Kriptográfia és Információbiztonság
7. előadás

MÁRTON Gyöngyvér

Sapientia Egyetem, Matematika-Informatika Tanszék
Marosvásárhely, Románia
mgyongyi@ms.sapientia.ro

2023

Miről volt szó az elmúlt előadáson?

- az RSA gyorsítása
- RSA: biztonsági problémák
- RSA: az e megválasztása
- RSA: Wiener feltörési algoritmusa
- Az SHA-1 (Secure Hash Algorithm) szerkesztése
- Üzenetintegritás
- Üzenet-hitelesítő kódok (Message Authentication Code): HMAC, CMAC
- Hitelesített titkosítás: GCM

Miről lesz szó?

- véges testek aritmetikája: az AES
- publikus kulcsú rendszerek biztonsága: feltételezések (assumptions)
- kvadratikus (négyzetes) maradékok
- kvadratikus maradék feltételezés
- a Jacobi szimbólum
- a kvadratikus maradék gyökének a meghatározása
- a Rabin titkosító, az SAEP rendszer

Véges testek aritmetikája

AES (Advanced Encryption Standard):

- Daemen és Rijmen, belga kriptográfusok tervezték 1997-ben, 2001 óta NIST standard
- kulcs-mérete: 128, 192, 256 bit, blokk-mérete: 128 bit,
- minden bájtot egy véges test, a $GF(2^8)$ test egy elemeként kezel, ahol az elem tulajdonképpen egy 0 és 1 együtthatójú polinom, például $a_7x^7 + a_6x^6 \ldots a_0$ polinom a $a_7a_6a_5a_4\ a_3a_2a_1a_0$ bájtnak felel meg

  $0x^7 + 1x^6 + 0x^5 + 0x^4 + 1x^3 + 0x^2 + 1x + 0 = x^6 + x^3 + x$ -nek megfelelő bináris alak: 0100 1010.
- a műveleteket a polinomok feletti műveleti szabályok szerint kell végezni, ahol az együtthatók esetében $\pmod 2$ kell számolni
- ha egy művelet elvégzése után nagyobb kitevőt kapunk mint x^7, akkor meg kell határozni az eredmény $m(x)$ szerinti osztási maradékát, ahol $m(x)$ egy 8-ad fokú irreducibilis polinom
- n-ed fokú irreducibilis polinom: olyan polinom, amely nem bontható fel két n-nél kisebb fokú polinom szorzatára
- $GF(2^8)$-ban 30 irreducibilis polinom van, ahol az AES a következővel dolgozik:

  $$x^8 + x^4 + x^3 + x + 1$$

Véges-testek aritmetikája

- az összeadás megfelel a $\oplus$ műveletnek, példa:

$$\begin{array}{ccccc} (x^7+x^3+x+1) & + & (x^3+x) & = & x^7+2x^3+2x+1 = x^7+1 \\ 10001011 & \oplus & 00001010 & = & 10000001 \end{array}$$

- a kivonás ekvivalens az összeadással, mert $1+1=1-1=0$, $1-0=1+0=1$, $0+1=0-1=1$, $0+0=0-0=0$.
- példa szorzásra $GF(2^4)$-ben, ahol legyen az irreducibilis polinom: $m(x)=x^3+x^2+1$:

$$(x^2+x)\cdot(x+1)=x^2+x+1, \text{ mert}$$

$(x^2+x)\cdot(x+1)=x^3+x^2+x^2+x=x^3+x$, amelynek $m(x)$ szerinti osztási maradéka: x^2+x+1.

- a szorzás azonban visszavezethető az x és hatványaival való szorzásra
- az osztáshoz multiplikatív inverzre van szükség, amelyet az adott irreducibilis polinom szerint kell venni, ez pedig a kiterjesztett Eukleidészi algoritmussal határozható meg

Véges-testek aritmetikája

8 biten $\pmod{m(x)}$ szerint az x-el való szorzás egy balra shift, és egy feltételhez kötött konstanssal való $\oplus$ művelet elvégzését jelenti, mert:

$$x^8 \pmod{m(x)} = m(x) - x^8$$

- legyen $f(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$.
- meghatározzuk:
  $xf(x) = (a_7x^8 + a_6x^7 + a_5x^6 + a_4x^5 + a_3x^4 + a_2x^3 + a_1x^2 + a_0x) \pmod{m(x)}$.
- megállapítható:

$$x \cdot f(x) = \begin{cases} (a_6a_5a_4a_3a_2a_1a_00), & \text{ha } a_7 = 0 \\ (a_6a_5a_4a_3a_2a_1a_00) \oplus (m(x) - x^8), & \text{ha } a_7 = 1 \end{cases}$$

$(x^6 + x^3 + x^2 + 1) \cdot (x^7 + x^2 + 1)$ szorzat meghatározásához, meg kell határozni:

$$(x^6 + x^3 + x^2 + 1) \cdot 1 = (01001101) \times (00000001)$$
$$(x^6 + x^3 + x^2 + 1) \cdot x^2 = (01001101) \times (00000010)$$
$$(x^6 + x^3 + x^2 + 1) \cdot x^7 = (01001101) \times (10000000)$$

Véges-testek aritmetikája

Szorzás: határozzuk meg $(x^6 + x^3 + x^2 + 1) \cdot (x^7 + x^2 + 1)$ szorzatot $m(x) = x^8 + x^4 + x^3 + x + 1$ szerint.
Binárisan ez azt jelenti, hogy mennyi: $(01001101) \times (10000101)$?

$$
\begin{aligned}
(01001101) \times (00000001) &= (01001101) \\
(01001101) \times (00000010) &= (10011010) \\
(01001101) \times (00000100) &= (00110100) \oplus (00011011) = (00101111) \\
(01001101) \times (00001000) &= (01011110) \\
(01001101) \times (00010000) &= (10111100) \\
(01001101) \times (00100000) &= (01111000) \oplus (00011011) = (01100011) \\
(01001101) \times (01000000) &= (11000110) \\
(01001101) \times (10000000) &= (10001100) \oplus (00011011) = (10010111)
\end{aligned}
$$

tehát:

$$
\begin{aligned}
&(01001101) \times (10000101) = \\
&(01001101) \times [(00000001) \oplus (00000100) \oplus (10000000)] = \\
&(01001101) \oplus (00101111) \oplus (10010111) = \\
&(11110101)
\end{aligned}
$$

ami megfelel: $x^7 + x^6 + x^5 + x^4 + x^2 + 1$-nek.

Véges-testek aritmetikája

Osztás: szorzunk a multiplikatív inverzzel

- a multiplikatív inverz meghatározására alkalmazzuk a kiterjesztett eukleidészi algoritmust, a polinomok körében.
- a kiterjesztett eukleidészi algoritmus, a polinomok körében meghatározza az $a(x)$ és $b(x)$ polinomok legnagyobb közös osztóját, a $d(x)$-t és azokat az $v(x)$ és $u(x)$ polinomokat, amelyekre fennáll a következő összefüggés:

$$a(x) \cdot v(x) + b(x) \cdot u(x) = d(x).$$

- azt mondjuk, hogy a $d(x)$ polinom az $a(x)$ és $b(x)$ polinomok legnagyobb közös osztója, ha fennáll, hogy $d(x)$ a legnagyobb olyan fokszámú polinom amelyik osztja $a(x)$-t és $b(x)$-et is.
- példa: $gcd(x^6 + x^5 + x^4 + x^2 + x + 1, x^5 + x^4 + x^3 + x^2 + x + 1) = x^3 + 1$.

Véges-testek aritmetikája

Osztás:

- ha az $a(x)$ és $b(x)$ polinomok legnagyobb közös osztója 1, akkor meghatározható az $a(x)$ multiplikatív inverze $b(x)$ szerint
- $x^6 + x^3 + x$ és $m(x) = x^8 + x^4 + x^3 + x + 1$ legnagyobb közös osztója 1 $\Rightarrow$ meghatározható $x^6 + x^3 + x$ multiplikatív inverze.
- $x^6 + x^3 + x$ multiplikatív inverze, $m(x)$ szerint:

$$x^7 + x^5 + x^3 + x + 1, \text{ mert}$$

$$(x^6 + x^3 + x) \cdot (x^7 + x^5 + x^3 + x + 1) = 1 \pmod{m(x)}.$$

A kiterjesztett eukleidészi algoritmus, a polinomok körében

Határozzuk meg $a(x) = x^8 + x^4 + x^3 + x + 1$ és $b(x) = x^6 + x^3 + x$ legnagyobb közös osztóját, majd azokat a $v(x)$ és $u(x)$ polinomokat, amelyekre fennáll a következő összefüggés: $a(x) \cdot v(x) + b(x) \cdot u(x) = 1$.

a	b	q	r	x_0	x_1	y_0	y_1
				1	0	0	1
$x^8 + x^4 + x^3 + x + 1$	$x^6 + x^3 + x$	x^2	$x^5 + x^4 + x + 1$	0	1	1	x^2
$x^6 + x^3 + x$	$x^5 + x^4 + x + 1$	$x + 1$	$x^4 + x^3 + x^2 + x + 1$	1	$x + 1$	x^2	$x^3 + x^2 + 1$
$x^5 + x^4 + x + 1$	$x^4 + x^3 + x^2 + x + 1$	x	$x^3 + x^2 + 1$	$x + 1$	$x^2 + x + 1$	$x^3 + x^2 + 1$	$x^4 + x^3 + x^2 + x$
$x^4 + x^3 + x^2 + x + 1$	$x^3 + x^2 + 1$	x	$x^2 + 1$	$x^2 + x + 1$	$x^3 + x^2 + 1$	$x^4 + x^3 + x^2 + x$	$x^5 + x^4 + 1$
$x^3 + x^2 + 1$	$x^2 + 1$	$x + 1$	x	$x^3 + x^2 + 1$	x^4	$x^5 + x^4 + 1$	$x^6 + x^3 + x^2 + 1$
$x^2 + 1$	x	x	1	$x^4 + x + 1$	$x^5 + x^3 + x^2 + 1$	$x^6 + x^3 + x^2 + 1$	$x^7 + x^5 + x^3 + x + 1$

Tehát:
$(x^8 + x^4 + x^3 + x + 1) \cdot (x^5 + x^3 + x^2 + 1) + (x^6 + x^3 + x) \cdot (x^7 + x^5 + x^3 + x + 1) = 1.$

AES-128 kulcsgenerálás

A $128 = 16 \cdot 8$ bites kulcsot oszloponként, egy 4×4-es kétdimenziós mátrix formába írjuk $\rightarrow RK_0$, majd minden $i = 1, \ldots 10$-re:

- w_0, w_1, w_2, w_3 -al jelölve RK_{i-1} oszlopait, meghatározzuk az nw_0, nw_1, nw_2, nw_3 *szavakat*:
  - $nw_0 = temp \oplus w_0$, ahol *temp*:
    - $rw = \textbf{RotWord}(w_3)$
    - $sw = \textbf{SubWord}(rw)$
    - $rcw = \textbf{Rcon}(i)$
    - $temp = sw \oplus rcw$
  - $nw_1 = nw_0 \oplus w_1$
  - $nw_2 = nw_1 \oplus w_2$
  - $nw_3 = nw_2 \oplus w_3$
- összefűzzük a kapott szavakat: $nw_0||nw_1||nw_2||nw_3 \rightarrow$ 128 bites értékből álló bitszekvencia
- oszloponként, egy 4×4-es kétdimenziós mátrix formába írjuk $\rightarrow RK_i$

AES-128 kulcsgenerálás, példa

- legyen a kulcs: $2b7e1516\ 28aed2a6\ abf71588\ 09cf4f3c$,
- 2 dimenziós mátrixba rendezve:

w_0	w_1	w_2	w_3
$2b$	28	ab	09
$7e$	ae	$f7$	cf
15	$d2$	15	$4f$
16	$a6$	88	$3c$

rw	sw	rcw	$temp$
$cf4f3c09$	$8a84eb01$	01000000	$8b84eb01$

- Az RK_1 szavai:

nw_0	nw_1	nw_2	nw_3
$a0fafe17$	$88542cb1$	$23a33939$	$2a6c7605$

- kétdimenziós mátrixba rendezve:

w_0	w_1	w_2	w_3
$a0$	88	23	$2a$
fa	54	$a3$	$6c$
fe	$2c$	39	76
17	$b1$	39	05

AES-128 kulcsgenerálás, RotWord, SubWord, Rcon

- **RotWord**, balra történő ciklikus eltolást hajt végre:
  $RotWord(a_0\ a_1\ a_2\ a_3) = (a_1\ a_2\ a_3\ a_0)$
- **SubWord**, alkalmazza bájtonként az S-boxot. Ha az átalakítandó bájt például a $4a$, akkor az S-box táblázat 4-ik sora és a-ik oszlopának kereszteződésénél található bájt lesz az új érték: $d6$.
- **Rcon** 4 kimeneti bájtja közül a legjobboldalibb bájt értéke egyenlő x^{i-1} (mod $m(x)$) hatványértékével, a további három bájt értéke, azaz a 3 legbaloldalibb bájt $0x00$ lesz, ahol $m(x) = x^8 + x^4 + x^3 + x + 1$

i	1	2	3	4	5	6	7	8	9	10
$Rcon(i)$	0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80	0x1b	0x36

$x \cdot x^{\mathbf{7}} = x^{\mathbf{8}} \equiv (x^{\mathbf{4}} + x^{\mathbf{3}} + x + 1) \pmod{m(x)} = \mathbf{0001\ 1011} = \mathbf{0}x1b$
$x \cdot (x^{\mathbf{4}} + x^{\mathbf{3}} + x + 1) \equiv (x^{\mathbf{5}} + x^{\mathbf{4}} + x^{\mathbf{2}} + x) \pmod{m(x)} = \mathbf{0011\ 0110} = \mathbf{0}x36$

AES titkosítás

- a $128 = 16 \cdot 8$ bites S bemenetet (*state*-et) oszloponként, egy 4×4-es, kétdimenziós mátrix formába írjuk
- meghatározzuk $SS =$**AddRoundKey**$(S,\ RK_0)$
- minden $i = 1, \ldots 9$-re:
  - $S1 =$ **SubBytes**(SS),
  - $S2 =$ **ShiftRows**$(S1)$,
  - $S3 =$ **MixColumns**$(S2)$,
  - $SS =$ **AddRoundKey**$(S3, RK_i)$,
- az 10. körben:
  - $S1 =$ **SubBytes**(SS),
  - $S2 =$ **ShiftRows**$(S1)$,
  - a titkosító függvény kimenete: **AddRoundKey**$(S2,\ RK_{10})$.

AES titkosítás, AddRoundKey, SubBytes, ShiftRows

- az **AddRoundKey** egy affin művelet, a bemeneti paraméterekre bájtonként alkalmazza a $\oplus$ műveletet,
- **SubBytes** egy nem lineáris helyettesítést végző művelet, a 4×4-es mátrix bájtjaira alkalmazza az S-boxot,
- **ShiftRows** egy lineáris keverő művelet, a 4×4-es mátrixot bájtjaira alkalmazza a következő ciklikus eltolásokat:

s_{11}	s_{12}	s_{13}	s_{14}	$\Rightarrow$	s_{11}	s_{12}	s_{13}	s_{14}
s_{21}	s_{22}	s_{23}	s_{24}		s_{22}	s_{23}	s_{24}	s_{21}
s_{31}	s_{32}	s_{33}	s_{34}		s_{33}	s_{34}	s_{31}	s_{32}
s_{41}	s_{42}	s_{43}	s_{44}		s_{44}	s_{41}	s_{42}	s_{43}

Megjegyzés: a lineáris és affin transzformációk lavina effektust eredményeznek: egyetlen bit változtatása a teljes szekvencia megváltozását vonja maga után.

AES titkosítás, MixColumns

- **MixColumns** egy lineáris keverő művelet, a 4×4-es mátrix bájtjaira alkalmazza a következő átalakításokat:

s_{11}	s_{12}	s_{13}	s_{14}	$\Rightarrow$	ns_{11}	ns_{12}	ns_{13}	ns_{14}
s_{21}	s_{22}	s_{23}	s_{24}		ns_{21}	ns_{22}	ns_{23}	ns_{24}
s_{31}	s_{32}	s_{33}	s_{34}		ns_{31}	ns_{32}	ns_{33}	ns_{34}
s_{41}	s_{42}	s_{43}	s_{44}		ns_{41}	ns_{42}	ns_{43}	ns_{44}

ahol

$$
\begin{aligned}
ns_{1i} &= (0x02 \bullet s_{1i}) \oplus (0x03 \bullet s_{2i}) \oplus s_{3i} \oplus s_{4i} \\
ns_{2i} &= s_{1i} \oplus (0x02 \bullet s_{2i}) \oplus (0x03 \bullet s_{3i}) \oplus s_{4i} \\
ns_{3i} &= s_{1i} \oplus s_{2i} \oplus (0x02 \bullet s_{3i}) \oplus (0x03 \bullet s_{4i}) \\
ns_{4i} &= (0x03 \bullet s_{1i}) \oplus s_{2i} \oplus s_{3i} \oplus (0x02 \bullet s_{4i}).
\end{aligned}
$$

- a $\bullet$ művelet szorzást jelent $\pmod{x^8 + x^4 + x^3 + x + 1}$ szerint a $GF(2^8)$ véges testben.

AES titkosítás, S-box

- az S-box bemenetének bitjeit egy polinom együtthatóinak tekintjük, és meghatározzuk a polinom multiplikatív inverzét $\pmod{x^8 + x^4 + x^3 + x + 1}$ szerint a $GF(2^8)$ véges testben, jelöljük ezt: $b = (b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)$-vel
- ezután egy affin transzformációt alkalmazunk, megkapva az S-box kimeneti bitjeit: $nb = (nb_7\ nb_6\ nb_5\ nb_4\ nb_3\ nb_2\ nb_1\ nb_0)$
  - minden $i = 0, \ldots, 7$ -re

    $nb_i = b_i \oplus b_{i+4 \pmod 8} \oplus b_{i+5 \pmod 8} \oplus b_{i+6 \pmod 8} \oplus b_{i+7 \pmod 8} \oplus c_i$,

    ahol c konstans:

    $$c = (c_7\ c_6\ c_5\ c_4\ c_3\ c_2\ c_1\ c_0) = (0110\ 0011) = 0x63$$

AES titkosítás, S-box példa

Határozzuk meg az S-box $\{4a\} = (0100\ 1010) = x^6 + x^3 + x$-ra alkalmazott értékét.

- $\{4a\}$ multiplikatív inverze: $b = x^7 + x^5 + x^3 + x + 1 = (1010\ 1011)$ lesz, mert $(x^6 + x^3 + x) \cdot (x^7 + x^5 + x^3 + x + 1) = 1 \pmod{x^8 + x^4 + x^3 + x + 1}$,
- a keresett érték

$$nb = (1101\ 0110) = \{d6\}, \text{ mert:}$$

- $nb_0 = b_0 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus c_0 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$
- $nb_1 = b_1 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_0 \oplus c_1 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1$
- $nb_2 = b_2 \oplus b_6 \oplus b_7 \oplus b_0 \oplus b_1 \oplus c_2 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$
- $nb_3 = b_3 \oplus b_7 \oplus b_0 \oplus b_1 \oplus b_2 \oplus c_3 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 0$
- $nb_4 = b_4 \oplus b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus c_4 = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 1$
- $nb_5 = b_5 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus c_5 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$
- $nb_6 = b_6 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus c_6 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$
- $nb_7 = b_7 \oplus b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus c_7 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 1$

AES-128 titkosítás, S-box

	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

AES-128 visszafejtés

- a blokk titkosítókra jellemzően a kör-kulcsokat fordított sorrendbe alkalmazza
- az alkalmazott függvényeket fordított sorrendben veszi a titkosításnál alkalmazott sorrendhez képest
- a SubBytes, ShiftRows, MixColumns, S-box függvények inverzeit használja $\rightarrow$ a titkosítás nem ugyanaz, mint a visszafejtés
  - titkosítási sorrend: SubBytes, ShiftRows, MixColumns, AddRoundkey
  - visszafejtési sorrend: AddRoundkey, InvMixColumns, InvShiftRows, InvSubBytes
- meg kell külön írni a titkosító és visszafejtő függvényt $\rightarrow$ hátrány, de lehet ezt optimalizálni

Publikus kulcsú rendszerek biztonsága

Az alábbi feltételezések/problémák, amelyeket csak informálisan adunk meg, a modern publikus kulcsú kriptorendszerek biztonságát szavatolják:

- faktorizáció feltételezés (factoring assumption): nincs hatékony algoritmus, amely meghatározná egy k-bites összetett szám prím osztóit. Ezen a feltételezésen alapszik az RSA-OAEP, RSA-PSS rendszerek biztonsága.
- kvadratikus (négyzetes) maradék feltételezés (quadratic residue assumption): nincs hatékony algoritmus, amely meghatározná egy kvadratikus maradék gyökét (négyzetgyökét) egy k-bites összetett modulus szerint. Ezen a feltételezésen alapszik az SAEP rendszer biztonsága.
- diszkrét logaritmus feltételezés (discrete logarithm assumption): nincs hatékony algoritmus, amely meghatározná a diszkrét logaritmus értékét egy adott k-bites modulus szerint (a hatványozás során alkalmazott hatványkitevőt). Ezen a feltételezésen alapszik a Diffie-Hellman kulcscsre, stb.
- diszkrét logaritmus feltételezés: nincs hatékony algoritmus, amely meghatározná hogy egy adott elliptikus görbe pontja milyen konstans értékkel volt beszorozva. Ezen a feltételezésen alapszik az elliptikus görbe krptográfia.

Másodfokú kongruenciák, kvadratikus maradékok

Határozzuk meg (mod 11) szerint a számok négyzetét:

$$\begin{array}{ll} 1^2 = 1 & 6^2 = 3 \\ 2^2 = 4 & 7^2 = 5 \\ 3^2 = 9 & 8^2 = 9 \\ 4^2 = 5 & 9^2 = 4 \\ 5^2 = 3 & 10^2 = 1 \end{array}$$

észrevesszük, hogy az alábbi kongruenciák megoldhatóak és minden esetben két megoldás van:

$$\begin{array}{lll} x^2 \equiv 1 \pmod{11}, & \text{megoldások: } 1, 10 \\ x^2 \equiv 4 \pmod{11}, & \text{megoldások: } 2, 9 \\ x^2 \equiv 3 \pmod{11}, & \text{megoldások: } 5, 6 \\ x^2 \equiv 5 \pmod{11}, & \text{megoldások: } 4, 7 \\ x^2 \equiv 9 \pmod{11}, & \text{megoldások: } 3, 8 \end{array}$$

Másodfokú kongruenciák, kvadratikus maradékok

Figyeljük meg, hogy igazak a következők is:

$$
\begin{array}{ll}
1^5 = 1 & 2^5 = 10 = (-1) \pmod{11} \\
4^5 = 1 & 6^5 = 10 = (-1) \pmod{11} \\
3^5 = 1 & 7^5 = 10 = (-1) \pmod{11} \\
5^5 = 1 & 8^5 = 10 = (-1) \pmod{11} \\
9^5 = 1 & 10^5 = 10 = (-1) \pmod{11}
\end{array}
$$

- az $1, 4, 3, 5, 9$ számok kvadratikus maradékok $\pmod{11}$ szerint;
- a $2, 6, 7, 8, 10,$ számok, pedig kvadratikus nemmaradékok

Másodfokú kongruenciák, kvadratikus maradékok

1. értelmezés

Az a számot aszerint nevezzük kvadratikus maradéknak (négyzetes maradék, négyzetes gyök, kvadratikus gyök), illetve kvadratikus nemmaradéknak $\pmod{n}$ *szerint, hogy az* $x^2 \equiv a \pmod{n}$ *kongruencia megoldható-e vagy sem, ahol* $(a, n) = 1$.

Vizsgáljuk a kvadratikus maradékokat egy P prímszám szerint:

- egy a szám akkor és csakis akkor kvadratikus maradék $\pmod{P}$ szerint, ahol P prímszám, ha: $a^{(P-1)/2} \equiv 1 \pmod{P}$
- egy a szám akkor és csakis akkor kvadratikus nemmaradék $\pmod{P}$ szerint, ahol P prímszám, ha: $a^{(P-1)/2} \equiv -1 \pmod{P}$
- ugyanannyi szám kvadratikus maradék, mint amennyi kvadratikus nem maradék, számuk: $\frac{P-1}{2}$, ahol P prímszám
- ha egy a szám kvadratikus maradék $\pmod{P}$ szerint, akkor az $x^2 \equiv a \pmod{P}$ kongruenciának két inkongruens megoldása van, ahol P prímszám,

Kvadratikus maradék feltételezés

Vizsgáljuk a kvadratikus maradékokat egy összetett N szám szerint: legyen $N = P \cdot Q$, ahol P, Q biztonságos prímek (safe prime, Blum integer), azaz $P, Q \equiv 3 \pmod 4$, és N legalább 1024 bites. Ekkor

- N ismeretében nem létezik hatékony algoritmus annak eldöntésére, hogy egy szám kvadratikus maradék vagy sem
- N ismeretében nem létezik hatékony algoritmus egy kvadratikus maradék gyökének (négyzetgyökének) a meghatározására
- P, Q ismeretében létezik hatékony algoritmus annak eldöntésére, hogy egy szám kvadratikus maradék vagy sem: Jacobi szimbólum meghatározásával
- P, Q ismeretében létezik hatékony algoritmus egy kvadratikus maradék gyökének a meghatározására: kínai maradéktétel segítségével

További megjegyzések:

- a kvadratikus maradékok száma N szerint: $(P-1) \cdot (Q-1)/4$
- Ha a P prímszám felírható: $P = 2 \cdot p + 1$ alakba, akkor a p-t Sophie Germain prímnek hívják.

Egy szám kvadratikus maradék-e?

- ha a modulus egy P prímszám, akkor a Legendre szimbólum, $\mathbb{L}_P(a)$ értéke jelzi, hogy az a szám kvadratikus maradék vagy sem $\pmod P$ szerint. A Legendre szimbólumot a következőképpen is jelöljük $\left(\frac{a}{P}\right)$, ahol

$$\mathbb{L}_P(a) = a^{(P-1)/2} \pmod P,$$

$$\mathbb{L}_P(a) = \left(\frac{a}{P}\right) = \begin{cases} 0, & \text{ha } a \equiv 0 \pmod P \\ 1, & \text{ha } a \not\equiv 0 \pmod P \text{ és } \exists\, x: \ a \equiv x^2 \pmod P \\ -1, & \text{ha } a \not\equiv 0 \pmod P \text{ és } \nexists \text{ ilyen } x \end{cases}$$

- ha a modulus egy N összetett szám, ahol $N = P_1^{\alpha_1} \cdot P_2^{\alpha_2} \dots P_n^{\alpha_n}$, akkor ebben az esetben a Jacobi szimbólumot $\mathbb{J}_N(a)$-t definiáljuk, a következőképpen:

$$\mathbb{J}_N(a) = \left(\frac{a}{N}\right) = \left(\frac{a}{P_1}\right)^{\alpha_1} \cdot \left(\frac{a}{P_2}\right)^{\alpha_2} \dots \left(\frac{a}{P_n}\right)^{\alpha_n},$$

- ha $N = P \cdot Q$, akkor:

$$\mathbb{J}_N(a) = \left(\frac{a}{N}\right) = \left(\frac{a}{P}\right) \cdot \left(\frac{a}{Q}\right),$$

Egy szám kvadratikus maradék-e?

- egy $N = P \cdot Q$ összetett szám esetében:
  - ha $\mathbb{J}_P(a) = 1$ és $\mathbb{J}_Q(a) = 1$, akkor $\mathbb{J}_N(a) = 1$ és a kvadratikus maradék (mod N) szerint
  - ha $\mathbb{J}_P(a) = -1$ és $\mathbb{J}_Q(a) = -1$, akkor $\mathbb{J}_N(a) = 1$ és a kvadratikus nemmaradék (mod N) szerint
  - ha $\mathbb{J}_P(a) = 1$ és $\mathbb{J}_Q(a) = -1$, akkor $\mathbb{J}_N(a) = -1$ és a kvadratikus nemmaradék (mod N) szerint
  - ha $\mathbb{J}_P(a) = -1$ és $\mathbb{J}_Q(a) = 1$, akkor $\mathbb{J}_N(a) = -1$ és a kvadratikus nemmaradék (mod N) szerint
- létezik hatékony algoritmus a Legendre, Jacobi szimbólumok meghatározására.

Jacobi szimbólum meghatározása

A Jacobi szimbólum meghatározása a Kvadratikus reciprocitás tétel alapján lehetséges, amely kimondja: (a tételnek több formája is ismert)

1. tétel

Ha P, Q páratlan prímszámok, akkor fennáll: $\left(\frac{P}{Q}\right) \cdot \left(\frac{Q}{P}\right) = (-1)^{\frac{(P-1)\cdot(Q-1)}{4}}$

Fennáll továbbá:

$$\left(\frac{0}{n}\right) = \begin{cases} 1, & \text{ha } n = 1 \\ 0, & \text{ha } n \neq 1 \end{cases} \qquad \left(\frac{2}{n}\right) = \begin{cases} 1, & \text{ha } n \equiv 1 \pmod 8 \\ 1, & \text{ha } n \equiv 7 \pmod 8 \\ -1, & \text{ha } n \equiv 3 \pmod 8 \\ -1, & \text{ha } n \equiv 5 \pmod 8 \end{cases}$$

$$\left(\frac{a}{n}\right) = \left(\frac{a \bmod n}{n}\right), \quad \text{ha } a \geq n$$

$$\left(\frac{a}{n}\right) = \left(\frac{2}{n}\right) \cdot \left(\frac{a \operatorname{div} 2}{n}\right), \quad \text{ha } a \equiv 0 \pmod 2$$

$$\left(\frac{a}{n}\right) = \begin{cases} -1 \cdot \left(\frac{n}{a}\right), & \text{ha } a \equiv 3 \pmod 4 \text{ és } n \equiv 3 \pmod 4 \\ \left(\frac{n}{a}\right), & \text{másképp} \end{cases}$$

Egy szám kvadratikus maradék-e? (NTL)

Feladat: Ellenőrizzük, tíz véletlenszerűen generált a számról, hogy kvadratikus maradék $\pmod N$ szerint, vagy sem, ahol $N = P \cdot Q$.

```
#include <NTL/ZZ.h>
using namespace std;
using namespace NTL;

int main() {
        long k = 16;
        ZZ P, Q, N, a;
        RandomPrime(P, k / 2);
        RandomPrime(Q, k / 2);
        cout << "P: " << P << endl << "Q: " << Q << endl;
        N = P * Q;
        for (int i = 0; i < 10; ++i) {
                a = RandomBnd(N);
                if (GCD(a, N) != 1) continue;
                cout << a << endl;
                long temp1 = Jacobi(a, P);
                long temp2 = Jacobi(a, Q);
                cout << temp1 << " " << temp2 << endl;
                if (temp1 == 1 && temp2 == 1) {
                        cout << "kvadratikus maradek N szerint";
                }
                else cout << "kvadratikus nemmaradek N szerint";
                cout << endl << endl;
        }
}
```

A kvadratikus maradék gyökének a meghatározása

- ha a modulus egy P prímszám, és $P \equiv 3 \pmod 4$, akkor az $x^2 \equiv a \pmod P$ kongruencia megoldása (2 megoldás), azaz a négyzetgyökök meghatározása a következőképpen történik:

$$x \equiv \pm a^{(P+1)/4} \pmod P$$

- Példa:
  - oldjuk meg az $x^2 = 5 \pmod{11}$ kongruenciát.
  - meghatározzuk $5^{(11+1)/4} \equiv 5^3 \equiv 4 \pmod{11}$
  - a kongruencia megoldása: $\pm 4 \pmod{11}$, azaz $4, 7$
  - ellenőrzés: $4^2 \equiv 5 \pmod{11}$ és $7^2 \equiv 5 \pmod{11}$.

A kvadratikus maradék gyökének a meghatározása

- ha a modulus egy összetett N szám és tudjuk, hogy $N = P \cdot Q$ és $P \equiv Q \equiv 3 \pmod 4$, akkor az $x^2 \equiv a \pmod N$, kongruencia megoldását (4 megoldás) a következőképpen határozzuk meg:
- először meghatározzuk a következő értékeket:

$$\begin{aligned} x_P &= a^{(P+1)/4} \pmod P, \\ x_Q &= a^{(Q+1)/4} \pmod Q, \end{aligned}$$

- majd alkalmazzuk a Kínai maradéktételt, ahol $x_1, -x_1, x_2, -x_2$ lesz a 4 lehetséges megoldás, ahol:

$$\begin{aligned} x_1 &= (P^{-1} \cdot P \cdot x_Q + Q^{-1} \cdot Q \cdot x_P) \pmod N, \\ x_2 &= (P^{-1} \cdot P \cdot x_Q - Q^{-1} \cdot Q \cdot x_P) \pmod N \end{aligned}$$

- P^{-1} érték P inverze $\pmod Q$ szerint
- Q^{-1} érték Q inverze $\pmod P$ szerint

A kvadratikus maradék gyökének a meghatározása

Példa:

- legyen $N = P \cdot Q = 2773$, ahol $P = 47$ és $Q = 59$
- oldjuk meg a $x^2 = 17 \pmod{2773}$ kongruenciát
- meghatározzuk P^{-1}, Q^{-1} értékeket:

$$\begin{aligned} P^{-1} &\equiv 54 \pmod{59}, && \text{mert fennáll:} && 47 \cdot 54 \equiv 1 \pmod{59} \\ Q^{-1} &\equiv 4 \pmod{47}, && \text{mert fennáll:} && 59 \cdot 4 \equiv 1 \pmod{47} \end{aligned}$$

- meghatározzuk:

$$\begin{aligned} x_P &= 17^{(P+1)/4} = 17^{12} \equiv 8 \pmod{47} \\ x_Q &= 17^{(Q+1)/4} = 17^{15} \equiv 28 \pmod{59} \\ x_1 &= 54 \cdot 47 \cdot 28 + 4 \cdot 59 \cdot 8 \pmod{2773} = 854 \\ x_2 &= 54 \cdot 47 \cdot 28 - 4 \cdot 59 \cdot 8 \pmod{2773} = 2624 \end{aligned}$$

- a négy megoldás:

$$\begin{aligned} 854, &\quad \text{ellenőrzés:} && 854^2 && \equiv 17 && \pmod{2773} \\ -854 \equiv 1919 \pmod{2773}, &\quad \text{ellenőrzés:} && 1919^2 && \equiv 17 && \pmod{2773} \\ 2624, &\quad \text{ellenőrzés:} && 2624^2 && \equiv 17 && \pmod{2773} \\ -2624 \equiv 149 \pmod{2773} &\quad \text{ellenőrzés:} && 149^2 && \equiv 17 && \pmod{2773} \end{aligned}$$

A kvadratikus maradék gyökének a meghatározása (NTL)

```
int main() {
        SetSeed(to_ZZ((long)time(0)));
        long k = 16;
        ZZ p, q, P, Q, N, a, mP, mQ, invP, invQ, m1, m2, m3, m4;
        //GenGermainPrime(p, k / 2), P = 2 * p + 1;
        //GenGermainPrime(q, k / 2), Q = 2 * q + 1;
        P = 47, Q = 59; // 5, 7, 11, 23, 47, 59, 83
        N = P * Q;
        cout << "P: " << P << " Q: " << Q << " N: " << N << endl << endl;
        for (;;) {
                a = RandomBnd(N);
                if (GCD(a, N) != 1) continue;
                cout << a << endl;
                long temp1 = Jacobi(a, P);
                long temp2 = Jacobi(a, Q);
                cout << temp1 << " " << temp2 << endl;
                if (temp1 == 1 && temp2 == 1) {
                        mP = PowerMod(a % P, (P + 1) / 4, P);
                        mQ = PowerMod(a % Q, (Q + 1) / 4, Q);
                        invP = InvMod(P % Q, Q);
                        invQ = InvMod(Q % P, P);
                        m1 = (P * invP * mQ + Q * invQ * mP) % N;
                        m2 = (P * invP * mQ + N - Q * invQ * mP) % N;
                        m3 = (-m1 + N) % N;
                        m4 = (-m2 + N) % N;
                        cout << m1 << " " << m2 << " " << m3 << " " << m4 << endl << endl;
                        break;
        } } }
```

```
#include <NTL/ZZ.h>
using namespace std;
using namespace NTL;
```

A Rabin titkosító rendszer

- 1979-ben publikálta Michael O. Rabin,
- biztonsága a faktorizációs feltételezésen és a kvadratikus maradékok feltételezésen alapszik
- azt a lehetőséget vizsgálja, amikor az RSA titkosítás során az $e = 2$ értéket választjuk
- ha $e = 2$, akkor nem létezik inverz, mert ϕ mindig páros, ezért $(e, \phi) \neq 1 \Rightarrow$ a visszafejtést más aritmetikai műveletek határozzák meg
- *Gen* a kulcsgeneráló algoritmus: $(P, Q) \xleftarrow{R} Gen(1^k)$, ahol
  - $P \equiv Q \equiv 3 \pmod 4$
  - $n = P \cdot Q$
  - $p_k = (n), s_k = (P, Q)$
  - publikus kulcs: n
  - privát kulcs: (P, Q)
- $Enc_{(n)}$ a rejtjelező algoritmus: $c \leftarrow m^2 \pmod n$,
- $Dec_{(P,Q)}$ a visszafejtő algoritmus: $m \leftarrow c^{\frac{1}{2}} \pmod n$.

A Rabin titkosító rendszer

- a visszafejtés során "négyzetgyököt", számolunk a négyzetes maradék alapján, amelyért alkalmazzuk a kínai maradéktételt:
  - $m_P = c^{(P+1)/4} \pmod{P}$,
  - $m_Q = c^{(Q+1)/4} \pmod{Q}$,
  - meghatározzuk P^{-1}-t, azaz P inverzét $\pmod{Q}$ szerint
  - meghatározzuk Q^{-1}-t, azaz Q inverzét $\pmod{P}$ szerint
  - $m_1 = (P^{-1} \cdot P \cdot m_Q + Q^{-1} \cdot Q \cdot m_P) \pmod{n}$,
  - $m_2 = (P^{-1} \cdot P \cdot m_Q - Q^{-1} \cdot Q \cdot m_P) \pmod{n}$
- a 4 megoldás: $m_1, m_2, m_3 = n - m_1, m_4 = n - m_2$
- a visszafejtés nem egyértelmű: 4 lehetséges visszafejtett szöveg közül kell kiválsztani a megfelelőt,

A Rabin titkosító rendszer, példa

- legyen $P = 11$, és $Q = 31 \Rightarrow n = 341$,
- legyen $m = 42$, a nyílt-szöveg,
- titkosítás: $c = 42^2 = 59 \pmod{341}$,
- visszafejtés:
  - $59^{(11+1)/4} = 9 \pmod{11}$,
  - $59^{(31+1)/4} = 20 \pmod{31}$,
  - meghatározzuk 31 inverzét mod 11 szerint: $31^{-1} = 5$, azaz fennáll: $31 \cdot 5 = 1 \pmod{11}$.
  - meghatározzuk 11 inverzét 31 szerint: $11^{-1} = 17$, azaz fennáll $11 \cdot 17 = 1 \pmod{31}$
  - $m_1 = (31 \cdot 31^{-1} \cdot 9 + 11 \cdot 11^{-1} \cdot 20) = 20 \pmod{341}$,
  - $m_2 = (31 \cdot 31^{-1} \cdot 9 - 11 \cdot 11^{-1} \cdot 20) = 42 \pmod{341}$,
  - $m_3 = 341 - m_1 = 321$,
  - $m_4 = 341 - m_2 = 299$,

A Rabin titkosító, biztonság

- a visszafejtés fenti képletét azért lehet alkalmazni, mert $P \equiv Q \equiv 3 \pmod 4$
- sokkal hatékonyabb, mint az RSA
- megmutatható, hogy a Rabin rendszer feltörése ugyanolyan nehézségű, mint a fakorizációs probléma, ez nem igaz a "textbook" RSA-ra
- az RSA-nál vett feltörési stratégiák egy része itt is alkalmazható
- újabb változata (2001, Boneh) ellenáll a CCA támadásnak (az egyik legerősebb támadási mód), illetve egyértelmű a visszafejtése
- hasonlóan az RSA-hoz, kulcscsere és hitelesítési protokollokban használják

Az SAEP rendszer

- a rendszer alkalmaz egy hash függvényt, legyen ez: $H : \{0,1\}^{l_r} \rightarrow \{0,1\}^{l_H}$,
- $M = \{0,1\}^{l_M}, C = \{0,1\}^{l_H+l_r}, K = \{0,1\}^{l_H+l_r}$, l_H, l_r pozitív egész számok, ahol $l_M < l_H$, és $k = l_H + l_r$,
- tipikus értékek: $k = 1024, l_M = 256, l_r = 640, l_H = 384$
- a $Gen(1^k)$ kulcsgeneráló algoritmus, meghatároz egy $(k+2)$ bites $n = P \cdot Q$ számot, ahol
  - az n legfelsőbb helyértékű két bitje 10,
  - P, Q pedig két $k/2+1$ bites prímszám, úgy hogy: $P \equiv Q \equiv 3 \pmod 4$,
  - nyilvános kulcs: n és titkos kulcs: (P, Q).

Az SAEP rendszer

az $Enc_n(m)$ titkosító algoritmus probabilisztikusan, polinomiális időben meghatározza a c rejtjelezett értéket:

- $x = m||0^{l_H - l_M}$, ahol $||$ összefűzést jelent,
- $r \xleftarrow{R} \{0,1\}^{l_r}$,
- $y = (x \oplus H(r))||r$,
- $c = y^2 \pmod{n}$.

megjegyzés: $y < 2^k < n/2$, ez fontos szerepet játszik a visszafejtés során.

Az SAEP rendszer

A $Dec_{(P,Q)}(c)$ visszafejtő algoritmus, polinomiális időben meghatározza az m értéket, ahol a visszafejtés során számos plusz tesztelést is végzünk, kezdetben, pedig úgy járunk el, ahogy a Rabin visszafejtésnél:

- meghatározzuk:
  - $m_P = c^{(P+1)/4} \pmod{P}$,
  - $m_Q = c^{(Q+1)/4} \pmod{Q}$,
- ellenőrizzük, hogy fennáll-e: $m_P^2 = c \pmod{P}$ vagy $m_Q^2 = c \pmod{Q}$; ha nem áll fenn, akkor visszautasítjuk a kapott c rejtjelezett értéket
- meghatározzuk az m_1, m_2, m_3, m_4 értékeket
  - meghatározzuk P^{-1}-t, azaz P inverzét $\pmod{Q}$ szerint
  - meghatározzuk Q^{-1}-t, azaz Q inverzét $\pmod{P}$ szerint
  - $m_1 = (P^{-1} \cdot P \cdot m_Q + Q^{-1} \cdot Q \cdot m_P) \pmod{n}$,
  - $m_2 = (P^{-1} \cdot P \cdot m_Q - Q^{-1} \cdot Q \cdot m_P) \pmod{n}$
  - $m_3 = n - m_1$
  - $m_4 = n - m_2$

Az SAEP rendszer

a $Dec_{(P,Q)}(c)$ visszafejtő algoritmus, folytatás:

- megvizsgáljuk, hogy az m_1, m_2, m_3, m_4 értékek közül melyek lesznek kisebbek, mint $n/2$; pontosan kettő lesz kisebb, mint $n/2$, jelöljük ezeket y_1, y_2-vel, mert a továbbiakban csak ezekkel dolgozunk
- felírjuk $y_1 = v_1||r_1$ $y_2 = v_2||r_2$, alakba, ahol v_1, v_2 az y_1, illetve y_2 első l_H darab bitje, r_1, r_2 pedig rendre az utolsó l_r darab bitje
- meghatározzuk: $x_1 = v_1 \oplus H(r_1)$, $x_2 = v_2 \oplus H(r_2)$
- felírjuk: $x_1 = m_1||t_1$, $x_2 = m_2||t_2$, ahol m_1, m_2 az x_1 illetve x_2 első l_M darab bitje
- ha t_1, t_2 mindegyike, vagy egyike sem áll csupa nullás bitekből, akkor visszautasítjuk a kapott c értéket
- meghatározzuk az m értékét: az az m_i érték lesz, ahol t_i csupa nullás bitekből áll, $i = 1, 2$.