ГЛАВА 4 Система Visual AES

Разработанный в рамках дипломной работы программный комплекс проектировался, как многофункциональная система визуализации, анализа и, собственно, реализации шифрования данных с использованием алгоритма AES. Основными компонентами (Рисунок 4.1) системы Visual AES являются:

- подсистема шифрования по стандарту AES – модули реализующие зашифровку/расшифровку с поддержкой большинства существующих режимов шифрования с обратной связью (ECB, CBC, CFB, OFB и CTR);
- подсистема визуализации, с помощью соответствующего графического материала, а также специально встроенного инструментария («Журнал», «Эксперт») раскрывающая суть алгоритма и принципов преобразований реализованных в нем;
- подсистема анализа, включающая в себя средства управления математическим аппаратом преобразований, проводимых в процессе шифрования и визуальная реализация известной атаки «Квадрат» на криптоалгоритм AES.

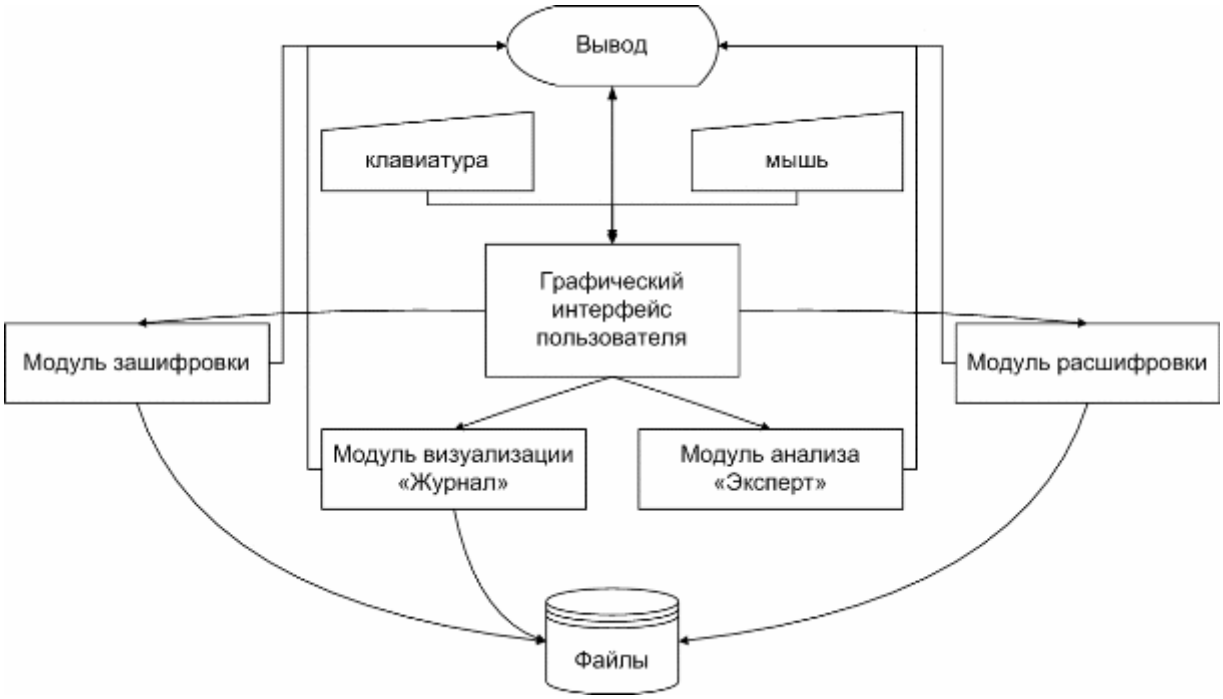


Рисунок 4.1 **«Схема взаимодействия модулей программы Visual AES»**

Схематично работа комплекса представлена на Рисунке 4.2. Для начала работы необходимо произвести первичную подготовку данных, сюда входит: загрузка проекций файлов соответственно открытого текста и шифротекста (вне зависимости от выбираемого направления зашифрования/расшифрования их размеры должны быть синхронизированы, что обусловливается условиями блочного криптоалгоритма) и ввод ключа, остальные настройки выбираются в зависимости от поставленной задачи.

В зависимости от дальнейшего выбора пользователем программы возможны следующие операции:

- зашифрование/расшифрование;
- подстройка математического аппарата преобразований для получения новых мутаций алгоритма, и соответственно, их исследования и, возможно, применения в обмене шифрованной информацией (файлами) в рамках системы Visual AES;
- наблюдение за промежуточными результатами раундовых преобразований и возможная их передача в другие системы, например, статистические или математические для дальнейшего анализа с привлечением дополнительных методов.

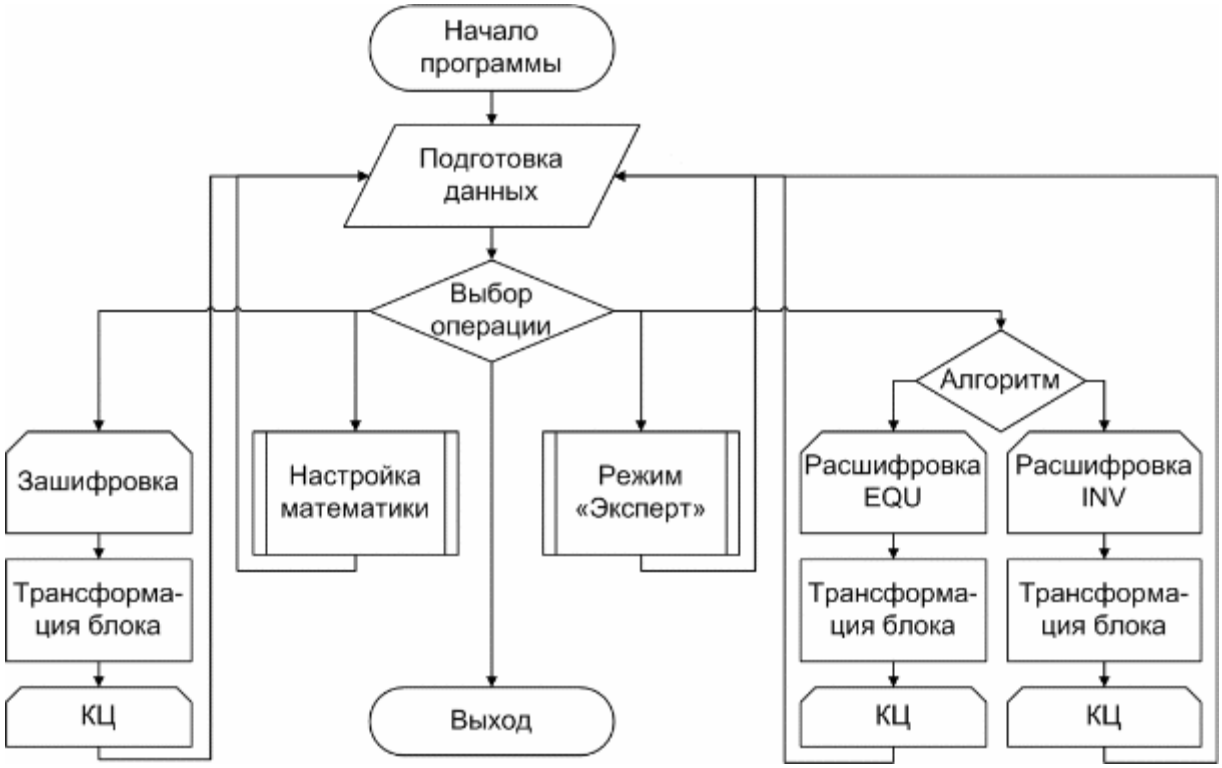


Рисунок 4.2 **«Схема программы Visual AES»**

Все результаты сформированные в ходе работы программы Visual AES предлагается сохранить в соответствующих файлах на внешних носителях либо передать с помощью средств DDE (Windows Clipboard) другим приложениям.

Модель данных, представленная на Рисунке 4.3 демонстрирует информационное взаимодействие модулей системы Visual AES:

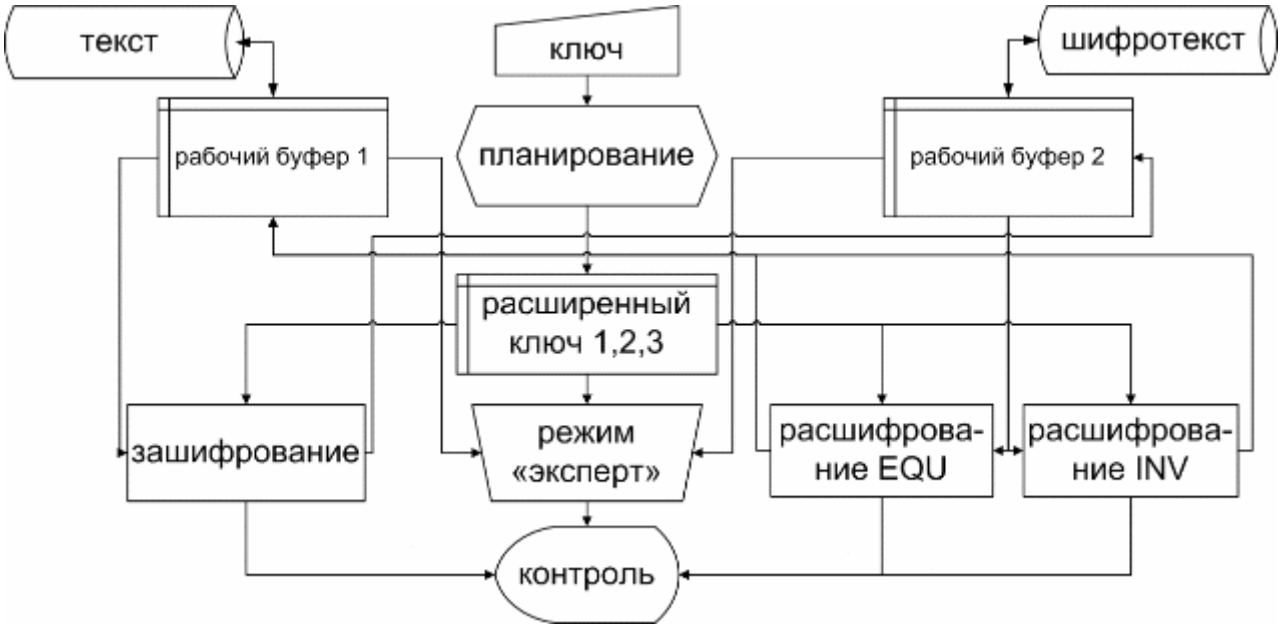


Рисунок 4.3 **«Схема данных программы Visual AES»**

Основными информационными блоками в работе программы являются буферы файлов открытого и шифротекста. На базе вводимого пользователем ключа шифрования система автоматически генерирует три ключевых последовательности, соответственно, для зашифрования, обратного и инверсного расшифрований (планирование подключей). Операции зашифрования ($РБ1 \Rightarrow РБ2$) и расшифрования ($РБ2 \Rightarrow РБ1$) перемещают текст

и шифротекст между буферами, что позволяет производить дополнительную проверку и убедиться в правильности результатов обращая операции.

Режим «Эксперт» обладает собственными рабочими буферами, что позволяет оперировать данными из исходных файлов не внося в них изменений (для сохранения результатов «экспертизы» можно воспользоваться DDE). При анализе атаки «Квадрат» создается закрытое множество $\lambda - set$ в динамической памяти, а ключевая информация доступна только для чтения, что также не оказывает разрушающих действий на исходные файлы.

Рисунок 4.4. иллюстрирует работу системы Visual AES. Стрелками показаны направления информационного взаимодействия между инструментами системы

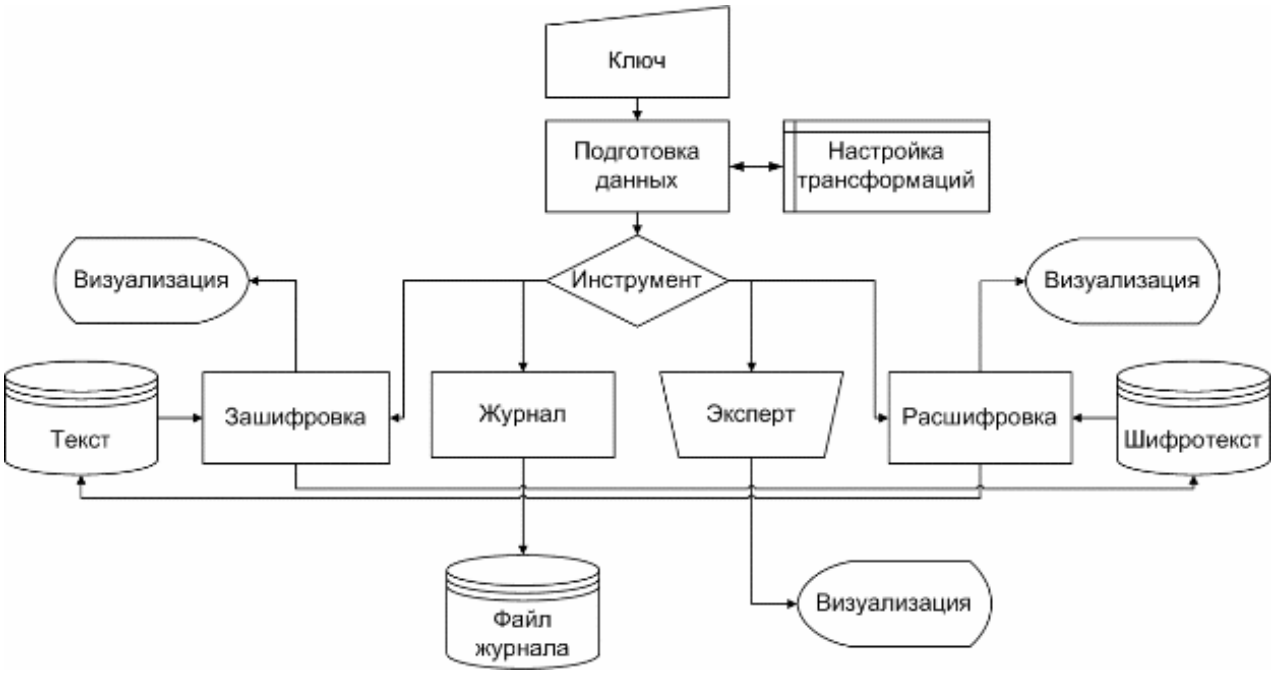


Рисунок 4.4 **«Схема работа системы Visual AES»**

Visual AES, как средство шифрования.

Как видно из Рисунка 4.5 система Visual AES, кроме реализации блочного преобра-

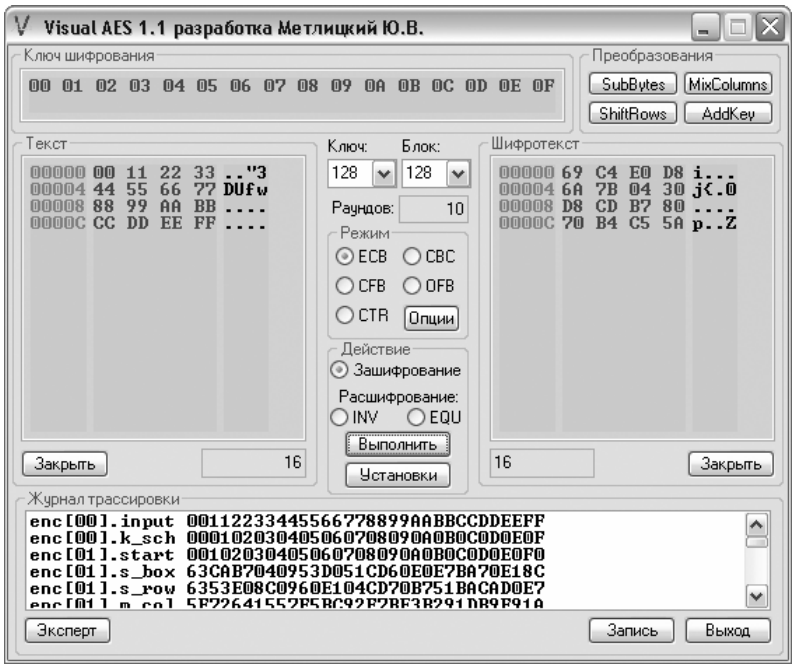


Рисунок 4.5 **«Основное окно программы Visual AES»**

зования по стандарту AES, обладает всеми необходимыми средствами для выполнения операций зашифрования/расшифрования (128, 192, 256–ти битных ключей и блоков данных независимо друг от друга, и, соответственно, 10, 12 и 14-ти раундовых преобразований) с применением основных режимов блочного шифрования с обратной связью. Ограничения на размер шифруемых данных зависят только от количества адресуемой виртуальной памяти доступной системе.

Режим электронной кодовой книги (Electronic Codebook, ECB)

В режиме ECB каждый блок открытого текста заменяется блоком шифротекста. А так как один и тот же блок открытого текста заменяется одним и тем же блоком шифротекста, теоретически возможно создать кодовую книгу блоков открытого текста и соответствующих шифротекстов. Но если размер блока составляет n бит, кодовая книга будет состоять из 2^n записей.

Режим ECB - простейший режим шифрования. Все блоки открытого текста шифруются независимо друг от друга. Это важно для шифрованных файлов с произвольным доступом, например, файлов баз данных. Если база данных зашифрована в режиме ECB, любая запись может быть добавлена, удалена, зашифрована или расшифрована независимо от любой другой записи (при условии, что каждая запись состоит из целого числа блоков шифрования). Кроме того, обработка может быть параллельной: если используются несколько шифровальных процессоров, они могут шифровать или расшифровывать различные блоки независимо друг от друга.

На Рисунке 4.2 изображается окно установок ECB и диаграмма операций производимых программой Visual AES в данном режиме.

Рисунок 4.6 **«Окно установок режима ECB программы Visual AES»**

К недостаткам режима ECB можно отнести то обстоятельство, что если у криптоаналитика есть открытый текст и шифротекст нескольких сообщений, он может, не зная ключа, начать составлять шифровальную книгу. В большинстве реальных ситуаций фрагменты сообщений имеют тенденцию повторяться. В различных сообщениях могут быть одинаковые битовые последовательности, В сообщениях, которые, подобно электронной почте, создаются компьютером, могут быть периодически повторяющиеся структуры. Сообщения могут быть высоко избыточными или содержать длинные строки нулей или пробелов.

К достоинствам режима ECB можно отнести возможность шифрования нескольких сообщений одним ключом без снижения надежности. По существу, каждый, блок можно рассматривать как отдельное сообщение, шифрованное тем же самым ключом. При расшифровании ошибки в символах шифротекста ведут к некорректному расшифрованию соответствующего блока открытого текста, однако не затрагивают остальной открытый текст. Но если бит шифротекста случайно потерян или добавлен, весь последующий шиф-

ротекст будет дешифрован некорректно, если только для выравнивания границ блоков не используется какое-нибудь выравнивание по границам блока.

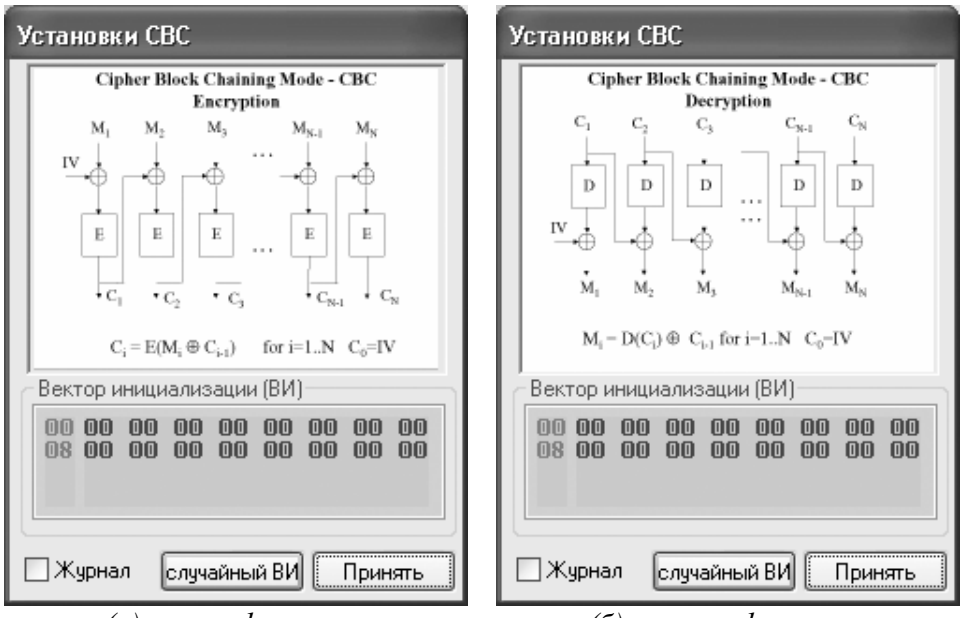
Большинство сообщений не делятся точно на n-битовые блоки шифрования - в конце обычно оказывается укороченный блок. Однако режим ECB требует использовать строго n-битовые блоки. Для решения этой проблемы используют дополнение *(padding),* Чтобы создать полный блок, последний блок дополняют некоторым стандартным шаблоном - нулями, единицами, чередующимися нулями и единицами.

Режим сцепления блоков шифротекста (Cipher Block Chaining, CBC)

Сцепление добавляет в блочный шифр механизм обратной связи: результаты шифрования предыдущих блоков влияют на шифрование текущего блока, т.е. каждый блок используется для модифицирования шифрования следующего блока. Каждый блок шифротекста зависит не только от шифруемого блока открытого текста, но и от всех предыдущих блоков открытого текста.

В режиме сцепления блоков шифротекста перед шифрованием над открытым текстом и предыдущим блоком шифротекста выполняется операция XOR. На Рисунке 4.7 (а) показан процесс шифрования в режиме CBC. Когда блок открытого текста зашифрован, полученный шифротекст сохраняется в регистре обратной связи. Следующий блок открытого текста перед шифрованием подвергается операции XOR с содержимым регистра обратной связи. Результат операции XOR используется как входные данные для следующего этапа процедуры шифрования. Полученный шифротекст снова сохраняется в регистре обратной связи, чтобы подвергнуться операции XOR вместе со следующим блоком открытого текста, и так до конца сообщения. Шифрование каждого блока зависит от всех предыдущих блоков.

Расшифрование выполняется в обратном порядке (Рисунок 4.7 (б)). Блок шифротекста расшифровывается обычным путем, но сохраняется в регистре обратной связи. Затем следующий блок расшифровывается и подвергается операции XOR с содержимым регистра обратной связи. Теперь следующий блок шифротекста сохраняется в регистре обратной связи и т.д. до конца сообщения.



(а) - зашифрование *(б) - расшифрование*

Рисунок 4.7 **«Окна установок режима CBC программы Visual AES»**

При шифровании в режиме CBC одинаковые блоки открытого текста превращаются в различающиеся друг от друга блоки шифротекста только в том случае, если различались какие-то предшествующие блоки открытого текста. Однако при шифровании двух

идентичных сообщений создается один и тот же шифротекст, Хуже того, два одинаково начинающихся сообщения будут шифроваться одинаково вплоть до первого различия.

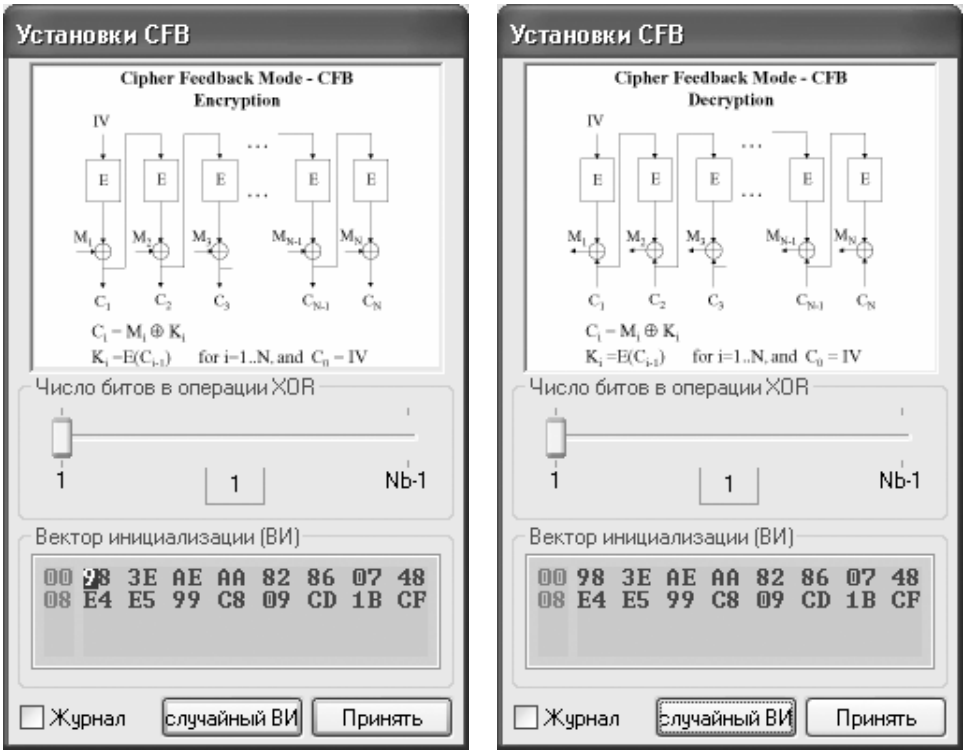
Чтобы избежать этого, можно зашифровать в первом блоке какие-то произвольные данные. Этот блок случайных данных называют вектором инициализации (ВИ) (Initialization Vector, IV, русский термин - синхропосылка), инициализирующей переменной или начальным значением сцепления. Вектор ВИ не имеет какого-то смыслового значения, он используется только для того, чтобы сделать каждое сообщение уникальным. Когда получатель расшифровывает этот блок, он использует его только для заполнения регистра обратной связи. В качестве вектора ВИ удобно использовать метку времени, либо какие-то случайные биты.

Если используется вектор инициализации, сообщения с идентичным открытым текстом после шифрования превращаются в сообщения с разными шифротекстами. Следовательно, злоумышленник не может попытаться повторить блок, и создание шифровальной книги затруднится. Хотя для каждого сообщения, шифруемого одним и тем же ключом, рекомендуется выбирать уникальный вектор ВИ, это требование необязательное.

Вектор ВИ не обязательно хранить в секрете, его можно передавать открыто - вместе с шифротекстом. Дополнение используется так же, как и в режиме ECB.

Режим обратной связи по шифротексту (Cipher-Feedback, CFB)

В режиме CBC начать шифрование до поступления полного блока данных невозможно. Для некоторых сетевых приложений это создает проблемы. Например, в защищенном сетевом окружении терминал должен иметь возможность передавать хосту каждый символ сразу после ввода. Если же данные нужно обрабатывать блоками в несколько байт, режим CBC просто не работает.



(а) - зашифрование *(б) - расшифрование*

Рисунок 4.8 **«Окна установок режима CFB программы Visual AES»**

В режиме CFB можно шифровать единицы данных размером не больше блока. Один из вариантов - шифрование по одному байту (этот режим называют 8-битовым CFB), с помощью 1 -битового CFB можно шифровать данные и побитово, но полное шифрование блочным шифром единственного бита требует много ресурсов.

Рисунок 4.8 (а, б) – демонстрация возможностей системы Visual AES по поддержке режима n-битового CFB, где $1 \le n < Nb$.

Как и в режиме CBC, первоначально очередь заполнена вектором инициализации ВИ. Очередь шифруется, затем выполняется операция XOR над n старшими (крайними левыми) битами результата и первым n-битовым символом открытого текста. В результате появляется первый n-битовый символ шифротекста. Теперь этот символ можно передать. Кроме того, полученные n битов попадают в очередь на место n младших битов, а все остальные биты сдвигаются на n позиций влево. Предыдущие n старших битов отбрасываются. Затем точно также шифруется следующие n битов открытого текста. Расшифрование выполняется в обратном порядке. Обе стороны - шифрующая и расшифровывающая -использует блочный алгоритм в режиме шифрования. Как и режим CBC, режим CFB сцепляет символы открытого текста с тем, чтобы шифротекст зависел от всего предыдущего открытого текста.

Для инициализации процесса шифрования в режиме CFB в качестве входного блока алгоритма можно использовать вектор инициализации ВИ. Как и в режиме CBC, хранить в тайне вектор ВИ не нужно. Однако вектор ВИ должен быть уникальным. (В отличие от режима CBC, где уникальность вектора ВИ необязательна, хотя и желательна). Если вектор ВИ в режиме CFB не уникален, криптоаналитик может восстановить соответствующий открытый текст. Вектор инициализации должен меняться в каждом сообщении. Например, вектором ВИ может служить порядковый номер, возрастающий в каждом новом сообщении и не повторяющийся все время жизни ключа. Если данные шифруются с целью последующего хранения, вектор ВИ может быть функцией индекса, используемого для поиска данных.

Режим обратной связи по выходу (Output-Feedback - OFB)

Режим OFB представляет собой метод использования блочного шифра в качестве синхронного потокового шифра. Этот режим подобен режиму CFB, за исключением того (Рисунок 4.9), что n битов предыдущего выходного блока сдвигаются в крайние правые позиции очереди. Расшифрование выполняется в обратном порядке. Такой режим называют n-битовым режимом OFB.

Блочный алгоритм работает в режиме шифрования как на шифрующей, так и на расшифровывающей сторонах. Такую обратную связь иногда называют внутренней, поскольку механизм обратной связи не зависит ни от потока открытого текста, ни от потока шифротекста.

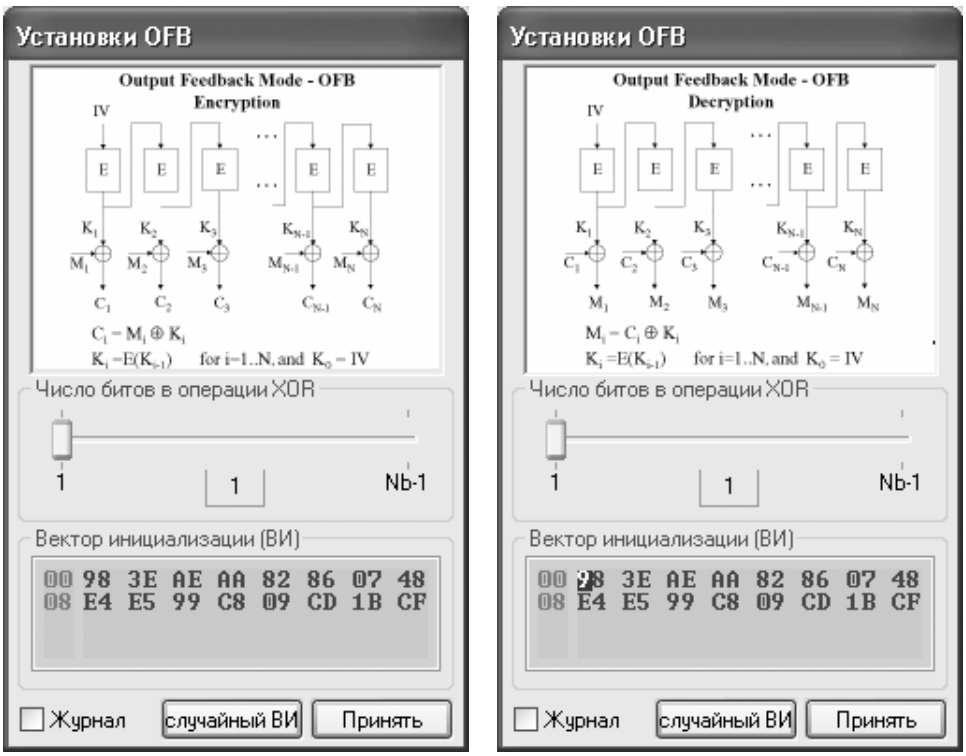
К достоинствам режима OFB относится то, что большую часть работы можно выполнить оффлайново, даже когда открытого текста сообщения еще вовсе не существует. Когда, наконец, сообщение поступает, для создания шифротекста над сообщением и выходом алгоритма необходимо выполнить операцию XOR.

В сдвиговый регистр OFB сначала надо загрузить вектор ВИ. Вектор должен быть уникальным, но сохранять его в тайне не обязательно.

Анализ режима OFB показывает, что OFB целесообразно использовать только если разрядность обратной связи совпадает с размером блока.

В режиме OFB над гаммой и текстом выполняется операция XOR. Эта гамма, в конце концов, повторяется. Существенно, чтобы она не повторялась для одного и того же ключа, в противном случае секретность не обеспечивается ничем.

Если разрядность обратной связи равна размеру блока, блочный шифр работает, выполняя перестановки m-битовых значений (где m - размер блока), и средняя длина цикла составляет $2^T - 1$ Когда разрядность обратной связи n меньше длины блока, средняя длина цикла снижается примерно до $2^{T/2}$. Для 128-битового шифра это число составляет только.

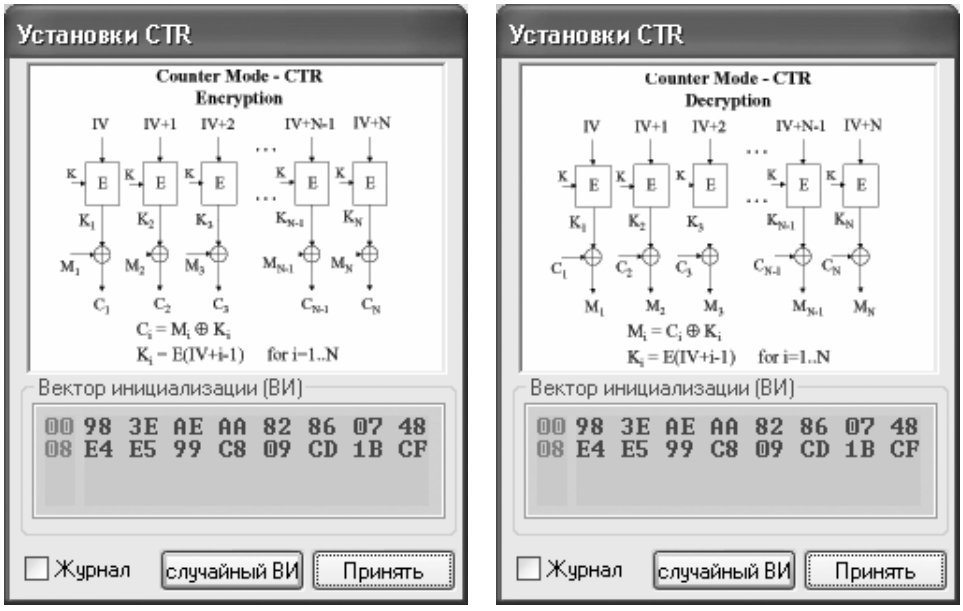


(а) - зашифрование *(б) - расшифрование*

Рисунок 4.9 **«Окна установок режима OFB программы Visual AES»**

Режим счетчика (Counter, CTR)

Блочные шифры в режиме счетчика используют последовательность чисел в качестве входов алгоритма (Рисунок 4.10). Для заполнения регистра используется счетчик, а не вывод алгоритма шифрования. После шифрования каждого блока значение счетчика возрастают на определенную константу, обычно на единицу. Свойства синхронизации и распространения ошибки этого режима точно такие же, как у режима OFB. Режим счетчика устраняет проблему л-битового выхода в режиме OFB, когда *n* меньше длины блока.



(а) - зашифрование *(б) - расшифрование*

Рисунок 4.10 **«Окна установок режима CTR программы Visual AES»**

К счетчику не предъявляют какие-то особые требования. В частности, он не должен пробегать последовательно все возможные значения. В качестве входа блочного алго-

ритма можно использовать генераторы случайных чисел, описанные в главах 16 и 17, независимо от их криптографической надежности.

Выбор режима шифрования

Если необходима главным образом простота и скорость, режим ECB можно рекомендовать как самый простой и быстрый режим работы блочного шифра. Помимо уязвимости к вскрытию с повторной передачей, алгоритм в режиме ECB проще всех других для криптоанализа. Поэтому использовать режим ECB для шифрования сообщений не рекомендуется.

Режим ECB удобно использовать для шифрования случайных данных, например, других ключей. Так как данные невелики по размеру и случайны, недостатки режима ECB в данном случае несущественны.

Для шифрования обычного открытого текста лучше всего использовать режимы CBC, CFB, OFB или CTR. Выбор режима зависит от потребностей.

Для шифрования файлов лучше всего подходит режим CBC. Надежность значительно возрастает; и хотя иногда появляются ошибки в символах хранимых данных, почти никогда не бывает сбоев синхронизации.

Visual AES, как средство визуализации (обучения).

Одной из важных функций системы Visual AES можно назвать наличие инструментария визуализации алгоритма AES, включающие в себя: графический материал раскрывающий его суть (выше уже были представлены изображения окон демонстрирующие суть различных режимов блочного шифрования), средства управления составом раундового преобразования (Рисунок 4.11), возможность ведения для каждого обрабатываемого блока подробного журнала раундовых преобразований.

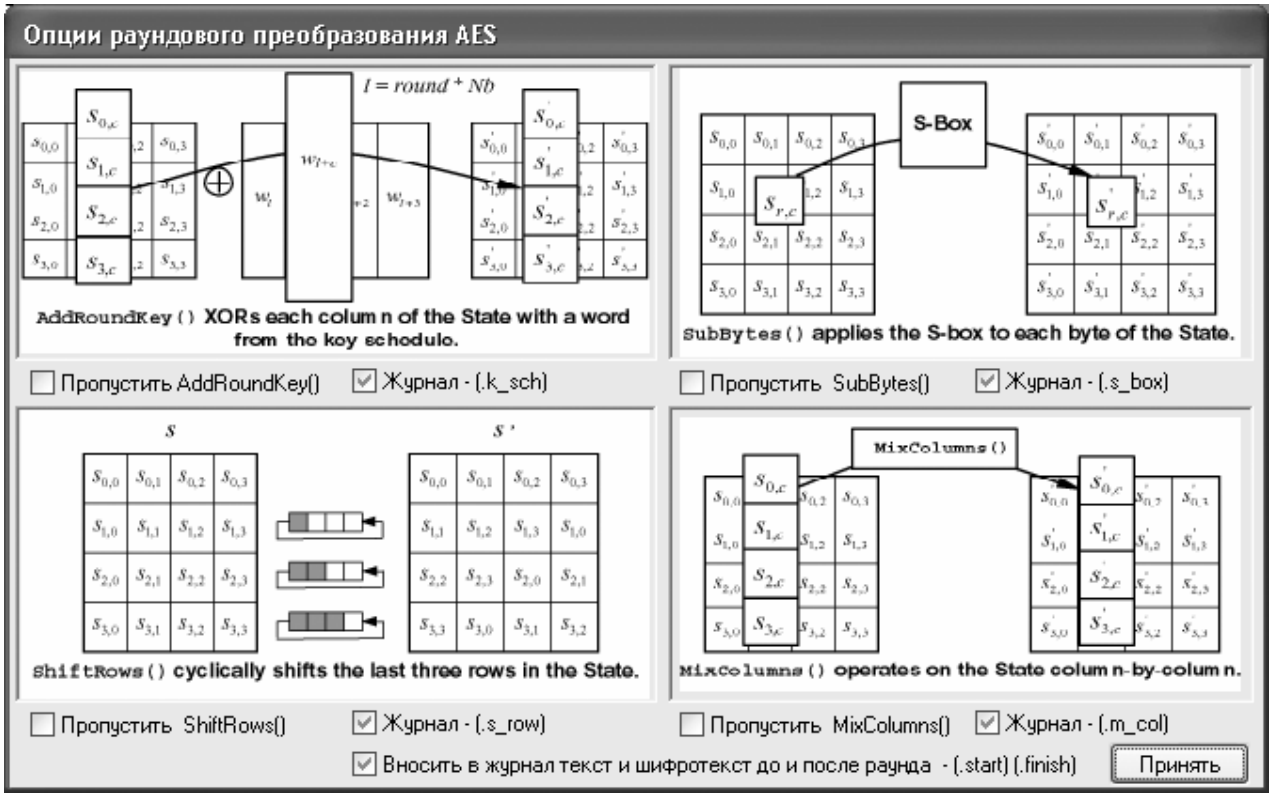


Рисунок 4.11 **«Окно Visual AES настроек визуализации раундовых преобразований»**

С помощью установок изображенных на Рисунке 4.11 можно управлять наполнением журнала и определять, какие операции будут включены в раундовое преобразование алгоритма AES и, соответственно, наблюдать за эволюцией данных в нестандартных условиях.

Журнал раундовых преобразований

Журнал раундовых преобразований посредством описанных выше установок программы Visual AES может содержать подробный отчет о трансформации данных каждого шифруемого и дешифруемого блока на всех этапах раундовых преобразований. Ниже приводятся листинги операций зашифрования, инвесного и эквивалентного расшифрований для эталонного ключа и блока предлагаемых в описании шифра его авторами:

Зашифрование

```
enc[00].input 00112233445566778899AABBCCDDEEFF
enc[00].k_sch 000102030405060708090A0B0C0D0E0F
enc[01].start 00102030405060708090A0B0C0D0E0F0
enc[01].s_box 63CAB7040953D051CD60E0E7BA70E18C
enc[01].s_row 6353E08C0960E104CD70B751BACAD0E7
enc[01].m_col 5F72641557F5BC92F7BE3B291DB9F91A
enc[01].k_sch D6AA74FDD2AF72FADAA678F1D6AB76FE
enc[02].start 89D810E8855ACE682D1843D8CB128FE4
enc[02].s_box A761CA9B97BE8B45D8AD1A611FC97369
enc[02].s_row A7BE1A6997AD739BD8C9CA451F618B61
enc[02].m_col FF87968431D86A51645151FA773AD009
enc[02].k_sch B692CF0B643DBDF1BE9BC5006830B3FE
enc[03].start 4915598F55E5D7A0DACA94FA1F0A63F7
enc[03].s_box 3B59CB73FCD90EE05774222DC067FB68
enc[03].s_row 3BD92268FC74FB735767CBE0C0590E2D
enc[03].m_col 4C9C1E66F771F0762C3F868E534DF256
enc[03].k_sch B6FF744ED2C2C9BF6C590CBF0469BF41
enc[04].start FA636A2825B339C940668A3157244D17
enc[04].s_box 2DFB02343F6D12DD09337EC75B36E3F0
enc[04].s_row 2D6D7EF03F33E334093602DD5BFB12C7
enc[04].m_col 6385B79FFC538DF997BE478E7547D691
enc[04].k_sch 47F7F7BC95353E03F96C32BCFD058DFD
enc[05].start 2472402369 66B3FA6ED2753288425B6C
enc[05].s_box 36400926F9336D2D9FB59D23C42C3950
enc[05].s_row 36339D50F9B539269F2C092DC4406D23
enc[05].m_col F4BCD45432E554D075F1D6C51DD03B3C
enc[05].k_sch 3CAAA3E8A99F9DEB50F3AF57ADF622AA
enc[06].start C81677BC9B7AC93B25027992B0261996
enc[06].s_box E847F56514DADDE23F77B64FE7F7D490
enc[06].s_row E8DAB6901477D4653FF7F5E2E747DD4F
enc[06].m_col 9816EE7400F87F556B2C049C8E5AD036
enc[06].k_sch 5E390F7DF7A69296A7553DC10AA31F6B
enc[07].start C62FE109F75EEDC3CC79395D84F9CF5D
enc[07].s_box B415F8016858552E4BB6124C5F998A4C
enc[07].s_row B458124C68B68A014B99F82E5F15554C
enc[07].m_col C57E1C159A9BD286F05F4BE098C63439
enc[07].k_sch 14F9701AE35FE28C440ADF4D4EA9C026
enc[08].start D1876C0F79C4300AB45594ADD66FF41F
enc[08].s_box 3E175076B61C04678DFC2295F6A8BFC0
enc[08].s_row 3E1C22C0B6FCBF768DA85067F6170495
enc[08].m_col BAA03DE7A1F9B56ED5512CBA5F414D23
enc[08].k_sch 47438735A41C65B9E016BAF4AEBF7AD2
enc[09].start FDE3BAD205E5D0D73547964EF1FE37F1
enc[09].s_box 5411F4B56BD9700E96A0902FA1BB9AA1
enc[09].s_row 54D990A16BA09AB596BBF40EA111702F
enc[09].m_col E9F74EEC023020F61BF2CCF2353C21C7
enc[09].k_sch 549932D1F08557681093ED9CBE2C974E
enc[10].start BD6E7C3DF2B5779E0B61216E8B10B689
enc[10].s_box 7A9F102789D5F50B2BEFFD9F3DCA4EA7
enc[10].s_row 7AD5FDA789EF4E272BCA100B3D9FF59F
enc[10].k_sch 13111D7FE3944A17F307A78B4D2B30C5
enc[10].final 69C4E0D86A7B0430D8CDB78070B4C55A
```

Инверсное расшифрование

```
inv[00].input 69C4E0D86A7B0430D8CDB78070B4C55A
inv[00].k_sch 13111D7FE3944A17F307A78B4D2B30C5
inv[01].start 7AD5FDA789EF4E272BCA100B3D9FF59F
inv[01].s_row 7A9F102789D5F50B2BEFFD9F3DCA4EA7
inv[01].s_box BD6E7C3DF2B5779E0B61216E8B10B689
inv[01].k_sch 549932D1F08557681093ED9CBE2C974E
inv[01].k_add E9F74EEC023020F61BF2CCF2353C21C7
inv[01].m_col 54D990A16BA09AB596BBF40EA111702F
inv[02].start 54D990A16BA09AB596BBF40EA111702F
inv[02].s_row 5411F4B56BD9700E96A0902FA1BB9AA1
inv[02].s_box FDE3BAD205E5D0D73547964EF1FE37F1
inv[02].k_sch 47438735A41C65B9E016BAF4AEBF7AD2
inv[02].k_add BAA03DE7A1F9B56ED5512CBA5F414D23
inv[02].m_col 3E1C22C0B6FCBF768DA85067F6170495
inv[03].start 3E1C22C0B6FCBF768DA85067F6170495
inv[03].s_row 3E175076B61C04678DFC2295F6A8BFC0
inv[03].s_box D1876C0F79C4300AB45594ADD66FF41F
inv[03].k_sch 14F9701AE35FE28C440ADF4D4EA9C026
inv[03].k_add C57E1C159A9BD286F05F4BE098C63439
inv[03].m_col B458124C68B68A014B99F82E5F15554C
inv[04].start B458124C68B68A014B99F82E5F15554C
inv[04].s_row B415F8016858552E4BB6124C5F998A4C
inv[04].s_box C62FE109F75EEDC3CC79395D84F9CF5D
inv[04].k_sch 5E390F7DF7A69296A7553DC10AA31F6B
inv[04].k_add 9816EE7400F87F556B2C049C8E5AD036
inv[04].m_col E8DAB6901477D4653FF7F5E2E747DD4F
inv[05].start E8DAB6901477D4653FF7F5E2E747DD4F
inv[05].s_row E847F56514DADDE23F77B64FE7F7D490
inv[05].s_box C81677BC9B7AC93B25027992B0261996
inv[05].k_sch 3CAAA3E8A99F9DEB50F3AF57ADF622AA
inv[05].k_add F4BCD45432E554D075F1D6C51DD03B3C
inv[05].m_col 36339D50F9B539269F2C092DC4406D23
inv[06].start 36339D50F9B539269F2C092DC4406D23
inv[06].s_row 36400926F9336D2D9FB59D23C42C3950
inv[06].s_box 247240236966B3FA6ED2753288425B6C
inv[06].k_sch 47F7F7BC95353E03F96C32BCFD058DFD
inv[06].k_add 6385B79FFC538DF997BE478E7547D691
inv[06].m_col 2D6D7EF03F33E334093602DD5BFB12C7
inv[07].start 2D6D7EF03F33E334093602DD5BFB12C7
inv[07].s_row 2DFB02343F6D12DD09337EC75B36E3F0
inv[07].s_box FA636A2825B339C940668A3157244D17
inv[07].k_sch B6FF744ED2C2C9BF6C590CBF0469BF41
inv[07].k_add 4C9C1E66F771F0762C3F868E534DF256
inv[07].m_col 3BD92268FC74FB735767CBE0C0590E2D
inv[08].start 3BD92268FC74FB735767CBE0C0590E2D
inv[08].s_row 3B59CB73FCD90EE05774222DC067FB68
inv[08].s_box 4915598F55E5D7A0DACA94FA1F0A63F7
inv[08].k_sch B692CF0B643DBDF1BE9BC5006830B3FE
inv[08].k_add FF87968431D86A51645151FA773AD009
inv[08].m_col A7BE1A6997AD739BD8C9CA451F618B61
inv[09].start A7BE1A6997AD739BD8C9CA451F618B61
inv[09].s_row A761CA9B97BE8B45D8AD1A611FC97369
inv[09].s_box 89D810E8855ACE682D1843D8CB128FE4
inv[09].k_sch D6AA74FDD2AF72FADAA678F1D6AB76FE
inv[09].k_add 5F72641557F5BC92F7BE3B291DB9F91A
inv[09].m_col 6353E08C0960E104CD70B751BACAD0E7
inv[10].s_row 63CAB7040953D051CD60E0E7BA70E18C
inv[10].s_box 00102030405060708090A0B0C0D0E0F0
inv[10].k_sch 000102030405060708090A0B0C0D0E0F
inv[10].final 00112233445566778899AABBCCDDEEFF
```

Эквивалентное расшифрование

```
equ[00].input 69C4E0D86A7B0430D8CDB78070B4C55A
equ[00].k_sch 13111D7FE3944A17F307A78B4D2B30C5
equ[01].start 7AD5FDA789EF4E272BCA100B3D9FF59F
equ[01].s_box BDB52189F261B63D0B107C9E8B6E776E
equ[01].s_row BD6E7C3DF2B5779E0B61216E8B10B689
equ[01].m_col 4773B91FF72F354361CB018EA1E6CF2C
equ[01].k_sch 549932D1F08557681093ED9CBE2C974E
equ[01].k_add 54D990A16BA09AB596BBF40EA111702F
equ[02].start 54D990A16BA09AB596BBF40EA111702F
equ[02].s_box FDE596F1054737D235FEBAD7F1E3D04E
equ[02].s_row FDE3BAD205E5D0D73547964EF1FE37F1
equ[02].m_col 2D7E86A339D9393EE6570A1101904E16
equ[02].k_sch 47438735A41C65B9E016BAF4AEBF7AD2
equ[02].k_add 3E1C22C0B6FCBF768DA85067F6170495
equ[03].start 3E1C22C0B6FCBF768DA85067F6170495
equ[03].s_box D1C4941F7955F40FB46F6C0AD68730AD
equ[03].s_row D1876C0F79C4300AB45594ADD66FF41F
equ[03].m_col 39DAEE38F4F1A82AAF432410C36D45B9
equ[03].k_sch 14F9701AE35FE28C440ADF4D4EA9C026
equ[03].k_add B458124C68B68A014B99F82E5F15554C
equ[04].start B458124C68B68A014B99F82E5F15554C
equ[04].s_box C65E395DF779CF09CCF9E1C3842FED5D
equ[04].s_row C62FE109F75EEDC3CC79395D84F9CF5D
equ[04].m_col 9A39BF1D05B20A3A476A0BF79FE51184
equ[04].k_sch 5E390F7DF7A69296A7553DC10AA31F6B
equ[04].k_add E8DAB6901477D4653FF7F5E2E747DD4F
equ[05].start E8DAB6901477D4653FF7F5E2E747DD4F
equ[05].s_box C87A79969B0219BC2526773BB016C992
equ[05].s_row C81677BC9B7AC93B25027992B0261996
equ[05].m_col 18F78D779A93EEF4F6742967C47F5FFD
equ[05].k_sch 3CAAA3E8A99F9DEB50F3AF57ADF622AA
equ[05].k_add 36339D50F9B539269F2C092DC4406D23
equ[06].start 36339D50F9B539269F2C092DC4406D23
equ[06].s_box 2466756C69D25B236E4240FA8872B332
equ[06].s_row 247240236966B3FA6ED2753288425B6C
equ[06].m_col 85CF8BF472D124C10348F545329C0053
equ[06].k_sch 47F7F7BC95353E03F96C32BCFD058DFD
equ[06].k_add 2D6D7EF03F33E334093602DD5BFB12C7
equ[07].start 2D6D7EF03F33E334093602DD5BFB12C7
equ[07].s_box FAB38A1725664D2840246AC957633931
equ[07].s_row FA636A2825B339C940668A3157244D17
equ[07].m_col FC1FC1F91934C9821OFBFB8DA340EB21
equ[07].k_sch B6FF744ED2C2C9BF6C590CBF0469BF41
equ[07].k_add 3BD92268FC74FB735767CBE0C0590E2D
equ[08].start 3BD92268FC74FB735767CBE0C0590E2D
equ[08].s_box 49E594F755CA638FDA0A59A01F15D7FA
equ[08].s_row 4915598F55E5D7A0DACA94FA1F0A63F7
equ[08].m_col 076518F0B52BA2FB7A15C8D93BE45E00
equ[08].k_sch B692CF0B643DBDF1BE9BC5006830B3FE
equ[08].k_add A7BE1A6997AD739BD8C9CA451F618B61
equ[09].start A7BE1A6997AD739BD8C9CA451F618B61
equ[09].s_box 895A43E485188FE82D121068CBD8CED8
equ[09].s_row 89D810E8855ACE682D1843D8CB128FE4
equ[09].m_col EF053F7C8B3D32FD4D2A64AD3C93071A
equ[09].k_sch D6AA74FDD2AF72FADAA678F1D6AB76FE
equ[09].k_add 6353E08C0960E104CD70B751BACAD0E7
equ[10].s_box 0050A0F04090E0308OD02070C01060B0
equ[10].s_row 00102030405060708090A0B0C0D0E0F0
equ[10].k_sch 000102030405060708090A0B0C0D0E0F
equ[10].final 00112233445566778899AABBCCDDEEFF
```

Обозначения:

`.input` – данные перед операцией шифрования;
`.start` – блок на начало раунда;
`.k_sch` – значение раундового ключа;
`.k_add` – блок после добавления раундового ключа;
`.s_box` – блок после операций (Inv)SubBytes;
`.s_row` – блок после операций (Inv)ShiftRows;
`.m_col` – блок после операций (Inv)MixColumns;
`.final` – результат шифрования.

Visual AES, как средство исследования.

Специфической функцией программы Visual AES является возможность управления математическим аппаратом раундовых преобразований с целью исследования алгоритма AES, а также получения новых средств шифрования на основе методов заложенных в нем.

Преобразование SubBytes()

На Рисунке 4.12 изображено окно настроек математических преобразований в рамках раундовой операции SubBytes. Как уже было сказано в главе 2, операции замены байт

и обратной замены байт для некоторого a в стандарте AES описываются следующими математическими выражениями:

$\lambda(f) \equiv (((X^4 + X^3 + X^2 + X + 1) \cdot f) \bmod (X^8 + X^4 + X^3 + X + 1) + X^6 + X^5 + X + 1) \bmod (X^8 + 1)$

$\lambda^{-1}(f) \equiv (((X^6 + X^3 + X) \cdot f) \bmod (X^8 + X^4 + X^3 + X + 1) + X^2 + 1) \bmod (X^8 + 1)$

$\sigma(a) = \lambda(a^{254})$, $\sigma^{-1}(a) = (\lambda^{-1}(a))^{254}$ - мультипликативная инверсия.

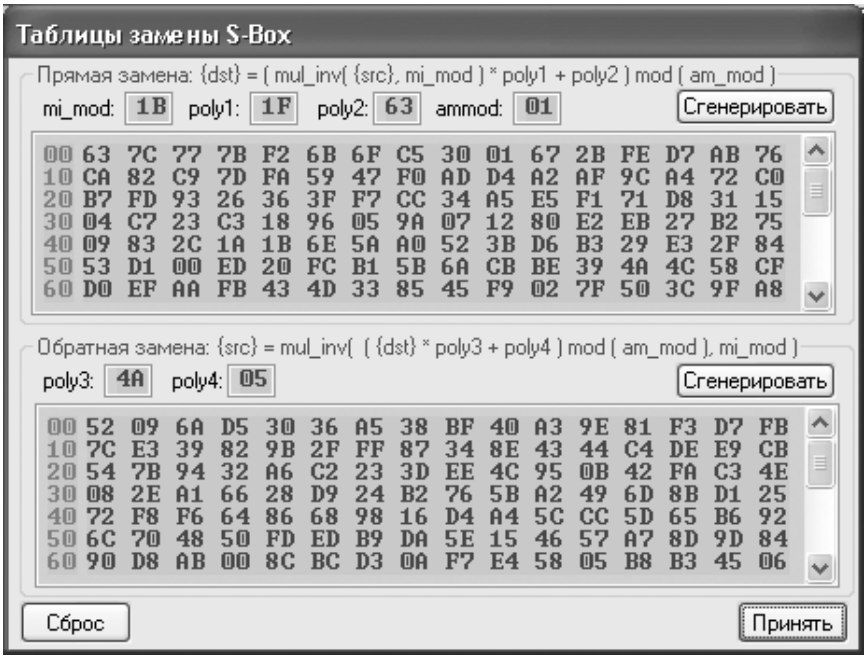


Рисунок 4.12 **«Окно Visual AES настроек преобразования SubBytes»**

В системе Visual AES предлагается параметризировать данные вычисления так:

$\lambda(f) \equiv ((poly1 \cdot f) \bmod (mi\_mod) + poly2) \bmod (am\_mod)$

$\lambda^{-1}(f) \equiv (((poly3) \cdot f) \bmod (mi\_mod) + poly4) \bmod (am\_mod)$

$[\, poly1, poly2\,], [\, poly3, poly4\,]$- произвольно задаваемые пары полиномов значения которых удовлетворяют условию обратимости замены байт; mi_mod - модуль вычисления мультипликативной инверсии; am_mod - модуль преобразования замены байт *affine map.*

Преобразование ShiftRows()

Раундовое преобразование сдвига строк в стандарте AES имеет строго определенную (см. глава 3) конфигурацию зависящую от размеров ключа и блока. В системе Visual AES существует возможность задания произвольных смещений для каждой из строк состояния– Рисунок 4.13.

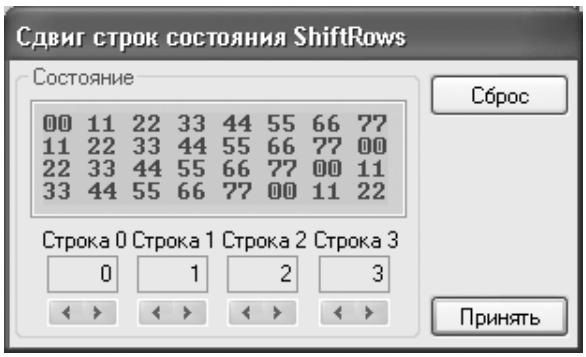


Рисунок 4.13 **«Окно Visual AES настроек преобразования ShiftRows»**

Преобразование MixColumns()

Преобразование перемешивания строк основывается на матричной операции умножения столбцов состояния представленных в виде четырехбайтных слов g на многочлены c и d для прямого и обратного преобразований соответственно, причем последние сдвигаются на n байт вправо, где n - номер столбца в состоянии, над которым производится операция.

$c = (X,1,1,X+1)$, $d = (X^3 + X^2 + X, X^3 + 1, X^3 + X^2 + 1, X^3 + X + 1)$

$\mu(g) \equiv c \cdot g \bmod(Y^4 + 1)$, $\nu(g) \equiv d \cdot g \bmod(Y^4 + 1)$

На Рисунке 4.14 показано соответствующее окно комплекса Visual AES, позволяющее задавать значения полиномов c и d побайтно, при сохранении в основной операции правил их сдвига.

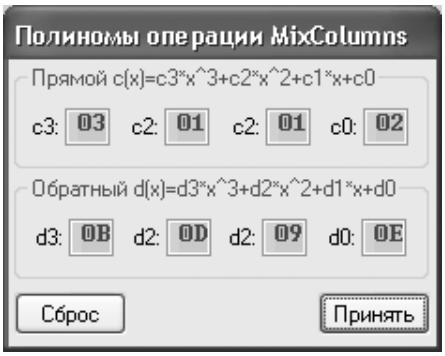


Рисунок 4.14 **«Окно Visual AES настроек преобразования MixColumns»**

Преобразование AddRoundKey()

Специализированное окно трансформаций AddRoundKey (Рисунок 4.15) позволяет изучить содержимое развернутых (раундовых) подключей для каждой из возможных операций шифрования: зашифрование, инверсное и эквивалентное расшифрования и, если это необходимо, средствами DDE произвести копирование этих данных в системный карман Windows.

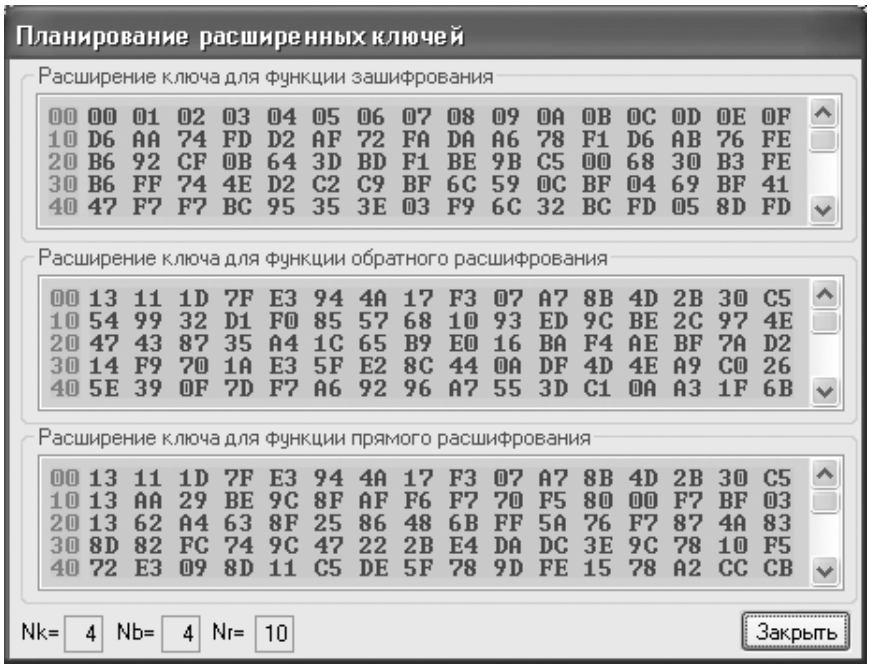


Рисунок 4.15 **«Окно Visual AES настроек преобразования AddRoundKey»**

Visual AES, как средство анализа.

Инструментарий «Эксперт» доступный из среды Visual AES, предназначен для аналитических наблюдений за раундовыми преобразованиями алгоритма AES над блоками данных. В качестве дополнительного к нему режима реализована единственная, на сегодняшний день, эффективная атака «Квадрат» представленная авторами шифра в качестве демонстрации стойкости алгоритма (Рисунок 4.16).

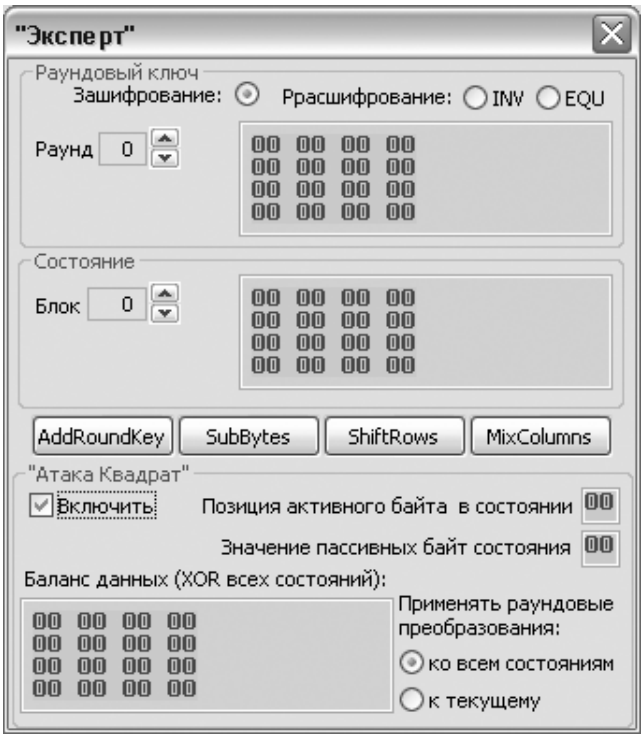


Рисунок 4.16 **«Инструмент пошаговой трассировки системыVisual AES»**

Пошаговая трассировка алгоритма AES

В режиме пошаговой трассировки инструмент «Эксперт» функционирует следующим образом: в зависимости от установок основной программы (ключа, входных файлов и выбранного алгоритма) возможно переключение между блоками (состояниями) соответствующего файла с помощью прокрутки «Блок» и элементами расширенного ключа - прокрутка «Раунд», для последующего применения к состоянию в произвольном порядке операций раундовых преобразований с учетом настройки их математики и наблюдением за ними.

Данные преобразования носят локальный характер и в исходных файлах они не отражаются, т.е. при переключении между блоками история трассировки теряется.

Атака «Квадрат»

Атака «Квадрат» была специально разработана для одноимённого шифра *SQUARE* (авторы J. Daemen, L. Knudsen, V. Rijmen). Атака использует при своем проведении байт-ориентированную структуру шифра. Учитывая, что *AES* унаследовал многие свойства шифра *SQUARE*, эта атака применима и к нему.

Атака «Квадрат» основана на возможности свободного подбора атакующим некоторого набора открытых текстов для последующего их зашифрования. Она независима от таблиц замен *S*-блоков, многочлена функции *MixColumns*( ) и способа разворачивания ключа. Эта атака для 6-раундового шифра *AES*, эффективнее, чем полный перебор по всему ключевому пространству. После описания базовой атаки на четырёх-раундовый *AES*, будет показано как эту атаку можно продлить на 5 и даже 6 раундов. Но уже для 7 раундов «Квадрат» становится менее эффективным, чем полный перебор.

Предпосылки.

Пусть Λ-набор - такой набор из 256 входных блоков (массивов *State*), каждый из которых имеет байты (назовём их *активными*), значения которых различны для всех 256 блоков. Остальные байты (будем называть их *пассивными*) остаются одинаковыми для всех 256 блоков из Λ-набора. То есть:

$$\forall x, y \in \Lambda : \begin{cases} x_{ij} \neq y_{ij}, \text{если байт с номером } ij \text{ активный;} \\ x_{ij} = y_{ij} \text{ в противном случае.} \end{cases}$$

Будучи подвергнутыми обработке функциями *SubBytes*( ) и *AddRounKey*( ) блоки Λ-набора дадут в результате другой Λ-набор с активными байтами в тех же позициях, что и у исходного. Функция *ShiftRows*( ) сместит эти байты соответственно заданным в ней смещениям в строках массивов *State*. После функции *MixColumns*( ) Λ-набор в общем случае необязательно останется Λ-набором (т.е. результат преобразования может перестать удовлетворять определению Λ-набора). Но, поскольку каждый байт результата функции *MixColumns*( ) является линейной комбинацией (с обратимыми коэффициентами) четырёх входных байт того же столбца

$b_{ij} = 2a_{ij} \oplus 3a_{(i+1)j} \oplus a_{(i+2)j} \oplus a_{(i+3)j}$,

столбец с единственным активным байтом на входе даст в результате на выходе столбец со всеми четырьмя байтами – активными.

Базовая атака «Квадрат» на 4 раунда.

Рассмотрим Λ-набор, во всех блоках которого активен только один байт. Иначе говоря, значение этого байта различно во всех 256 блоках, а остальные байты одинаковы (скажем, равны нулю). Проследим эволюцию этого байта на протяжении трёх раундов. В первом раунде функция *MixColumns*( ) преобразует один активный байт в столбец из 4 активных байт. Во втором раунде эти 4 байта разойдутся по 4 различным столбцам в результате преобразования функцией *ShiftRows*( ). Функция же *MixColumns*( ) следующего, третьего раунда преобразует эти байты в 4 столбца, содержащие активные байты. Этот набор всё ещё остаётся Λ-набором до того самого момента, когда он поступает на вход функции *MixColumns*( ) третьего раунда.

Основное свойство Λ-набора, используемое здесь, то, что поразрядная сумма по модулю 2 всех блоков такого набора всегда равна нулю. Действительно, поразрядная сумма неактивных (с одинаковыми значениями) байт равна нулю по определению операции поразрядного *XOR*, а активные байты, пробегая все 256 значений, также при поразрядном суммировании дадут нуль. Рассмотрим теперь результат преобразования функцией *MixColumns*( ) в третьем раунде байтов входного массива данных *a* в байты выходного массива данных *b*. Покажем, что и в этом случае поразрядная сумма всех блоков выходного набора будет равна нулю, то есть:

$$b = MixColumns(a), a \in \Lambda \overset{\oplus}{} b_{ij} = \overset{\oplus}{a \in \Lambda} \left(2a_{ij} \oplus 3a_{(i+1)j} \oplus a_{(i+2)j} \oplus a_{(i+3)j}\right) =$$

$$= \overset{2\oplus}{a \in \Lambda} a_{ij} \oplus \overset{3\oplus}{a \in \Lambda} a_{(i+1)j} \oplus \overset{\oplus}{a \in \Lambda} a_{(i+2)j} \oplus \overset{\oplus}{a \in \Lambda} a_{(i+3)j}.$$

Таким образом, все данные на входе четвёртого раунда сбалансированы (т. е. их полная сумма равна нулю). Этот баланс в общем случае нарушается последующим преобразованием данных функцией *SubBytes*( ).

Мы предполагаем далее, что четвёртый раунд является последним, то есть в нём нет функции *MixColumns*( ). Тогда каждый байт выходных данных этого раунда зависит только от одного байта входных данных. Если обозначить через *a* байт выходных данных четвёртого раунда, через *b* байт входных данных и через *k* – соответствующий байт раундового ключа, то можно записать:

$a_{ij} = SubBytes(b_{ij}) \oplus k_{ij}$.

Отсюда, предполагая значение k_{ij}, можно по известному a_{ij} вычислить b_{ij}, а затем проверить правильность догадки о значении k_{ij}: если значения байта b_{ij}, полученные при данном k_{ij} не будут сбалансированы по всем блокам (то есть не дадут при поразрядном суммировании нулевой результат), значит догадка неверна. Перебрав максимум 2^8 вариантов байта раундового ключа, мы найдем его истинное значение.

По такому же принципу могут быть определены и другие байты раундового ключа. За счёт того, что поиск может производиться отдельно (читай – параллельно) для каждого байта ключа, скорость подбора всего значения раундового ключа весьма велика. А по значению полного раундового ключа, при известном алгоритме его развёртывания, не составляет труда восстановить сам начальный ключ шифрования.

Добавление пятого раунда в конец базовой атаки «Квадрат».

Если будет добавлен пятый раунд, то значения b_{ij} придётся вычислять уже на основании выходных данных не четвёртого, а пятого раунда. И дополнительно кроме байта раундового ключа четвёртого раунда перебирать ещё значения четырёх байт столбца раундового ключа для пятого раунда. Только так мы сможем выйти на значения сбалансированных байт b_{ij} входных данных четвёртого раунда.

Таким образом, теперь нам нужно перебрать 2^{40} значений - 2^{32} вариантов для 4 байт столбца раундового ключа пятого раунда и для каждого из них 2^8 вариантов для одного байта четвёртого раунда. Эту процедуру нужно будет повторить для всех четырёх столбцов пятого раунда. Поскольку при подборе «верного»[1] значения байта раундового ключа четвёртого раунда количество «неверных»[2] ключей уменьшается в 2^8 раз, то работая одновременно с пятью Λ-наборами, можно с большой степенью вероятности правильно подобрать все 2^{40} бита. Теперь поиск может производиться отдельно (т.е. параллельно) для каждого столбца каждого из пяти Λ-наборов, что опять же гораздо быстрее полного перебора всех возможных значений ключа.

Добавление шестого раунда в начало базовой атаки «Квадрат».

Основная идея заключается в том, чтобы подобрать такой набор блоков открытого текста, который на выходе после первого раунда давал бы Λ-набор с одним активным байтом. Это требует предположения о значении четырёх байт ключа, используемых функцией *AddRoundKey*( ) перед первым раундом.

Для того, чтобы на входе второго раунда был только один активный байт достаточно, чтобы в первом раунде один активный байт оставался на выходе функции *MixColumns*( ). Это означает, что на входе *MixColumns*( ) первого раунда должен быть такой столбец, байты *a* которого для набора из 256 блоков в результате линейного преобразования

$b_i = 2a_i \oplus 3a_{i+1} \oplus a_{i+2} \oplus a_{i+3}$, $0 \le i \le 3$,

где *i* – номер строки, для одного определённого *i* давали 256 различных значений, в то время как для каждого из остальных трёх значений *i* результат этого преобразования должен оставаться постоянным. Следуя обратно по порядку приложения функций преобразования в первом раунде, к *ShiftRows*( ) данное условие нужно применить к соответственно разнесённым по столбцам 4 байтам. С учётом применения функции *SubBytes*( ) и сложения с предполагаемым значением 4-байтового раундового ключа можно смело составлять уравнения и подбирать нужные значения байт открытого текста, подаваемых на зашифрование для последующего анализа результата:

$b_{ij} = 2SubBytes(a_{ij} \oplus k_{ij}) \oplus 3SubBytes(a_{(i+1)(j+1)} \oplus k_{(i+1)(j+1)}) \oplus$

$\oplus SubBytes(a_{(i+2)(j+2)} \oplus k_{(i+2)(j+2)}) \oplus SubBytes(a_{(i+3)(j+3)} \oplus k_{(i+3)(j+3)})$,

[1] Кавычки здесь означают то, что *верным* это значение может быть названо лишь с некоторой (но довольно большой) степенью вероятности.

[2] То же.

$0 \le i, j \le 3$.

Таким образом, получаем следующий алгоритм взлома. Имеем всего 2^{32} различных значений a для определённых i и j. Остальные байты для всех блоков одинаковы (пассивные байты). Предположив значения четырёх байт k ключа первого раунда, подбираем (исходя из вышеописанного условия) набор из 256 блоков. Эти 256 блоков станут Λ-набором после первого раунда. К этому Λ-набору применима базовая атака для 4 раундов. Подобранный с её помощью один байт ключа последнего раунда фиксируется. Теперь подбирается новый набор из 256 блоков для того же значения 4 байт k ключа первого раунда. Опять осуществляется базовая атака, дающая один байт ключа последнего раунда. Если после нескольких попыток значение этого байта не меняется, значит мы на верном пути. В противном случае нужно менять предположение о значении 4 байт k ключа первого раунда. Такой алгоритм действий достаточно быстро приведёт к полному восстановлению всех байт ключа последнего раунда.

Реализация данной атаки средствами инструмента «Эксперт» выглядит следующим образом (см. Рисунок 4.16, «Атака квадрат»): при активации данного режима в динамической памяти создается Λ-набор, между элементами которого можно переключаться с помощью прокрутки «Блок». Параметры Λ-набора можно задать тут же (позиция активного байта в состоянии и значения пассивных байт). Выбор раундового ключа осуществляется с помощью прокрутки «Раунд». Далее, по нажатии одной из кнопок соответствующих одной из раундовых трансформаций, она будет применена ко всем элементам Λ-набора. Для контроля баланса данных выводиться поразрядная сумма всех состояний Λ-набора.

В 4-х раундовом алгоритме данная реализация инструментария имеет очень высокую эффективность, но с увеличением числа раундов необходимы дополнительные средства для хранения сопроводительной информации и инструментарий статистического анализа.