УДК 003.26:004.056.55 **Кваснiков В.П., Кінзерявий В.М., Гнатюк С.О., Кінзерявий О.М.**

БЛОКОВИЙ СИМЕТРИЧНИЙ КРИПТОАЛГОРИТМ «LUNA»

У даній статті запропоновано новий перспективний блоковий симетричний криптоалгоритм «LUNA». Наведено основні поняття та терміни, що використовуються для опису даного алгоритму, формалізовано основні параметри, операції та процедури. Крім того, визначено коло подальших досліджень даного криптографічного алгоритму, що полягає у перевірці його стійкості до відомих методів криптоаналізу, оцінці швидкісних параметрів та проведенні статистичного тестування.

Ключові слова: симетрична криптографія, блоковий шифр, криптоалгоритм, ключ, шифротекст, процедура розширення ключів, процедура шифрування.

Вступ. Одним із найнадійніших методів захисту інформаційних ресурсів інформаційно-комунікаційних систем (ІКС) є використання криптографічних засобів, основною перевагою яких є охорона безпосередньо самої інформації, а не доступу до неї. Сучасна криптографія містить чотири основні розділи – це симетричні криптосистеми, асиметричні криптосистеми, системи електронного цифрового підпису та системи розподілу ключів. Проте, не зважаючи на різноманітність сучасних криптосистем, далеко не всі володіють необхідним рівнем безпеки для забезпечення високого рівня захищеності електронних інформаційних ресурсів. На сьогодні єдиним доведеним криптографічним методом з абсолютною (безумовною) стійкістю є метод Вернама («одноразового блокноту») [1], а більшість інших володіють лише практичною чи обчислювальною стійкістю. Стійкість таких методів базується на теорії складності і оцінюється виключно на якийсь певний момент часу і послідовно з двох позицій. Перш за все оцінюється: обчислювальна складність повного перебору, а потім – відомі на даний момент уразливості та їх вплив на обчислювальну складність. У кожному конкретному випадку можуть також існувати додаткові критерії оцінки стійкості. Крім того, стрімкий розвиток обчислювальних засобів формулює нові вимоги до стійкості та швидкодії криптосистем – старі криптоалгоритми відходять у історію, а їх місце займають нові, які пройшли певний відбір (наприклад, конкурси AES [2, 3], NESSIE [4], український відкритий конкурс криптографічних алгоритмів [5]) і довели свою спроможність забезпечити захист критичних інформаційних ресурсів на певний період часу у майбутньому.

Метою роботи є розробка ефективного блокового симетричного алгоритму шифрування для забезпечення конфіденційності електронних інформаційних ресурсів, що передаються в ІКС. В основу запропонованого алгоритму були закладені ідеї, що використані в сучасних алгоритмах шифрування Rijndael (AES) [6] та «Калина» [7].

1. Термінологія, що використовується

Біт – двійковий розряд, що може приймати значення «0» або «1». *Байт* – одиниця кількості інформації (частина машинного слова), яка складається із восьми бітів, кожний з яких має свій ваговий коефіцієнт. *Відкритий текст* – блок даних розміром 128 бітів (16 байтів), 256 бітів (32 байта), або 512 бітів (64 байта), що підлягає зашифровуванню, а також блок даних того ж розміру після розшифровування (розміри відкритого тексту й шифротексту збігаються). *Шифротекст* – блок даних розміром 128 бітів (16 байтів), 256 бітів (32 байта), або 512 бітів (64 байта) після зашифровування, або цей же блок даних, що підлягає розшифруванню (розміри відкритого тексту й шифротексту збігаються). *Ключ шифрування (секретний ключ, K)* – блок даних розміром 128 бітів (16 байтів), 256 бітів (32 байта), 512 бітів (64 байта), що використовується в якості встановлюваного секретного параметра в процедурі зашифрування або розшифрування. *Алгоритм шифрування (шифр)* – опис послідовності операцій перетворення відкритого тексту в шифротекст із застосуванням ключа шифрування, або опис відповідного зворотного перетворення. *Довжина блоку* (N_b) – довжина блоку даних відкритого тексту, виражена в 128-бітових елементах (в залежності від довжини відкритого тексту (шифротексту) приймає значення 1, 2 або 4). *Довжина ключа* (N_k) – довжина ключа, виражена в 128-бітових елементах (в залежності від довжини секретного

ключа тексту приймає значення 1, 2 або 4). *Зашифровування* – застосування алгоритму шифрування для одержання шифротексту з відкритого тексту з використанням ключа шифрування. *Розшифрування* – застосування алгоритму шифрування для одержання відкритого тексту із шифротексту з використанням ключа шифрування. *Шифрування* – зашифровування або розшифрування. *Процедура розширення підключів* – формування із ключа шифрування − підключів, для виконання перетворень. *Поточний стан* (*S*) – блок даних, що використовується на проміжних етапах виконання процедури зашифрування або розшифрування (розмір поточного стану збігається з розміром відкритого тексту). *Підключ* (K_i) – блок даних, що отриманий із ключа шифрування в результаті виконання процедури розширення підключів. Розмір підключа збігається з розміром блоку відкритого тексту. *Цикл шифрування (циклове перетворення*) – ітеративна процедура, що здійснює перетворення поточного стану на вході процедури в поточний стан на її виході із застосуванням відповідного підключа. *Кількість циклів шифрування* (N_r) – число циклових перетворень при шифруванні, обумовлене довжиною ключа й розміром блоку. *Таблиця підстановки* – таблиця заміни (підстановки) байтових значень, що реалізує нелінійне перетворення. *Таблиця зворотної підстановки* – таблиці заміни байтових значень, зворотна до таблиці підстановки. <<< – циклічний по бітний зсув вліво. >>> – циклічний по бітний зсув вправо. [+] – додавання 32-бітових слів за модулем 2^{32}. $\oplus$ – побітове додавання двійкових векторів за модулем 2.

2. Опис алгоритму

2.1. Позначення та параметри алгоритму

Подання вхідних і вихідних даних. До вхідних, вихідних даних алгоритму «LUNA» відносяться відкритий текст, шифротекст і ключ шифрування (секретний ключ), які представляються у вигляді байтових або бітових рядків заданої довжини $16 \times N_b$ байтів ($128 \times N_b$ бітів). Байтові рядки довжиною $n = 16 \times N_b$ представляються в такій формі: $B_0, B_1, B_2, B_3, ..., B_{n-1}$. Відповідний цьому бітовий рядок довжиною $8 \times n = 128 \times N_b$ бітів представляється як $v_0, v_1, v_2, ..., v_7, v_8, v_9, ..., v_{15}, ..., v_{8n-8}, v_{8n-7}, ..., v_{8n-1}$, з такою відповідністю:

$B_0 \to v_0, v_1, ..., v_7$, $B_1 \to v_8, v_9, ..., v_{15}$, …, $B_{n-1} \to v_{8n-8}, v_{8n-7}, ..., v_{8n-1}$.

Поточний стан шифру. Заповнення поточного стану. При виконанні зашифрування або розшифрування операції виконуються над двовимірним масивом байт, названим поточним станом шифру. Поточний стан можна представити у вигляді матриці $16 \times N_b$ байтів (16 рядків по N_b байт). Кожен байт у поточному стані має два індекси: номер рядка (*r*, $0 \le r \le 15$) і номер стовпчика (*c*, $0 \le c < N_b$). Кожен байт може бути адресований як $S_{r,c}$ або $S[r,c]$. Перед початком зашифрування відкритий текст копіюється в поточний стан шифру. Після завершення процедури зашифрування шифротекст копіюється з поточного стану.

Для кожного з можливих довжин блоку даних (128, 256 або 512 біт) відкритий текст представляють байтовим рядком $in_0, in_1, ..., in_n$ ($n = 16 \times N_b$, для 128, 256 та 512 бітних блоків даних $N_b = 1$, $N_b = 2$, $N_b = 4$ відповідно), а отриманий шифротекст – $out_0, out_1, ..., out_n$. На рис. 1-3 представлене заповнення поточного стану перед початком зашифрування та після його закінчення в залежності від довжини блоку даних. Аналогічно, при розшифруванні шифротекст позначається байтовим рядком $in_0, in_1, ..., in_n$, а отриманий відкритий текст – $out_0, out_1, ..., out_n$, заповнення відповідає рис. 1-3.

Поліном що не приводиться. Операція лінійного розсіювання алгоритму «LUNA» використовує поліноміальне подання байтів у полі GF(2^8), утворене поліномом, що не приводиться. У якості поліному, що не приводиться, обраний поліном, який використовується в шифрі «Калина» [7]: $m(x) = x^8 + x^4 + x^3 + x^2 + 1$.

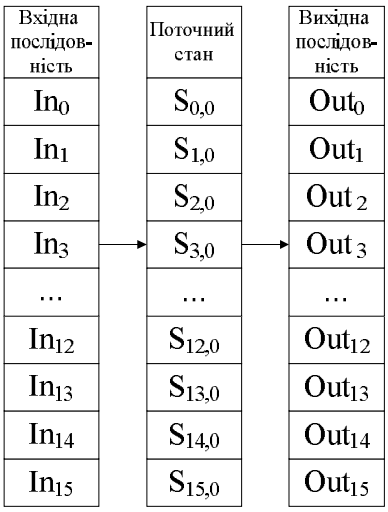

Рис. 1. Заповнення поточного стану для довжини блоку 128 біт

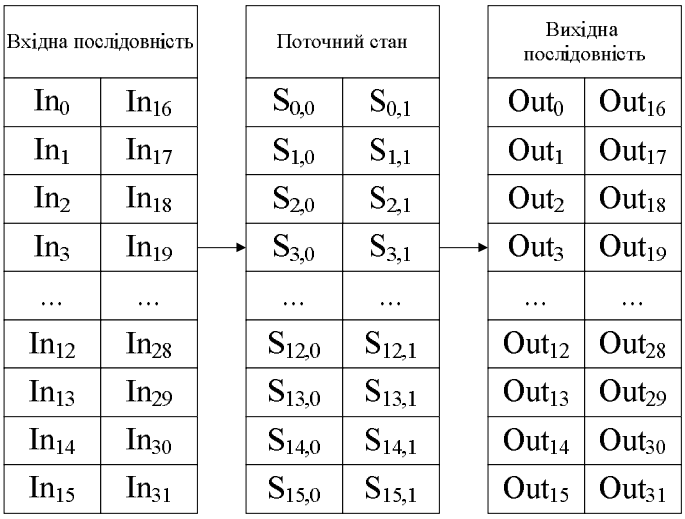

Рис. 2. Заповнення поточного стану для довжини блоку 256 біт

Розмір блоку та довжина ключа. Алгоритм шифрування «LUNA» підтримує довжину блоку в 128, 256 та 512 бітів з підтримкою секретного ключа довжиною 128, 256 та 512 бітів. Допустимі комбінації розміру блоку та довжини ключа шифрування наведені в табл. 1.

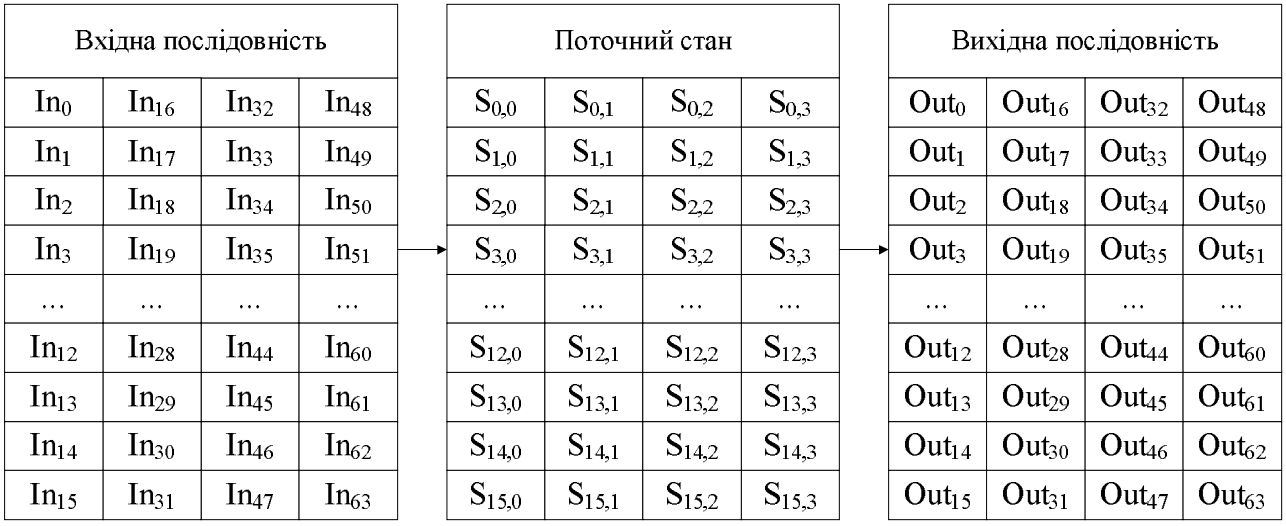

Рис.3. Заповнення поточного стану для довжини блоку 512 біт

Кількість циклів шифрування. Алгоритм шифрування є ітеративною процедурою, що складається з блоків додаткового і зворотного йому перемішування та двох різних ітеративних послідовних перетворень, що шифрують (циклових перетворень). На вхід кожного циклового перетворення подається поточний стан і відповідний підключ. Відкритий текст копіюється в поточний стан перед початком зашифрування, а по його завершенні в поточному стані знаходиться шифротекст. Кількість циклів шифрування (N_r) залежить від довжини секретного ключа (див. табл. 2).

Комбінації розмірів блоку даних та секретного ключа Таблиця 1

Розмір блоку даних, біт	Розмір секретного ключа, біт
128 ($N_b = 1$)	128 ($N_k = 1$), 256 ($N_k = 2$), 512 ($N_k = 4$)
256 ($N_b = 2$)	256 ($N_k = 2$), 512 ($N_k = 4$)
512 ($N_b = 4$)	512 ($N_k = 4$)

Кількість циклів шифрування алгоритму «LUNA» (N_r) від параметрів N_b та N_k Таблиця 2

Довжина блоку(біт)	Розмір ключа (біт)		
	128 ($N_k = 1$)	256 ($N_k = 2$)	512 ($N_k = 4$)
128 ($N_b = 1$)	8	12	20
256 ($N_b = 2$)	–	12	20
512 ($N_b = 4$)	–	–	20

2.2. Процедури зашифрування та розшифрування

Процедура зашифрування. На вхід процедури подається відкритий текст і підключі шифрування. Відкритий текст представляють блоком даних, що описує поточний стан шифру в залежності від параметра N_b (див. рис. 1-3). Над поточним станом спочатку виконують операцію додаткового перемішування даних *(ExtraMix)*, після чого його N_r разів обробляють двома різними цикловими перетвореннями, а на при кінці виконують зворотну операцію до додаткового перемішування даних *(InvExtraMix)*. Між $2 \times N_k$ і $N_r - 2 \times N_k$ цикловими перетвореннями виконуються додатково операції *ExtraMix* та *InvExtraMix* відповідно. Після закінчення зашифровування отриманий шифротекст копіюється з поточного стану шифру. На рис. 4. наведений псевдокод процедури зашифрування.

```
void Luna_Enc( byte in[16][Nb], byte out[16][Nb],
               byte subkey[Nr+4][16][Nb] )
{
  byte state[16][Nb] = in;

  ExtraMix( state, subkey[0] );

  int indexkey = 1;
  for( int round = 1; round <= Nr/2; round++ )
  {
    AddRoundKeyMod2( state, subkey[indexkey] );
    SubBytes( state );
    MixColumns( state );
    indexkey++;

    AddRoundKeyMod32( state, subkey[indexkey] );
    SubBytes( state );
    MixColumns( state );
    indexkey++;

    if( round == Nk )
    {
      ExtraMix( state, subkey[indexkey] );
      indexkey++;
    }

    if( round == ( Nr/2 - Nk ) )
    {
      InvExtraMix( state, subkey[indexkey] );
      indexkey++;
    }
  }

  InvExtraMix( state, subkey[Nr+3] );

  out = state;
}
```

Рис.4. Псевдокод процедури зашифрування

```
void Luna_Dec( byte in[16][Nb], byte out[16][Nb],
               byte subkey[Nr+4][16][Nb] )
{
  byte state[16][Nb] = in;

  ExtraMix( state, subkey[Nr+3] );

  int indexkey = Nr+2;
  for( int round = 1; round <= Nr/2; round++ )
  {
    InvMixColumns( state );
    InvSubBytes( state );
    InvAddRoundKeyMod32( state, subkey[indexkey] );
    indexkey--;

    InvMixColumns( state );
    InvSubBytes( state );
    InvAddRoundKeyMod2( state, subkey[indexkey] );
    indexkey--;

    if( round == Nk )
    {
      ExtraMix( state, subkey[indexkey] );
      indexkey--;
    }

    if( round == ( Nr/2 - Nk ) )
    {
      InvExtraMix( state, subkey[indexkey] );
      indexkey--;
    }
  }

  InvExtraMix( state, subkey[0] );

  out = state;
}
```

Рис.5. Псевдокод процедури розшифрування

Процедури розшифрування. Процедура розшифрування є зворотною до процедури зашифрування. На вхід процедури подається шифротекст і підключі шифрування (підключи подаються у зворотному порядку). Над поточним станом виконуються оборотні зашифруванню циклові перетворення. Після закінчення процедури розшифрування, відкритий текст копіюється з поточного стану шифру. Псевдокод процедури розшифрування наведений на рис. 5.

2.3. Операції, що використовуються при шифруванні

AddRoundKeyMod2, InvAddRoundKeyMod2. Під операцією *AddRoundKeyMod2* мається на увазі побітове додавання за модулем 2 відповідних бітів підключа та поточного стану, а під операцією *InvAddRoundKeyMod2* – їхнє зворотне перетворення. Для виконання операції, підключ представляють у вигляді матриці відповідного розміру аналогічно представленню відкритого тексту у вигляді поточного стану шифру (див. п.2.1). Приклад подання даної операції наведено на рис. 6 для 512-бітного блоку даних. Оскільки операція додавання за модулем 2 є своєю власною інверсію, то операції *AddRoundKeyMod2* та *InvAddRoundKeyMod2* ідентичні.

AddRoundKeyMod32, InvAddRoundKeyMod32. Під операцією *AddRoundKeyMod32* мається на увазі додавання за модулем 2^{32} 32-бітних фрагментів поточного стану та підключа, а під операцією *InvAddRoundKeyMod32* – їхня різниця взята за модулем 2^{32}. Для виконання операції, підключ подається у вигляді матриці відповідного розміру аналогічно представленню відкритого тексту у вигляді поточного стану шифру (див. п.2.1). Поточний стан блоку даних і підключ розбиваються на $4\times N_b$ 32-бітні блоки, причому кожний стовпчик ділиться на чотири 32-бітні фрагменти. Над утвореними блоками поточного стану виконується операція додавання (для *AddRoundKeyMod32*) або віднімання (для *InvAddRoundKeyMod32*) за модулем 2^{32} з відповідним фрагментом ключа (рис. 7). Результат знову розбивається на 8 бітні блоки і підставляється в поточний стан.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,}$	$K_{0,0}$	$K_{0,1}$	$K_{0,2}$	$K_{0,}$	$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,}$	$K_{1,0}$	$K_{1,1}$	$K_{1,2}$	$K_{1,}$	$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,}$	$K_{2,0}$	$K_{2,1}$	$K_{2,2}$	$K_{2,}$	$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,}$
$S_{,0}$	$S_{,1}$	$S_{,2}$	$S_{,}$	$K_{,0}$	$K_{,1}$	$K_{,2}$	$K_{,}$	$S_{,0}$	$S_{,1}$	$S_{,2}$	$S_{,}$
…	…	…	…	…	…	…	…	…	…	…	…
$S_{12,0}$	$S_{12,1}$	$S_{12,2}$	$S_{12,}$	$K_{12,0}$	$K_{12,1}$	$K_{12,2}$	$K_{12,}$	$S_{12,0}$	$S_{12,1}$	$S_{12,2}$	$S_{12,}$
$S_{13,0}$	$S_{13,1}$	$S_{13,2}$	$S_{13,}$	$K_{13,0}$	$K_{13,1}$	$K_{13,2}$	$K_{13,}$	$S_{13,0}$	$S_{13,1}$	$S_{13,2}$	$S_{13,}$
$S_{14,0}$	$S_{14,1}$	$S_{14,2}$	$S_{14,}$	$K_{14,0}$	$K_{14,1}$	$K_{14,2}$	$K_{14,}$	$S_{14,0}$	$S_{14,1}$	$S_{14,2}$	$S_{14,}$
$S_{15,0}$	$S_{15,1}$	$S_{15,2}$	$S_{15,}$	$K_{15,0}$	$K_{15,1}$	$K_{15,2}$	$K_{15,}$	$S_{15,0}$	$S_{15,1}$	$S_{15,2}$	$S_{15,}$

Рис.6. Приклад операції *AddRoundKeyMod2* для блоку даних розміром 512 біт

Наприклад, для виконання операції *AddRoundKeyMod32* над фрагментами $S_{0,0}-S_{0,3}$ та $K_{0,0}-K_{0,3}$ необхідно виконати такі дії: 1) Об'єднати їх у 32-бітні фрагменти: $S(1)=2^{24}\cdot S_{0,3}+2^{16}\cdot S_{0,2}+2^{8}\cdot S_{0,1}+S_{0,0}$, $K(1)=2^{24}\cdot K_{0,3}+2^{16}\cdot K_{0,2}+2^{8}\cdot K_{0,1}+K_{0,0}$. 2) Скласти за модулем 2^{32}: $S(1)=\left(S(1)+K(1)\right)\bmod 2^{32}$. 3) Результат п. 2 розбити на чотири 8-бітні частини і занести в поточний стан: $S_{0,0}=S(1)\bmod 2^{8}$, $S_{1,0}=\left(S(1)/2^{8}\right)\bmod 2^{8}$, $S_{2,0}=\left(S(1)/2^{16}\right)\bmod 2^{8}$, $S_{3,0}=\left(S(1)/2^{24}\right)\bmod 2^{8}$.

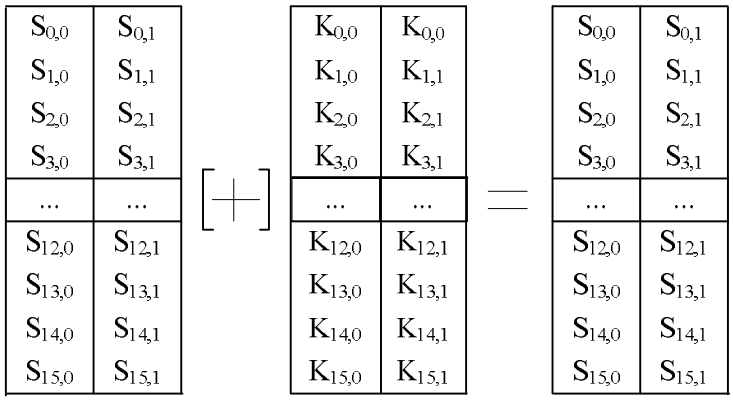

Рис.7. Приклад операції *AddRoundKeyMod32* для блоку даних розміром 256 біт

SubBytes, InvSubBytes. Перетворення *SubBytes* являє собою нелінійну заміну кожного байта поточного стану на відповідне йому значення таблиці підстановок, а *InvSubBytes* зворотне до *SubBytes* перетворення. У алгоритмі шифрування «LUNA» використовується одна таблиця підстановок та одна зворотна таблиця (див. табл. 3-4). Таблиці замін алгоритму шифрування «LUNA» взяті з алгоритму шифрування AES [6]. Виконання заміни байта на байт (через таблицю підстановок) полягає в тому, що поточне значення байта задає адресу в таблиці підстановки (старші 4 біти поточного стану визначають рядок, молодші 4 біти – колонку), по якій необхідно взяти нове значення байта, яке і буде результатом виконання підстановки. Наприклад, якщо потрібно замінити байт = {68} за допомогою табл. 3, то результат заміни необхідно шукати на перетині рядка з індексом «6» та колонки з індексом «8», у результаті отримаємо {45}.

Таблиця підстановок Таблиця 3

	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	1	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	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	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	0e	32	3a	0a	49	6	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	8
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	3	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

Запропоноване значення таблиці підстановок не є обов'язковим для алгоритму шифрування «LUNA» і може змінюватись, проте нові таблиці підстановок потрібно обирати таким чином, щоб вірогідність лінійної та диференціальної характеристики [7-10] була найменшою (оскільки вони суттєво впливають на стійкість проти лінійного та диференціального криптоаналізу [8, 9]). Вибір таблиць підстановок AES для використання в алгоритмі шифрування «LUNA» обумовлено тим, що вірогідність лінійної та диференціальної характеристики тут оптимальна.

Зворотна таблиця підстановок Таблиця 4

	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

MixColumns, InvMixColumns. Операція *MixColumns* являє собою лінійне перетворення блока даних, а *InvMixColumns* – зворотна до *MixColumns* операція. Лінійні перетворення в криптоалгоритмах вирішують завдання розсіювання (поширення кожного вхідного біта на як можна більшу кількість вихідних бітів). Основним показником, що характеризує якість розсіювання є число галузей активізації [7, 10]. Тому, у якості лінійного перетворення були обрані МДВ-коди (коди із максимально допустимою відстанню), що гарантує максимально досяжне число галузей активізації (якщо МДВ-перетворення покриває M байтів, то число галузей активізації $M+1$) [7, 10]. Саме тому було обрано таке перетворення, щоб число галузей активізації дорівнювало 17 (для порівняння в шифрі «Калина» – 9 [7]). Даний показник впливає на стійкість проти лінійного й диференціального криптоаналізу [7, 10].

У алгоритмі шифрування «LUNA» для операцій *MixColumns* та *InvMixColumns* кожна 16-байтна колонка поточного стану розглядається як поліном над полем GF(2^8) з 16 термами. Кожну колонку поточного стану у вигляді полінома перемножують за модулем $x^{16}+1$ з фіксованим поліномом *c(x)* (для *MixColumns*) або з поліномом *d(x)* (для *InvMixColumns*), де

$$c(x) = \{01\}x^{15} + \{3A\}x^{14} + \{37\}x^{13} + \{5F\}x^{12} + \{E2\}x^{11} + \{A6\}x^{10} + \{77\}x^9 + \{69\}x^8 + \{61\}x^7 + \{C2\}x^6 + \{83\}x^5 + \{AA\}x^4 + \{4F\}x^3 + \{2D\}x^2 + \{1F\}x^1 + \{3B\};$$

$$d(x) = \{55\}x^{15} + \{09\}x^{14} + \{EC\}x^{13} + \{9A\}x^{12} + \{F8\}x^{11} + \{F5\}x^{10} + \{F5\}x^9 + \{3B\}x^8 + \{61\}x^7 + \{37\}x^6 + \{AA\}x^5 + \{F2\}x^4 + \{18\}x^3 + \{27\}x^2 + \{DA\}x^1 + \{BA\}.$$

Представлені поліноми *c(x)* та *d(x)* не є обов'язковими для алгоритму шифрування «LUNA» і можуть бути змінені на інші. Рис. 8 показує принцип виконання перетворення *MixColumns* для поточного стану шифру.

Слід зазначити, що при $N_b = 2$ та $N_b = 4$ перед виконанням операцій *MixColumns, InvMixColumns* відбувається перемішування байтів рядків поточного стану за таким алгоритмом: кожен *i*-ий ($\overline{0,...,15}$) рядок циклічно зсувається вліво на $i \bmod N_b$ байтів для операції *MixColumns*, та циклічно зсувається вправо на $(N_b - i) \bmod N_b$ байтів для операції *InvMixColumns*.

ExtraMix, InvExtraMix. Операції *ExtraMix* та *InvExtraMix* використовуються для додаткового перемішування даних. *InvExtraMix* є оборотною до *ExtraMix*. Для виконання цих операцій використовують стандартні операції, що були раніше описані. Для операцій *ExtraMix* та *InvExtraMix*, підключ подається у вигляді матриці відповідного розміру аналогічно представленню відкритого тексту у вигляді поточного стану шифру (див. п.2.1).

Поточний стан блоку даних і підключ розбиваються на $4\times N_b$ 32-бітні блоки, причому кожний стовпчик ділиться на 4 32-бітні фрагменти. За допомогою фрагментів кожного стовпчика підключа відбувається перемішування відповідних фрагментів стовпчиків поточного стану.

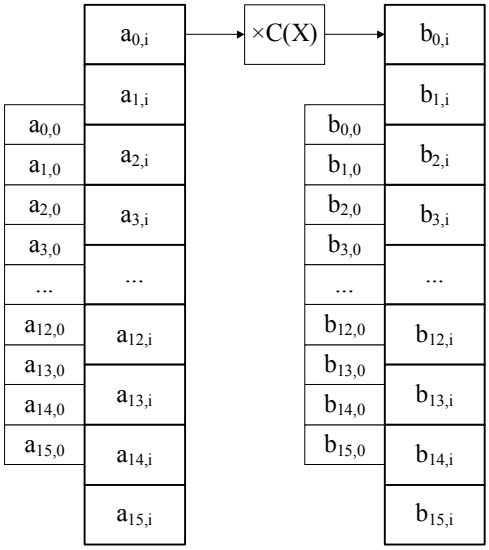

Рис.8. Порядок виконання перетворення *MixColumns* для поточного стану шифру

$$t = a(0),\ u = a(1),$$
$$t = t \oplus k(2),\ u = u \oplus k(3),$$
$$t = t[+]u,\ t = SubBytes(t),$$
$$u = u <<< 7,\ u = u[+]t,$$
$$t = t >>> 1,\ u = SubBytes(u),$$
$$t = t[+]k(0),\ u = u[+]k(1)$$
$$a`(2) = a(2) \oplus t,\ a`(3) = a(3) \oplus u,$$
$$t = a`(3),\ u = a`(2),$$
$$t = t \oplus k(0),\ u = u \oplus k(1),$$
$$t = t[+]u,\ t = SubBytes(t),$$
$$u = u <<< 7,\ u = u[+]t,$$
$$t = t >>> 1,\ u = SubBytes(u),$$
$$t = t[+]k(3),\ u = u[+]k(2)$$
$$a`(1) = a(1) \oplus t,\ a`(0) = a(0) \oplus u.$$

Рис.9. Операція *ExtraMix* над кожним стовпчиком поточного стану

$$t = a`(3),\ u = a`(2),$$
$$t = t \oplus k(0),\ u = u \oplus k(1),$$
$$t = t[+]u,\ t = SubBytes(t),$$
$$u = u <<< 7,\ u = u[+]t,$$
$$t = t >>> 1,\ u = SubBytes(u),$$
$$t = t[+]k(3),\ u = u[+]k(2)$$
$$a(1) = a`(1) \oplus t,\ a(0) = a`(0) \oplus u,$$
$$t = a(0),\ u = a(1),$$
$$t = t \oplus k(2),\ u = u \oplus k(3),$$
$$t = t[+]u,\ t = SubBytes(t),$$
$$u = u <<< 7,\ u = u[+]t,$$
$$t = t >>> 1,\ u = SubBytes(u),$$
$$t = t[+]k(0),\ u = u[+]k(1)$$
$$a(2) = a`(2) \oplus t,\ a(3) = a`(3) \oplus u.$$

Рис.10. Операція *InvExtraMix* над кожним стовпчиком поточного стану

Якщо позначити 32-бітні фрагменти стовпчика поточного стану як $a(3), a(2), a(1), a(0)$, а фрагменти стовпчика підключа як $k(3), k(2), k(1), k(0)$ – то на рис. 9-10 зображено відповідні операції перемішування кожного стовпчика поточного стану за допомогою відповідного стовпчика підключа. Кожен з результуючих 32-бітних фрагменти $a`(3), a`(2), a`(1), a`(0)$ знову розбивається на 8 бітні блоки і підставляється в поточний стан.

2.4. Процедура розширення підключів

Для одержання підключів з секретного ключа використовується процедура розширення ключа. Потрібно отримати $N_r + 4$ підключів розміром $16 \times N_b$ байтів, для цього виконуються стандартні операції, що були раніше описані. На рис. 11 наведений псевдокод процедури отримання підключів, який можна описати так: **1)** Секретний ключ та константи *C1*, *C2*, *C3*, *C4* (значення констант в залежності від параметру N_r наведені на рис. 12) представляють у вигляді матриці розміром $16 \times N_r$ аналогічно поданню відкритого тексту у вигляді поточного стану шифру. **2)** За допомогою секретного ключа і констант *C1*, *C2*, *C3*, *C4* вираховуються допоміжні елементи *u*, *t*, *v*, *m*. **3)** Упродовж $(N_r + 4) + 10$ циклів за допомогою допоміжних елементів *u*, *t*, *v*, *m* обчислюються $(N_r + 4) + 10$ значення псевдоключів з розміром $16 \times N_r$. Після кожного циклу значення елементів *u*, *t*, *v*, *m* змінюються. **4)** $(i + 10)$-й псевдоключ $(i = \overline{0, \ldots, N_r + 3})$ стане K_i-м підключем. Але оскільки розмір згенерованих псевдоключів може не співпадати з розміром вхідного блоку даних при $N_b \le N_r$, то в таких випадках потрібно з псевдоключа взяти лише N_b перших стовпчиків.

Висновки. Таким чином, у даній роботі наведено опис основних параметрів, операцій та процедур запропонованого авторами блокового симетричного криптоалгоритму «LUNA». З огляду на архітектуру та функціональні особливості, даний криптоалгоритм безперечно має перспективу впровадження в існуючі ІКС з метою захисту електронних інформаційних ресурсів. Проте, необхідними передумовами практичного застосування будь-якого шифру є перевірка його стійкості до відомих методів криптоаналізу, оцінка швидкісних характеристик та дослідження його статистичних властивостей. Саме це і окреслює чіткі напрямки подальших досліджень криптоалгоритму «LUNA».

```
void Rozwur_Key( byte Key[16][Nr],
                 byte subkey[Nr+4][16][Nb],
                 byte C1[16][Nr], byte C2[16][Nr],
                 byte C3[16][Nr], byte C4[16][Nr] )
{
  byte tempkey[(Nr+4)+10][16][Nr];
  byte u[16][Nr] = C1, t[16][Nr] = C2,
  byte v[16][Nr] = C3, m[16][Nr] = C4;
  AddRoundKeyMod2( u, Key );
  SubBytes( u );
  AddRoundKeyMod32( t, Key );
  SubBytes( t );
  InvAddRoundKeyMod32( v, Key );
  SubBytes( v );
  ExtraMix( m, Key );
  int indexkey = 0;
  for( int index = 1; index <= (Nr + 4) + 10; index++ )
  {
    AddRoundKeyMod32( u, t );
    SubBytes( u );
    MixColumns( u );
    ExtraMix( u, v );
    tempkey[index] = u;
    ExtraMix ( t, m );
    AddRoundKeyMod32( v, m );
    ExtraMix( m , C1 );
    u = t;
    t = v;
    v = m;
    AddRoundKeyMod2( m, u );
    ExtraMix( m , t );
    if( ( index > 10 )
    { for( int j = 0; j < Nb; j ++ )
      {
        for( int x = 0; x < 16; x ++ )
        {
          subkey[indexkey][x][j] =  tempkey[index][c][j];
          indexkey++;
        }
      }
```

Для $N_r = 1$

C1 = {5454545454545454 5454545454545454 }
C2 = {7272727272727272 7272727272727272 }
C3 = {6363636363636363 6363636363636363}
C4 = {1234512345123451 1234512345123451 }

Для $N_r = 2$

C1 = {5454545454545454 5454545454545454 1616161616161616 1616161616161616 }
C2 = {7272727272727272 7272727272727272 4949494949494949 4949494949494949}
C3 = {6363636363636363 6363636363636363 5252525252525252 5252525252525252 }
C4 = {1234512345123451 1234512345123451 8787878787878787 8787878787878787}

Для $N_r = 4$

C1 = {5454545454545454545454545454545454 1616161616161616161616161616161616 4564564564564564456456456456456 4 7427427427427427742742742742742 7 }
C2 = {7272727272727272727272727272727272 4949494949494949494949494949494949 6586586586586586658658658658658 6 8138138138138138138138138138138 }
C3 = {6363636363636363636363636363636363 5252525252525252525252525252525252 5195195195195195195195195195195195 4784784847847847847847847847847847 }
C4 = {1234512345123451123451234512345123451

```
    }
  }
}
```

Рис.11. Псевдокод процедури розширення підключів

```
878787878787878787878787878787
951951951951951995195195195195
241241212412412424124124124124124 }
```

Рис.12. Значення констант *C1*, *C2*, *C3*, *C4* в залежності від параметру N_r

ЛІТЕРАТУРА:

1. Shannon C. Communication Theory of Secrecy Systems, Bell Systems Technical Journal, 1949. — Vol. 28. — P. 656-715.
2. Randomness Testing of the Advanced Encryption Standard Candidate Algorithms [Electronic resource] : Electronic data. — Gaithersburg, Maryland, USA : NIST, 1999. — Mode of access: World Wide Web. — URL: http://csrc.nist.gov/publications/nistir/ir6390.pdf — Description based on screen.
3. Randomness Testing of the Advanced Encryption Standard Finalist Candidates [Electronic resource] : Electronic data. — Gaithersburg, Maryland, USA : NIST, 2000. — Mode of access: World Wide Web. — URL: http://csrc.nist.gov/publications/nistir/ir6483.pdf — Description based on screen.
4. Конкурс NESSIE (Новые европейские алгоритмы подписи, обеспечения целостности и шифрования) [Электронный ресурс] С.П. Панасенко. — Режим доступа: http://old.cio-world.ru/bsolutions/e-safety/340556.
5. Положення про проведення відкритого конкурсу криптографічних алгоритмів [Електронний ресурс] // Інститут кібернетики ім. В.М. Глушкова НАНУ; ДСТСЗІ. — Режим доступу: http://www.dstszi.gov.ua /dstszi/control/ru/publish/article;jsessionid=EE63A37FEF8F5B34030F1E 38D7247DBC?art_id=48387&cat_id=92733.
6. Advanced Encryption Standard (AES) [Electronic resource] : FIPS 197. — Electronic data (1 file: 279 457 byte). — Gaithersburg, Maryland, USA : NIST, 2001. — Mode of access: World Wide Web. — URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. — Description based on screen.
7. Горбенко І.Д. Перспективний блоковий симетричний шифр «КАЛИНА». Основні положення та специфікація / І.Д. Горбенко, В.І. Долгов, Р.В. Олійников та ін. // Прикладная радиоэлектроника. — 2007. — Т. 6, № 2. — С. 195-208.
8. Lai X. Markov ciphers and differential cryptanalysis / X. Lai, J. Massey, S. Murphy // Advances in Cryptology — EUROCRYPT'91, Proceedings. — Springer Verlag, 1991. — P. 17-38.
9. Matsui M. Linear cryptanalysis methods for DES cipher [Electronic resource] EUROCRYPT, Springer Verlag, 1998. — Mode of access: World Wide Web. — URL: http://www.cs.bgu.ac.il/~crp042/Handouts/Matsui.pdf.
10. Алексейчук А.Н. Оценки практической стойкости блочного шифра «Калина» относительно методов разностного, линейного криптоанализа и алгебраических атак, основанных на гомоморфизмах / А.Н. Алексейчук, Л.В. Ковальчук, Е.В. Скрынник, А.С. Шевцов // Прикладная радиоэлектроника. — 2008. — Т.7, № 3. — С. 203-209.

Надійшла: 14.10.2011

Рецензент: д.т.н., проф. Хорошко