HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

AES algoritam

CCERT-PUBDOC-2003-08-37

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.

Rezultat toga rada ovaj je dokument koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.lss.hr - laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

AES (engl. *Advanced Encryption Standard*) je jedan od kriptografskih algoritama za zaštitu digitalnih podataka. AES standard temelji se na simetričnom Rijndael algoritmu, a kao standard razvijen je da bi postupno zamijenio DES, čija sigurnost u današnje vrijeme više nije dovoljna.
Slično kao i DES, AES je razvijen u privatnom sektoru, no u suradnji s američkom vladom. AES je definiran i prihvaćen od strane NIST-a (engl. *National Institute of Standards and Technology*) u FIPS 197 dokumentu, te se kao takav može koristiti u američkim državnim i drugim institucijama.
U ovom trenutku, što se tiče uporabe na Internetu, DES, pa i drugi simetrični algoritmi još uvijek su zastupljeni u većoj mjeri, no vremenom će i AES pronaći svoje mjesto, pogotovo zato što DES kao takav ne predstavlja adekvatno rješenje u sustavima koji zahtijevaju najviše razine sigurnosti.

2. Način rada

Za razliku od npr. DES i IDEA algoritama koji podatke šifriraju u 64-bitnim blokovima, AES, odnosno Rijndael algoritam, šifrira 128-bitne blokove podataka. Duljina ključa može biti 128, 192, ili 256 okteta. Sam Rijndael algoritam je oblikovan tako da je moguće šifriranje podataka u blokovima različitih duljina i s različitim duljinama ključeva, no to nije definirano kroz standard. Obzirom na ključeve, uobičajeno je algoritme nazivati AES-128, AES-192 i AES-256.

2.1. Matrica stanja

Interno, sve operacije AES algoritma provode se na dvodimenzionalnom nizu okteta, odnosno matrici stanja (engl. *state*). Matrica stanja se sastoji od četiri retka koji sadrže određeni broj okteta. Taj broj okteta predstavlja ukupnu duljinu bloka (u bitovima) podijeljenu sa 32. U konkretnom slučaju, kod AES algoritma to znači 4 okteta (Rijndael algoritam dopušta i druge vrijednosti).
Šifriranje, odnosno dešifriranje, se provodi tako da se ulazni blok podataka kopira u matricu stanja nad kojom se zatim provode razne operacije, kako će biti pokazano u nastavku dokumenta. Konačno, završna vrijednost matrice stanja se kopira u izlazni šifrirani blok podataka (Slika 1).

ulaz

$u_{0,0}$	$u_{0,1}$	$u_{0,2}$	$u_{0,3}$
$u_{1,0}$	$u_{1,1}$	$u_{1,2}$	$u_{1,3}$
$u_{2,0}$	$u_{2,1}$	$u_{2,2}$	$u_{2,3}$
$u_{3,0}$	$u_{3,1}$	$u_{3,2}$	$u_{3,3}$

matrica stanja

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

izlaz

$i_{0,0}$	$i_{0,1}$	$i_{0,2}$	$i_{0,3}$
$i_{1,0}$	$i_{1,1}$	$i_{1,2}$	$i_{1,3}$
$i_{2,0}$	$i_{2,1}$	$i_{2,2}$	$i_{2,3}$
$i_{3,0}$	$i_{3,1}$	$i_{3,2}$	$i_{3,3}$

Slika 1: Ulazni i izlazni podaci te matrica stanja

Ulazni blok podataka kopira se u matricu stanja po sljedećoj formuli:

$$stanje[r,s]=ulaz[r+4s],\ za\ 0 \le r < 4,\ i\ 0 \le s < 4$$

Također, na sličan način se izlazni podaci kopiraju iz matrice stanja:

$$izlaz[r+4s]=stanje[r,s],\ za\ 0 \le r < 4,\ i\ 0 \le s < 4$$

Četiri okteta u svakom stupcu matrice stanja formiraju 32-bitnu riječ. Na taj način matricu stanja moguće je interpretirati kao jednodimenzionalni niz 32-bitnih riječi $w_0,\ldots,w_3$. Formalno se te riječi definiraju na sljedeći način:

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

2.2. Ključevi

Kao što je ranije spomenuto, ulazni i izlazni blokovi AES algoritma su duljine 128 okteta, dok duljina ključa može biti 128, 192 ili 256 okteta. Ključ se prikazuje kao određeni broj 32-bitnih riječi koji, ovisno o duljini, varira, pa može biti predstavljan s 4, 6 ili 8 riječi. Također, broj koraka od kojih se

algoritam sastoji ovisi o duljini ključa. Tablica 1 daje prikaz ovisnosti broja koraka algoritma o duljini ključa, odnosno o implementaciji algoritma (broj koraka i duljina ulaznih/izlaznih blokova dani su u 32-bitnim riječima). N_k i N_w redom označavaju duljinu ključa i duljinu bloka podataka u riječima, dok N_r označava broj koraka algoritma.

	N_k	N_w	N_r
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Tablica 1: Duljine ključeva, blokova i broj koraka inačica AES algoritma

2.3. Algoritam

Svaki od koraka algoritma predstavlja funkciju koja se sastoji od četiri transformacije, koje se provode nad oktetima:

1. Zamjena okteta na temelju supstitucijske tablice (S-blok).
2. Posmak redaka u matrici stanja.
3. Miješanje podataka unutar svakog stupca matrice stanja.
4. Dodavanje podključa u matricu stanja.

Šifriranje se provodi tako da se ulazni blok kopira u matricu stanja kako je ranije opisano, te se zatim provodi inicijalno dodavanje podključa u matricu. Nakon toga, matrica stanja se transformira 10, 12 ili 14 puta, ovisno o duljini ključa, s time da je posljednji korak donekle različit od prethodnih. Konačno, matrica stanja se kopira u izlazni blok, na način koji je također opisan prethodno.
Algoritam se može definirati na sljedeći način:

```
početak
  oktet stanje[4,Nw]

  stanje = ulaz
  dodaj_podključ (stanje, w[0,Nw-1])

  za korak=1 do korak=Nr-1 radi
        zamjena_okteta(stanje)
        posmak_redaka(stanje)
        mijesanje_stupaca(stanje)
        dodaj_podključ(stanje,w[korak*Nw,(korak+1)*Nw-1])
  kraj
  zamjena_okteta(stanje)
  posmak_redaka(stanje)
  mijesanje_stupaca(stanje)
  dodaj_podključ(stanje,w[Nr*Nw,(Nr+1)*Nw-1]

  izlaz = stanje
kraj
```

2.3.1. Zamjena okteta

Funkcija `zamjena_okteta()` obavlja nelinearnu transformaciju koja nezavisno za svaki oktet vrši promjenu korištenjem supstitucijske tablice (Tablica 2) što predstavlja S-blok algoritma. S-blok se konstruira korištenjem dviju transformacija:

1. Izračunati multiplikativnu inverziju u konačnom polju GF(2^8) (dodatak A); element `[00]` mapira se sam na sebe.
2. Primijeniti sljedeću transformaciju nad bitovima matrice stanja:
   $b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$, $0 \le i < 8$,
   gdje b_i i-ti bit okteta, a c_i je i-ti bit okteta c vrijednosti `[63]` odnosno `[01100011]` u binarnom prikazu.

X – četiri viša bita okteta	Y – četiri niža bita okteta															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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

Tablica 2: S-blok – supstitucijske vrijednosti za oktet XY (heksadecimalni prikaz)

2.3.2. Posmak redaka

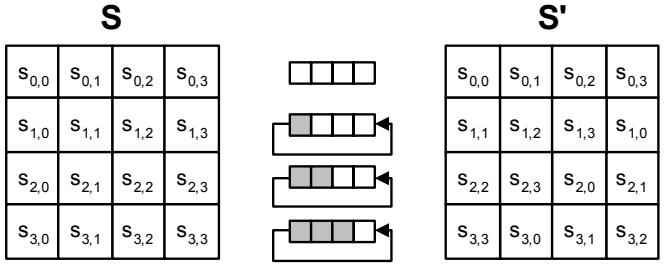
Funkcija `posmak_redaka (stanje)` predstavlja transformaciju gdje se okteti u zadnja tri retka matrice stanja ciklički pomiču za određeni broj okteta (posmak). Prvi redak (r_0) se ne posmiče. Formalno, posmak se može opisati na sljedeći način:

$stanje'_{r,s} = stanje_{r,(s+posmak(r,Nw)) \bmod Nw}$, za $0 < r < 4$ i $0 \leq s < N_w$

gdje vrijednost varijable $posmak(r,N_w)$ ovisi o broju retka r na sljedeći način:

$posmak(1,N_w)=1 \quad posmak(2,N_w)=2 \quad posmak(3,N_w)=3$

Ovakav način posmaka kao posljedicu ima pomicanje okteta na "niže" pozicije u retku (manja vrijednost s u danom retku), dok se okteti sa nižih pozicija posmiču na kraj retka (viša vrijednost s u danom retku).
Slika 2 prikazuje tako definirani posmak redaka.



Slika 2: Rotacija redaka matrice stanja koju uzrokuje funkcija posmak_redaka()

2.3.3. Miješanje podataka u stupcima matrice stanja

Funkcija `miješanje_stupaca()` provodi transformacije unutar matrice stanja nad svakim pojedinim stupcem (Slika 3). Stupci se promatraju kao polinomi četvrtog reda, te se multipliciraju s konstantnim polinomom `a(x)`, a rezultat se svodi na odgovarajući stupanj korištenjem modulo (x^4+1) operacije. Konstantni polinom `a(x)` definiran je na sljedeći način:

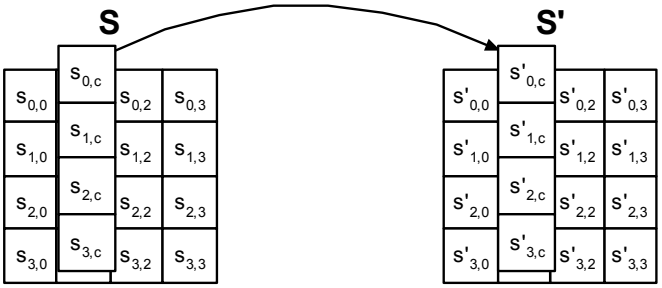
$a(x)=[03]x^3+[01]x^2+[01]x+[02]$

Matrica stanja modificira se na sljedeći način:

$s'(x)=a(x)*s(x)$

Odnosno:

$s'_{0,c} = ([02] \otimes s_{0,c}) \oplus ([03] \otimes s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$
$s'_{1,c} = s_{0,c} \oplus ([02] \otimes s_{1,c}) \oplus ([03] \otimes s_{2,c}) \oplus s_{3,c}$
$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus ([02] \otimes s_{2,c}) \oplus ([03] \otimes s_{3,c})$
$s'_{3,c} = ([03] \otimes s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus ([02] \otimes s_{2,c})$



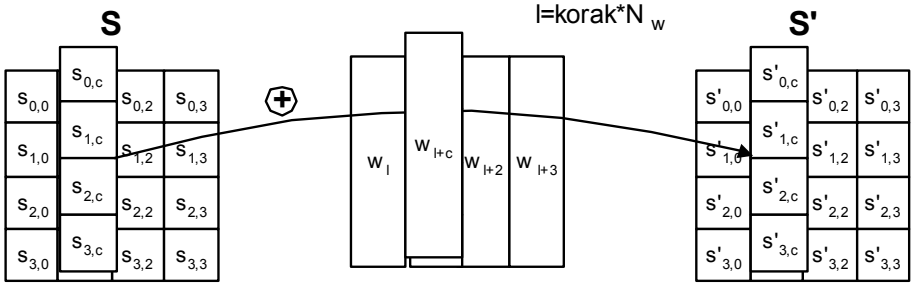
Slika 3: Funkcija miješanje_stupaca() provodi transformaciju svakog stupca matrice stanja

2.3.4. Dodavanje podključa u matricu stanja

Funkcija `dodaj_podključ()` transformira matricu stanja dodavanjem podključa, odnosno izvođenjem XOR operacije nad bitovima. Svaki podključ sastoji se od N_w riječi iz linearnog niza dobivenog ekspanzijom ključa (poglavlje 2.4). Dodavanje tih riječi provodi se na sljedeći način:

$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}]$ XOR $[w_{korak*Nw+c}]$, za $0 \leq c < N_w$

gdje su w_i riječi iz niza opisanog u poglavlju 2.4, dok varijabla `korak` predstavlja korak algoritma i može poprimiti vrijednosti $0 \leq korak \leq N_r$. Prilikom šifriranja, prvo se provodi inicijalno dodavanje podključa (`korak = 0`). Funkcija `dodaj_podključ()` zatim se primjenjuje u sljedećih N_r koraka šifriranja. Slika 4 prikazuje postupak dodavanja podključa u matricu stanja.



Slika 4: Dodavanje podključa izvođenjem XOR operacije nad svakim stupcem matrice stanja

2.4. Ekspanzija ključa

AES algoritam koristi ključ za šifriranje `K` te provodi njegovu ekspanziju da bi generirao odgovarajuće podključeve. Ekspanzijom ključa generiraju se 32-bitne riječi. Točan broj riječi koje se dobivaju ekspanzijom može se izračunati sljedećom formulom:

$N_w * (N_r + 1)$,

gdje, kako je već ranije rečeno, N_w predstavlja duljinu bloka podataka (u 32-bitnim riječima), dok N_r predstavlja broj koraka algoritma. Rezultat ekspanzije ključa jest linearni niz 32-bitnih riječi koje se mogu označiti sa `w[i]`, gdje je `i`:

$0 \leq i < N_w(N_r + 1)$

ALGORITAM ZA EKSPANZIJU IZGLEDA OVAKO:

```
početak
  riječ tmp

  i = 0
```

```
dok je i < Nk
       w[i] = riječ(ključ[4*i], ključ[4*i+1], ključ[4*i+2], ključ[4*i+3])
       i = i+1
kraj

i = Nk

dok je i < Nb*(Nr+1)
       temp = w[i-1]
       ako je (i mod Nk = 0)
              temp = zamijeni_riječ(rotiraj_riječ(temp)) XOR Rconst[i]
       inače ako (Nk > 6 AND (i mod Nk) = 4)
              temp = zamijeni_riječ(temp)
       w[i] = w[i-Nk] XOR temp
       i = i + 1
kraj
kraj
```

Funkcija `zamijeni_riječ()` kao argument uzima 32-bitnu riječ i primjenjuje S-blok (Tablica 2) permutaciju na svaki od četiri okteta te zatim kao izlaz generira 32-bitnu riječ. Funkcija `rotiraj_riječ()` uzima riječ oblika $a_0a_1a_2a_3$, a kao rezultat rotacije vraća riječ $a_1a_2a_3a_0$. Riječ `Rconst` je konstantna za svaki od koraka i sadrži sljedeću vrijednost (ovisnu o koraku): 02^{i-1},00,00,00, gdje indeks i počinje od 1.

Može se uočiti da se prvih Nk riječi ekspandiranog ključa generira iz glavnog ključa za šifriranje. Svaka sljedeća riječ, `w[i]`, dobiva se izvođenjem XOR operacija prethodne riječi `w[i-1]` i riječi N_k pozicija prije, `w[i-Nk]`. Za riječi čiji je indeks višekratnik od N_k, provodi se transformacija od `w[i-1]` prije izvođenja XOR operacije. Transformacija podrazumijeva rotaciju koju slijedi permutacija kroz S-blok, te zatim izvođenje XOR operacije s konstantom koraka `Rconst[i]`.

Vidljivo je da se postupak ekspanzije ključa za 256-bitne ključeve (N_k=8) donekle razlikuje od ekspanzije 128 i 192-bitnih ključeva. Ukoliko je N_k=8 a vrijednost `i-4` je višekratnik od N_k (od broja 8) nad `w[i-1]` provodi se samo permutacija kroz S-blok.

2.5. Dešifriranje

Postupak šifriranja opisan u prethodnim poglavljima može biti invertiran tako da provodi suprotnu funkciju, odnosno dešifriranje. Pri tome se koriste inverzije funkcija `zamjena_okteta()`, `posmak_redaka()` i `mijesanje_stupaca()`, opisanih redom u poglavljima 2.3.1, 2.3.2, 2.3.3. Također, koristi se i nepromijenjena funkcija `dodaj_podključ()` opisana u poglavlju 2.3.4.

Algoritam za dešifriranje može se opisati na sljedeći način:

```
početak
  oktet stanje[4,Nw]

  stanje = ulaz
  dodaj_podključ(stanje, w[Nr*Nw,(Nr+1)*Nw-1])

  za korak = Nr-1 do korak = 1
       inv_posmak_redaka(stanje)
       inv_zamjena_okteta(stanje)
       dodaj_podključ(stanje, w[korak*Nb,(korak+1)*Nb-1])
       inv_mijesanje_stupaca(stanje)
  kraj
  inv_posmak_redaka(stanje)
  inv_zamjena_okteta (stanje)
  dodaj_podključ (stanje,w[0, Nb-1])
```

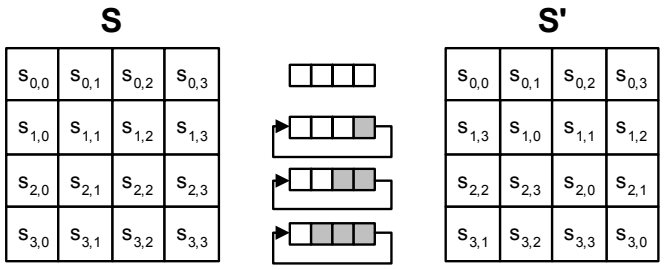
```
  izlaz = stanje
kraj
```

2.5.1. Inverzni posmak redaka

Funkcija `inv_posmak_redaka()` predstavlja inverziju funkcije `posmak_redaka()` (poglavlje 2.3.2). Okteti u zadnja tri retka matrice stanja kružno se posmiču za određeni broj okteta (posmak). Prvi redak se ne posmiče. Donja tri retka se posmiču kružno za N_w-`posmak(r,`N_w`)` okteta, gdje vrijednost `posmak(r,`N_w`)` ovisi o broju retka, kako je dano u poglavlju 2.3.2.
Inverzni posmak vrši se na sljedeći način:

$$stanje'_{(r,s+posmak(r,Nw))\bmod Nw} = stanje'_{r,s}, \text{za } 0 < r < 4 \text{ i } 0 \le s < N_w$$

Slika 5 prikazuje transformaciju matrice stanja koju inducira inverzni posmak redaka.



Slika 5: Rotacija redaka matrice stanja koju uzrokuje funkcija inv_posmak_redaka()

2.5.2. Inverzna zamjena okteta

Funkcija `inv_zamjena_okteta()` predstavlja inverznu supstitucijsku transformaciju u odnosu na funkciju `zamjena_okteta()` (poglavlje 2.3.1), odnosno na svaki oktet tablice stanja primjenjuje se inverzni S-blok.
Tablica 3 prikazuje S-blok (supstitucijske vrijednosti za svaki pojedini oktet u heksadecimalnom obliku) koji se koristi u funkciji `inv_zamjena_okteta()`.

X – četiri viša bita okteta	Y – četiri niža bita okteta															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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

Tablica 3: S-blok – supstitucijske vrijednosti za oktet XY (heksadecimalni prikaz)

2.5.3. Inverzno miješanje podatka u stupcima matrice stanja

Funkcija `inv_miješanje_stupaca()` jest inverzija funkcije `miješanje_stupaca()` (poglavlje 2.3.3). Funkcija vrši transformaciju svakog pojedinog stupca u matrici stanja, promatrajući

svaki stupac kao polinom četvrtog stupnja. Kao i kod funkcije `miješanje_stupaca()`, vrši se multiplikacija s konstantnim polinomom $a^{-1}(x)$ modulo x^4+1. Polinom $a^{-1}(x)$ je sljedećeg oblika:

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

Matrica stanja modificira se na sljedeći način:

$s'(x)=a(x)^{-1}*s(x)$

Odnosno:

$$s'_{0,c}=([0e]\otimes s_{0,c})\oplus([0b]\otimes s_{1,c})\oplus([0d]\otimes s_{2,c})\oplus([09]\otimes s_{3,c})$$
$$s'_{1,c}=([09]\otimes s_{0,c})\oplus([0e]\otimes s_{1,c})\oplus([0b]\otimes s_{2,c})\oplus([0d]\otimes s_{3,c})$$
$$s'_{2,c}=([0d]\otimes s_{0,c})\oplus([09]\otimes s_{1,c})\oplus([0e]\otimes s_{2,c})\oplus([0b]\otimes s_{3,c})$$
$$s'_{3,c}=([0b]\otimes s_{0,c})\oplus([0d]\otimes s_{1,c})\oplus([09]\otimes s_{2,c})\oplus([0e]\otimes s_{3,c})$$

2.5.4. Alternativni način dešifriranja

Opisani način dešifriranja u odnosu na šifriranje mijenja slijed transformacija, dok redoslijed generiranja podključeva za šifriranje i dešifriranje ostaje isti. AES omogućava i drugi način dešifriranja kod kojeg slijed transformacija matrice stanja ostaje isti (uz korištenje inverznih funkcija), a mijenja se redoslijed generiranja podključeva.

3. Sigurnost

AES standard definira tri duljine ključa: 128 bita, 192 bita i 256 bitova. Obzirom na duljinu ključa, razlikuje se i broj koraka koji se izvode tijekom procesa šifriranja odnosno dešifriranja. U ovom trenutku duljine ključeva zadovoljavaju sigurnosne zahtjeve u većini, ako ne i svim primjenama.
Osim toga, AES se temelji na Rijndael algoritmu, koji omogućava šifriranje blokova podataka različitih duljina te primjenu ključeva koji također mogu imati različitu duljinu, što omogućava buduća unaprjeđenja standarda, ukoliko to bude potrebno.
U algoritmu do sad nisu pronađeni nesigurni ili potencijalno nesigurni ključevi (kao npr. kod DES-a).
Nadalje, prilikom razvoja Rijndael algoritma posebna pažnja posvećena je tome da algoritam bude K-siguran i hermetički. K-sigurnost znači da je algoritam siguran ukoliko sve moguće strategije napada na njega imaju očekivanje trajanje i zahtjeve za memorijski prostor identične ili veće u odnosu na ostale algoritme koji šifriraju blokove podataka iste duljine. Hermetičnost znači da algoritam ne sadrži druge slabosti koje nisu prisutne niti u ostalim algoritmima koji koriste ekvivalentne blokove i duljine ključeva.
Iako AES nije osjetljiv na napade korištenjem linearne i diferencijalne kriptoanalize, postoje naznake da je osjetljiv na algebarske napade, a posebno na novu XSL (engl. *eXtended Sparse Linearization*) metodu napada. Pokazuje se da se takvim napadom mogu postići rezultati mnogo bolji nego napadima primjenom sile (engl. *brute force*).

4. Zaključak

AES standard, odnosno Rijndael algoritam, predstavljen je kao novi standard za zaštitu i šifriranje podataka, prihvaćen od strane NIST-a, te se predviđalo da će u najskorije vrijeme postati *de facto* standard svjetski standard za komercijalne aplikacije koje koriste kriptografske metode. Do današnjeg vremena, iako se algoritam pokazuje vrlo sigurnim to se, međutim, još nije dogodilo.
U najnovije vrijeme pronađene su kriptografske metode koje teoretski omogućavaju i razbijanje AES algoritma u vremenu bržem od napada primjenom sile. U ovom trenutku takve napade još uvijek je praktički nemoguće izvesti, no oni pokazuju da ovaj standard ima svoje nedostatke. Za očekivati je da će u budućnosti ipak biti predstavljen drugi standard koji bi trebao ispuniti sve zahtjeve koje postavlja moderna kriptografija.

Dodatak A: Matematička podloga

Svi okteti AES algoritma interpretiraju se kao elementi konačnog polja (Galoisovo polje). Oktet se može prikazati kao niz bitova $b_7b_6b_5b_4b_3b_2b_1b_0$. Okteti se također mogu prikazati kao elementi konačnog polja korištenjem sljedeće notacije:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0 = \sum_{i=0}^{7} b_i x^i$$

Tako se npr. oktet 0110 0011 (ili u heksadecimalnom prikazu 63) može prikazati kao element konačnog polja $x^6 + x^5 + x + 1$.

Zbrajanje

Zbrajanje dvaju elemenata u konačnom polju postiže se izvođenjem XOR ($\oplus$) operacije s odgovarajućim bitovima polinoma. Na primjer:

$$(x^6 + x^4 + x + 1) \oplus (x^7 + x^6 + x^2 + 1) = x^7 + x^4 + x^2 + x$$

Multiplikacija

U polinomskom prikazu, multiplikacija u Galoisovom polju GF(2^8) odgovara multiplikaciji polinoma te primjenom modulo operacije s nedjeljivim polinomom osmog stupnja (polinom je nedjeljiv ukoliko je djeljiv samo s jedinicom i samim sobom). U implementaciji AES algoritma nedjeljivi polinom koji se koristi jest:

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

Npr. multiplikacija okteta $57 \otimes 83 = c1$:

$$(x^6 + x^4 + x^2 + x + 1) \otimes (x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 +$$

$$x^2 + x + x^6 + x^4 + x^2 + x + 1$$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

dalje je:

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1.$$

Modulo operacija korištenjem polinoma $m(x)$ osigurava da će rezultat multiplikacije biti polinom stupnja manjeg od 8, odnosno moći će biti prikazan kao oktet.

Ovako definirana multiplikacija je asocijativna, a konstanta 01 predstavlja multiplikativni identitet. Također, za svaki polinom $b(x)$ različit od nul polinoma može se naći inverzni polinom koji se označava sa $b^{-1}(x)$. Korištenjem proširenog Euklidovog algoritma mogu se izračunati polinomi $a(x)$ i $c(x)$ sa sljedećim svojstvom:

$$b(x)a(x) + m(x)c(x) = 1.$$

Zato je

$$a(x) \otimes b(x) \bmod m(x) = 1,$$

što znači:

$$b^{-1}(x) = a(x) \bmod m(x).$$

Nadalje, za svaki $a(x)$, $b(x)$ i $c(x)$ u polju vrijedi da je:

$$a(x) \otimes (b(x) + c(x)) = a(x) \otimes b(x) + a(x) \otimes c(x)$$

Polinomi s koeficijentima koji su također elementi konačnog polja

Moguće je definirati polinome četvrtog stupnja s koeficijentima koji su također elementi konačnog polja. Polinom

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

može se prikazati kao riječ $a_0,\ldots,a_3$. Ovi polinomi razlikuju se od prethodno definiranih polinoma pošto su koeficijenti u ovom slučaju također elementi konačnog polja; odnosno koeficijenti su u ovom slučaju okteti a ne bitovi.

Zbrajanje ovakvih polinoma četvrtog stupnja identično je zbrajanju polinoma opisanih ranije, odnosno svodi se na izvođenje XOR operacija nad odgovarajućim oktetima u svakoj od riječi.

Uz dane polinome $a(x)$ i $b(x)$, zbrajanje se izvodi na sljedeći način:

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

Multiplikacija polinoma s koeficijentima koji su također elementi konačnog polja donekle se razlikuje i odvija se u dva koraka. U prvom koraku polinomi se algebarski ekspandiraju:

$$c(x) = a(x) \otimes b(x) = c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x^1 + c_0$$

gdje je:

$$c_0 = a_0 \otimes b_0,$$
$$c_1 = a_1 \otimes b_0 \oplus a_0 \otimes b_1,$$
$$c_2 = a_2 \otimes b_0 \oplus a_1 \otimes b_1 \oplus a_0 \otimes b_2,$$
$$c_3 = a_3 \otimes b_0 \oplus a_2 \otimes b_1 \oplus a_1 \otimes b_2 \oplus a_0 \otimes b_3,$$
$$c_4 = a_3 \otimes b_1 \oplus a_2 \otimes b_2 \oplus a_1 \otimes b_3,$$
$$c_5 = a_3 \otimes b_2 \oplus a_2 \otimes b_3,$$
$$c_6 = a_3 \otimes b_3.$$

Rezultirajući polinom `c(x)` ne predstavlja četiri oktetnu riječ, zbog čega je potrebno provesti i drugi korak multiplikacije, odnosno reducirati polinom `c(x)` korištenjem modulo operacije s polinomom četvrtog stupnja. U implementaciji AES algoritma polinom koji se koristi jest:

$$x^4 + 1$$

Tako da je:

$$x^i \bmod (x^4 + 1) = x^{i \bmod 4}.$$

Modularni produkt `a(x)` i `b(x)`, `a(x)*b(x)` jest polinom četvrtog stupnja `d(x)` definiran na sljedeći način:

$$d(x) = d_3 x^3 + d_2 x^2 + d_1 x + d_0,$$

gdje su redom:

$$d_0 = (a_0 \otimes b_0) \oplus (a_3 \otimes b_1) \oplus (a_2 \otimes b_2) \oplus (a_1 \otimes b_3),$$
$$d_1 = (a_1 \otimes b_0) \oplus (a_0 \otimes b_1) \oplus (a_0 \otimes b_2) \oplus (a_3 \otimes b_3),$$
$$d_2 = (a_2 \otimes b_0) \oplus (a_1 \otimes b_1) \oplus (a_0 \otimes b_2) \oplus (a_3 \otimes b_3),$$
$$d_3 = (a_3 \otimes b_0) \oplus (a_2 \otimes b_1) \oplus (a_1 \otimes b_2) \oplus (a_0 \otimes b_3).$$

Ukoliko je `a(x)` fiksni polinom, `d(x)` je moguće definirati i u matričnoj formi na sljedeći način:

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

Pošto x^4+1 nije nedjeljiv na konačnom polju GF(2^8), multiplikacija s fiksnim polinomom četvrtog stupnja nije nužno reverzibilna. Ipak, AES algoritam definira polinom četvrtog stupnja koji ima svoj inverziju:

$$a(x) = [03]x^3 + [01]x^2 + [01]x + [02]$$
$$a^{-1}(x) = [0b]x^3 + [0d]x^2 + [09]x + [0e].$$