kriptosustav, dok nova ideja s kraja 20. stoljeća se naziva asimetrični kriptosustav. Hibridni kriptosustavi koji su i tema ovog rada su kombinacija ta dva sustava i danas naraširenji sustavi za sigurnu izmjenu podataka.

U prvom poglavlju uvode se osnovni pojmovi, definiraju se kriptosustav i njegove podijele. Također uvode se osnovni matematički pojmovi i neki rezultati vezani uz te pojmove, a koji su bitni za rad. Drugo poglavlje bavi se simetričnim kriptosustavima, dok se treće bavi asimetričnim kriptosustavima.Četvrtom poglavlje bavi se hibridnim kriptosustavima, dok se peto bavi sigurnošću kriptosustava.

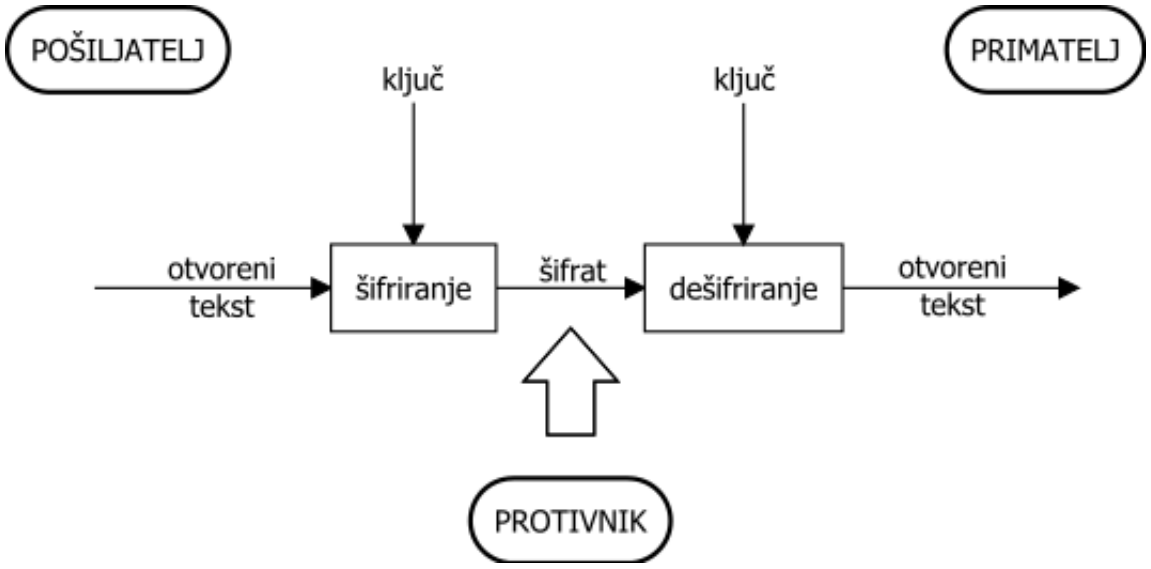
Poglavlje 1

Osnovni pojmovi

1.1 Kriptosustav

Kriptografski sustav ili kraće kriptosustav je konkretna realizacija sustava u kojem je ostvarena tražena sigurnost izmjena poruka. Kriptosustavi građeni su od više dijelova, a biti će dan primjer općeg modela kriptosustava s kratkim opisom toka podataka. U svakom kriptosustavu postoji osoba koja šalje poruku i koja prima poruku te njih možemo označiti s pošiljatelj i primatelj (u literaturi često se označavaju s imenima Alice i Bob). Pošiljatelj pokušava primatelju poslati poruku, tekst koji u izvornom obliku zovemo otvoreni tekst. Međutim kako ne želimo da tekst po nesigurnom kanalu putuje u izvornom obliku potrebna je njegova modifikacija, te tu modifikaciju nazivamo šifriranje, a šifrirani otvoreni tekst nazivamo šifrat. Šifrianje se vrši uz jednu dodatnu informaciju koja se naziva ključ. Nakon šifriranja tekst je spreman za putovanje nesigurnim kanalima. Pretpostavljamo da su kanali nesigurni jer su dostupni na pregled trećoj osobi u našem pojednostavljenom sustavu. Toj osobi poruka nije namijenjena, no kako je ima priliku pročitati, pa makar to bilo u obliku šifrata, tu osobu nazivamo protivnik. Protivnik sam po sebi možda nema nikakva saznanja o pošiljatelju i primatelju, niti da je šifrat poruka između njih, niti možda uopće pokušava pročitati poruku, no zbog same mogućnosti da on može doći do šifrata u našem sustavu on se smatra protivnikom, jer postoji mogućnost, bez obzira koliko ona bila mala, da on pokuša i uspije pročitati poruku u izvornom obliku. Nakon putovanja poruke nesigurnim kanalom poruka dolazi do primatelja, koji je zatim pomoću ključa transformira ponovno u otvoreni tekst. Taj postupak zove se dešifriranje. Nakon dešifriranja primatelj može pročitati poruku koju mu je pošiljatelj poslao. Šifriranje i dešifriranje su kriptografske primitive, a njihova specifikacija je kriptografski algoritam. Kriptografski protokol je dogovoreni način izmjene poruka između sudionika kriptosustava. Ako na kriptografske primitive gledamo kao na osnovne građevne jedinice kriptosustava, tada na kriptofgrafski protokol možemo gledati kao na odabir primitiva i njihovo slaganje u određenom redosli-

jedu kako bi se postigla željena sigurnost.



Slika 1.1: Model kriptosustava

1.2 Podjela kriptosustava

Glavna podjela kriptosustava je prema sljedeća tri kriterija:

1. podjela prema obradi teksta
   Tekst se može obrađivati znak po znak, dok se isto tako može odrediti veličina bloka znakova, a zatim se tekst može obrađivati blok po blok.

2. podjela prema tipu operacije nad tekstom
   Znak (blok znakova) može se zamjenjivati s nekim drugim znakom (blokom znakova) ili se isto tako znakovi (blokovi znakova) mogu ispremješati (ispermutirati). Prvi tip operacije zove se supstitucija, dok se drugi zove transpozicija.

3. podjela prema odabiru ključeva (tajnih i javnih)
   Postoje dva tipa kriptosustava s obzirom na odabir ključeva. Ukoliko je ključ za šifriranje jednak ključu za dešifriranje, tada takav sustav zovemo simetrični kriptosustav. U ovom sustavu poželjno je da je ključ poznat samo pošiljatelju i primatelju te se zato i ključ u ovakvom sustavu zove tajni ključ.

Kriptosustavi kod kojih je ključ za šifriranje različit od ključa za dešifriranje i teško izračunljiv (u nekom razumnom vremenu) zove se asimetrični kriptosustav. Ključ za šifriranje u ovom sustavu zove se javan ključ (dostupan svima i svi mogu pomoću njega šifrirati poruke), dok je ključ za dešifriranje privatni (ili tajni, ali ne u smislu kao kod simetričnih kriptosustava) te poruku može dešifrirati samo osoba koja ga posjeduje.

1.3 Matematički pojmovi

Definicija 1.3.1. ***Grupa*** $G = \{a, b, \ldots\}$ *jest neprazan skup* G *u kome je svakom uređenom paru* $a, b \in G$ *pridružen jedan i samo jedan element iz* G*, koga zovemo produktom elemenata* a *i* b *i pišemo* $a \cdot b$*. Pri tome to pridruživanje, koje zovemo grupovnom operacijom ili množenjem, zadovoljava sljedeće uvjete (aksiome)*

1. $$a \cdot (b \cdot c) = (a \cdot b) \cdot c \quad \textit{(asocijativnost)}$$

   za sve $a, b, c \in G$

2. *Postoji bar jedan element* $e \in G$ *takav da je*

   $$a \cdot e = e \cdot a = a$$

   za svako $a \in G$*. Element* e *s ovim svojstvom zovemo neutralnim ili jediničnim elementom ili jedinicom grupe.*

3. *Za svaki element* $a \in G$ *postoji bar jedan element* $a^{-1} \in G$ *takav, da je*

   $$a \cdot a^{-1} = a^{-1} \cdot a = e$$

   Element a^{-1} *s ovim svojstvom zovemo inverznim elementom elementa* a*.*

Grupa G *zove se* ***komutativna*** *ili* ***Abelova****, ako je* $a \cdot b = b \cdot a$ *za svaki par elemenata* $a, b \in G$*. Abelova grupa u kojoj je grupna operacija označena sa* $+$ *zove se još i aditivna Abelova grupa. U aditivnoj grupi element* a^{-1} *označava se sa* $-a$*, dok jedinicu grupe zovemo nulom i označavamo je sa* 0*.*

Definicija 1.3.2. ***Prstenom*** R *nazivamo aditivnu Abelovu grupu* R *s bar dva različita elementa u kojoj je svakom uređenom paru elemenata* $a, b \in G$ *pridružen jedan jedini element* $a \cdot b \in R$*, koji zovemo produktom elemenata* a *i* b*. Pri tome množenje u* R *zadovoljava ove uvjete (aksiome)*

1. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ *(asocijativnost)*

2. $a \cdot (b + c) = a \cdot b + a \cdot c$ *(lijevi zakon distributivnosti)*

3. $(b + c) \cdot a = b \cdot a + c \cdot a$ *(desni zakon distributivnosti)*

za sve $a, b, c \in R$

U prstenu R $a + b$ *zove se zbroj, dok se* $a \cdot b$ *zove produkt elemenata* a *i* b. *Funkcija koja uređenom paru* $a, b \in G$ *pridružuje zbroj* $a + b \in R$ *zove se zbrajanje, a funkcija koja paru* $a, b \in G$ *pridružuje produkt* $a \cdot b \in R$ *naziva se množenje u* R.
Prsten se zove komutativan ako je $a \cdot b = b \cdot a$ *za svaki par* $a, b \in R$.

Definicija 1.3.3. *Neka je* R *prsten, te neka je* $0 \neq x \in R$ *takav da vrijedi*

$$x \cdot y = 0$$

za neki $0 \neq y \in R$. *Tada* x *zovemo* ***djelitelj nule***.

Definicija 1.3.4. ***Integralna domena*** *je komutativan prsten koja nema djelitelja nule, tj.*

$$a \cdot b \Rightarrow a = 0 \quad ili \quad b = 0$$

Definicija 1.3.5. *Neka je* $a \neq 0$ *neinvertibilan element neke integralne domene. Kažemo da je* a ***ireducibilan*** *ako se ne može zapisati kao produkt dva neinvertibilna elementa.*

Definicija 1.3.6. ***Tijelom*** Φ *nazivamo prsten sa svojstvom da skup svih od nule različitih elemenata iz* Φ, *tj. skup* $\Phi \setminus \{0\}$, *obrazuje grupu s obzirom na množenje. Jedinični element te grupe označavamo s* 1.
Komutativno tijelo zove se ***polje***. *Ako su* $\mathbb{K}$ *i* $\mathbb{L}$ *polja, takva da je* $\mathbb{K} \subseteq \mathbb{L}$, *onda kažemo da je* $\mathbb{K}$ *potpolje od* $\mathbb{L}$.

Definicija 1.3.7. ***Vektorskim prostorom*** *nad tijelom* Φ *zovemo aditivnu Abelovu grupu* $X = \{x, y, \ldots\}$, *u kojoj definiramo množenje s elementima iz* Φ, *tj. za svaki par* $x \in X$ *i* $\lambda \in \Phi$ *definiramo je* λx *i to je element iz* X. *Pri tome vrijedi:*

$$\begin{array}{rcll} \alpha \cdot (x + y) & = & \alpha \cdot x + \alpha \cdot y & (\alpha \in \Phi; x, y \in X) \\ (\alpha + \beta) \cdot x & = & \alpha \cdot x + \beta \cdot y & (\alpha, \beta \in \Phi; x \in X) \\ \alpha \cdot (\beta \cdot x) & = & (\alpha \cdot \beta) \cdot x & (\alpha, \beta \in \Phi; x \in X) \\ 1 \cdot x & = & x & (x \in X) \end{array}$$

Definicija 1.3.8. ***Karakteristika polja*** $\mathbb{K}$ *je najmanji prirodan broj n takav da je*

$$1 + 1 + \ldots + 1 = n \cdot 1 = 0$$

gdje su 0 i 1 *neutralni elemnti za zbrajanje, odnosno množenje u* $\mathbb{K}$*. Ako je* $n \cdot 1 \neq 0$ *za svaki prirodan broj n, tada kažemo da je polje karakteristike 0.*

Definicija 1.3.9. *Polje koje ima konačan broj elemenata nazivamo* ***konačno polje*** *ili* ***Galoisovo polje****. Konačno polje sa q elemenata označava se sa* $\mathbb{F}_q$ *ili sa* $GF(q)$*.*

Lema 1.3.10. *Svaka konačna integralna domena čini polje.*

Dokaz. Za dokaz ove tvrdnje treba pokazati da za svaki ne-nul element konačne integralne domene postoji multiplikativni inverz. Neka je $a \neq 0$ element konačne integralne domene. Promotrimo niz $a, a^2, a^3, \ldots$. Prema pretpostavci integralna domena ima konačan broj elemenata, tako za neke $m, n \in \mathbb{N}$ takve da je $m < n$ vrijedi:

$$a^m = a^n, \quad a^m \neq 0$$

te neka su m i n najmanji za koje ta jednakost vrijedi. Tada je:

$$0 = a^m - a^n = a^m(1 - a^{n-m})$$

Kako je $a^m \neq 0$, a ujedno i element integralne domene, to znači da je $0 = 1 - a^{n-m}$, tj.

$$1 = a^{n-m} = a \cdot a^{n-m-1}$$

Pa a ima multiplikativni inverz, tj. konačna integralna domena je polje. □

Definicija 1.3.11. *Cijeli broj b djeljiv je cijelim brojem a* ($a \neq 0$) *(odnosno a dijeli b), ako postoji cijeli broj x takav da je* $b = a \cdot x$*, te to zapisujemo kao* $a \mid b$*. U slučaju da b nije djeljiv sa a, to zapisujemo sa* $a \nmid b$*.*

Definicija 1.3.12. *Prirodan broj* $n > 1$ *zove se prost broj (ili prim broj) ukoliko je djeljiv samo sa brojem 1 i samim sobom. Broj koji nije prost naziva se složen broj.*

Propozicija 1.3.13. *Neka je* $\mathbb{F}_k$ *konačno polje. Tada je karakteristika polja p prost broj.*

Dokaz. Pretpostavimo suprotno, neka je karakteristika polja p složen broj, tj. $p = a \cdot b$ gdje su $a, b < p$, pa vrijedi:

$$a \cdot b = p = \underbrace{1 + 1 + \ldots + 1}_{\text{p puta}} = \underbrace{(1 + 1 + \ldots + 1)}_{\text{a puta}} \cdot \underbrace{(1 + 1 + \ldots + 1)}_{\text{b puta}} = 0$$

jer je p karakteristika polja. Iz $a \cdot b = 0$ proizlazi da je $a = 0$ ili $b = 0$, što je u kontradikciji sa pretpostavkom da je p karakteristika polja.
Dodatno, potrebno je dokazati da zaista vrijedi:

$$a \cdot b = 0 \Rightarrow a = 0 \quad \text{ili} \quad b = 0$$

Pa pretpostavimo da tvrdnja ne vrijedi, tj. neka vrijedi:

$$a \cdot b = 0,\ a \neq 0, b \neq 0$$

S obzirom da su elementi a i b različiti od nule, tada prema definiciji postoje njihovi inverzi s obzirom na množenje (a^{-1} i b^{-1}). Ako gornju jendadžbu pomnožimo s njima dobit ćemo:

$$1 = 1 \cdot 1 = a^{-1} \cdot a \cdot b \cdot b^{-1} = a^{-1} \cdot 0 \cdot b^{-1} = 0$$

Iz čega proizlazi da je jedinica za množenje i zbrajanje jednaka, pa je tako i jedini element polja, što je u kontradikciji s definicijom polja. (Polje sadrži barem dva različita elemenata). □

Propozicija 1.3.14. *Neka je $\mathbb{F}_q$ konačno polje karakteristike p. Tada $\mathbb{F}_q$ sadrži potpolje $\mathbb{F}_p$ i nad njime čini vektorski prostor.*

Dokaz. Kako je $\mathbb{F}_q$ konačno polje iz propozicije 1.3.13 slijedi da je karakteristika polja $\mathbb{F}_q$ prost broj, tj. p je prost broj. Promatrajmo sada skup:

$$\mathbb{F}_p = \{0, 1_{\mathbb{F}}, 1_{\mathbb{F}} + 1_{\mathbb{F}}, \ldots, \underbrace{1_{\mathbb{F}} + 1_{\mathbb{F}} + \ldots + 1_{\mathbb{F}}}_{(p-1)puta}\} = \{0 \cdot 1_{\mathbb{F}}, 1 \cdot 1_{\mathbb{F}}, \ldots, (p-1) \cdot 1_{\mathbb{F}}\}$$

Uočimo da je $\mathbb{F}_p$ izomorfan skup skupu $\mathbb{Z}_p$ (izomorfizam može biti funkcija identiteta, gdje zbrajanje i množenje na $\mathbb{F}_p$ definiramo analogno kao i na $\mathbb{Z}_p$). Kako je $\mathbb{Z}_p$ polje, tako je i $\mathbb{F}_p$ polje, pa samim time i potpolje od $\mathbb{F}_q$. Odavdje se može zaključiti kako $\mathbb{F}_q$ čini vektorski prostor nad $\mathbb{F}_p$ s obzirom da je $\mathbb{F}_p \subseteq \mathbb{F}_q$ (zaključuje se na temelju definiranosti svih operacija i zadovoljavanju svih uvjeta, sve se naslijeđuje iz polja $\mathbb{F}_q$). □

Napomena 1.3.15. *Nadalje, ako dimenziju od $\mathbb{F}_q$ kao vektorskog prostora označimo sa n, sada je jasno da $\mathbb{F}_q$ ima točno p^n elemenata, tj. $q = p^n$. Općenito za bilo koji prosti broj p i bilo koji prirodan broj n postoji jedinstveno (do na izomorfizam) konačno polje s p^n elemenata. Jedna realizacija takvog polja je $\mathbb{Z}_p[x]/f(x)$ gdje je $f(x)$ ireducibilan polinom n-tog stupnja nad $\mathbb{Z}_p[x]$.*

Korolar 1.3.16. *$Z_p[x]/f(x)$ je polje.*

Dokaz. $Z_p[x]/f(x)$ je skup koji sadrži polinome stupnja $n-1$ ili manjeg unutar kojeg je zbrajanje $(+)$ definirano kao obično zbrajanje polinoma (koeficijenti se zbrajaju kao u polju $\mathbb{Z}_p$), dok je množenje $(\bullet)$ definirano kao obično množenje polinoma modulo $f(x)$, tj kao rezultat se gleda ostatak pri djeljenju s polinomom $f(x)$.
Skup $Z_p[x]/f(x)$ je konačan jer su polinomi nad konačnim poljem $\mathbb{Z}_p$ i do stupnja $n-1$. Također je i integralna domena, jer za bilo koja dva proizvoljna polinoma $a(x), b(x) \in Z_p[x]/f(x)$ vrijedi

$$a(x) \bullet b(x) \equiv 0 \bmod f(x) \rightarrow a(x) = 0 \quad \text{ili} \quad b(x) = 0$$

Kada bi vrijedilo suprotno, to bi značilo da postoje polinomi $a(x)$ i $b(x)$ stupnja manjeg od $n-1$, nad poljem $\mathbb{Z}_p$ takvi da vrijedi:

$$a(x) \cdot b(x) = f(x)$$

tj. polinom $f(x)$ ne bi bio ireducibilan, što je pretpostavka. Pa je $Z_p[x]/f(x)$ i integralna domena.
Prema lemi 1.3.10 $Z_p[x]/f(x)$ polje. □

Definicija 1.3.17. *Neka je K polje karakteristike različite od 2 i 3. Eliptička krivulja E nad poljem K je skup svih točaka* $(x, y) \in K \times K$ *koje zadovoljavaju uvjet:*

$$y^2 = x^3 + ax + b$$

zajedno s još jednim elementom koji zovemo "točka u beskonačnosti" i označavamo sa $\mathcal{O}$*. Također koeficijenti* a, b *su elementi polja K i zadovoljavaju uvjet* $4 \cdot a^3 + 27 \cdot b^2 \neq 0$ *što je ekvivalentno uvjetu da polinom* $x^3 + ax + b$ *nema višestrukih nultočaka.*

Definicija 1.3.18. *Neka je E eliptička krivulja nad poljem K. Definiramo binarnu operaciju zbrajanja (u oznaci +) nad točkama skupa E prema sljedećim pravilima:*

1. $P + \mathcal{O} = \mathcal{O} + P = P$ *za svaki* $P \in E$,
2. *Ako je* $P = (x, y) \in E$, *tada je* $(x, y) + (x, -y) = \mathcal{O}$. *Točka* $(x, -y)$ *označava se i sa* $-P$. *Primjetimo da ako je točka* $P \in E$, *tada vrijedi i da je* $-P \in E$.
3. *Neka je* $P = (x_1, y_1) \in E$ *i* $Q = (x_2, y_2) \in E$ *gdje* $P \neq -Q$, *tada je* $P + Q = (x_3, y_3)$ *gdje je*

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}$$

te je

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

Neka je $P \in E$ i k neki cijeli broj. Sa $k * P$ ili $[k]P$ označavati ćemo sljedeću sumu:

$$k * P = [k]P = \underbrace{P + P + \ldots + P}_{k-puta}$$

Skup $(E, +)$, tj. skup E s binarnom operacijom $+$ nad njezinim elementima čini Abelovu grupu. Iz definicije zbrajanja je jasna egzistencija neutralnog elementa i inverza. Kako se zbrajanje svodi na operacije iz polja K, tako su asocijativnost i komutativnost zbrajanja naslijeđena iz operacija polja K.

Definicija 1.3.19. *Neka su a i b cijeli brojevi različiti od nule. Neka je c cijeli broj. Broj c zovemo zajednički djelitelj brojeva a i b ako vrijedi $c \mid a$ i $c \mid b$. Nadalje c zovemo najveći zajednički djelitelj od a i b, ako je c najveći cijeli broj koji ih djeli (u oznaci $c = (a, b)$ ili $c = \mathrm{nzd}(a, b)$).*

Definicija 1.3.20. *Dva prirodna broja n i m zovemo relativno prostim brojevima ukoliko im je najveći zajednički djelitelj broj 1.*

Definicija 1.3.21. *Neka je funkcija $\varphi : \mathbb{N} \to \mathbb{N}$ dana sljedećim pravilom:*

$$\varphi(n) = |\{a \in \{1, 2, \ldots, n-1\} \mid nzd(a, n) = 1\}|$$

Funkcija φ zove se Eulerova funkcija.

Propozicija 1.3.22. *Eulerova funkcija zadovoljava sljedeća svojstva:*

(a) Neka je p prost broj. Tada vrijedi:

$$\varphi(p) = p - 1$$

(b) Neka je $n = p \cdot q$, gdje su p i q prosti brojevi. Tada vrijedi:

$$\varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$$

Dokaz. (a) slijedi direktno iz definicije prostog broja.
Kako bi dokazali (b) uočimo da zato što su p i q prosti brojevi, jedini brojevi koji nisu relativno prosti s brojem n biti će višekratnici brojeva p i q manji od n, a to su redom:

$$0, 1 \cdot p, 2 \cdot p, \ldots, (q-1) \cdot p, 1 \cdot q, 2 \cdot q, \ldots, (p-1) \cdot q$$

te ih je točno $1 + (q-1) + (p-1) = p + q - 1$. Odavdje se vidi da relativno prostih brojeva sa n, a koji su manji od n ima točno $n - (p + q - 1) = p \cdot q - p - q + 1 = (p-1) \cdot (q-1)$, pa je

$$\varphi(n) = (p-1) \cdot (q-1) = \varphi(p) \cdot \varphi(q)$$

□

Definicija 1.3.23. *Ako cijeli broj m,* $(m \neq 0)$ *dijeli razliku* $a-b$*, kažemo da je a kongruentan b modulo m i zapisujemo to* $a \equiv b(\mathrm{mod} m)$*. Ako* $a - b$ *nije djeljivo sa m, tada kažemo da a nije kongruentan b modulo m i zapisujemo to* $a \not\equiv b(\mathrm{mod} m)$

Poglavlje 2

Simetrični Kriptosustavi

2.1 DES

DES (*eng. Data Encryption Standard*, *hrv. standard šifriranja podataka*) najpoznatija je simetrična blokovna šifra. Razvijana je prvom dijelu 1970-ih godina unutar tvrtke IBM. Algoritam je službeno objavljen 1976. godine kao dio FIPS-a (*eng. Federal Information Processing Standards*), tj. kao standard razvijen od američke vlade na području računalnih znanosti. Dizajn DES-a povezuje se sa dva opća koncepta, produktne šifre i Feistelove šifre. Osnovna ideja produktne šifre je izgradnja kompleksnije funkcije šifriranja kombiniranjem jednostavnih šifara koje pojedinačno ne nude dovoljno sigurnosti. Pod osnovne operacije misli se na transpoziciju, aritmetičke operacije, modularno množenje, `xor` (isključivo ili) operaciju i supstituciju. Horst Feistel kao dio IBMovog tima razvijao je DES, te je osmislio algoritam koji se naziva Feistelova šifra i koji se osim u DES-u koristi u nekim drugim simetričnim blokovnim šiframa kao osnovni koncept.

Napomene

U ovom poglavlju tablicama permutacija i tablicom odabira biti će prikazane permutacije i preslikavanje. Kako notacija nije standardna u ovom potpoglavlju će biti opisana.
Neka je funkcija **X** zadana sljedećom tablicom:

X			
$x_{0,1}$	$x_{1,2}$	$\dots$	$x_{1,n}$
$x_{1,1}$	$x_{2,2}$		
	$\vdots$	$\ddots$	
$x_{m-1,1}$	$x_{m-1,2}$	$\dots$	$x_{m-1,n}$

Tada je funkcija $\mathbf{X} : \{1, 2, \ldots, m \cdot n\} \to \{1, 2, \ldots, m \cdot n\}$ definirana sa:

$$X(k) = x_{i,j} \quad \left\{ \begin{array}{l} k = i * n + j \\ i \in \{0, 1, \ldots, m-1\} \\ j \in \{1, 2, \ldots, n\} \end{array} \right\}$$

Neka je $B = (b_1 b_2 \ldots b_l)$ blok bitova. Tada je $\mathbf{X}(B) = (b_{\mathbf{X}(1)} b_{\mathbf{X}(2)} \ldots b_{\mathbf{X}(m \cdot n)})$, no postoje 3 slučaja:

- $l = m \cdot n$
  Svakom bitu je pridružena njegova vrijednost.

- $l > m \cdot n$
  Kardinalni broj domene funkcije $\mathbf{X}$ je manji od broja bitova, pa se višak bitova odbacuje.

- $l < m \cdot n$
  Kardinalni broj domene funkcije $\mathbf{X}$ je veći od broja bitova, pa se duljina bloka proširuje do $m \cdot n$ bitova. To vrijedi samo ako je funkcija definirana tako da za svaki y iz domene funkcije $\mathbf{X}$ vrijedi $\mathbf{X}(y) \in \{1, 2, \ldots, l\}$

Neka su $A = (a_1 a_2 \ldots a_p)$ i $C = (c_1 c_2 \ldots c_q)$ blokovi bitova jednake duljine ($p = q$). Tada je sa $A \oplus B$ definirano *bit-po-bit* zbrajanje modulo 2 (u bazi 2), tj. zbrajanje u bazi 2 sa zanemarivanjem ostatka (ekvivalentno operaciji `xor`).

Feistelova šifra

Feistelova šifra iterativna je šifra koja mapira blok otvorenog teksta duljine $2t$ bita u šifrat. Otvoreni tekst podijeljen je u dvije grupe (bloka) (L_0, R_0) gdje svaki od blokova, lijevi blok L_0 i desni blok R_0, ima po t bita. 2 bloka otvorenog teksta (L_0, R_0) šifriraju se u šifrat (R_r, L_r) kroz iterativni postupak od r runda gdje je $r \geq 1$. Za svaki $1 \leq i \leq r$, tijekom i-te runde mapira se $(L_{i-1}, R_{i-1}) \mapsto (L_i, R_i)$, na sljedeći način:

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f(R_{i-1}, K_{i-1})$$

gdje je svaki potključ K_i dobiven iz ključa K. Nakon što se završi izvršavanje svih r rundi Feistelove šifre dobiva se blok (L_r, R_r), no zbog lakšeg dešifriranja, tj. analognog postupka šifriranju s ključevima u obrnutom redoslijedu kod rezultata mijenja se poredak blokova u (R_r, L_r), te se taj blok uzima kao konačni rezultat Feistelove šifre.

Šifriranje

DES je Feistelova šifra koja kao ulaz prima otvoreni tekst (jedan blok otvorenog teksta) i ključ. Duljina bloka otvorenog teksta i ključa je $n = 64$ bita, a kao izlaz algoritam daje šifrat također duljine 64 bita. Ključ K je dugađak 64 bita, no njegov efektivni dio, tj. dio koji se koristi u algoritmu je duljine 56 bita. Preostalih 8 bitova (8,16,...,56,64) mogu se koristiti kao kontrolni bitovi.
Neka je sa $O = (o_1 o_2 \dots o_{64})$ označen blok otvorenog teksta koji treba šifrirati. Tok algoritma je sljedeći:

- **Početna permutacija:**
  Bitovi otvorenog teksta ispermutiraju se početnom permutacijom **PP**.

$$\mathbf{PP}(O) = (o_{\mathbf{PP}(1)} \dots o_{\mathbf{PP}(64)})$$

- **16 runda Feistelove šifre:**
  Nakon permutiranja bloka, blok se dijeli na dva početna bloka (L_0, R_0), gdje je $L_0 = (o_{\mathbf{PP}(1)} \dots o_{\mathbf{PP}(32)})$, a $R_0 = (o_{\mathbf{PP}(33)} \dots o_{\mathbf{PP}(64)})$, koji služe kao ulaz u Feistelovu šifru, gdje se zatim u svakoj rundi ponavlja sljedeći postupak (u oznakama za i-tu rundu):

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_{i-1}) \end{aligned}$$

  Funkcija f ima dva ulaza, prvi je desni od dobivenih blokova iz prošle runde (duljine 32 bita), dok je drugi 48-bitni podključ K_i generiran za i-tu rundu. Funkcija f kao izlaz daje niz od 32 bita. Kako se generiraju podključevi za svaku rundu i što točno radi funkcija f objašnjeno je u sljedećim potpoglavljima. Nakon 16 rundi Feistelove šifre i mijenjanja poretka blokova dobivamo rezultat (R_{16}, L_{16})

- **Inverz početne permutacije:**
  Blok (R_{16}, L_{16}) ispermutira se inverzom početne permutacije $\mathbf{PP}^{-1}$ ($\mathbf{PP}^{-1}(R_{16}L_{16})$) te se dobiva šifrat bloka otvorenog teksta.

permutacije **PP** i $\mathbf{PP}^{-1}$ definirane su tablicama koje su dane na slici 2.1.

Generiranje podključeva

Neka je sa $K = (k_1 k_2 \dots k_{64})$ označen početan ključ, te neka su sa $K_i = (k_{i,1} k_{i,2} \dots k_{i,48})$ označeni podključevi koji se generiraju za svaku rundu iz početnog ključa K, $i \in \{1, \dots, 16\}$ s obzirom da algoritam ima 16 runda. Postupak generiranja je gotovo isti za svaku rundu, a samo generiranje ovisi o dvije permutacije. Permutaciji izbora 1 (**PI1**) i permutaciji izbora 2 (**PI2**) koje su definirane tablicama na slici 2.2

PP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

PP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Slika 2.1: Tablice permutacije

PI1

C	57	49	41	33	25	17	9
	1	58	50	42	34	26	18
	10	2	59	51	43	35	27
	19	11	3	60	52	44	36
D	63	55	47	39	31	23	15
	7	62	54	46	38	30	22
	14	6	61	53	45	37	29
	21	13	5	28	20	12	4

PI2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Slika 2.2: Tablice permutacije izbora

Permutacija **PI1** podijeljena je na dva dijela (**C** i **D**) te ima 56 elemenata. Kao što je već spomenuto efektivni dio ključa nije 64 bita, već njih 56, pa se pomoću **PI1** eliminiraju točno ti suvišni, kontrolni, bitovi (primjetimo da $\mathbf{PI1}(y) \notin \{8, 16, 24, 32, 40, 48, 56, 64\}$). U svakoj rundi generiraju se nova dva bloka (C_i, D_i) gdje je svaki od njih duljine 28 bita. Označimo sa $C_i = (c_{i,1}c_{i,2}\ldots c_{i,27}c_{i,28})$ gdje je $c_{i,j}$ j-ti bit u i-toj rundi bloka. Analogno definiramo D_i. Kao početno stanje definiramo $C_0 = (k_{\mathbf{PI1}(1)}\ldots k_{\mathbf{PI1}(28)})$ i $D_0 = (k_{\mathbf{PI1}(29)}\ldots k_{\mathbf{PI1}(56)})$. Blokove C_1 i D_1 dobivamo sa cikličkim pomakom bitova za jedno mjesto ulijevo blokova C_0 i D_0, respektivno, tj.

$$\begin{aligned} C_1 &= (c_{1,1}c_{1,2}\ldots c_{1,27}c_{1,28}) = (c_{0,2}c_{0,3}\ldots c_{0,28}c_{0,1}) \\ D_1 &= (d_{1,1}d_{1,2}\ldots d_{1,27}d_{1,28}) = (d_{0,2}d_{0,3}\ldots d_{0,28}d_{0,1}) \end{aligned}$$

Analogno vrijedi za svaku rundu (C_i, D_i) dobiva se iz (C_{i-1}, D_{i-1}) cikličkim pomakom bitova ulijevo. U svakoj rundi rade se dva pomaka ulijevo, osim u 1., 2., 9. i 16. rundi kada

se radi jedan pomak ulijevo. Kada izgeneriramo blokove C_i i D_i za i-tu rundu, gledamo na njih kao na cjelinu, tj. na blok $H = h_1h_2 \dots h_{55}h_{56}$. Podključ K_i dobivamo iz bloka H i permutacije izbora 2

$$K_i = \mathbf{PI2}(H) = (h_{\mathbf{PI2}(1)}, \dots, h_{\mathbf{PI2}(56)}) = (k_{i,1}k_{i,2} \dots k_{i,48})$$

Fukcija f

Funkcija f u i-toj rundi kao parametre prima R_i i K_i gdje je R_i blok od 32 bita, dok je K_i blok od 48 bita. Ulaskom u funkciju prvo se blok R_i proširi na 48 bita pomoću preslikavanja **E**. Funkcija odabira **E** dana je tablicom na slici 2.3

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Slika 2.3: Tablice odabira bitova **E**

$\mathbf{E}(R_i) = r_{i,1} \dots r_{i,48}$. Nakon proširivanja bloka R_i, prošireni blok zbrajamo sa podključem generiranim za i-tu rundu i dobivamo novi blok od 48 bita. Taj blok dijeli se na 8 blokova od po 6 bita koje označavamo sa B_j.

$$B = B_1B_2 \dots B_8 = \mathbf{E}(R_i) \oplus K_i$$

Svaki od blokova B_j proslijeđujemo $\mathbf{S}_j$-kutiji. Svaka **S**-kutija može se smatrati funkcijom koja prima 6 bita i vraća 4 bita. Svaka **S**-kutija definirana je matricom dimenzija 4×16 koja sadrži elemente iz skupa $\{0, 1, \dots, 15\}$. Primjer jedne **S**-kutije dan je na slici 2.4

Neka je $B = (b_1, e_1, e_2, e_3, e_4, b_2)$ proizvoljan blok od 6 bita koji ulazi u neku **S**-kutiju. Bitove iz B grupiramo u dvije grupe , tj. zapišemo kao uređen par $(b_1b_2, e_1e_2e_3e_4)$. Uvijek grupiramo prvi i zadnji bit, te srednja 4. b_1b_2 u bazi 2 čine broj između 0 i 3, dok $e_1e_2e_3e_4$ u bazi 2 čine broj između 0 i 15. Ako redove u promatranoj **S**-kutiji numeriramo redom sa 0,1,2,3, a stupce redom sa $0, 1, \dots, 15$ tada bi uređen par $(b_1b_2, e_1e_2e_3e_4)$ označavao točno jedan redak i točno jedan stupac, tj. jedan element promatrane **S**-kutije. Kako je već navedeno u **S**-kutijama nalaze se elementi iz skupa $\{0, 1, \dots, 15\}$ koji se svi mogu zapisati u 4

$\mathbf{S}_1$															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Slika 2.4: **S**-kutija

bita. Taj element, tj. tih 4 bita su izlaz iz **S**-kutije za ulaznih 6 bita.

Svaka od 8 **S**-kutija vraća 4 bita što je ukupno 32 bita. Označimo sa B'_j 4-bitni blok koji vraća $\mathbf{S}_j$-kutija. Nakon **S** kutija dobiva se blok $B' = B'_1 B'_2 \dots B'_8$. Dobiveni B' ispermutiramo prema tablici **P** sa slike 2.5, te se taj blok $\mathbf{P}(B')$ vraća kao izlaz funkcije f.

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Slika 2.5: Tablica permutacije **P**

Navedenim postupkom opisana je jedna runda Feistelove šifre unutar DES-a. Nakon 16 runda DES završava, te se za ulazni blok otvorenog teksta dobiva blok šifrata.

Dešifriranje

Dešifriranje DES-a potpuno je analogno šifriranju s jedinom razlikom u obrnutom rasporedu podključeva. Šifrirani blok stavimo kao ulaz u algoritam, te kao ključ u prvoj rundi stavlja se K_{16}, u drugoj rundi kao ključ stavlja se K_{15},... dok se u zadnjoj rundi kao ključ stavlja K_1. Kao izlaz dobiva se blok otvorenog teksta od kojeg je dobiven šifrirani blok.

2.2 AES

1998. godine NIST (*eng. National Institute of Standards and Technology*) objavio je otvoreni natječaj za novu blokovni algoritam koji bi se zvao AES (*eng. Advanced Encryption Standard*). Zahtjevi na prijavljene algoritme bili su:

- Veličina bloka (otvorenog teksta) mora biti 128 bita.
- Algoritam po mogućnosti treba raditi sa varijabilnim veličinama ključa (128,192 i 256 bita).
- Algoritam treba biti brži od trostrukog DES-a.

Očito je da se natječajem željela stvoriti sigurnija i brža šifra od DES-a (T-DES-a). Iz tog razloga je i jedan od zahtjeva bio da je ključ veći nego kod DES-a, te da veličina ključa bude varijabilna, kako bi se u budućnosti od napada provjere cijelog prostora ključeva branilo samo povećavanjem tog prostora, tj. povećanjem duljine ključa. Također zbog veličine ključa u DES-u (56 bita), DES od njegovog uvođenja nije smatran za potpuno siguran algoritam (postojale su spekulacije da neke organizacije imaju dovoljno *jaka* računala s kojima mogu u razumnom vremenu provjeriti prostor svih ključeva ili da su poznavale neki drugi brz i efikasan napad na DES). Natječaj za AES je bio javan baš iz razloga kako bi se takve sumnje odbacile, a također kako bi svi mogli sudjelovati (smatralo se kako bi se povećavanjem konkurencije kvaliteta natjecanja također trebala povećati).
2000. godine kao pobjednički algoritam odabran je *Rijndael* algoritam dvojice belgijskih kriptografa (Joan Daemen i Vincent Rijmen). Kako duljina ključa varira između 128, 192 i 256 bita, tako se za svaku duljinu ključa algoritam drukčije označava, AES-128,AES-192 i AES-256.

Napomene

U uvodu u matematičke pojmove pokazano je da je $\mathbb{Z}_p[x]/f(x)$ polje. Kako se u AES algoritmu promatraju bajtovi (nizovi od osam bitova), njih se promatra kao polje polinoma s 8 članova, nad $\mathbb{Z}_2$ (svaki član polinoma predstavlja jedan bit iz bajta). Točnije promatra se polje $GF(2^8)$ kao reprezentacija pomoću polinomijalne baze gdje je kao ireducibilni polinom $f(x)$ uzet:

$$f(x) = x^8 + x^4 + x^3 + x + 1$$

Neka je b proizvoljan bajt određen sa $b = (b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8)$ (vodeća znamenka je b_1), njega ćemo nadalje zapisivati i u obliku $\{b_1b_2b_3b_4b_5b_6b_7b_8\}$. Također, 4 bita u heksadecimalnom zapisu su prikazana jednim znakom, pa ćemo često i taj bajt zapisivati

u heksadecimalnom obliku $\{h_1h_2\}$ gdje je h_1 heksadecimalna znamenka određena sa grupom bitova $b_1b_2b_3b_4$ (b_1 je vodeća znamenka), a h_2 heksadecimalna znamenka određena sa grupom bitova $b_5b_6b_7b_8$ (b_5 je vodeća znamenka). Naprimjer bajt $\{11000101\}$ zapisivat ćemo i kao $\{c5\}$. Nadalje svaki polinom iz promatranog polja određen je sa 8 bita, tj. jednim bajtom, pa ćemo te polinome u nekim slučajevima zapisivati pomoću pokazanog heksidecimalnog prikaza. Naprimjer, polinom $x^7 + x^6 + x^3 + x + 1$ možemo shvatiti kao $(1, 1, 0, 0, 1, 0, 1, 1) = \{cb\}$. Tako množenje polinoma možemo zapisati i pomiću te notacije, pa ćemo $(x^7 + x^6 + x^3 + x + 1) \cdot (x^2 + 1)$ zapisivati i kao $\{cb\} \bullet \{05\}$.
Također promatrat će se riječi (nizovi od 32 bita ili 4 bajta), tj. operacije nad 4 bajta koji će biti zapisani u obliku $[a_0, a_1, a_2, a_3]$ gdje je svaki od bajta koeficijent četveročlanog polinoma $a_3x^3 + a_2x^2 + a_1x + a_0$. Zbrajanje tih polinoma definirano je kao obično zbrajanje polinoma (koeficijenti uz članove istih potencija se zbrajaju, što su u ovom slučaju elementi polja $GF(2^8)$, pa se zbrajaju bit-po-bit u bazi 2). Množenje je također definirano kao obično množenje polinoma, s time da rezultat može biti polinom stupnja većeg od 3, zbog toga rezultat množenja se reducira, te se gleda ostatak rezultata pri djeljenju s polinomom $x^4 + 1$. Polinom $x^4 + 1$ nije ireducibilan, što znači da množenje nije nužno invertibilno (ne formira se polje, pa nema svaki element inverz), no polinom s kojim će se množiti u algoritmu biti će invertibilan, pa je taj mogući problem zaobiđen.
Ako su $a(x)$ i $b(x)$ polinomi 3 stupnja te je njihov modularni produkt produkt (u oznaci $a(x) \otimes b(x)$) označen sa $d(x)$ tada je $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$ gdje su:

$$\begin{aligned}
d_0 &= (a_0 \bullet b_0) \otimes (a_3 \bullet b_1) \otimes (a_2 \bullet b_2) \otimes (a_1 \bullet b_3) \\
d_1 &= (a_1 \bullet b_0) \otimes (a_0 \bullet b_1) \otimes (a_3 \bullet b_2) \otimes (a_2 \bullet b_3) \\
d_2 &= (a_2 \bullet b_0) \otimes (a_1 \bullet b_1) \otimes (a_0 \bullet b_2) \otimes (a_3 \bullet b_3) \\
d_3 &= (a_3 \bullet b_0) \otimes (a_2 \bullet b_1) \otimes (a_1 \bullet b_2) \otimes (a_0 \bullet b_3)
\end{aligned}$$

Množenje možemo zapisati i matrično:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Šifriranje

Slikom 2.6 dan je pseudokod algoritma šifriranja, koji je u nastavku pojašnjen u detalje.

in je ulazna varijabla, tj. niz bitova (bajtova) koji predstavljaju blok otvorenog teksta, **out** je izlazna varijabla, koja predstavlja niz šifriranih bitova (šifrat). **state** je matrica stanja, te će biti objašnjenja u sljedećem potpoglavlju. $\mathbf{N}_b$ je broj riječi (niz od 32 bita) koji se šifrira (u ovom slučaju je fiksan $\mathbf{N}_b = 4$). $\mathbf{N}_r$ je broj rundi izvršavanja algoritma,

```
AES(byte in[4·Nb], byte out[4·Nb], word w[Nb·(Nr+1)] )
begin
  byte state[4,Nb]

  state = in

  AddRoundKey(state,w[0,Nb − 1])

  for round = 1 to Nr − 1
    SubBytes(state)
    ShisftRows(state)
    MixColumns(state)
    AddRoundKey(state, w[round·Nb, (round+1)Nb-1])
  endfor

  SubBytes(state)
  ShisftRows(state)
  AddRoundKey(state, w[Nr · Nb, (Nr + 1) · Nb − 1])

  out = state
end
```

Slika 2.6: Psuedokod AES algoritma

te on ovisi o duljini ključa (verziji algoritma). Iz pseudokoda vidljivo je da je glavna ideja algoritma vrlo jednostavna. Blok otvorenog teksta kopira se u pomoćnu matricu **state**, nad kojom se zatim u petlji i jednom izvan izvršavaju funkcije: **AddRoundKey()**, **SubBytes()**,**ShiftRows()** i **MixColumns()**. Funkcije su objašnjenje u sljedećim potpoglavljima.

Matrica stanja `state`

Iz algoritma je vidljivo da se sve operacije vrše nad matricom **state** koja predstavlja trenutno stanje bloka otvorenog teksta prilikom šifriranja. Matrica je dimenzija 4×4 a u svakom elementu sadržava jedan bajt. Redove matrice označavaju se sa $r \in \{0,1,2,3\}$, dok se stupci označavaju sa $c \in \{0,1,2,3\}$. Prije početka izvršavanja algoritma blok otvorenog teksta kopira se u matricu **state** iz ulaznog niza bajtova **in**, te se na kraju iz matrice **state** šifrat kopira u izlazni niz bajtova **out**. Varijablu **state** kraće ćemo označavati sa s. Prvo

kopiranje izvršava se prema sljedećem rasporedu:

$$s[r, c] = in[r + 4 \cdot c] \quad \text{za } 0 \leq r, c < 4$$

Drugo kopiranje izvršava se prema sljedećem rasporedu:

$$out[r + 4 \cdot c] = s[r, c] \quad \text{za } 0 \leq r, c < 4$$

Shemu preslikavanja možemo vidjeti na slici 2.7

Slika 2.7: Kopiranje

U svakom stupcu matrice **state** 4 byte-a formiraju 32-bitnu riječ gdje broj reda r označava svaki pojedini byte unutar riječi. Matrica stanja može biti smatrana i jednodimenzionalnim poljem 32-bitnih riječi (po stupcima), w_0, w_1, w_2, w_3 gdje su:

$$w_0 = s_{0,0}s_{1,0}s_{2,0}s_{3,0} \quad w_2 = s_{0,2}s_{1,2}s_{2,2}s_{3,2}$$
$$w_1 = s_{0,1}s_{1,1}s_{2,1}s_{3,1} \quad w_3 = s_{0,3}s_{1,3}s_{2,3}s_{3,3}$$

SubBytes() transformacija

SubBytes() transformacija je nelinearna supstitucija koja obuhvaća svaki bajt posebno koristeći **S**-kutiju (tablicu supstitucije). **S**-kutija koja je invertibilna konstruirana je kompozicijom dvije transformacije:

1. Pronalazak multiplikativnog inverza u konačnom polju $GF(2^8)$, element {00} preslikava se u samog sebe.

2. Afina transformacija (nad $GF(2)$):

$$b'_i = b_i \oplus b_{(i+4) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+6) mod 8} \oplus b_{(i+7) mod 8} \oplus c_i \quad 0 \leq i < 8$$

gdje je b_i i-ti bit bajta nad kojim se transformacija izvršava, a c_i je i-ti bit bajta c čija je vrijednost {63} ili {01100011}

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	D1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Slika 2.8: **S**-kutija

S-kutiju možemo zapisati u heksidecimalnom formatu kao tablicu koja je prikazana na slici 2.8.

Recimo da u matrici **state** imamo element $s_{r,c} = \{53\}$, to znači da bi transformacija **SubBytes()** bila određena **S**-kutijom, tj. redom u **S**-kutiji s indeksom "5" i stupcem s indeksom "3", pa bi transformacijom dobili element $s'_{r,c} = \{ed\}$. Općenito vrijedi

$$\{xy\} = s_{r,c} \xrightarrow{\texttt{SubBytes()}} s'_{r,c}$$

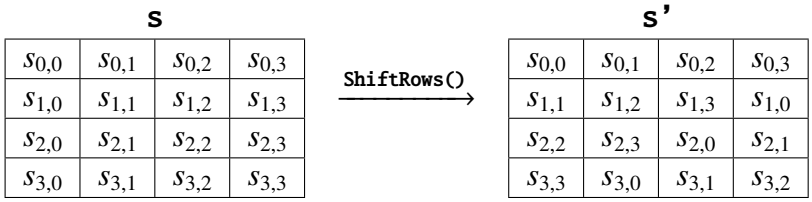
gdje je $s'_{r,c}$ određen redom x i stupcem y **S**-kutije.

ShiftRows() transformacija

Unutar **Shiftrows()** transformacije, bajtovi u zadnja 3 reda matrice **state** ciklički se pomiču (u svakom redu posebno) za različit broj pomaka ulijevo. Prvi red ostaje isti. Pomaci se izvršavaju prema formuli:

$$s'_{r,c} = s_{r,(c+r)mod4} \quad 0 \leq r, c < 4$$

To jest, svaki element se ciklički pomiče ulijevo ovisno o tome u kojem se redu nalazi. Redovi su numerirani od 0 do 3, tako da u prvom redu nema pomaka, u drugom redu svaki element se pomiče za jedno mjesto ulijevo (ciklički), u drugom redu za dva mjesta, a u trećem redu za tri mjesta. Skicu **ShiftRows()** transformacije možemo vidjeti i na slici 2.9.



Slika 2.9: **ShiftRows()** transformacija

Mixcolumns() transformacija

Transformacija **Mixcolumns()** djeluje na matricu stanja **state** tako da na svaki stupac gleda kao na četveročlani polinom s koeficijentima nad $GF(2^8)$ množeni modulo $x^4 + 1$ s fiksnim polinomom $a(x)$ danim s formulom:

$$a(x) = \{03\} \cdot x^3 + \{01\} \cdot x^2 + \{01\} \cdot x + \{02\}$$

Kao što je već pojašnjeno to množenje $s'(x) = a(x) \otimes s(x)$ možemo zapisati kao matrično množenje:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Kao rezultat množenja 4 bajta u stupcu zamijenjeni su sljedećim

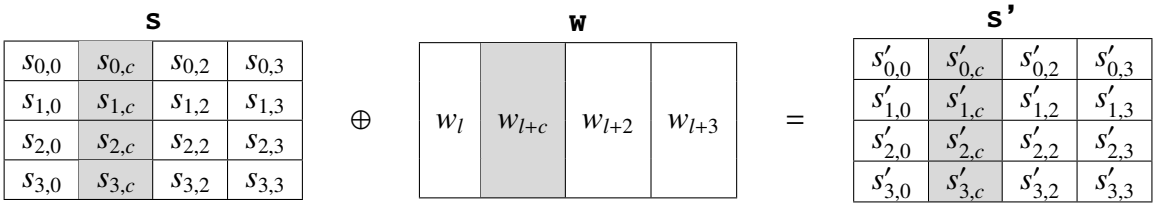
$$\begin{aligned} s'_{0,c} &= (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} &= s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c}) \\ s'_{3,c} &= (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}) \end{aligned}$$

AddRoundKey() transformacija

Unutar **AddRoundKey()** funkcije ključ runde (potključ) i riječi iz matrice stanja zajedno čine bitovnu `xor` operaciju (*zbrajaju se*) te se tako dobiva nova promijenjena matrica stanja. Riječi se zbrajaju prema sljedećem pravilu:

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{4 \cdot round+c}] \quad \text{za } 0 \leq c < 4$$

gdje je sa $[w_i]$ označena i-ta riječ ključa (objašnjeno u sljedećem potpoglavlju), dok je varijabla *round* vrijednost iz skupa $\{0, 1, 2, \ldots N_r\}$. Skica izvršavanja **AddRoundKey()** funkcije može se vidjeti na slici 2.10 gdje je $l = \mathbf{N_b} \cdot round$.



Slika 2.10: **AddRoundKey()** transformacija

Proširivanje ključa

AES algoritam za generiranje potključeva za svaku rundu izvršavanja uzima kao ulaz početni ključ K i koristi ga za proširivanje te tako generira $\mathbf{N_b} \cdot (\mathbf{N_r} + 1)$ 32-bitne riječi. Algoritam na ulazu zahtjeva $\mathbf{N_b}$ riječi, te svaka runda zahtjeva $\mathbf{N_b}$ riječi koje čine jedan potključ. Rezultat je linearno polje w riječi (duljine 4 bajta) koje čine sve potključeve, te je i-ta riječ označena sa $[w_i]$, gdje je i u rasponu $0 \leq i < \mathbf{N_b} \cdot (\mathbf{N_r} + 1)$

Proširivanje početnog ključa u potključeve za sve runde AES algoritma opisano je pseudokom na slici 2.11. U nastavku pseudokod je objašnjen u detalje.

Funkcija **SubWord()** je funkcija koja kao ulaz prima riječ duljine 4 bajta koji pojedinačno prolaze kroz **S**-kutiju (slika 2.8) te kao izlaz daje novu riječ od 4 bajta. Funkcija **RotWord()** kao ulaz prima riječ $[a_0, a_1, a_2, a_3]$ te nad njom izvodi cikličku permutaciju te vraća riječ $[a_1, a_2, a_3, a_0]$. Konstanta **i**-te runde **Rcon[i]** sadržava vrijednost danu sa $[\{02\}^{i-1}, \{00\}, \{00\}, \{00\}]$. Množenje sa $\{02\}$ na razini bajtova može biti implementirano kao pomak ulijevo te zatim zbrajanje bit-po-bit (`xor`) sa $\{1b\}$. Iz pseudokada može se vidjeti kako se točno proširuje ključ do potrebne duljine. Prvih $\mathbf{N_k}$ (broj riječi u ključu) riječi proširenog ključa puni se sa početnim ključem, dok je svaka sljedeća riječ **w**[**i**] jednaka **w[i] xor w[i - N$_k$]**. Riječi koje se nalaze na pozicijama koje su višekratnici

```
KeyExpansion(byte key[4·Nk], word w[Nb·(Nr + 1)], Nk)
begin
  word temp

  i = 0

  while(i < Nk)
    w[i] = word(key[4·i], key[4·i + 1], key[4·i + 2], key[4·i + 3])
    i = i + 1
  end while

  i = Nk

  while(i < Nb·(Nr + 1))
    temp = w[i-1]
    if i mod Nk = 0
      temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
    else if (Nk > 6 and i mod Nk = 4)
      temp = SubWord(temp)
    end if
    w[i] = w[i - Nk] xor temp
    i = i + 1
  end while
end
```

Slika 2.11: Psuedokod proširivanja ključa

od N_k transformiraju se prije opisane **xor** operacije. Prvo se nad njima izvrši funkcija **RotWord()**, pa zatim **SubWord()**, te se zatim nad tom transformiranom riječi još izvrši operacija **xor** sa konstantom runde **Rcon[i]**. Primjetimo da postoji mala razlika kod proširivanja ključa kada koristimo algoritam AES-256.

Dešifriranje

Postupak dešifriranja AES algoritma dan je pseudokom na slici 2.12, a svodi se na invertiranje transformacija korištenih prilikom šifriranja, s razlikom u rasporedu. Inverzne transformacije primjenjuju se u obrnutom redoslijedu s potključevima u obrnutom redoslijedu. Inverzne transformacije ukratko su opisane u sljedećem potpoglavlju.

```
InvAES(byte in[4 · Nb], byte out[4 · Nb], word w[Nb · (Nr+1)])
begin
  byte state[4,Nb]

  state = in

  AddRoundKey(state,w[Nr · Nb,(Nr + 1) · Nb-1])

  for round = Nr - 1 downto 1
    InvShiftRows(state)
    InvSubBytes(state)
    AddRoundKey(state, w[round·Nb, (round+1)Nb-1])
    InvMixColumns(state)
  end for

  InvShiftRows(state)
  InvSubBytes(state)
  AddRoundKey(state, w[0, Nb − 1]

  out = state
end
```

Slika 2.12: Psuedokod inverznog AES algoritma

Inverzne transformacije

Transformacija **InvAddRoundKey()** inverzna je transofrmacija transformacije **AddRoundKey()**, no kako je transformacija **AddRoundKey()** u biti **xor** operacija, tako je ona inverz sama sebi, pa su te dvije transformacije jednake.
InvShiftRows() transformacija inverzna je transformacija transformacije **ShiftRows()**. Kako je transformacija **ShiftRows()** jednak cikličkom pomicanju redova (svakog reda posebno) ulijevo, tako je transformacija **InvShiftRows()** jednaka cikličkom pomicanju redova udesno (svakog reda posebno), točno za onoliko mjesta, za koliko su pomaknuti transformacijom **ShiftRows()**. Transformacija je dana formulom:

$$s'_{r,(c+r) \bmod N_b} = s_{r,c} \quad r \in \{0,1,2,3\}, c \in \{0,1,\ldots N_b - 1\}$$

Transformacija **InvMixColumns()** inverzna je transformacija transformacije **MixColumns**,

te kako je transformacija **MixColumns** svedena na množenje fiksnog polinoma $a(x)$ s polinomima nad poljem $GF(2^8)$ modulo $x^4 + 1$ ($a(x) = \{03\} \cdot x^3 + \{01\} \cdot x^2 + \{01\} \cdot x + \{02\}$ je fiksan i invertibilan polinom), tako je inverzna operacija množenje inverznog polinoma $a^{-1}(x)$ polinomima nad poljem $GF(2^8)$ modulo $x^4 + 1$ gdje je

$$a^{-1}(x) = \{0b\} \cdot x^3 + \{0d\} \cdot x^2 + \{09\} \cdot x + \{0e\}$$

a transformacija $s'(x) = a^{-1}(x) \otimes s(x)$ može se zapisati pomoću matrica na sljedeći način:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Kao rezultat množenja u stupcu rezultata dobivamo sljedeća 4 bajta:

$$\begin{aligned} s'_{0,c} &= (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c}) \\ s'_{1,c} &= (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c}) \\ s'_{2,c} &= (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c}) \\ s'_{3,c} &= (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c}) \end{aligned}$$

Transformacija **InvSubBytes()** inverzna je transformacija transformacije **SubBytes()** koje se svodi na **S**-kutiju danu slikom 2.8 (pronalazak multiplikativnog inverz i zatim afina transformacija nad elementom polja $GF(2^8)$). Tako se i transformacija **InvSubytes()** također može dati (inverznom) **S**-kutijom. (prvo afina transformacija s istim elementom kao i kod **SubBytes()** transformacije, te zatim pronalazak multiplikativnog inverza). Inverzna **S**-kutija dana je slikom 2.13.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Slika 2.13: Inverzna **S**-kutija

Poglavlje 3

Asimetrični kriptosustavi

3.1 RSA

Već spomenuta glavna razlika između simetričnih i asimetričnih kriptosustava je u ključevima. Kod simetričnih kriptosustava kao što smo vidjeli ključ za šifriranje je jednak ključu za dešifriranje, dok kod asimetričnih kriptosustava ključ za šifriranje nije jednak ključu za dešifriranje. Tako je i pretpostavka kod simetričnih kriptosustava da su pošiljatelj i primatelj unaprijed upoznati sa ključem enkripcije tj. dekripcije. Takva pretpostavka na model se na prvi pogled ne čini problematična, no ona podrazumijeva da su se pošiljatelj i primatelj unaprijed dogovorili o ključu. Pretpostavka je također da su to učinili na siguran način, jer u suprotnom se njihov kriptosustav ne bi mogao smatrati sigurnim. No tu je temeljno pitanje koji je to siguran način dijeljenja informacija bio? Ako se radilo o osobnom susretu, tada se postavlja pitanje je li simetrični kriptosustav koji koriste uopće potreban, jer su možda sve potrebne informacije izmijenili prilikom osobnog susreta. No i ovdje se postavlja pitanje: što ako osobni susret nije moguć? Pa tako postoji mogućnost da pošiljatelj i primatelj znaju za neki alternativni način sigurne komunikacije pomoću kojeg su podijelili informaciju ključa. No i ovdje je očito pitanje zašto bi onda uopće koristili simetrični kriptosustav, kada imaju već postojeći sigurni sustav izmjena informacija? Rješenje tog problema ponuđeno je asimetričnim kriptosustavima (kriptosustavima s javnim ključem) Jedan od prvih (a i najpoznatijih) asimetričnih kriptosustava objavljen je 1977. godine te se zove RSA prema svojim tvorcima Ronaldu Rivestu, Adi Shamiru i Leonardu Adlemanu. Kasnije se ustanovilo da je ekvivalentan kriptosustav prvi izumio Clifford Cocks par godina ranije radeći za britansku obavještajnu agenciju, no ta je informacija ostala tajna do 1997. godine. Snaga RSA algoritma temelji se na problemu faktorizacije koji se danas smatra *teškim* problemom.

Algoritam

RSA algoritam može se podijeliti na dva dijela:

(a) **Generiranje ključeva**

1. Nasumično odabrati dva velika prosta broja p i q.
2. Izračunati vrijednosti $n = p \cdot q$ i $\varphi(n)$ gdje je $\varphi(n)$ Eulerova funkcija, tj. broj brojeva u nizu $1, 2, \ldots, n$ koji su relativno prosti s n. U ovom slučaju je:

$$\varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1) = n - p - q + 1$$

3. Odabrati broj $e < n$ relativno prost s brojem $(p-1) \cdot (q-1)$, te izračunati njegov multiplikativni inverz modulo $(p-1) \cdot (q-1)$, tj. modulo $\varphi(n)$ Ako označimo multiplikativni inverz sa d, tada vrijedi:

$$d \cdot e \equiv 1 (\mathrm{mod} \varphi(n))$$

Brojevi $[e, n]$ formiraju javni ključ koji bi trebao biti javno dostupan, dok brojevi $[d, n]$ formiraju tajni ključ koji bi trebao biti poznat samo osobi koja je vlasnik odgovarajućeg javnog ključa. U slučaju da je vrijednost n poznata, ponekad se koristi skraćena notacija za javni i tajni ključ, te se zapisuju kao e i d, respektivno.

(b) **Šifriranje i dešifriranje**
Pretpostavimo da pošiljatelj šalje poruku primatelju s javnim ključem $[e, n]$. RSA algoritam ne radi s bitovima i bajtovima, nego s prirodnim brojevima modulo n, pa tako otvoreni tekst X prikažemo kao niz brojeva modulo n $(x_1, x_2, \ldots, x_m)$. Šifriranje otvorenog teksta svodi se na potenciranje brojeva otvorenog teksta s vrijednošću e modulo n. Neka je x jedan od brojeva otvorenog teksta X, tada je šifrat C jednak:

$$C = x^e \bmod n$$

Šifrat C je također broj manji od n, te se on šalje primatelju. Primatelj posjeduje tajni ključ d te se postupak dešifriranja svodi na potenciranje šifrata sa tajnim ključem, tj.

$$x = C^d \bmod n = (x^e)^d \bmod n = x^{e \cdot d} \bmod n$$

No zašto vrijedi $x = x^{e \cdot d} \bmod n$? Prema Eulerovom teoremu znamo da vrijedi $x^{\varphi(x)} = 1 \bmod n$. Također, prisjetimo se da je $e \cdot d = 1 \bmod \varphi(n)$. To znači da je $e \cdot d - 1 = k \cdot \varphi(n)$, za neki cijeli broj k. Tako da sada imamo:

$$x^{e \cdot d} = x^{(e \cdot d - 1)+1} = x \cdot x^{e \cdot d - 1} = x \cdot x^{k \cdot \varphi(n)}$$

No kako je $x^{\varphi(x)} = 1 \bmod n$, tako je i $x^{k \cdot \varphi(n)} = (x^{\varphi(n)})^k = 1^k \bmod n$, pa vrijedi

$$x \cdot x^{k \cdot \varphi(n)} = x \cdot 1^k = x \bmod n$$

tj. vrijedi tvrdnja da je:

$$x = x^{e \cdot d} \bmod n$$

Snaga RSA kriptosustava temelji se na težini problema faktorizacije, koji se smatra *teškim* matematičkim problemom. Faktorizacija je teža što su brojevi veći, pa se tako preporuča da broj n bude barem 2048 bita dugačak, tj. da p i q budu otprilike 1024 bita dugački. Također, određeni izbori e i d smatraju se *slabi*, tj. da oslabljuju sigurnost sustava, pa se tako ne preporučuje rad sa malim e ($e = 2^{16} + 1$ smatra se dovoljno dobrim izborom za sve upotrebe).

Digitalni potpis i autentikacija pomoću RSA

Ponuđeni kriptosustav s javnim ključem rješava problem prve nesigurne komunikacije spomenut na početku poglavlja, no postoji mogućnost otvaranja novog problema. Javni ključevi su javno dostupni, te bilo tko čiji je ključ objavljen može primati poruke na siguran način. Poruku može poslati bilo tko, pa tako i protivnik u našem kriptosustavu, te na taj način doći do željenih informacija. Iz tog razloga prilikom slanja poruka osmišljen je još jedan sigurnosni mehanizam koji se zove digitalni potpis. Analogon u stvarnom svijetu bio bi mu vlastoručni potpis, tako da osoba koja prima poruku može provjeriti ishodište (autora) poruke bez obzira na njen sadržaj.
Pretpostavimo da pošiljatelj ima svoj javni ključ $[e_1, n_1]$ i tajni ključ $[d_1, n_1]$, te da primatelj ima javni ključ $[e_2, n_2]$ i tajni ključ $[d_2, n_2]$. Neka je x otvoreni tekst koji pošiljatelj želi poslati primatelju. Prema RSA algoritmu pošiljatelj šifrira tekst javnim ključem primatelja, no ukoliko pošiljatelj želi ostaviti svoj digitalni potpis postupak je sljedeći:

a) Pošiljatelj *potpisuje* otvoreni tekst svojim tajnim ključem (postupak analogan šifriranju kod RSA)
$$s = x^{d_1} \bmod n_1$$

b) Pošiljatelj šalje primatelju šifrirani otvoreni tekst C i potpisani otvoreni tekst s, te primatelj dešifrira tekst svojim tajnim ključem, te provjerava dešifrirati

tekst s javnim ključem osobe koja joj je poslala poruku. Ukoliko su oba teksta jednaka tada je provjereno i ishodište poruke.

3.2 Eliptičke krivulje u kriptografiji

Korištenje eliptičkih krivulja za šifriranje/dešifriranje naziva se kriptografija eliptičkih krivulja (*eng.* Elliptic curve cryptography (ECC)). U ovu svrhu na koordinatne točke gleda se kao na cijele brojeve nad konačnim poljem ili preciznije kao na polje generirano velikim prostim brojem. Računanje se izvodi nad cijelim brojevima modulo prosti broj p.
Za neki prosti broj p, eliptička krivulja E nad poljem $GF(p)$ dana je jednadžbom:

$$E : y^2 = x^3 + a \cdot x + b \mod p$$

gdje su x, y, a i b elementi skupa $\{1, 2, 3, \ldots, p-1\}$. S obzirom da brojevi a, b i p određuju jednadžbu krivulje E, krivulja se može zapisati i u obliku $E : [a, b, p]$. Općenito eliptičke krivulje u kriptografiji koriste se za asimetrične kriptosustave, generirajući javni i tajni ključ, od kojih je obično javni za šifriranje, dok je tajni za dešifriranje.

Generiranje ključeva

Ključevi se generiraju prema sljedećem postupku:

1. Odabere se početna točka B na krivulji
2. Odabere se nasumičan cijeli broj $k, (k < p)$ (tajni ključ) te se ne otkriva
3. Izračuna se točka K (javni ključ) skalarnim množenjem $K = k * B$

Javni i tajni ključ mogu se zapisivati kao:

$$\text{javni ključ: } [x_K, y_K, x_B, y_B, a, b, p] \quad \text{i tajni ključ: } [k, x_B, y_B, a, b, p]$$

Problem pronalaska privatnog ključa iz javnog ključa temelji se na problemu diskretnog logaritma za eliptičke krivulje. Problem diskretnog logaritma bavi se rješavanjem jednadžbe:

$$K = k * B \mod p$$

gdje su K i B točke na eliptičkoj krivulji, p prost broj, a k nepoznati cijeli broj. Radi usporedbe sigurnosti NIST (*eng.* National Institute of Standard and Technology) i ANSI

(*eng.* American National Standards Institute) za duljinu ključa korištenog kod RSA algoritma preporučuju 1024 bita, dok za duljinu ključa kod eliptičkih krivulja preporučuju 160 bita, što bi značilo da algortimi koji koriste eliptičke krivulje za generiranje ključeva istu sigurnost postižu kao i RSA sa 6 puta manjim ključem, čime se dobiva na memoriji i brzini.

Šifriranje i dešifriranje

Osim za generiranje ključeva eliptičke krivulje mogu poslužiti za šifriranje i dešifriranje poruke, tako čineći kriptosustav s javnim ključem.

Pretpostavimo da pošiljatelj pokušava poslati poruku primatelju koji ima sljedeći javni i tajni ključ:

$$\text{javni ključ: } [K, B, a, b, p] \quad \text{i tajni ključ: } [k, B, a, b, p]$$

Također se pretpostavlja da je poruka koja se želi poslati reprezentirana nekom točkom M na eliptičkoj krivulji $E : [a, b, p]$. Pošiljatelj može šifrirati poruku M i poslati je primatelju na sljedeći način:

- Pošiljatelj odabire nasumičan cijeli broj r i izračunava šifrat $[c_0, c_1]$ na sljedeći način:

$$c_0 = r \cdot B \quad \text{i} \quad c_1 = M + r \cdot K$$

- Pošiljatelj pošalje šifrat $[c_0, c_1]$ primatelju.
- Kako bi dešifrirao šifrat primatelj prvo pomnoži prvu komponentu šifrata sa svojim tajnim ključem, te zatim taj produkt oduzme od druge komponente:

$$c_1 - k \cdot c_0 = (M + r \cdot K) - k \cdot (r \cdot B) = (M + r \cdot K) - r \cdot (k \cdot B) = M + r \cdot K - r \cdot K = M$$

Poglavlje 4

Hibridni kriptosustavi

U prethodna dva poglavlja pojašnjeni su pojmovi simetričnih i asimetričnih kriptosustava, te su dani neki primjeri konkretnih algoritma za oba kriptosustava. Međutim, bez obzira na pomno osmišljavanje navedenih kriptosustava, neke mane nisu bile izbjegnute. Simetrični sustavi imaju već spomenutu manu u svojem dizajnu te ju je nemoguće zaobići, a to je sigurna izmjena ključa između pošiljatelja i primatelja, dok asimetrični kriptosustavi imaju manu drukčijeg karaktera. Asimetrični kriptosustavi su računski vrlo zahtjevni, te nisu najpogodniji za izmjenu velikih podatkovnih datoteka, tj. dugačke poruke. Iz tih razloga proizašla je ideja kombiniranja oba sustava te stvaranje novog koji bi imao prednosti oba sustava, a koji bi zaobišao njihove mane. Takav sustav naziva se hibridni kriptosustav, te danas postoji širok broj hibridnih kriptosustava, no glavna ideja njihove realizacije je uvijek ista:

1. Generiranje simetričnog ključa

2. Izmjena simetričnog ključa pomoću asimetričnog kriptosustava

3. Izmjena poruka pomoću simetričnog kriptosustava

Ideja je vrlo jednostavna, a na najbolji način iskorištava prednosti simetričnih i antisimetričnih kriptosustava, te zaobilazi njihove mane. Zbog svoji svojstva, tj. prednosti, hibridni sustavi postali su najraširenij kriptosustavi te se koriste kao sigurnosni mehanizam komunikacije za gotovo sve usluge dostupne putem interneta i šire.

4.1 Primjeri

RSA+AES

Neka je poruka koju pošiljatelj želi poslati primatelju sljedeća:" `Ovo je primjer` ".Pretpostavimo da je primatelj posjeduje javni ključ, te da je on poznat pošiljatelju. Neka je javni ključ primatelja sljedeći:

$$[e,n] = [65537,\ \begin{matrix}75875780556311055322345919831602861672 4\\ 99653445911088214708880116661497463385 5\\ 76882192405513758813328354680663782324 4\\ 47816600683066954416280055123272122 61]\end{matrix}$$

Pošiljatelj će prvo izgenerirati ključ za simetrični kriptosustav (AES128) kojim bi htio izmjenjivati poruke sa primateljem. Neka je izgenerirao sljedeći ključ:

`0b8e162f9bc3ee9c47f9e2f7d2d2e200`

Nakon generiranja pošiljatelj želi primatelju poslati taj ključ na siguran način. U RSA sustavu poruke su interpretirane sa brojevima u bazi 10, a kako je AES128 ključ 128-bitni broj, prebacivanjem u bazu 10 na njega možemo gledati kao na poruku. Označimo taj 128 bitni broj u baz 10 sa x. Bitno je da je $x < n$ zbog jedinstvenosti njegovog prikaza. Ukoliko to nije slučaj ključ moramo slati u 2 ili više poruka u asimetričnom kriptosustavu. U našem slučaju ključ možemo poslati u jednoj poruci. Dobijemo da je

$$x = 153592640924879824956011646935023539 20$$

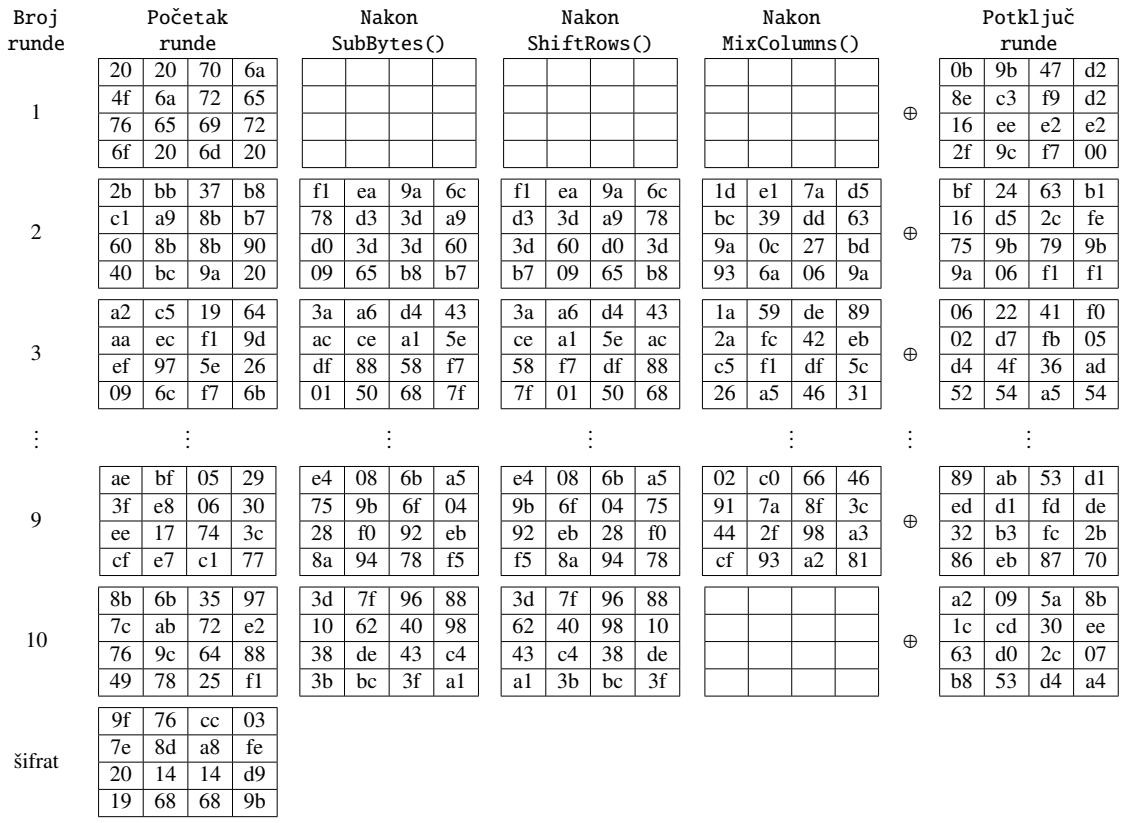
Sada poznavajući javni ključ $[e,n]$ poruku šifriramo tako da broj x potenciramo sa eksponentom e, te od dobivenog broja gledamo ostatak pri djeljenju sa n. Na taj način dobijemo šifrat y:

$$y = x^e \bmod n = \begin{matrix}16028112516031755846138418682189144171 20\\ 79014195579671449790165703738147944379 44\\ 83712390960143381556858854913272917852 82\\ 19877408941460970169364029153786 06\end{matrix}$$

Šifrat pošaljemo primatelju koji zatim pomoću tajnog ključa d:

$$d = \begin{matrix}28602333928066048518219569852751097816792101231079 13\\ 75001050228616220195436607821908076545897618297975 59\\ 80799614629116971065179430644305927477697389463848\end{matrix}$$

dešifrira šifrat te tako dobiva broj x ($y^d = (x^e)^d = x^{e \cdot d} \equiv x (\mathrm{mod} n)$). Primatelj broj x prebacuje u bazu 16, te tako dobiva ključ za AES algoritam pomoću kojeg će dalje komunicirati s pošiljateljem. Pošiljatelj nakon što je poslao ključ za AES128 algoritam može započeti sa šifriranjem i slanjem poruke putem simetričnog kriptosustava. Kako algoritam AES128 šifrira blokove duljine 128 bita, tj. 16 bajta, tada je otvoreni tekst prvo potrebno pretvoriti u blokove duljine 16 bajta. Prema proširenom ASCII (*eng.* American Standard Code for Information Interchange) standardu jednom znaku odgovara jedan bajt. ASCII se bazira na engleskoj abecedi, no u otvorenom tekstu ne nalaze se slova koji nisu iz engleske abecede, pa je pomoću njega moguće svaki znak otvorenog teksta zamijeniti s jednim bajtom. Kako otvoreni tekst ima 16 znakova (bajtova), tako cijela poruka čini jedan blok. Poruka se šifrira pomoću već opisanog AES128 algoritma čiji se prikaz može vidjeti na slici 4.1. Primatelj prima šifrat i dešifrira ga s dobivenim ključem (postupak dešifriranja je



Slika 4.1: Prikaz AES128 algoritma

jednak postupku šifriranja, ali u obrnutom rasporedu transformacija i potključeva, tako da su međurezultati po rundama jednaki). Naravno primatelj sada također može slati poruke

pošiljatelju na siguran način. U praksi broj n iz RSA algoritma je otprilike 4 puta duži, no ovdje je prikazan primjer sa manjim brojem n zbog duljine ispisa.

SSL/TLS

Sljedeći primjer hibridnog kriptosustava bit će prikazan SSL protokol (*eng.* Secure Sockets Layer) zbog njegove sveprisutnosti u komunikaciji putem interneta. SSL je razvijen sredinom 1990-ih godina u tvrtci Netscape, a kao jedan od izumitelja ističe se egipatsko američki kriptograf Taher El Gamal. SSL je stvoren kako bi se koristio u kombinaciji s internet preglednikom *Navigator*. Naknadno je organizacija IETF (*eng.* Internet Engineering Task Force), koja razvija internet standarde, preuzela odgovornost za daljnje razvijanje, te je 1999. godine objavila verziju poznatiju kao TLS 1.0 (*eng.* Transport Layer Security). Razlike između ta sva protokola nisu velike, no TLS kao noviji protokol i nasljednik SSL-a nudi više različitih algoritama za uspostavu komunikacije i samo komuniciranje. Razvoj i unaprijeđivanje alata koji implementiraju SSL i TLS i odgovarajućih kriptografskih biblioteka danas je mogući putem otvorenog OpenSSL projekta.
SSL se u biti sastoji od dva protokola:

1. *Protokol rukovanja*. Pomoću ovog protokola dvije strane dogovaraju se oko realizacije sigurnog SSL kanala putem kojeg će izmijenjivati podatke. Detaljnije, protokol se može korsiti za:

    - Dogovor oko kriptografskih algoritma pomoću koji će se uspostaviti siguran kanal;
    - Autentikaciju sugovornika;
    - Ustanovljenje ključeva potrebnih za komunikaciju.

2. *Protokol prepiske*. Ovaj protokol implementira sigurnost kanala, što uključuje:

    - Oblikovanje podataka (npr. podjela u blokove);
    - Računanje MAC-a podataka;
    - Šifriranje podataka.

MAC (*eng.* Message Authentication Code) je u biti kriptografski kontrolni zbroj koji se šalje zajedno sa porukom, a koji osigurava da poruka nije mijenjana, te osigurava saznanje o ishodištu poruke, tj. provjeru ishodišta.

Opis protokola rukovanja

Prikazan protokol rukovanja je pojednostavljena verzija, koja je zasnovana na paradigmi server-klijent. Klijent želi uspostaviti komunikaciju sa serverom i koristiti neku njegovu uslugu. Zato se u ovom pojednostavljenom prikazu protokola provjerava samo autentičnost servera (serveru nije toliko bitno tko koristi njegovu uslugu). Protokol slijedi:

Zahtjev klijenta. Ova poruka klijenta prema serveru započinje komunikaciju i zahtjev za zasnivanje zaštićenog SSL kanala. Kao dio poruke klijent šalje podatke, koji između ostalog sadrže:

- ID sesije, koji služi kao jedinstveni broj za identifikaciju sesije;
- pseudo nasumičan broj r_c, koji će služiti kao dokaz svježine u komunikaciji (tj. dokaz da je sugovornik zaista trenutno aktivan);
- popis kriptografskih algoritama koje podržava.

Odgovor servera. Server odgovara slanjem podataka klijentu, uključujući:

- ID sesije;
- pseudo nasumičan broj r_s, koji će služiti kao dokaz serverove svježine u komunikaciji;
- algoritam kojim će komunicirati, a koji je server odabrao sa klijentove liste;
- kopiju certifikata javnog ključa, uključujući podatke za njegovu provjeru (certifikat je podatak koji povezuje javni ključ sa njegovom svrhom uporabe. Sadrži četiri ključne informacije, a to su: ime vlasnika, vrijednost javnog ključa, vrijeme trajanja ključa, digitalni potpis).

U ovom trenutku klijent bi trebao provjeriti certifikat koji mu je poslao server, te se tako osigurati da je u komunikaciji sa željenom stranom.

K_p-sigurno slanje. Klijent ponovno generira pseudonasumičan broj K_p kojeg šifrira sa serverovim javnim ključem i pošalje ga šifriranog serveru. Iz broja K_p izvađaju se ključevi pomoću kojih se osigurava komunikacija (sesija), te zato mora biti poslan na siguran način. I klijent i server u ovom trenutku su u poziciji da mogu izvesti glavni ključ K_m. Za to koriste podataka K_p koji samo oni posjeduju i koji nije poznat nikome drugome, te brojeve r_c i r_s. Iz glavnog ključa K_m izvađaju se svi ostali ključevi za ostvarenje algoritama koji se koriste u protokolu.

Završetak klijenta. Klijent računa MAC svih izmijenjenih poruka do tog trenutka, te taj podatak šalje serveru.

Završetak servera. Server provjerava je li MAC koji je primio od klijenta ispravan, te nakon provjere računa MAC svih izmijenjenih poruka do tog trenutka, te taj podatak šalje klijentu.

Uočimo kako je gornjim postupkom osigurana autentikacija servera s obzirom da tko god je poslao poruku *Završetak servera* poznata mu je informacija K_m, jer mu je morala biti poznata informacija K_p. Onaj kojem je poznata informacija K_p (osim klijentu), poznati mu je i tajni ključ koji odgovara javnom ključu iz certifikata iz poruke *Odgovor servera*. Ta informacija je poznata samo izvornom serveru. Server je sigurno i aktivan s obzirom da je ključ K_m generiran iz pseudonausmičnih vrijednosti K_p i r_c koje je poslao klijent.
Naravno postoji i verzija protokola rukovanja u kojoj se i provjerava autentikacija klijenta, no ona neće ovdje biti razmatrana.

Protokol prepiske

Protokol prepiske započinje nakon završetka protokola rukovanja, te nakon što su i klijent i server na dogovoreni način izgenerirali ključeve potrebne za komunikaciju iz glavnog ključa K_m. Generirani ključevi su:

- K_{ECS} za simetrično šifriranje od klijenta prema serveru,
- K_{ESC} za simetrično šifriranje od servera prema klijentu,
- K_{MCS} za MAC od klijenta prema serveru,
- K_{MSC} za MAC od servera prema klijentu.

SSL protokol prepiske specificira proces korištenja tih ključeva za zaštitu izmijenjenog prometa podataka između servera i klijenta. Na primjer za podatke koje klijent šalje serveru, proces je sljedeći:

1. Izračunavanje MAC-a nad podacima koji se šalju (i još nekim dodatnim podacima) koristeći ključ K_{MCS}.
2. Dodavanja dobivenog MAC-a na podatke koji se šalju i eventualno nadodavanje podataka do višekratnika veličine bloka koji se šalje prema algoritmu.
3. Šifriranje dobivene poruke ključem K_{ECS}.

Heartbleed bug

Procijenjeno je da aktivne internet stranice koje koriste OpenSSL alate čine gotovo 66% tržišnog udjela na internetu. OpenSSL također se koristi za zaštitu e-mail servera, chat servera, VPN (*eng.* virtual private network) servera, te u mnogo klijentskih aplikacija. Također neki operativni sustavi dolaze već opremljeni sa OpenSSL alatima.
07.04.2014. javno je objavljena informacija da je pronađena pogreška u implementaciji tada jedne od posljednje objavljenih verzija OpenSSL-a. Pogreška je nazvana *Heartbleed bug*, a omogućavala je napadačima prisluškivanje komunikacije na ranjivim verzijama alata OpenSSL-a. Tajni ključevi davatelja usluga su također bili kompromitirani, pa su samim time napadači mogli doći do povjerljivih informacija o njihovim korisnicima. Pogrešku su pronašli sigurnosni inžinjeri tvrtke Codenomicon, a nezavisno od njih otkrila ju je i Neel Mehta iz tvrtke Google. Pogreška je uklonjena novom verzijom OpenSSL-a koja je izdana na dan objavljivanja Heartbleed bug-a, no verzija s greškom je bila u izdana i u uporabi duže od 2 godine.

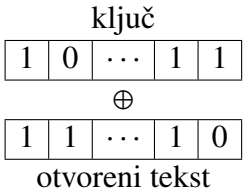
Poglavlje 5

Sigurnost kriptosustava

Do sada su navedeni kriptosustavi i njihova specifikacija, no temeljno je pitanje zašto su oni sigurni ili se smatraju takvima? Tom pitanju je posvećeno ovo poglavlje.

5.1 DES i AES

Prema specifikaciji DES-a i AES-a može se uočiti sličnost u ideji algoritma šifriranja. Otvoreni tekst se pomoću operacija transformira određeni broj runda (u DES-u radi se se o funkciji f, u AES-u o transformacijama **`AddRoundKey()`**,**`MixColumns()`**, **`ShiftRows()`** i **`SubBytes()`**). U oba algoritma dva faktora su ključna za sigurnost, a to su ključ i **S**-kutije. Uloga ključa je jasna, te je cilj da se pomoću ključa otvoreni tekst transformira operacijom **`xor`**.



Uloga **S**-kutije nelinearna transformacija, zajedno sa spomenutim transformacijama, po nekoliko rundi i s različitim ključevima dovoljno transformiraju otvoreni tekst tako da se smatra da je u oba slučaja (i AES i DES) generalno gledajući najbolji napad *brute force*, tj. napad provjeravanjem cijelog prostora ključeva. Za takav, sveobuhvatan, napad na DES potrebno je provjeriti 2^{56} mogućih ključeva, dok je za AES potrebno provjeriti 2^{128}, 2^{192} ili 2^{256} mogućih ključeva, ovisno o duljini ključa. Iz tog razloga DES već više od 15 godina nije preporučljiv za upotrebu, jer se smatra da je prostor ključeva premalen. Kao alternativa ponuđen je trostruki DES (3DES) koji tri puta izvršava DES, te se tako dobiva 48 rundi Feistelove šifre, a prema standardu jedna od opcija je i da svaki DES ima svoj ključ, čime se dobiva duljina ključa od 168 bita. Zbog toga je jedan od uvjeta na natječaju za AES bio

da bude brži od 3DES-a.

Iako je koncept DES-a i AES-a isti, a to je transformirati otvoreni tekst ponavljajući transformacije kroz određeni broj runda s različitim potključevima, razlika u realizaciji je velika. Kod AES-a na blokove otvorenog teksta se gleda kao na elemente konačnog polja, te su operacije kojima se blokovi transformiraju u biti operacije koje taj skup *blokova* čine poljem, pa su samim time invertibilne. Tako je i proces dešifriranja samim time ništa drugo nego invertiranje u obrnutom redoslijedu. U DES-u to nije slučaj, no Feistelova šifra (mreža) konstruriana je na taj način da je postupak dešifriranja jednak postupku šifriranja s obrnutim redoslijedom ključeva (uočimo da nema inverznih operacija, nego su operacije potpuno jednake). Ako se prisjetimo, šifrat dobiven DES-om dan je u sljedećem obliku: $\mathbf{PP^{-1}}(R_{16}, L_{16})$. Primjenom početne permutacije dobiva se (R_{16}, L_{16}). Ako sa (L'_i, R'_i) označavamo blokove prilikom dešifriranja, tada je $(R_{16}, L_{16}) = (L'_0, R'_0)$, pa je:

$$\begin{aligned} L'_1 &= R'_0 = L_{16} = R_{15} \\ R'_1 &= L'_0 \oplus f(R'_0, K'_1) = R_{16} \oplus f(L_{16}, K_{16}) = R_{16} \oplus f(R_{15}, K_{16}) \end{aligned}$$

A kako smo prilikom šifriranja dobili da je $R_{16} = L_{15} \oplus f(R_{15}, K_{16})$, tako supstitucijom dobivamo:

$$R_{16} \oplus f(R_{15}, K_{16}) = \overbrace{L_{15} \oplus \underbrace{f(R_{15}, K_{16}) \oplus f(R_{15}, K_{16})}_{[0,0,\ldots,0]}}^{R_{16}} = L_{15}$$

Tako smo nakon prve runde dešifriranja DES-a dobili blok $(L'_0, R'_0) = (R_{15}, L_{15})$, te analogno daljnjim postupkom po rundama dobivamo blokove $(R_{15}, L_{15}), (R_{14}, L_{14}), \ldots, (R_1, L_1)$, a nakon zadnje runde dobivamo (R_0, L_0), kojem izmjenimo raspored i nad njim upotrijebimo inverz početne permutacije te dobijemo

$$\mathbf{PP^{-1}}(L_0, R_0) = O$$

tj. dobivamo otvoreni tekst. Dakle sigurnost DES-a i AES-a bazira se na dovoljno velikom broju jednostavnih transformacija u kombinaciji s tajnom informacijom ključa.

5.2 RSA i ECC

U prijašnjim poglavljima pokazani su algoritmi RSA i ECC, te je spomenuto da se baziraju na problemima koji se smatraju *teškima*, no nigdje nije spomenuto zašto? Problemi na kojima se baziraju ti algoritmi su vrlo jednostavni i lagano shatljivi, te je jasno kako ih je moguće riješiti, pa se nameće pitanje što ih čini *teškima*? Bez obzira što je način rješavanja poznat, za rješavanje je današnjim računalima potrebno previše vremena da bi se rješenja

isplatilo tražiti na taj način.
Možemo zaključiti da je glavna mjera za određivanje težine problema vrijeme utrošeno na rješavanje istog. No, opet, vrijeme rješavanja ovisi o tome koliko brzo računalo računa, tj. koliko brzo izvršava zadane operacije, te isti problem se može smatrati izuzetno teškim na starijem računalu, dok na novom računalu može biti lagan, tj. brzo rješiv. Preciznije, glavna mjera za određivanje težine problema je broj operacija koji je potrebno izvršiti da bi riješili problem. Iz tog razloga problem se ne promatra na nekom konkretnom računalu nego se promatra teoretski model računala. Pa ćemo za početak definirati taj model, koji se naziva Turingov stroj, prema svojem izumitelju, Alanu Turingu.

Definicija 5.2.1. ***Turingov stroj*** *uređena je sedmorka* $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$*, gdje su* Q, Σ, Γ *konačni skupovi i vrijedi:*

1. Q *je skup svih stanja,*
2. Σ *je skup ulaznih znakova koji ne sadrži* ***prazan znak*** $\sqcup$*,*
3. Γ *je abeceda trake i sadrži znak* $\sqcup$*, te vrijedi* $\Sigma \subseteq \Gamma$*,*
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D, S\}$ *je funkcija prijelaza,*
5. q_0 *je početno stanje,*
6. $q_{DA} \in Q$ *je stanje prihvaćanja*
7. $q_{NE} \in Q$ *je stanje odbijanja, gdje vrijedi* $q_{NE} \neq q_{DA}$

Turingov stroj se u pravilu osim formalno, definira i opisno, jer iz formalne definicije nije najjasnije što bi Turingov stroj trebao predstavljati, tj. kako bi trebala izgledati njegova realizacija. Turingov stroj možemo zamisliti kao stroj sastavljen od beskonačne trake koja predstavlja memoriju, te glave za čitanje koja čita znakove sa trake. Osim čitanja, glava može i pisati po traci, te se također pomicati ulijevo ili udesno po traci. Također Turingov stroj uvijek se nalazi u nekom stanju $q \in Q$. Slijedi kratak opis rada:

Prije početka rada Turingov stroj nalazi se u početnom stanju q_0, na traci je napisana ulazna riječ (niz ulaznih znakova), dok je na svim ostalim mjestima na traci upisan znak $\sqcup$. Glava za čitanje uobičajeno se nalazi na krajnje lijevom znaku ulazne riječi, no prema dogovoru može se pretpostaviti da se nalazi na nekoj drugoj poziciji. Glava čita znak sa trake iznad kojeg se nalazi i na temelju stanja u kojem se nalazi, te pročitanog znaka ($Q \times \Gamma$) Turingov stroj odlučuje u kojem će stanje promijeniti trenutno stanje, koji znak će upisati na traku umjesto pročitanog znaka (može se upisati isti znak), te hoće li glavu pomaknuti ulijevo, udesno ili će ostati na istoj poziciji ($Q \times \Gamma \times \{L, D, S\}$). Odluka o upisivanju znaka i pomicanju glave Turingovog stroja određena je funkcijom prijelaza δ. Ukoliko se Turingov

stroj nađe u jednom od naglašenih, završnih stanja q_{DA} ili q_{NE} odmah prestaje s radom te prihvaća ili odbacuje ulaznu riječ. Ako se nađe u stanju q_{DA} riječ je prihvaćena, a ako se nađe u stanju q_{NE} riječ je odbijena. Također postoji mogućnost da Turingov stroj nikada ne dosegne jedno od završnih stanja, te u tom slučaju može raditi u beskonačno, nikada ne stajući.

Definicija 5.2.2. *Kažemo da Turingov stroj prepoznaje riječ $w \in \Gamma^*$ ako stane u završnom stanju q_{DA} kada je na početku rada na traci stroja bila upisana samo riječ w. Za proizvoljan Turingov stroj T, sa $L(T)$ označavamo skup svih riječi koje prepoznaje.*
Kažemo da je jezik $L \subseteq \Gamma^*$ prepoznatljiv ukoliko postoji Turingov stroj M koji prepoznaje sve njegove riječi.
Kažemo da je jezik $L \subseteq \Gamma$ odlučiv ukoliko postoji Turingov stroj M koji ga prepoznaje, te za svaku riječ $w \in \Gamma^* \setminus L$ Turingov stroj M stane u završnom stanju q_{NE}.

Uočimo da je definicijom funkcije prijelaza δ definiran rad Turingovog stroja, te kako je δ definirana tako da je za svaki uređeni par $(q, a) \in Q \times \Gamma$ određena točno jedna uređena trojka $(q', a', X) \in Q \times \Gamma \times \{L, D, S\}$. Iz tog razloga takav Turingov stroj naziva se još i **deterministički** Turingov stroj. Osim determinističkog Turingovog stroja postoji i **nedeterministički** Turingov stroj koji je definiran na sljedeći način:

Definicija 5.2.3. ***Nedeterministički Turingov stroj*** *uređena je sedmorka $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$, gdje su Q, Σ, Γ konačni skupovi i vrijedi:*

1. *Q je skup svih stanja,*
2. *Σ je skup ulaznih znakova koji ne sadrži **prazan znak** $\sqcup$,*
3. *Γ je abeceda trake i sadrži znak $\sqcup$, te vrijedi $\Sigma \subseteq \Gamma$,*
4. *$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, D, S\})$ je funkcija prijelaza,*
5. *q_0 je početno stanje,*
6. *$q_{DA} \in Q$ je stanje prihvaćanja*
7. *$q_{NE} \in Q$ je stanje odbijanja, gdje vrijedi $q_{NE} \neq q_{DA}$*

Jedina razlika u definicijama determinističkog i nedeterminističkog Turingovog stroja je u funkciji prijelaza. Vidimo da kod nedeterminističkog Turingovog stroja funkcija prijelaza kao ulaz uzima uređeni par iz skupa $Q \times \Gamma$, te vraća neki podskup skupa $Q \times \Gamma \times \{L, D, S\}$, dok deterministički stroj vraća samo jedan element skupa $Q \times \Gamma \times \{L, D, S\}$, što znači da prijelaz nije jednoznačno određen, te za isti ulazni uređeni par nedeterministički Turingov

stroj može reagirati na više načina. Na svaki od tih načina možemo gledati kao na novu granu stabla koje se generira radom Turingovog stroja. Kako se svaka grana razvija posebno, svaka od njih može i stati u stanju prihvaćanja, stanju odbijanja ili ne mora stati. Iz tog razloga potrebno je posebno definirati da stroj prihvaća ulaznu riječ ako samo u jednoj grani u konačno mnogo koraka stane u stanju q_{DA}.

Dva navedena modela računala bitna su za daljnja razmatranja vremenske složenosti problema, pa ćemo definirati što je to vremenska složenost Turingovog stroja.

Definicija 5.2.4. *Neka je M deterministički Turingov stroj koji staje za svaki ulaz. Vremenska složenost stroja M je funkcija $f : \mathbb{N} \to \mathbb{N}$, gdje je $f(n)$ maskimalan broj koraka koje stroj M napravi za svaki ulaz duljine n. Ako je $f(n)$ vremenska složenost od M, tada još i kažemo da se M izvršava u vremenu $f(n)$ ili da je M jedan $f(n)$ Turingov stroj.*

Vremenska složenost nedeterminističkog Turingovog stroja definira se analogno, jedina je razlika što je kod nedeterminističkog stroja funkcija $f(n)$ maksimalan broj koraka svih grana koje stroj M napravi za svaki ulaz duljine n.

Definicija 5.2.5. *Neka su $f, g : \mathbb{N} \to \mathbb{R}^+$ funkcije.Definiramo da je $f(n) = O(g(n))$ ako postoje prirodni brojevi c i n_0, takvi da za svaki $n \geq n_0$ vrijedi:*

$$f(n) \leq c \cdot g(n)$$

Kažemo još i da je funkcija g gornja asimptotska međa za funkciju f.

Sada možemo definirati klasu vremenske složenosti:

Definicija 5.2.6. *Neka je $t : \mathbb{N} \to \mathbb{R}^+$ funkcija. Definiramo klasu vremenske složenosti $TIME(t(n))$ kao skup svih jezika koji su odlučivi na nekom $O(t(n))$ Turingovom stroju. Specijalno ako govorimo o deterministčkim Turingovim strojevima klasu ćemo označavati sa $DTIME(t(n))$, a ako govorimo o nedeterminističkim Turingovim strojevima, klasu ćemo označavati sa $NTIME(t(n))$.*

Definicija 5.2.7. **P** *je klasa svih jezika koji su odlučivi na polinomnim deterministčkim Turingovim strojvima, drugim riječima:*

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} DTIME(n^k)$$

Bez obzira što se cijela teorija koju smo do sada iznijeli bazira na modelu računala, klasa **P** je vrlo važna jer je invarijanta za sve modele računanja koje su polinomno ekvivalentne deterministčkom Turingovom stroju, te ugrubo odgovara klasi svih problema

koji su rješivi u razumnom vremenu na stvarnim računalima. Ako je problem u klasi **P**, tada postoji metoda koja je vremenske složenosti n^k, gdje je k konstanta. Naravno, ako je vremenska složenost metode n^{100} jasno je da takva metoda neće biti od neke koristi. Unatoč tome, u praksi se pokazalo korisnim dokazati da je neki problem dio klase **P**, jer za problem za koji je pronađena polinomijalna vremenska složenost n^k, ta se složenost od možda početne nepraktične svede na praktičnu, pronalaskom još brže metode (s manjim parametrom k). Kod mnogih problema moguće je izbjeći rješavanje problema na način da se provjeravaju sva moguća rješenja (*brute force*), te doći do rješenja polinomno vremenski složenom metodom. No, za veliki broj problema pokušaji izbjegavanja *brute force* nisu bili uspješni, te polinomno složene vremenske metode kojima bi se ti problemi rješavali do danas nisu poznate. Sljedećom definicijom bit će dana klasa takvih problema.

Definicija 5.2.8. **NP** *je klasa svih jezika koji su odlučivi na polinomnim nedeterminističkim Turingovim strojevima, drugim riječima:*

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} NTIME(n^k)$$

Nije poznato zašto nisu pronađena polinomna rješenja za te probleme, tako da je moguće da će biti otkrivena u budućnosti, kao što je moguće da takva rješenja jednostavno ne postoje.
Za klasu **NP** dokazana je karakterizacija koja se zove teorem o certifikatu, a govori sljedeće:

$$L \in \mathbf{NP} \Leftrightarrow (\exists R \in \mathbf{P})(\exists k \in \mathbb{N}) L = \{x : (\exists c)(|c| = O(|x|^k) \wedge R(x, c))\}$$

to jest govori da ukoliko je neki jezik iz klase **NP** tada za svaku njegovu riječ x postoji polinomno vremenski složen Turingov stroj koji kao ulaz prima x i riječ c te ih prihvaća. Do riječi c moguće je doći u polinomnom vremenu. Drugim riječima, problem (jezik) je iz **NP** ukoliko se u polinomnom vremenu može provjeriti je li x njegovo rješenje.

Klase **P** i **NP** možemo promatrati i na sljedeći način:

P	=	{ klasa svih problema koji se mogu riješiti *brzo* }
NP	=	{ klasa svih problema čija se rješenja mogu provjeriti *brzo* }

Iz definicije determinističkih i nedeterminističkih Turingovih strojeva očit je odnos između ove dvije klase: $\mathbf{P} \subseteq \mathbf{NP}$, no pitanje na koje matematičari već četeresetak godina bezuspješno pokušavaju odgovoriti, je točan odnos između ove dvije klase. Vrijedi li $\mathbf{P} = \mathbf{NP}$ ili vrijedi $\mathbf{P} \neq \mathbf{NP}$? Čvrsto se vjeruje da vrijedi $\mathbf{P} \neq \mathbf{NP}$, no još nitko nije tu tvrdnju uspio dokazati niti opovrgnuti.

Problem faktorizacije cijelih brojeva i problem diskretnog logaritma dva su primjera problema koji su dio klase **NP**, a za koje nije poznato jesu li i dio klase **P**. Za oba problema

postoje algoritmi koji ih rješavaju, no ti algoritmi nisu polinomijalne vremenske složenosti, već eksponencijalne, što ih za velike brojeve čini praktički neupotrebljivim.

Bibliografija

[1] *The Heartbleed Bug*, 2014, http://www.heartbleed.com/, [Online; accessed 25-8-2014].

[2] Andrej Dujella i Marcel Maretić, *Kriptografija*, Element, 2007.

[3] PUB FIPS, *46-3: Data Encryption Standard (DES)*, National Institute of Standards and Technology **25** (1999), br. 10.

[4] Keith M. Martin, *Everyday Cryptography: Fundamental Principles and Applications*, Oxford University Press, 2012.

[5] Alfred J. Menezes, Paul C. Van Oorschot i Scott A. Vanstone, *Handbook of Applied Cryptography. Series on discrete mathematics and its applications*, CRC press, 1997.

[6] Rolf Oppliger, *Contemporary cryptography*, Artech House, 2011.

[7] NIST FIPS Pub, *197: Advanced encryption standard (AES)*, Federal Information Processing Standards Publication **197** (2001).

[8] Mark Stamp, *Information security: principles and practice*, John Wiley & Sons, 2011.

[9] Mladen Vuković, *Složenost algoritama*, (2013), http://web.math.pmf.unizg.hr/~vukovic/Diplomski-kolegiji/SA/SA-skripta-2013-verzija-4.pdf.

[10] P.K. Yuen, *Practical cryptology and Web security*, Pearson Education, 2006.

Sažetak

Kriptosustav je realizacija modela za sigurnu izmjenu poruka. Jedna od podjela kriptosustava je prema odabiru ključeva. U ovom radu prikazana je ta podjela, podjela na simetrične i asimetrične kriptosustave, te je dana specifikacija najpoznatijih simetričnih i asimetričnih algoritama. Također prikazani su hibridni kriptosustavi kao kombinacija simetričnih i asimetričnih kriptosustava. Na kraju se pokazuje zašto se ti kriptosustavi smatraju sigurnim sustavima za izmjenu poruka.

Summary

The cryptosystem is implementation of secure message exchange model. One of classification criteria for cryptosystems is the choice of keys. Classification of cryptosystems to asymmetric and symmetric is presented in this work and specification of most known symmetric and asymmetric algorithm is given. Also, hybrid cryptosytems as combination of symmetric and asymmetric cryptosystem are shown. At the end it is shown why these cryptosystems are regarded as safe for message exchange.

Životopis

Josip Iveković rođen je 5.12.1988. godine u Zagrebu. Roditelji su mu Mario Iveković i Janja Iveković (rođ. Škreblin). 2003. godine završio je osnovnu školu bana Josip Jelačića u Podsusedu. Iste godine upisuje opću XI. gimanziju u Zagrebu koju završava 2007. godine s vrlo dobrim uspjehom. Preddiplomski studij Matematike upisuje na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu 2007. godine. 2011. godine završava preddiplomski studij sa dobrim uspjehom, te upisuje diplomski studij Računarstvo i matematika.