Лабораторная работа 7. ИЗУЧЕНИЕ КРИПТОСИСТЕМЫ МАК-ЭЛИС

Цель работы

Изучить преобразования, выполняемые при шифровании и дешифровании сообщений в системе Мак-Элис, а так же простейшие попытки ее взлома.

Задание

Для работы используются специальные программы MAGMA.exe и McEliece.exe.

1. Сгенерировать порождающую матрицу с применением кода Гоппы.
2. Изучить алгоритмы генерирования открытого и закрытого ключей, а так же шифрования и дешифрования сообщения.
3. Произвести атаку на криптосистему.

Порядок выполнения

1. Запустить программу MAGMA.exe.
2. Выбрать параметры криптосистемы Мак-Элис из предложенного диапазона:

   k – длина информационного сообщения от 2 до 22 бит,

   n – длина кодового слова 8, 16 или 32 бита,

   t – количество искусственно вводимых ошибок от 2 до 4.
3. Сгенерировать для выбранных параметров порождающую матрицу с применением кода Гоппы;
4. Запустить программу McEliece.exe.
5. Проверить, верны ли начальные данные для выбранных параметров при генерировании кода.
6. Сгенерировать несингулярную матрицу S и перестановочную матрицу P. Рассчитать произведение матриц. Определить закрытый и открытый ключи.
7. Ввести двоичное сообщение, которое мы хотим зашифровать. Сгенерировать случайный вектор ошибок Z. Найти криптограмму.
8. Расшифровать сообщение, произведя при этом необходимые промежуточные вычисления. Убедится в правильности дешифрования.
9. Произвести атаку. Сделать выводы об эффективности атаки. При успешной атаке, убедится в правильности восстановленного сообщения. Если атака не удалась, изменить параметры системы и произвести атаку вновь.

Отчет

1. Титульный лист.

2. Параметры криптосистемы.
3. Параметры кода Гоппы.
4. Сообщение, случайный двоичный вектор ошибок и криптограмма.
5. Дешифрованное сообщение.
6. При успешной атаке номера случайно выбранных столбцов. Случайный двоичный вектор и криптограмма, ограниченные выбранными столбцами. Получившееся сообщение. Расчет веероятности успешной атаки для разных значений n, k, t. Выводы.

Описание выполнения работы

Для выполнения работы используются программа MAGMA, позволяющая генерировать порождающую матрицу кода Гоппы, и специально разработанная программа, содержащаяся в файле McEliece.exe.

Для наглядности процессов генерации ключей, шифрования и дешифрования будем рассматривать поля $GF(2^3)$, $GF(2^4)$, $GF(2^5)$. Поля больших порядков не используем, иначе размеры порождающей матрицы будут настолько велики, что работать с ней будет неудобно.

В общем случае, в качестве многочлена Гоппы можно выбрать любой полином. Однако для простоты мы будем использовать неприводимые многочлены, т.к. для любого неприводимого полинома g(z) над полем $GF(2^m)$ степени “t” существует код Гоппы с параметрами:

$n=2^m$– длина кодового слова,

$k\geq 2^m-mt$ - количество информационных символов,

$d \geq 2t+1$ - т.е. он сможет исправлять не менее чем t ошибок.

Рассмотрим следующий пример. В качестве неприводимого полинома в поле GF(2) выберем многочлен G(z)= z^2+z+1. Корни этого многочлена лежат в поле $GF(2^2)$ и, следовательно, в полях $GF(2^4)$, $GF(2^6)$... Предполагая, что m не делится на 2, и выбирая $L=GF(2^m)$, получаем неприводимый код Гоппы с параметрами: $n=2^m$, $k\geq 2^m-2m$.

Положим m=5 . Так как 5 не делится на 2, можно получить код Гоппы с параметрами: $n=2^5=32$бита, $k\geq 2^5-2\cdot 5=22$ бита, $d \geq 2\cdot 2+1=5$.

Подобрав для рассматриваемых полей неприводимые полиномы, можно получить коды Гоппы с различными параметрами. Результаты подбора сведены в таблицу 1.

Таблица 1. Параметры кодов Гоппы, реализуемых в программе

Поле $GF(2^m)$	Длина кода Гоппыn	Число исправляемых ошибок t_A	Неприводимый полином g(z)	Число информационных символов k
$GF(2^3)$	8	2	z^2+z+1	2
$GF(2^4)$	16	3	z^3+z+1	4
			z^3+z^2+1	4

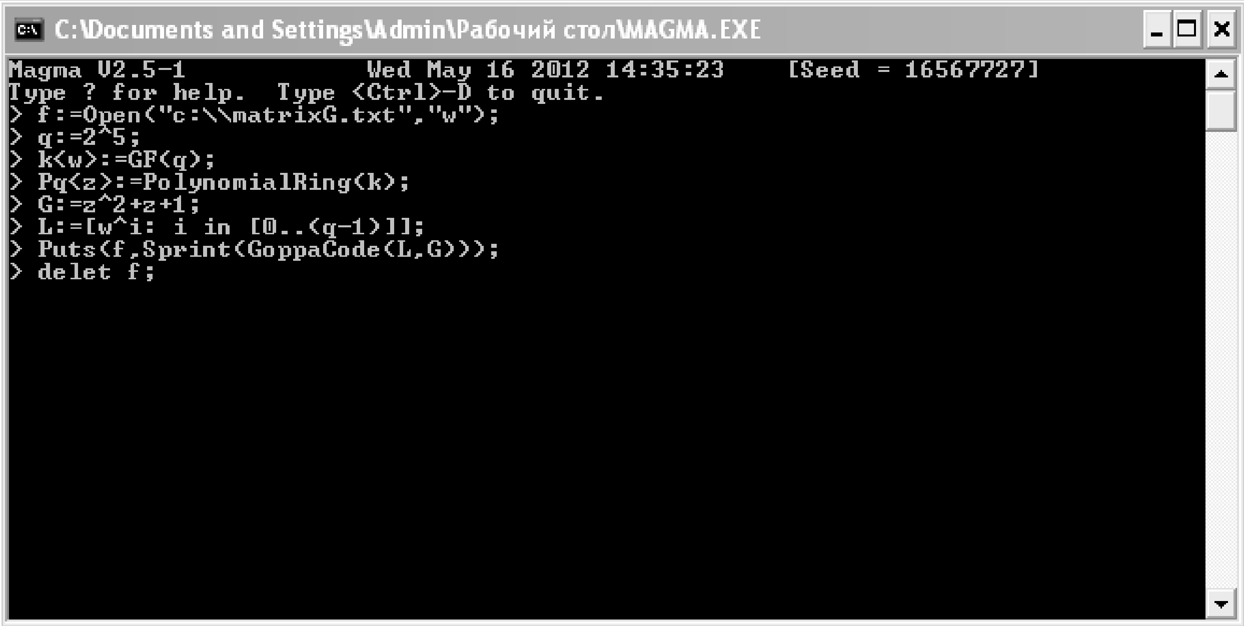
GF(2^5)	32	2	z^2+z+1	22
		3	z^3+z+1	17
			z^3+z^2+1	17
		4	z^4+z+1	12
			z^4+z^3+1	12
			z^4+z^3+z^2+z+1	12

Сначала из таблицы 1 необходимо выбрать любой набор параметров для криптосистемы Мак-Элис: *(n, k.* t_A*)*. Например, возьмем длину кодового слова $n=32$ бита, длина информационного сообщения $k=22$ бита, а число искусственно водимых ошибок $t_A=2$.

Чтобы сгенерировать порождающую матрицу *G* с помощью MAGMA, необходимо запустить программу и в открывшемся окне ввести следующие команды:

```
f:= Open("c:\\matrixG.txt","w");
q:=2^m;
k<w>:=GF(q);
Pq<z>:=PolynomialRing(k);
G:=g(z);
L:=[w^i: i in [0..(q-1)]];
Puts(f, Sprint(GoppaCode(L,G)));
delete f;
```

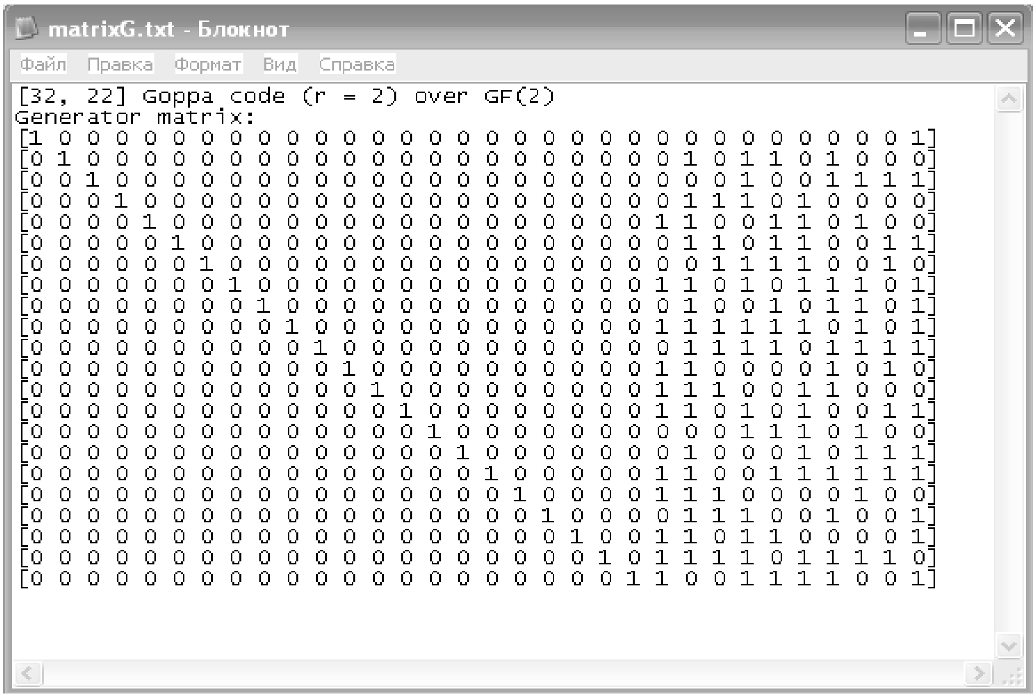
Где вместо ***m*** подставляем степень поля, для выбранных параметров; вместо ***g(z)*** неприводимый многочлен, для соответствующего поля. Для нашего примера m=5 и *g(z)*= z^2+z+1. (Рис. 1).




```
Magma V2.5-1          Wed May 16 2012 14:35:23    [Seed = 16567727]
Type ? for help.  Type <Ctrl>-D to quit.
> f:=Open("c:\\matrixG.txt","w");
> q:=2^5;
> k<w>:=GF(q);
> Pq<z>:=PolynomialRing(k);
> G:=z^2+z+1;
> L:=[w^i: i in [0..(q-1)]];
> Puts(f,Sprint(GoppaCode(L,G)));
> delet f;
```

Рис. 1. Генерирование порождающей матрицы

В результате выполнения этих команд на диске c:\\ появится текстовый файл с именем “matrixG.txt”. Открыв его можно увидеть порождающую матрицу и параметры системы. Однако внесение какие-либо изменений в этот файл не допускается! Если потребуется матрица других размеров, необходимо выполнить выше упомянутые команды, изменяя соответствующие параметры. Пример такого текстового файла изображен на рис. 2.




```
[32, 22] Goppa code (r = 2) over GF(2)
Generator matrix:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0]
[0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1]
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0]
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 0 0]
[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1]
[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 0 1]
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 1]
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1]
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0]
[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1]
```

Рис. 2. Пример порождающей матрицы

[32, 22] – размер матрицы G. Соответственно $n = 32, k = 22$.

$r = 2$ число гарантированно исправляемых кодом ошибок, т.е. $t_A = 2$.

Так как нужная нам порождающая матрица сгенерирована, MAGMA можно закрыть.

Теперь запускается McEliece.exe. Для начала работы надо сгенерировать ключи. На вкладке «Генерирование ключей» необходимо нажать на кнопку «Порождающая матрица G». В поле «Начальные данные» появится размерность порождающей матрицы и количество исправляемых ошибок, а в поле справа сама матрица, которая была сгенерирована ранее. Далее генерируются матрицы *S* и *P,* нажатием соответствующих кнопок. Вычисляется матрица $\hat{G}=S\cdot G\cdot P$, путем последовательного умножения матриц: сначала перемножаются матрицы *G* и *P*, затем S и (G*P). (Рис. 3).

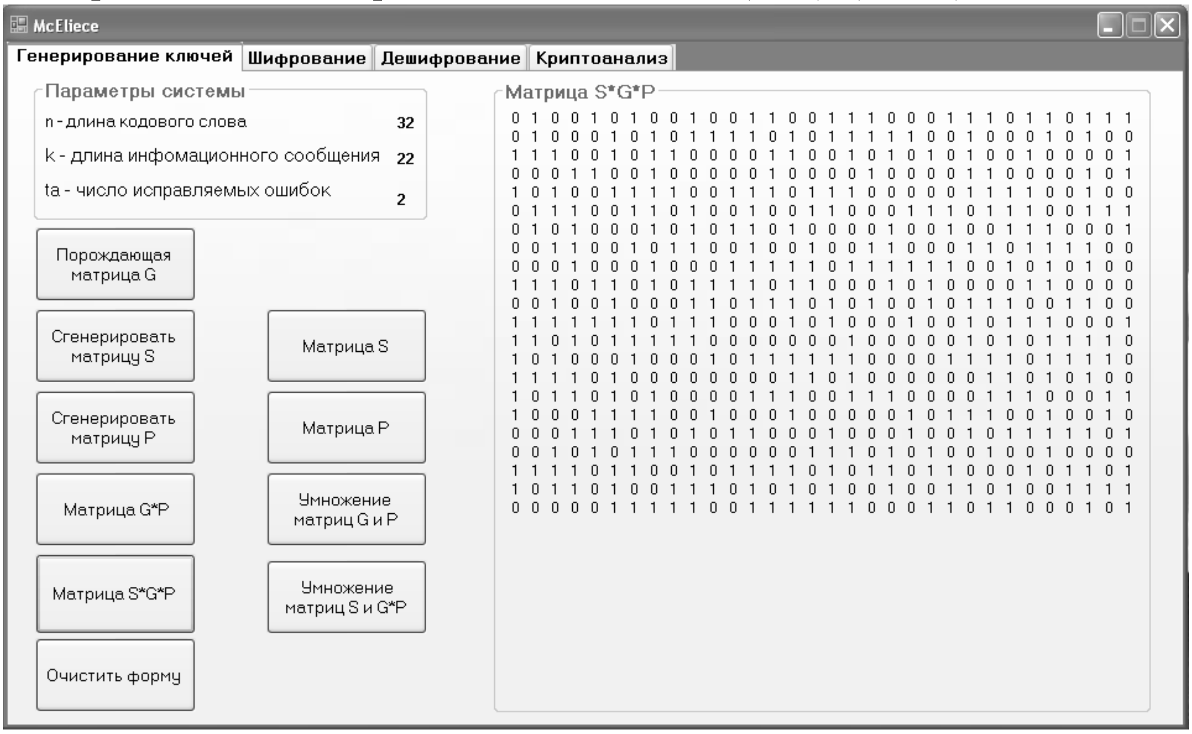


Рис. 3. Вычисление матрицы $\hat{G}$=S*G*P

Для просмотра матриц *S* и *P*, не генерируя при этом новые матрицы, необходимо нажать на кнопки «Матрица S» и «Матрица P» соответственно.

Для проверки умножения матриц, а так же просмотра изменения размерности, следует нажать на кнопки «Умножение матриц G и P» и «Умножение матриц S и G*P».

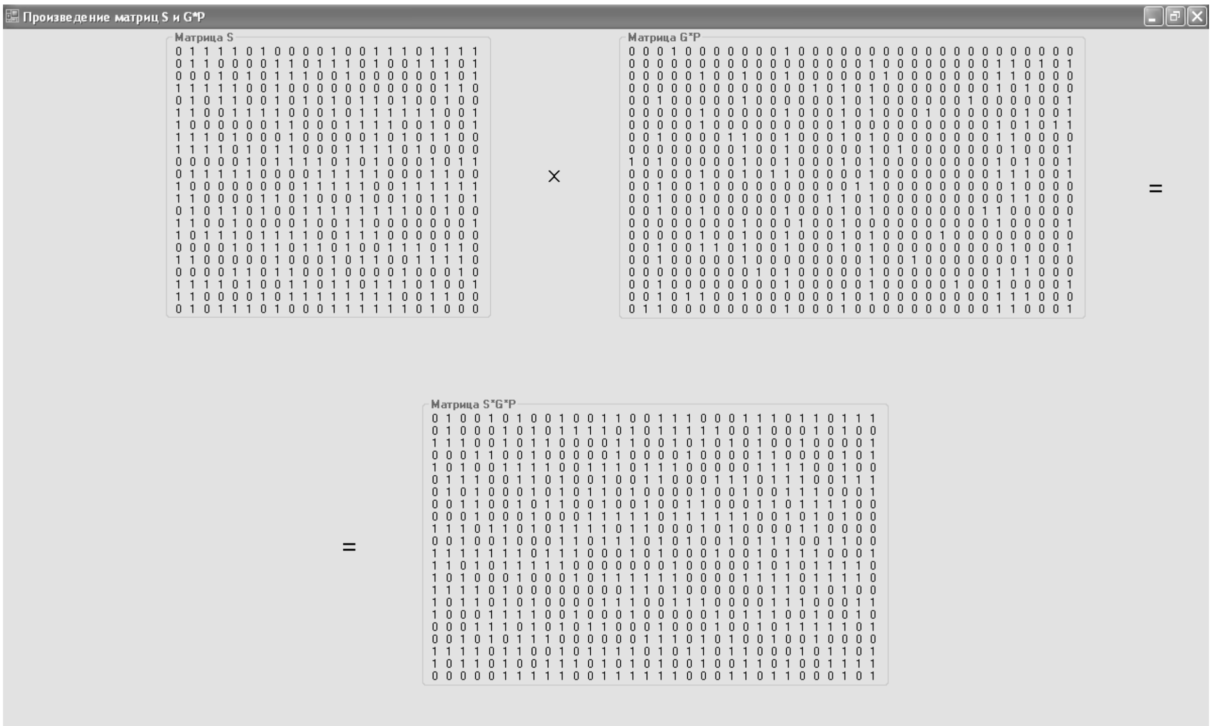


Рис. 4. Перемножение матриц S и D*P

В случае, если необходимо сгенерировать матрицу G с другими параметрами, следует нажать кнопку “Очистить форму”. Будет очищено поле вывода матриц и поле “Параметры системы”. А так же из памяти программы сотрутся все сгенерированные ранее матрицы.

Сгенерировав ключи, переходим на вкладку “Шифрование”.

Чтобы зашифровать сообщение, его сначала преобразуют в двоичную последовательность, а затем разбивают на блоки длиной k символов. В данной же работе реализован процесс шифрования одного такого блока. Т.е. в поле для ввода сообщения надо ввести не буквенный текст, а двоичную последовательность длины k.

Например, введем такую последовательность: 1100001010100000001111. Ее длина *22* символа, т.к. в нашем случае $k=22$.

Затем генерируется случайный двоичный вектор Z и находится криптограмма, предварительно вычислив произведение сообщения и матрицы $\hat{G}=S*G*P$. (Рис. 5).

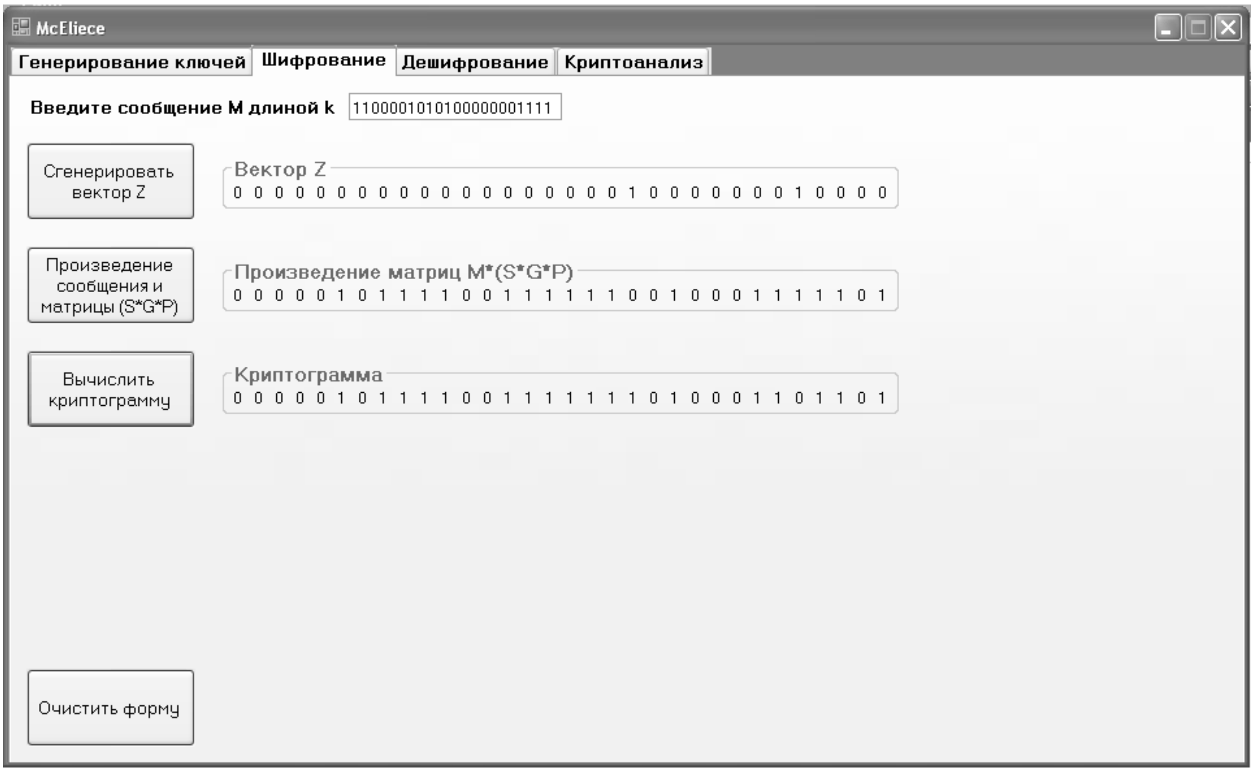


Рис. 5. Шифрование сообщения

Если необходимо ввести другое сообщение, надо нажать “Очистить форму”, вследствие чего поля вывода результатов очистятся.

Для дешифрования сообщения, необходимо перейти на вкладку “Дешифрование”. Сначала надо нажать на кнопку «Вычислить вектор C’», в результате чего появится вектор $\hat{C}_i=C_i{\cdot}P_A{}^{-1}$,а так же матрица, обратная *P*.

Теперь используя известный алгоритм декодирования для кода Гоппы с порождающей матрицей G, исправим не более t_A ошибок в *C*`, что даст некоторый двоичный вектор *M*` длины *k*. Но так как этот алгоритм достаточно сложен, в программе исправление ошибок реализовано за счет вычитания двоичного вектора ошибок *Z*.

Для восстановления сообщения нажимаем на соответствующую кнопку. Выводится матрица, обратная *S*, и расшифрованное сообщение. (Рис. 6).

Как видно, расшифрованное сообщение 11000010101000000001111 такое же, как и последовательность, зашифрованная ранее.

Рис. 6. Дешифрование сообщения

Последний этап работы, атака на криптосистему. Для этого надо перейти на вкладку «Криптоанализ» и нажать «Атака». После нажатия кнопки система начинает случайно выбирать k столбцов матрицы $\hat{G}=S*G*P$, проверяя при этом необходимые условия для взлома. Если матрица, составленная из выбранных k столбцов, окажется несингулярной и вектор Z, ограниченный этими столбцами, окажется нулевым, то сообщение будет восстановлено, а полученные результаты выведены на экран.

Если одно из условий не выполняется, программа выбирает другой набор столбцов и снова проверяет условия, до тех пор, пока они не выполнятся.

Для рассматриваемого примера атака прошла успешно. Видно, что полученное сообщение совпадает с сообщением, которое зашифровывали ранее. (Рис. 7).

На самом деле процесс взлома происходит немного иначе. Взломщику не известен вектор ошибок Z, так он абсолютно случайный. Поэтому процесс подбора столбцов происходит до тех пор, пока не появится сообщение с известным характером избыточности.

В нашем случае для простоты полагается, что злоумышленнику известны биты, в которых появляется ошибка.

Не трудно заметить, что атака займет какое-то время. К тому же при определенных параметрах систему не удастся взломать. В этом случае следует нажать кнопку “Стоп” для остановки процесса взлома и корректного выхода из программы.

Отметим, что при успешной атаке набор выбранных столбцов является не единственным. Если нажать кнопку “Очистить форму” (в результате которой все поля очистятся) и затем опять “Атака”. То при повторной успешной атаке можно обнаружить другую комбинацию столбцов.

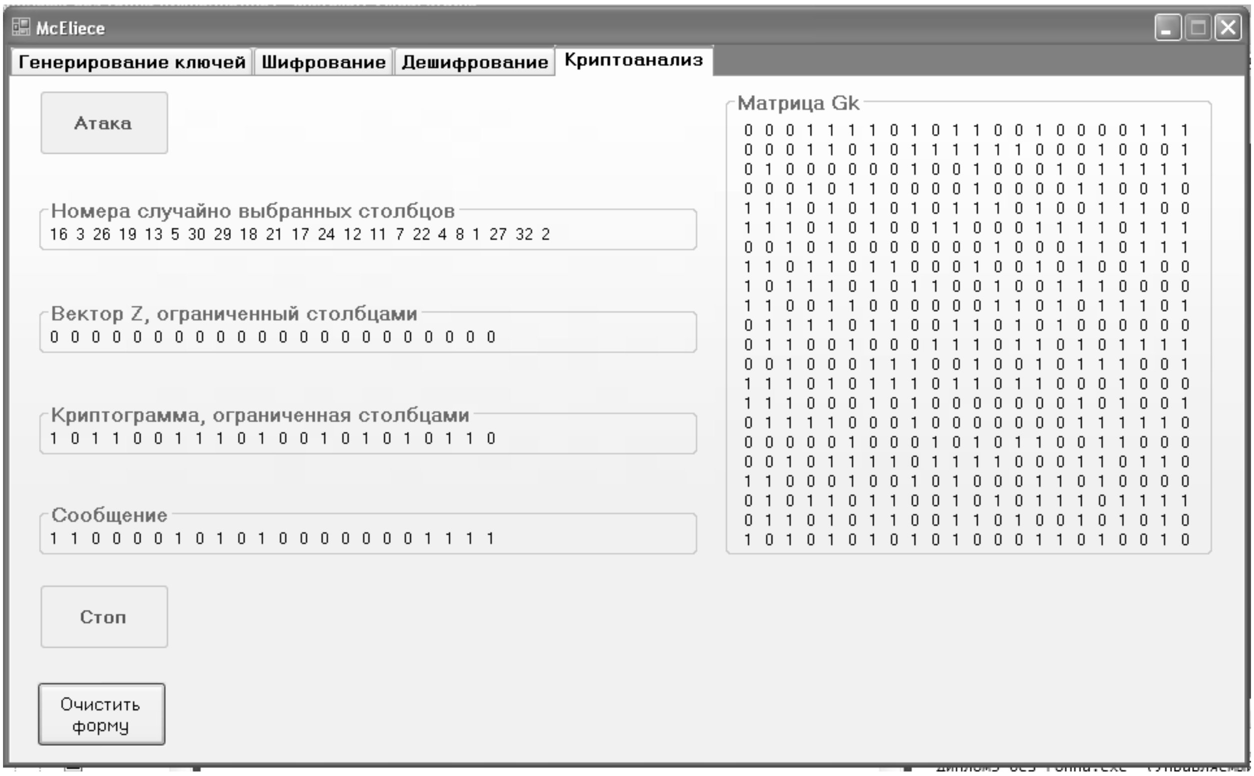


Рис. 7. Атака на криптосистему

Пример отчета

1. Параметры криптосистемы Мак-Элис:
   длина кодового слова n = 32 бита,
   длина информационного сообщения k = 22 бита,
   число искусственно водимых ошибок $t_A = 2$.
2. Параметры кода Гоппы:
   длина $n=2^5=32$бита,
   размерность $k \geq 2^5-2\cdot 5=22$ бита,
   минимальное расстояние $d \geq 2\cdot 2+1=5$.
3. Сообщение: 1100001010100000001111
   Вектор Z: 00000000000000000001000000010000
   Криптограмма: 00000101111001111111010001101101
4. Восстановленное сообщение: 1100001010100000001111
5. Номера столбцов: 16; 3; 26; 19; 13; 5; 30; 29; 18; 21; 17; 24; 12; 11; 7; 22; 4; 8; 1; 27; 32; 2
   Нулевой вектор Z: 0000000000000000000000
   Восстановленное сообщение: 1100001010100000001111

6. Расчеты вероятности успеха атаки на КС Мас-Элис.

(см. образец отчета)

Контрольные вопросы

1. Что является открытым и закрытым ключом криптосистемы Мак-Элис?
2. Алгоритм шифрования и дешифрования криптосистемы Мак-Элис.
3. В чем суть производимой в лабораторной работе атаки на систему? При каких параметрах системы вероятность атаки увеличивается?
4. Как выбирают параметры криптосистемы Мак-Элис?
5. В чем плюсы и минусы криптосистемы Мак-Элис?