1. Цель работы

Реализовать потоковую систему шифрования ZUC.

2. Описание алгоритма

2.1. ZUC

ZUC — слово-ориентированный потоковый шифр. На вход подается 128-битные ключ и инициализирующий вектор. На выходе получаем 32-битное слово. Данный выходной поток используется для шифрования и расшифрования.

Выполнение алгоритма ZUC можно разделить на 2 режима: инициализирующий и рабочий, которые будут описаны ниже. На рисунке 1 представлена схема алгоритма.

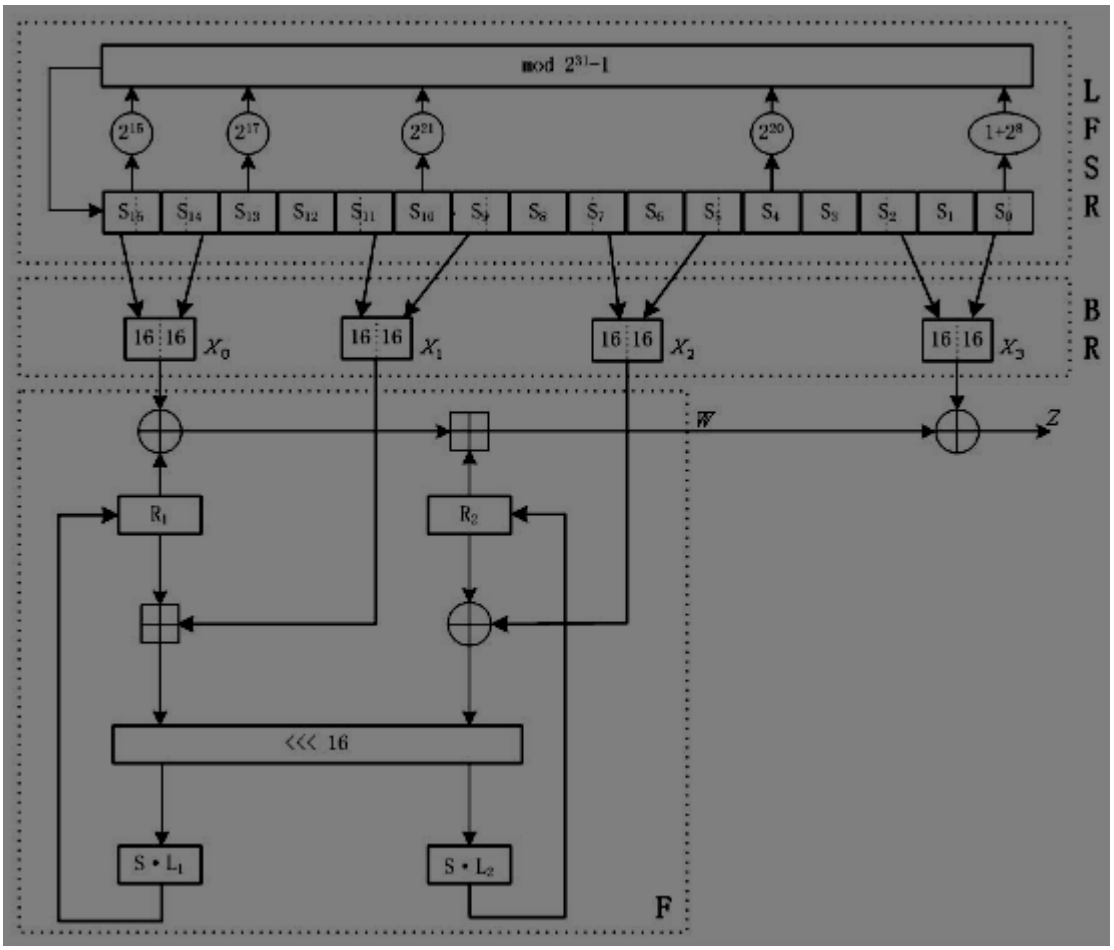


Рис. 1 Схема алгоритма ZUC

Схема имеет три уровня: регистр сдвига с линейной обратной связью (LFSR или РСЛОС), средний уровень — бит-реорганизация (BR), а нижний уровень — представляет из себя нелинейную функцию F.

2.2. Регистр сдвига с линейной обратной связью (РСЛОС)

РСЛОС имеет 16 31-битных ячеек $(s_{0,}s_{1,}\dots,s_{15})$. Каждая ячейка может принимать одно значение из следующего набора $\{1,2,3,\dots,2^{31}-1\}$. Регистр имеет два режима, о которых упоминалось ранее.

Рассмотрим инициализирующий режим.

На вход РСЛОС получает 31-битное слово *u*, полученное удалением младшего бита 32-битного слова *W* из выхода нелинейной функции F, т. е. *u = W >> 1*. Данный режим можно представить в виде следующего псевдокода:

$$v = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1+2^2)s_0 mod(2^{31}-1)$$
$$s_{16} = (v+u) mod(2^{31}-1)$$
$$\text{Если} s_{16} = 0, \text{то} s_{16} = 2^{31}-1$$
$$s_0 = s_{1,} s_1 = s_{2,} s_2 = s_{3,} \dots, s_{15} = s_{16}$$

Рабочий режим отличается тем, что на вход ничего не подается.

$$v = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1+2^2)s_0 mod(2^{31}-1)$$
$$\text{Если} s_{16} = 0, \text{то} s_{16} = 2^{31}-1$$
$$s_0 = s_{1,} s_1 = s_{2,} s_2 = s_{3,} \dots, s_{15} = s_{16}$$

Т.к. операции выполняются над двоичным полем, то умножение на соответствующие степени двойки можно выполнить следующим образом.

$$v = (s_{15} \ll 15) + (s_{13} \ll 17) + (s_{10} \ll 21) + (s_4 \ll 20) + (s_0 \ll 8) + s_0 mod(2^{31}-1)$$

2.3. Бит-реорганизация

На данном уровне происходи извлечение из РСЛОС 4 32-битных слова, которые получаются следующим образом.

$$x_0 = ((s_{15} \wedge 0x7FFF8000) \ll 1) + (s_{14} \wedge 0xFFFF)$$
$$x_1 = ((s_{11} \wedge 0xFFFF) \ll 16) + (s_9 \gg 15)$$
$$x_2 = ((s_7 \wedge 0xFFFF) \ll 16) + (s_5 \gg 15)$$
$$x_3 = ((s_2 \wedge 0xFFFF) \ll 16) + (s_0 \gg 15)$$

Первые три значения используются в функции F, а последнее будет сложено по модулю 2 с результатом этой функции.

2.4. Нелинейная функция F

Данная функция имеет 2 ячейки памяти R_1и R_2, которые на начальном этапе равны нулю. На вход функции F подаются три числа $x_{0,}x_{1,}x_2$, которые были описаны ранее (п. 2.3). Далее представлен псевдокод функции F.

$$W = (x_0 \otimes R_1) + R_2 mod 2^{32}$$
$$W_1 = R_1 + X_1 mod 2^{32}$$
$$W_2 = R_2 \otimes X_2$$
$$R_1 = S(L_1((W_1 \ll 16) + (W_2 \gg 16)))$$
$$R_1 = S(L_2((W_2 \ll 16) + (W_1 \gg 16)))$$

S — представляет собой фиксированную таблицу размером 32×32.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	3E	72	5B	47	CA	E0	00	33	04	D1	54	98	09	B9	6D	CB
1	7B	1B	F9	32	AF	9D	6A	A5	B8	2D	FC	1D	08	53	03	90
2	4D	4E	84	99	E4	CE	D9	91	DD	B6	85	48	8B	29	6E	AC
3	CD	C1	F8	1E	73	43	69	C6	B5	BD	FD	39	63	20	D4	38
4	76	7D	B2	A7	CF	ED	57	C5	F3	2C	BB	14	21	06	55	9B
5	E3	EF	5E	31	4F	7F	5A	A4	0D	82	51	49	5F	BA	58	1C
6	4A	16	D5	17	A8	92	24	1F	8C	FF	D8	AE	2E	01	D3	AD
7	3B	4B	DA	46	EB	C9	DE	9A	8F	87	D7	3A	80	6F	2F	C8
8	B1	B4	37	F7	0A	22	13	28	7C	CC	3C	89	C7	C3	96	56
9	07	BF	7E	F0	0B	2B	97	52	35	41	79	61	A6	4C	10	FE
A	BC	26	95	88	8A	B0	A3	FB	C0	18	94	F2	E1	E5	E9	5D
B	D0	DC	11	66	64	5C	EC	59	42	75	12	F5	74	9C	AA	23
C	0E	86	AB	BE	2A	02	E7	67	E6	44	A2	6C	C2	93	9F	F1
D	F6	FA	36	D2	50	68	9E	62	71	15	3D	D6	40	C4	E2	0F
E	8E	83	77	6B	25	05	3F	0C	30	EA	70	B7	A1	E8	A9	65
F	8D	27	1A	DB	81	B3	A0	F4	45	7A	19	DF	EE	78	34	60

Рис. 2 Таблица S_0

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	55	C2	63	71	3B	C8	47	86	9F	3C	DA	5B	29	AA	FD	77
1	8C	C5	94	0C	A6	1A	13	00	E3	A8	16	72	40	F9	F8	42
2	44	26	68	96	81	D9	45	3E	10	76	C6	A7	8B	39	43	E1
3	3A	B5	56	2A	C0	6D	B3	05	22	66	BF	DC	0B	FA	62	48
4	DD	20	11	06	36	C9	C1	CF	F6	27	52	BB	69	F5	D4	87
5	7F	84	4C	D2	9C	57	A4	BC	4F	9A	DF	FE	D6	8D	7A	EB
6	2B	53	D8	5C	A1	14	17	FB	23	D5	7D	30	67	73	08	09
7	EE	B7	70	3F	61	B2	19	8E	4E	E5	4B	93	8F	5D	DB	A9
8	AD	F1	AE	2E	CB	0D	FC	F4	2D	46	6E	1D	97	E8	D1	E9
9	4D	37	A5	75	5E	83	9E	AB	82	9D	B9	1C	E0	CD	49	89
A	01	B6	BD	58	24	A2	5F	38	78	99	15	90	50	B8	95	E4
B	D0	91	C7	CE	ED	0F	B4	6F	A0	CC	F0	02	4A	79	C3	DE
C	A3	EF	EA	51	E6	6B	18	EC	1B	2C	80	F7	74	E7	FF	21
D	5A	6A	54	1E	41	31	92	35	C4	33	07	0A	BA	7E	0E	34
E	88	B1	98	7C	F3	3D	60	6C	7B	CA	D3	1F	32	65	04	28
F	64	BE	85	9B	2F	59	8A	D7	B0	25	AC	AF	12	03	E2	F2

Рис. 3 Таблица S_1

Преобразования L_1, L_2 выглядят следующим образом.

$$L_1(x) = x \otimes (x \lll_{32} 2) \otimes (x \lll_{32} 10) \otimes (x \lll_{32} 18) \otimes (x \lll_{32} 24)$$
$$L_2(x) = x \otimes (x \lll_{32} 8) \otimes (x \lll_{32} 14) \otimes (x \lll_{32} 22) \otimes (x \lll_{32} 30)$$

2.5. Начальная инициализация

После загрузки ключа и инициализирующего вектора, необходимо задать начальное состояние регистру, но для этого еще понадобится таблица фиксированных векторов:

$d_0 = 0x44\text{DF}$
$d_1 = 0x26\text{BC}$
$d_2 = 0x626\text{B}$
$d_3 = 0x135\text{E}$
$d_4 = 0x5789$
$d_5 = 0x35\text{E}2$
$d_6 = 0x7135$
$d_7 = 0x09\text{AF}$
$d_8 = 0x4\text{D}78$
$d_9 = 0x2\text{F}13$
$d_{10} = 0x6\text{BC}4$
$d_{11} = 0x1\text{AF}1$
$d_{12} = 0x5\text{E}23$
$d_{13} = 0x3\text{C}4\text{D}$
$d_{14} = 0x789\text{A}$
$d_{15} = 0x47\text{AC}$

Таким образом, начальное состояние регистра определяется следующим видом:

$s_i = (key_i \ll 23) + (d_i \ll 8) + initVector_i$

3. Описание реализации алгоритма

3.1. Класс Zuc

Данный класс является основным. Содержит методы:

`public void loadKey(String keyFile);`

Метод предназначен для загрузки ключа и инициализирующего вектора, которые должны быть записаны в отдельных строках.

`public void encrypt(String input, String output);`

Метод для шифрования. В параметрах указывается название файла, откуда считать данные и куда записать результат. Файл побайтово считывается и складывается по модулю 2 с результатом потока.

```
int stream = register.generateStream();
byte[] z = this.split(stream);
int m, c, index = 0;
long fileSize = file.length();
for (int i = 0; i < fileSize;) {
            if (index < 4) {
                  m = in.read();
                  c = m ^ z[index];
                  index++;     i++;
                  out.write((byte) c);
            } else {
                  stream = register.generateStream();
                  z = this.split(stream);   index = 0;
            }
}
```

`public void decrypt(String input, String output);`

Метод для расшифровывания. В качестве параметра указывается название файла, откуда считать данные и куда необходимо записать результат. Полностью совпадает с функцией шифрования. Но чтобы расшифровать, необходимо установить регистр в такое состояние, с которого начиналось шифрование.

3.2. Класс LFSR

Данный класс предназначен для работы с регистром. В конструктор передаются массив байт ключа и инициализирующего вектора. Содержит следующие методы:

`public void initMode(String uVector);`

Метод необходим для инициализирующего режима. В качестве параметра передается вектор, полученный на выходе функции F. После чего выполняется обновление состояния регистра.

```
int fVector = (int)(((long) this.registeresState[15] << 15) +
                                ((long) this.registeresState[13] << 17) +
                                ((long) this.registeresState[10] << 21) +
                                ((long) this.registeresState[4]  << 20) +
                                ((long) this.registeresState[0]  << 8 ) +
                                ((long) this.registeresState[0])) % 0x7fffffff;
if (fVector == 0) {
            fVector = 0x7fffffff;
}
this.statesUpdate(fVector);
```

`public void statesUpdete(int vector);`

Метод обновляет состояние регистра путем сдвига каждого значения в соседнее. В последнюю ячейку записывается аргумент метода.

```
for (int i = 0; i < 15; i++) {
            this.registeresState[i] = this.registeresState[i+1];
}
this.registeresState[15] = vector;
```

`public void workMode();`

Метод необходим для рабочего режима. Аналогичен инициализирующему методу. Не имеет входных аргументов.

`public int generateStream();`

Метод возвращает получаемые на выходе генератора значения.

```
this.workMode();
int[] x = BitReorganization.run(registeresState);
int z = f.run(x) ^ x[3];
return z;
```

3.3. Класс BitReorganization

Класс предназначен для реорганизации бит. Содержит единственный статический метод, выполняющий операции, описанные в п. 2.3. Возвращает массив из полученных значений.

3.4. Класс FFunction

Класс является аналогом нелинейной функции F. Содержит методы, описанные в п. 2.4. На вход подается массив значений, полученный в результате работы метода класса п. 3.3. Возвращает 32-битное слово.

4. Примеры работы программы

В качестве примера возьмем текстовый файл. Ключ укажем как «0123456789abcdef», а инициализирующий вектор - «fedcba9876543210». Результат работы алгоритма приведены на рисунках ниже.

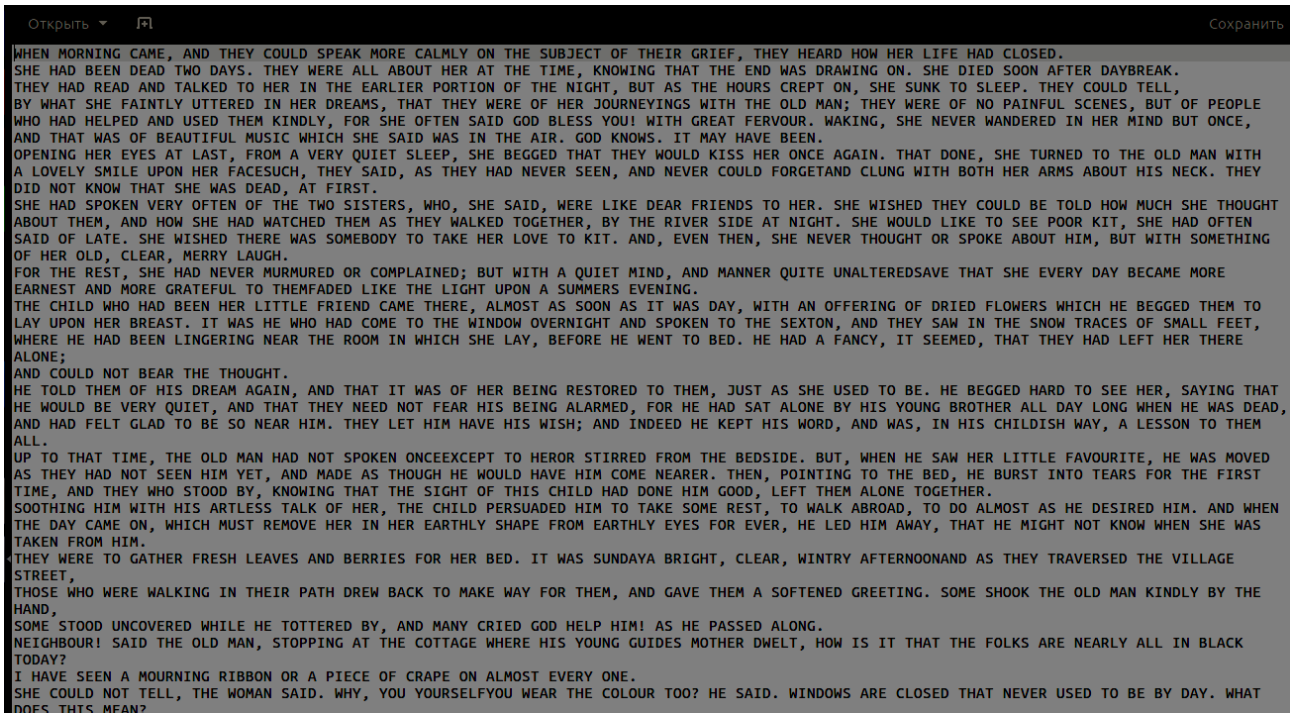


Рис. 4 Пример входных данных

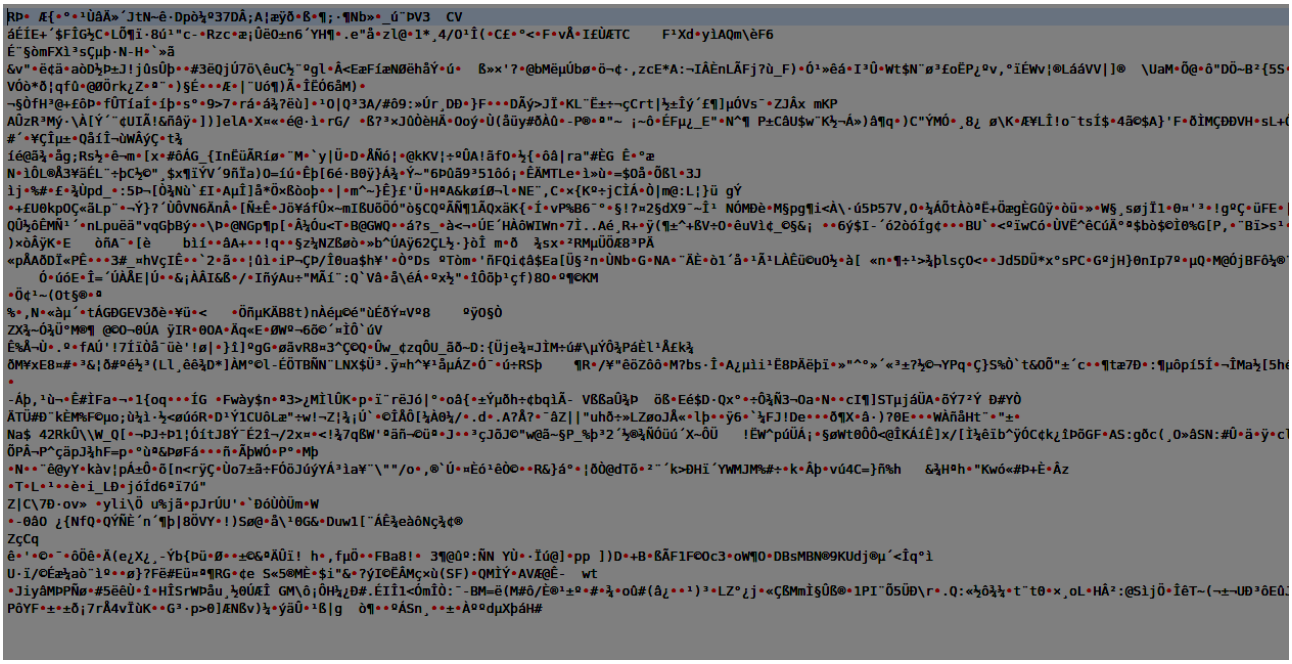
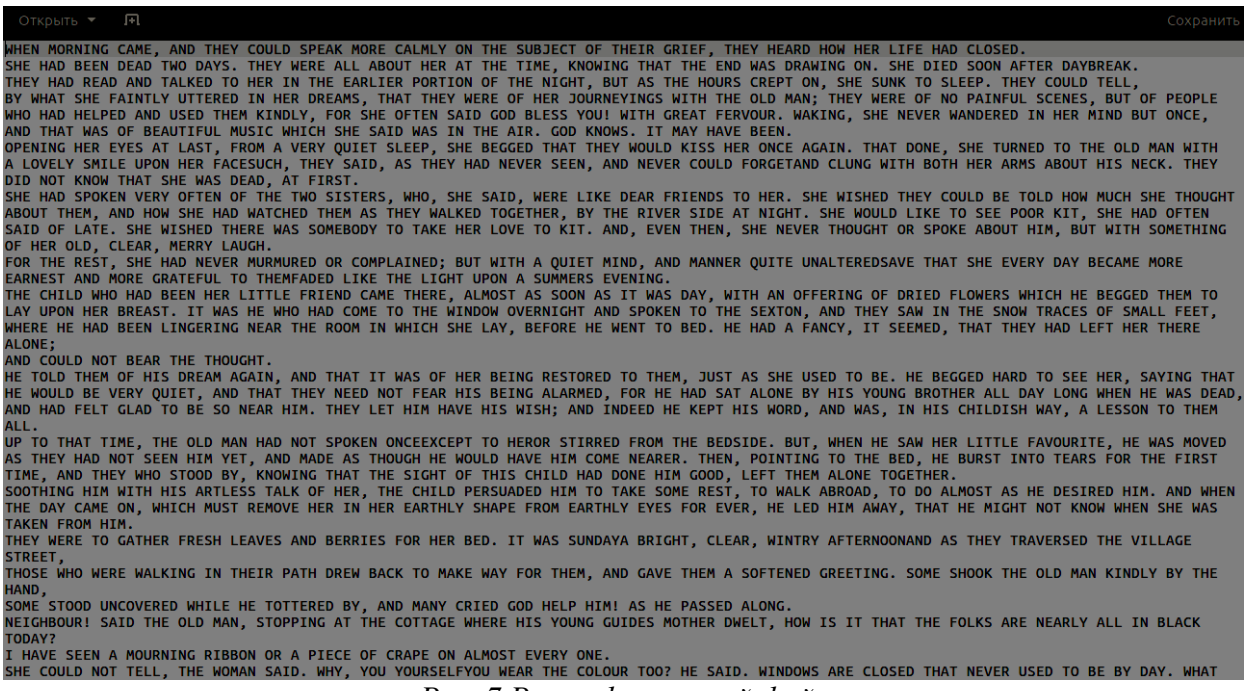


Рис. 6 Зашифрованный текст

Расшифрованный файл совпадает с исходным.



WHEN MORNING CAME, AND THEY COULD SPEAK MORE CALMLY ON THE SUBJECT OF THEIR GRIEF, THEY HEARD HOW HER LIFE HAD CLOSED.
SHE HAD BEEN DEAD TWO DAYS. THEY WERE ALL ABOUT HER AT THE TIME, KNOWING THAT THE END WAS DRAWING ON. SHE DIED SOON AFTER DAYBREAK.
THEY HAD READ AND TALKED TO HER IN THE EARLIER PORTION OF THE NIGHT, BUT AS THE HOURS CREPT ON, SHE SUNK TO SLEEP. THEY COULD TELL,
BY WHAT SHE FAINTLY UTTERED IN HER DREAMS, THAT THEY WERE OF HER JOURNEYINGS WITH THE OLD MAN; THEY WERE OF NO PAINFUL SCENES, BUT OF PEOPLE
WHO HAD HELPED AND USED THEM KINDLY, FOR SHE OFTEN SAID GOD BLESS YOU! WITH GREAT FERVOUR. WAKING, SHE NEVER WANDERED IN HER MIND BUT ONCE,
AND THAT WAS OF BEAUTIFUL MUSIC WHICH SHE SAID WAS IN THE AIR. GOD KNOWS. IT MAY HAVE BEEN.
OPENING HER EYES AT LAST, FROM A VERY QUIET SLEEP, SHE BEGGED THAT THEY WOULD KISS HER ONCE AGAIN. THAT DONE, SHE TURNED TO THE OLD MAN WITH
A LOVELY SMILE UPON HER FACESUCH, THEY SAID, AS THEY HAD NEVER SEEN, AND NEVER COULD FORGETAND CLUNG WITH BOTH HER ARMS ABOUT HIS NECK. THEY
DID NOT KNOW THAT SHE WAS DEAD, AT FIRST.
SHE HAD SPOKEN VERY OFTEN OF THE TWO SISTERS, WHO, SHE SAID, WERE LIKE DEAR FRIENDS TO HER. SHE WISHED THEY COULD BE TOLD HOW MUCH SHE THOUGHT
ABOUT THEM, AND HOW SHE HAD WATCHED THEM AS THEY WALKED TOGETHER, BY THE RIVER SIDE AT NIGHT. SHE WOULD LIKE TO SEE POOR KIT, SHE HAD OFTEN
SAID OF LATE. SHE WISHED THERE WAS SOMEBODY TO TAKE HER LOVE TO KIT. AND, EVEN THEN, SHE NEVER THOUGHT OR SPOKE ABOUT HIM, BUT WITH SOMETHING
OF HER OLD, CLEAR, MERRY LAUGH.
FOR THE REST, SHE HAD NEVER MURMURED OR COMPLAINED; BUT WITH A QUIET MIND, AND MANNER QUITE UNALTEREDSAVE THAT SHE EVERY DAY BECAME MORE
EARNEST AND MORE GRATEFUL TO THEMFADED LIKE THE LIGHT UPON A SUMMERS EVENING.
THE CHILD WHO HAD BEEN HER LITTLE FRIEND CAME THERE, ALMOST AS SOON AS IT WAS DAY, WITH AN OFFERING OF DRIED FLOWERS WHICH HE BEGGED THEM TO
LAY UPON HER BREAST. IT WAS HE WHO HAD COME TO THE WINDOW OVERNIGHT AND SPOKEN TO THE SEXTON, AND THEY SAW IN THE SNOW TRACES OF SMALL FEET,
WHERE HE HAD BEEN LINGERING NEAR THE ROOM IN WHICH SHE LAY, BEFORE HE WENT TO BED. HE HAD A FANCY, IT SEEMED, THAT THEY HAD LEFT HER THERE
ALONE;
AND COULD NOT BEAR THE THOUGHT.
HE TOLD THEM OF HIS DREAM AGAIN, AND THAT IT WAS OF HER BEING RESTORED TO THEM, JUST AS SHE USED TO BE. HE BEGGED HARD TO SEE HER, SAYING THAT
HE WOULD BE VERY QUIET, AND THAT THEY NEED NOT FEAR HIS BEING ALARMED, FOR HE HAD SAT ALONE BY HIS YOUNG BROTHER ALL DAY LONG WHEN HE WAS DEAD,
AND HAD FELT GLAD TO BE SO NEAR HIM. THEY LET HIM HAVE HIS WISH; AND INDEED HE KEPT HIS WORD, AND WAS, IN HIS CHILDISH WAY, A LESSON TO THEM
ALL.
UP TO THAT TIME, THE OLD MAN HAD NOT SPOKEN ONCEEXCEPT TO HEROR STIRRED FROM THE BEDSIDE. BUT, WHEN HE SAW HER LITTLE FAVOURITE, HE WAS MOVED
AS THEY HAD NOT SEEN HIM YET, AND MADE AS THOUGH HE WOULD HAVE HIM COME NEARER. THEN, POINTING TO THE BED, HE BURST INTO TEARS FOR THE FIRST
TIME, AND THEY WHO STOOD BY, KNOWING THAT THE SIGHT OF THIS CHILD HAD DONE HIM GOOD, LEFT THEM ALONE TOGETHER.
SOOTHING HIM WITH HIS ARTLESS TALK OF HER, THE CHILD PERSUADED HIM TO TAKE SOME REST, TO WALK ABROAD, TO DO ALMOST AS HE DESIRED HIM. AND WHEN
THE DAY CAME ON, WHICH MUST REMOVE HER IN HER EARTHLY SHAPE FROM EARTHLY EYES FOR EVER, HE LED HIM AWAY, THAT HE MIGHT NOT KNOW WHEN SHE WAS
TAKEN FROM HIM.
THEY WERE TO GATHER FRESH LEAVES AND BERRIES FOR HER BED. IT WAS SUNDAYA BRIGHT, CLEAR, WINTRY AFTERNOONAND AS THEY TRAVERSED THE VILLAGE
STREET,
THOSE WHO WERE WALKING IN THEIR PATH DREW BACK TO MAKE WAY FOR THEM, AND GAVE THEM A SOFTENED GREETING. SOME SHOOK THE OLD MAN KINDLY BY THE
HAND,
SOME STOOD UNCOVERED WHILE HE TOTTERED BY, AND MANY CRIED GOD HELP HIM! AS HE PASSED ALONG.
NEIGHBOUR! SAID THE OLD MAN, STOPPING AT THE COTTAGE WHERE HIS YOUNG GUIDES MOTHER DWELT, HOW IS IT THAT THE FOLKS ARE NEARLY ALL IN BLACK
TODAY?
I HAVE SEEN A MOURNING RIBBON OR A PIECE OF CRAPE ON ALMOST EVERY ONE.
SHE COULD NOT TELL, THE WOMAN SAID. WHY, YOU YOURSELFYOU WEAR THE COLOUR TOO? HE SAID. WINDOWS ARE CLOSED THAT NEVER USED TO BE BY DAY. WHAT

Рис. 7 Расшифрованный файл

Размеры файлов в течение все работы приведены на рисунке 8.

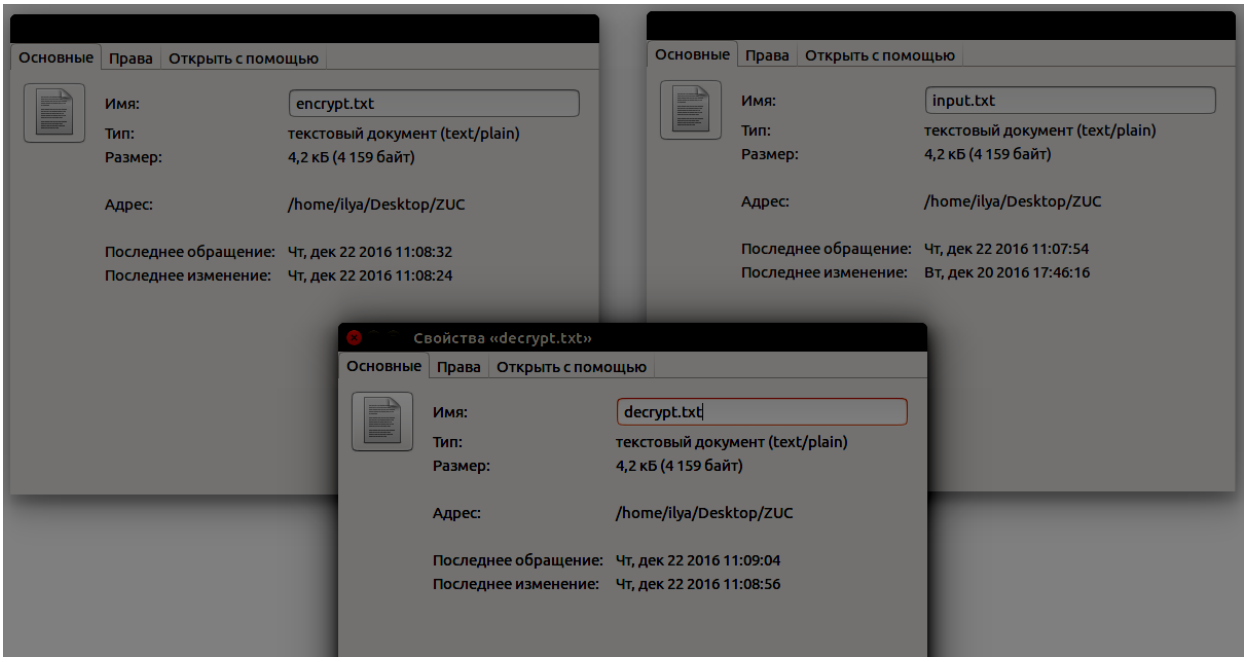


Рис. 8 Сравнение размеров файлов

Размеры всех файлов совпадают, т. к. шифрование производилось побайтово, и никакие лишние байты не записывались.

5. Статистические тесты

Результат тестов приведены на рисунке 9. Ниже приведено краткое описание тестов.

5.1. Частотный тест

Заключается в подсчете распределения «единиц» в случайной последовательности длиной N.

$$f(S^N) = \frac{2}{\sqrt{(N)}}(\sum_{i=1}^{N} s_i - \frac{N}{2})$$

Полученное значение должно принадлежать отрезку $[-3; 3]$.

5.2. Последовательный тест

Зависит от параметра L — длина блока. Вся последовательность длиной N разбивается на блоки длиной L. Определяется число появлений двоичного представления целого числа i для $0 \leqslant i \leqslant 2^L - 1$.

$$f(S^N) = \frac{L2^L}{N} \sum_{i=1}^{2^L-1} (n_i(s^N) - \frac{N}{L2^L})^2$$

Полученное значение аппроксимируется распределением хи-квадрат с $2^L - 1$степенями свободы. При этом рекомендуемая длина исследуемой последовательности не должна быть меньше $5 \times L \times 2^L$бит.

5.3. Тест серий

На вход подается параметр L. Определяется количество 0-серий длины i и, аналогично, 1-серий, где $0 \leqslant i \leqslant L$.

$$f(S^N) = \sum_{b\in\{0,1\}} \sum_{i=1}^{L} \left(\frac{(n_i^b(s^N) - N/2^{(i+2)})^2}{N/2^{(i+2)}}\right)$$

Аппроксимируется критерием хи-квадрат с 2L степенями свободы.

5.4. Автокорреляционный тест

На вход подается значение t, определяющее задержку последовательности. Данный тест определяет, насколько последовательность похожа сама на себя при различных сдвигах. Чем меньше корреляция, тем лучше.

5.5. Универсальный тест

На вход подается параметр L — длина исследуемых блоков. Создается таблица размером 2^L. В ней хранятся последние места встречи данного блока. Первые $Q \geq 10 \times 2^L$блоков являются инициализирующими. Далее $K \geq 1000 \times 2^L$блоков являются исследуемыми. Статистика считается по следующей формуле:

$$f = \frac{1}{K}\left(\sum_{i=Q+1}^{Q+K} \log_2(i - Tab[s(i)])\right)$$

Значение f должно быт близким к математическому ожиданию. Сравнить полученные значения можно с таблицей, приведенной самим автором теста.

Ниже приведены результаты тестов алгоритма ZUC. Все тесты кроме теста серий были успешно пройдены.

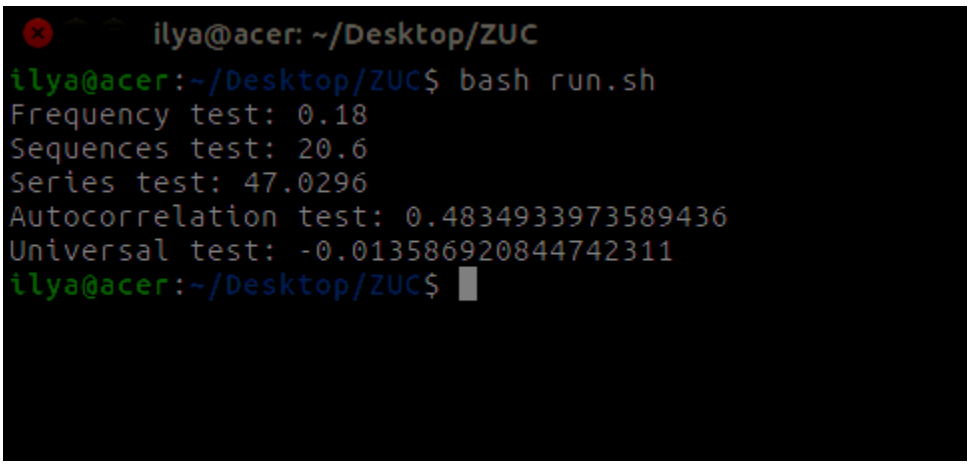


Рис. 9 Результаты статистических тестов