Симметричные криптосистемы: ГОСТ 28147-89

Лекция

Ревизия: 0.1

Последняя модификация: Vlad Kovtun
© NRJETIX 2000 - 2008
Дата последней модификации: 31.01.2011 21:22:00

История изменений

25.10.2009 – Версия 0.1. Первичный документ. Ковтун В.Ю.

Содержание

Лекция 3. Симметричные блочные криптосистемы: ГОСТ 28147-89. Часть 2

Вопросы

1. Введение.
2. Описание симметричной блочной криптосистемы ГОСТ 28147-89.
3. Особенности реализации.

Введение

То, что информация имеет ценность, люди осознали очень давно – недаром переписка сильных мира сего издавна была объектом пристального внимания их недругов и друзей. Тогда-то и возникла задача защиты этой переписки от чрезмерно любопытных глаз. Древние пытались использовать для решения этой задачи самые разнообразные методы, и одним из них была тайнопись – умение составлять сообщения таким образом, чтобы его смысл был недоступен никому кроме посвященных в тайну. Есть свидетельства тому, что искусство тайнописи зародилось еще в доантичные времена. На протяжении всей своей многовековой истории, вплоть до совсем недавнего времени, это искусство служило немногим, в основном верхушке общества, не выходя за пределы резиденций глав государств, посольств и – конечно же ! – разведывательных миссий. И лишь несколько десятилетий назад все изменилось коренным образом – информация приобрела самостоятельную коммерческую ценность и стала широко распространенным, почти обычным товаром. Ее производят, хранят, транспортируют, продают и покупают, а значит – воруют и подделывают – и, следовательно, ее необходимо защищать. Современное общество все в большей степени становится информационно–обусловленным, успех любого вида деятельности все сильней зависит от обладания определенными сведениями и от отсутствия их у конкурентов. И чем сильней проявляется указанный эффект, тем больше потенциальные убытки от злоупотреблений в информационной сфере, и тем больше потребность в защите информации. Одним словом, возникновение индустрии обработки информации с железной необходимостью привело к возникновению индустрии средств защиты информации.

Среди всего спектра методов защиты данных от нежелательного доступа особое место занимают криптографические методы. В отличие от других методов, они опираются лишь на свойства самой информации и не используют свойства ее материальных носителей, особенности узлов ее обработки, передачи и хранения. Образно говоря, криптографические методы строят барьер между защищаемой информацией и реальным или потенциальным злоумышленником из самой информации. Конечно, под криптографической защитой в первую очередь – так уж сложилось исторически – подразумевается шифрование данных. Раньше, когда эта операция выполнялось человеком вручную или с использованием различных приспособлений, и при посольствах содержались многолюдные отделы шифровальщиков, развитие криптографии сдерживалось проблемой реализации шифров, ведь придумать можно было все что угодно, но как это реализовать… Появление цифровых электронно-вычислительных машин, приведшее в конечном итоге к созданию мощной информационной индустрии, изменило все коренным образом и в этой сфере. С одной стороны, взломщики шифров получили в свои руки чрезвычайно мощное орудие, с другой стороны, барьер сложности реализации исчез и для создателей шифров открылись практически безграничные перспективы. Все это определило стремительный прогресс криптографии в последние десятилетия.

Описание симметричной блочной криптосистемы ГОСТ 28147-89

Термины и обозначения

Описание стандарта шифрования Украины, Российской Федерации и других стран бывшего СССР, содержится в очень интересном документе ГОСТ СССР, озаглавленном «Алгоритм криптографического преобразования данных ГОСТ 28147-89». То, что в его названии вместо термина «шифрование» фигурирует более общее понятие «***криптографическое преобразование***», вовсе не случайно. Помимо нескольких тесно связанных между собой процедур шифрования, в документе описан один построенный на общих принципах с ними алгоритм выработки ***имитовставки***. Последняя является не чем иным, как криптографической контрольной комбинацией,

то есть кодом, вырабатываемым из исходных данных с использованием секретного ключа с целью ***имитозащиты***, или защиты данных от внесения в них несанкционированных изменений.

На различных шагах алгоритма ГОСТа, данные, которыми они оперируют, интерпретируются и используются различным образом. В некоторых случаях элементы данных обрабатываются как массивы независимых битов, в других случаях – как целое число без знака, в третьих – как имеющий структуру сложный элемент, состоящий из нескольких более простых элементов. Поэтому во избежание путаницы следует договориться об используемых обозначениях.

Элементы данных в данной лекции обозначаются заглавными латинскими буквами с наклонным начертанием (например, $\mathbf{X}$). Через $|\mathbf{X}|$ обозначается размер элемента данных $\mathbf{X}$ в битах. Таким образом, если интерпретировать элемент данных $\mathbf{X}$ как целое неотрицательное число, можно записать следующее неравенство: $0 \le \mathbf{X} < 2^{|\mathbf{X}|}$.

Если элемент данных состоит из нескольких элементов меньшего размера, то этот факт обозначается следующим образом: $\mathbf{X} = (x_0, x_1, \ldots, x_{n-1}) = X_0 \,||\, X_1 \,||\, \ldots \,||\, X_{n-1}$. Процедура объединения нескольких элементов данных в один называется ***конкатенацией*** данных и обозначается символом ||. Естественно, для размеров элементов данных должно выполняться следующее соотношение: $|\mathbf{X}| = |X_0| + |X_1| + \cdots + |X_{n-1}|$. При задании сложных элементов данных и операции конкатенации, составляющие элементы данных перечисляются в порядке возрастания старшинства. Иными словами, если интерпретировать составной элемент и все входящие в него элементы данных как целые числа без знака, то можно записать следующее равенство:

$$(X_0, X_1, \ldots, X_{n-1}) = X_0 \,||\, X_1 \,||\, \ldots \,||\, X_{n-1} = X_0 + 2^{|X_0|}(X_1 + 2^{|X_1|}(\ldots(X_{n-2} + 2^{|X_{n-2}|} X_{n-1})\ldots)).$$

В алгоритме элемент данных может интерпретироваться как массив отдельных битов, в этом случае биты обозначаем той же самой буквой, что и массив, но в строчном варианте, как показано на следующем примере:

$$\mathbf{X} = (x_0, x_1, \ldots, x_{n-1}) = x_0 + 2x_1 + \cdots + 2^{n-1} x_{n-1}.$$

Если над элементами данных выполняется некоторая операция, имеющая логический смысл, то предполагается, что данная операция выполняется над соответствующими битами элементов. Иными словами $A \bullet B = (a_0 \bullet b_0, a_1 \bullet b_1, \ldots, a_{n-1} \bullet b_{n-1})$, где $n = |A| = |B|$, а символом «$\bullet$» обозначается произвольная бинарная логическая операция; как правило, имеется ввиду операция ***исключающего или***, она же – операция суммирования по модулю 2: $a \oplus b = (ab) \bmod 2$.

Логика построения шифра и структура ключевой информации ГОСТа

Если внимательно изучить оригинал ГОСТа 28147–89, можно заметить, что в нем содержится описание алгоритмов нескольких уровней. На самом верхнем находятся практические алгоритмы, предназначенные для шифрования массивов данных и выработки для них имитовставки. Все они опираются на три алгоритма низшего уровня, называемые в тексте ГОСТа ***циклами***. Эти фундаментальные алгоритмы упоминаются в данной лекции как ***базовые циклы***, чтобы отличать их от всех прочих циклов. Они имеют следующие названия и обозначения, последние приведены в скобках и смысл их будет объяснен позже:

- цикл зашифровывания (32-З);
- цикл расшифровывания (32-Р);
- цикл выработки имитовставки (16-З).

В свою очередь, каждый из базовых циклов представляет собой многократное повторение одной единственной процедуры, называемой для определенности далее в настоящей работе ***основным шагом криптопреобразования***.

Таким образом, чтобы разобраться в ГОСТе, надо понять три следующие вещи:

а) что такое ***основной шаг*** криптопреобразования;

б) как из ***основных шагов*** складываются ***базовые циклы***;

в) как из трех ***базовых циклов*** складываются все практические алгоритмы ГОСТа.

Прежде чем перейти к изучению этих вопросов, следует поговорить о ключевой информации, используемой алгоритмами ГОСТа. В соответствии с принципом Кирхгофа, которому удовлетворяют все современные известные широкой общественности шифры, именно ее секретность обеспечивает секретность зашифровыванного сообщения. В ГОСТе ключевая информация состоит из двух структур данных. Помимо собственно ***ключа***, необходимого для всех шифров, она содержит еще и ***таблицу замен***. Ниже приведены основные характеристики ключевых структур ГОСТа.

1. ***Ключ*** является массивом из восьми 32-битных элементов кода, далее в настоящей работе он обозначается символом $\mathbf{K} = \{K_i\}_{0 \le i \le 7}$. В ГОСТе элементы ключа используются как 32-разрядные целые числа без знака: $0 \le K_i < 2^{32}$. Таким образом, размер ключа составляет 32·8=256 бит или 32 байта.

2. ***Таблица замен*** является матрицей 8×16, содержащей 4-битовые элементы, которые можно представить в виде целых чисел от 0 до 15. Строки ***таблицы замен*** называются ***узлами замен***, они должны содержать различные значения, то есть каждый ***узел замен*** должен содержать 16 различных чисел от 0 до 15 в произвольном порядке. В настоящей лекции таблица замен обозначается символом $H = \{H_{i,j}\}_{0 \le j \le 15}^{0 \le i \le 7}, 0 \le H_{i,j} \le 15$. Таким образом, общий объем таблицы замен равен: 8 узлов × 16 элементов/узел × 4 бита/элемент = 512 бит или 64 байта.

Основной шаг криптопреобразования

Основной шаг криптопреобразования по своей сути является оператором, определяющим преобразование 64-битового блока данных. Дополнительным параметром этого оператора является 32-битовый блок, в качестве которого используется какой-либо элемент ключа. Схема алгоритма основного шага приведена на рисунке 1. Ниже даны пояснения к алгоритму основного шага:

Шаг 0. Определяет исходные данные для основного шага криптопреобразования:

- $\mathbf{N}$ –преобразуемый 64-битовый блок данных, в ходе выполнения шага его младшая (N_1) и старшая (N_2) части обрабатываются как отдельные 32-битовые целые числа без знака. Таким образом, можно записать $\mathbf{N} = (N_1, N_2)$.
- $\mathbf{X}$ –32-битовый элемент ключа;

Шаг 1. Сложение с ключом. Младшая половина преобразуемого блока складывается по модулю 2^{32} с используемым на шаге элементом ключа, результат передается на следующий шаг;

Шаг 2. Поблочная замена. 32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков кода: $\mathbf{S} = (S_0, S_1, S_2, \ldots, S_7)$.

Далее значение каждого из восьми блоков заменяется на новое, которое выбирается по таблице замен следующим образом: значение блока S_i заменяется на S_i-тый по порядку элемент (нумерация с нуля) i-того узла замен (т.е. i-той строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. Теперь становится понятным размер таблицы замен: число строк в ней равно числу 4-битных элементов в 32-битном блоке данных, то есть восьми, а число столбцов равно числу различных значений 4-битного блока данных, равному, как известно, 2^4, шестнадцати.

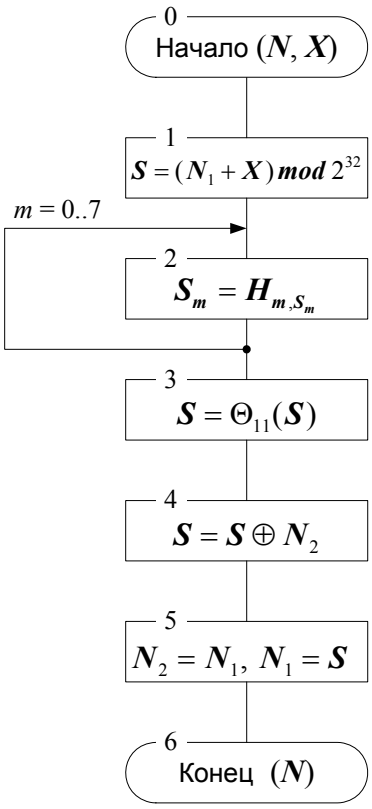


Рис. 1. Схема основного шага криптопреобразования алгоритма ГОСТ 28147-89

Шаг 3.Циклический сдвиг на 11 бит влево. Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий шаг. На схеме алгоритма символом Θ_{11} обозначена функция циклического сдвига своего аргумента на 11 бит в сторону старших разрядов.

Шаг 4. Побитовое сложение: значение, полученное на шаге 3, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.

Шаг 5. Сдвиг по цепочке: младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.

Шаг 6. Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага криптопреобразования.

Базовые циклы криптографических преобразований

Как отмечено в начале настоящей лекции, ГОСТ относится к классу блочных шифров, то есть единицей обработки информации в нем является блок данных. Следовательно, вполне логично ожидать, что в нем будут определены алгоритмы для криптографических преобразований, то есть для зашифровывания, расшифровывания и «учета» в контрольной комбинации одного блока данных. Именно эти алгоритмы и называются ***базовыми циклами*** ГОСТа, что подчеркивает их фундаментальное значение для построения этого шифра.

Базовые циклы построены из ***основных шагов*** криптографического преобразования, рассмотренного в предыдущем разделе. В процессе выполнения основного шага используется только один элемент ключа, в то время как ключ ГОСТ содержит восемь таких элементов. Следовательно, чтобы ключ был использован полностью, каждый из базовых циклов должен многократно выполнять основной шаг с различными его элементами. Вместе с тем кажется вполне естественным, что в каждом базовом цикле все элементы ключа должны быть использованы одинаковое число раз, по соображениям стойкости шифра это число должно быть больше одного.

Все сделанные выше предположения, опирающиеся просто на здравый смысл, оказались верными. Базовые циклы заключаются в многократном выполнении *основного шага* с использованием разных элементов ключа и отличаются друг от друга только числом повторения шага и порядком использования ключевых элементов. Ниже приведен этот порядок для различных циклов.

1. Цикл зашифровывания 32-З:

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,\ K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$
$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,\ K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$

2. Цикл расшифровывания 32-Р:

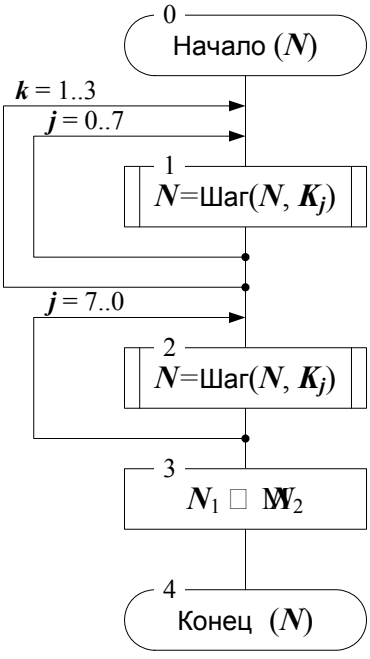
$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,\ K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,$
$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,\ K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$

3. Цикл выработки имитовставки 16-З:

$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,\ K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$

Каждый из циклов имеет собственное буквенно-цифровое обозначение, соответствующее шаблону «$n-X$», где первый элемент обозначения (n), задает число повторений основного шага в цикле, а второй элемент обозначения (X), буква, задает порядок зашифровывания («З») или расшифровывания («Р») в использовании ключевых элементов. Этот порядок нуждается в дополнительном пояснении:

Цикл расшифровывания должен быть обратным циклу зашифровывания, то есть последовательное применение этих двух циклов к произвольному блоку должно дать в итоге исходный блок, что отражается следующим соотношением: $Ц_{32-Р}(Ц_{32-З}(T)) = T$, где T – произвольный 64-битный блок данных, $Ц_X(T)$ – результат выполнения цикла X над блоком данных T. Для выполнения этого условия для алгоритмов, подобных ГОСТу, необходимо и достаточно, чтобы порядок использования ключевых элементов соответствующими циклами был взаимно обратным. В справедливости записанного условия для рассматриваемого случая легко убедиться, сравнив приведенные выше последовательности для циклов 32-З и 32-Р. Из сказанного вытекает одно интересное следствие: свойство цикла быть обратным другому циклу является взаимным, то есть цикл 32-З является обратным по отношению к циклу 32-Р. Другими словами, зашифровывание блока данных теоретически может быть выполнено с помощью цикла расшифровывания, в этом случае расшифровывание блока данных должно быть выполнено циклом зашифровывания. Из двух взаимно обратных циклов любой может быть использован для зашифровывания, тогда второй должен быть использован для расшифровывания данных, однако стандарт ГОСТ28147-89 закрепляет роли за циклами и не предоставляет пользователю права выбора в этом вопросе.



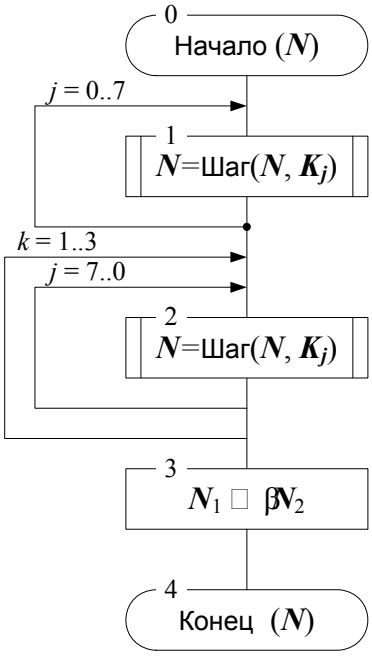


Рис. 2а. Схема цикла зашифровывания 32-З. Рис. 2б. Схема цикла расшифровывания 32-Р

Цикл выработки имитовставки вдвое короче циклов шифрования, порядок использования ключевых элементов в нем такой же, как в первых 16 шагах цикла зашифровывания, в чем нетрудно убедиться, рассмотрев приведенные выше последовательности, поэтому этот порядок в обозначении цикла кодируется той же самой буквой «З».

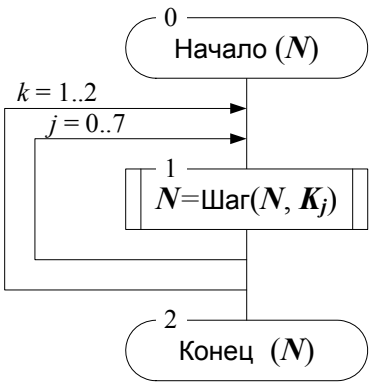


Рис. 2в. Схема цикла выработки имитовставки 16-З

Схемы базовых циклов приведены на рисунках 2а-в. Каждый из них принимает в качестве аргумента и возвращает в качестве результата 64-битный блок данных, обозначенный на схемах $\mathbf{N}$. Символ Шаг$(\mathbf{N}, \mathbf{X})$ обозначает выполнение основного шага криптопреобразования для блока $\mathbf{N}$ с использованием ключевого элемента $\mathbf{X}$. Между циклами шифрования и вычисления имитовставки есть еще одно отличие, не упомянутое выше: в конце базовых циклов шифрования старшая и младшая часть блока результата меняются местами, это необходимо для их взаимной обратимости.

Основные режимы шифрования

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

- простая замена,
- гаммирование,
- гаммирование с обратной связью,
- и один дополнительный режим выработки имитовставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита, на которые разбивается массив, подвергаемый криптографическому преобразованию, именно поэтому ГОСТ относится к блочным шифрам. Однако в двух режимах гаммирования есть возможность обработки неполного блока данных размером меньше 8 байт, что существенно при шифровании массивов данных с произвольным размером, который может быть не кратным 8 байтам.

Прежде чем перейти к рассмотрению конкретных алгоритмов криптографических преобразований, необходимо пояснить обозначения, используемые на схемах в следующих разделах:

T_o, $T_ш$ –массивы соответственно открытых и зашифрованных данных;

T_i^o, $T_i^ш$ – i-тые по порядку 64-битные блоки соответственно открытых и зашифрованных данных: $T_o = \left(T_1^o, T_2^o, \ldots, T_n^o\right)$, $T_ш = \left(T_1^ш, T_2^ш, \ldots, T_n^ш\right)$, $1 \le i \le n$, последний блок может быть неполным: $\left|T_i^o\right| = \left|T_i^ш\right| = 64$, $1 \le i \le n$, $1 \le \left|T_n^o\right| = \left|T_n^ш\right| < 64$;

n –число 64-битных блоков в массиве данных;

$Ц_X$ – функция преобразования 64-битного блока данных по алгоритму базового цикла «X».

Теперь опишем основные режимы шифрования.

Простая замена

Зашифровывание в данном режиме заключается в применении цикла 32-З к блокам открытых данных, расшифровывание – цикла 32-Р к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга. Схемы алгоритмов зашифровывания и расшифровывания в режиме простой замены приведены на рисунках 3а и 6 соответственно, они тривиальны и не нуждаются в комментариях.

Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифровыванию или расшифровыванию, должен быть кратен 64 битам: $|T_o| = |T_ш| = 64 \cdot n$, после выполнения операции размер полученного массива данных не изменяется.

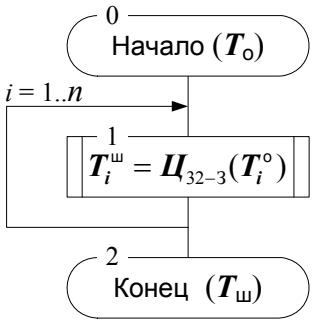


Рис. 3а. Алгоритм зашифровывания данных в режиме простой замены

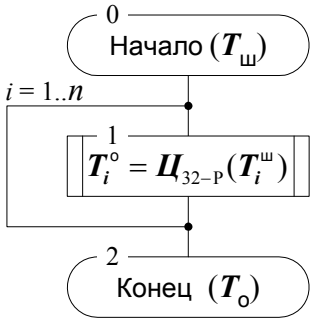


Рис. 3б. Алгоритм расшифровывания данных в режиме простой замены

Режим шифрования простой заменой имеет следующие особенности:

1. Так как блоки данных шифруются независимо друг от друга и от их позиции в массиве, при зашифровывании двух одинаковых блоков открытого текста получаются одинаковые блоки шифротекста и наоборот. Отмеченное свойство позволит криптоаналитику сделать заключение о тождественности блоков исходных данных, если в массиве зашифрованных данных ему встретились идентичные блоки, что является недопустимым для серьезного шифра.
2. Если длина шифруемого массива данных не кратна 8 байтам или 64 битам, возникает проблема, чем и как дополнять последний неполный блок данных массива до полных 64 бит. Эта задача не так проста, как кажется на первый взгляд, поскольку очевидные решения типа «дополнить неполный блок нулевыми битами» или, более обще, «дополнить неполный блок фиксированной комбинацией нулевых и единичных битов» могут при определенных условиях дать в руки криптоаналитика возможность методами перебора определить содержимое этого самого неполного блока, и этот факт означает снижение стойкости шифра. Кроме того, длина шифротекста при этом изменится, увеличившись до ближайшего целого, кратного 64 битам, что часто бывает нежелательным.

На первый взгляд, перечисленные выше особенности делают практически невозможным использование режима простой замены, ведь он может применяться только для шифрования массивов данных с размером кратным 64 битам, не содержащим повторяющихся 64-битных блоков. Кажется, что для любых реальных данных гарантировать выполнение указанных условий невозможно. Это почти так, но

есть одно очень важное исключение: вспомните, что размер ключа составляет 32 байта, а размер таблицы замен – 64 байта. Кроме того, наличие повторяющихся 8-байтовых блоков в ключе или таблице замен будет говорить об их весьма плохом качестве, поэтому в реальных ключевых элементах такого повторения быть не может. Таким образом мы выяснили, что режим простой замены вполне подходит для шифрования ключевой информации, тем более, что прочие режимы для этой цели менее удобны, поскольку требуют наличия дополнительного синхронизирующего элемента данных – синхропосылки (см. следующий раздел). Наша догадка верна, ГОСТ предписывает использовать режим простой замены исключительно для шифрования ключевых данных.

Гаммирование

Как же можно избавиться от недостатков режима простой замены? Для этого необходимо сделать возможным шифрование блоков с размером менее 64 бит и обеспечить зависимость блока шифротекста от его номера, иными словами, ***рандомизировать*** процесс шифрования. В ГОСТе это достигается двумя различными способами в двух режимах шифрования, предусматривающих ***гаммирование***. ***Гаммирование*** – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифровыванных (открытых) данных. Для наложения гаммы при зашифровывании и ее снятия при расшифровывании должны использоваться взаимно обратные бинарные операции, например, сложение и вычитание по модулю 2^{64} для 64-битных блоков данных. В ГОСТе для этой цели используется операция побитного сложения по модулю 2, поскольку она является обратной самой себе и к тому же наиболее просто реализуется. Гаммирование решает обе упомянутые проблемы; во первых, все элементы гаммы различны для реальных шифруемых массивов и, следовательно, результат зашифровывания даже двух одинаковых блоков в одном массиве данных будет различным. Во вторых, хотя элементы гаммы и вырабатываются одинаковыми порциями в 64 бита, использоваться может и часть такого блока с размером, равным размеру шифруемого блока.

Теперь перейдем непосредственно к описанию режима гаммирования. Гамма для этого режима получается следующим образом: с помощью некоторого алгоритмического рекуррентного генератора последовательности чисел (РГПЧ) вырабатываются 64-битные блоки данных, которые далее подвергаются преобразованию по циклу 32-3, то есть зашифровыванию в режиме простой замены, в результате получаются блоки гаммы. Благодаря тому, что наложение и снятие гаммы осуществляется при помощи одной и той же операции побитового исключающего или, алгоритмы зашифровывания и расшифровывания в режиме гаммирования идентичны, их общая схема приведена на рисунке 5.

РГПЧ, используемый для выработки гаммы, является рекуррентной функцией: $\Omega_{i+1} = f(\Omega_i)$, где Ω_i – элементы рекуррентной последовательности, f – функция преобразования. Следовательно, неизбежно возникает вопрос о его инициализации, то есть об элементе Ω_0. В действительности, этот элемент данных является параметром алгоритма для режимов гаммирования, на схемах он обозначен как $\mathbf{S}$, и называется в криптографии ***синхропосылкой***, а в ГОСТе – ***начальным заполнением*** одного из регистров устройства шифрования. По определенным соображениям разработчики ГОСТа решили использовать для инициализации РГПЧ не непосредственно синхропосылку, а результат ее преобразования по циклу 32-3: $\Omega_0 = Ц_{32-3}(\mathbf{S})$. Последовательность элементов, вырабатываемых РГПЧ, целиком зависит от его начального заполнения, то есть элементы этой последовательности являются функцией своего номера и начального заполнения РГПЧ: $\Omega_i = f_i(\Omega_0)$, где $f_i(X) = f(f_{i-1}(X))$, $f_0(X) = X$. С учетом преобразования по алгоритму простой замены добавляется еще и зависимость от ключа:

$\mathbf{\Gamma}_i = Ц_{32-3}(\Omega_i) = Ц_{32-3}(f_i(\Omega_0)) = Ц_{32-3}(f_i(Ц_{32-3}(\mathbf{S}))) = \varphi_i(\mathbf{S}, \mathbf{K})$, где $\mathbf{\Gamma}_i$ – i-ый элемент гаммы, $\mathbf{K}$ – ключ.

Таким образом, последовательность элементов гаммы для использования в режиме гаммирования однозначно определяется ключевыми данными и синхропосылкой. Естественно, для обратимости процедуры шифрования в процессах за- и

расшифровывания должна использоваться одна и та же синхропосылка. Из требования уникальности гаммы, невыполнение которого приводит к катастрофическому снижению стойкости шифра, следует, что для шифрования двух различных массивов данных на одном ключе необходимо обеспечить использование различных синхропосылок. Это приводит к необходимости хранить или передавать синхропосылку по каналам связи вместе с зашифрованными данными, хотя в отдельных особых случаях она может быть предопределена или вычисляться особым образом, если исключается шифрование двух массивов на одном ключе.

Теперь подробно рассмотрим РГПЧ, используемый в ГОСТе для генерации элементов гаммы. Прежде всего надо отметить, что к нему не предъявляются требования обеспечения каких-либо статистических характеристик вырабатываемой последовательности чисел. РГПЧ спроектирован разработчиками ГОСТа исходя из необходимости выполнения следующих условий:

- период повторения последовательности чисел, вырабатываемой РГПЧ, не должен сильно (в процентном отношении) отличаться от максимально возможного при заданном размере блока значения 2^{64};
- соседние значения, вырабатываемые РГПЧ, должны отличаться друг от друга в каждом байте, иначе задача криптоаналитика будет упрощена;
- РГПЧ должен быть достаточно просто реализуем как аппаратно, так и программно на наиболее распространенных типах процессоров, большинство из которых, как известно, имеют разрядность 32 бита.
- Исходя из перечисленных принципов создатели ГОСТа спроектировали весьма удачный РГПЧ, имеющий следующие характеристики:
- в 64-битовом блоке старшая и младшая части обрабатываются независимо друг от друга: $\Omega_i = (\Omega_i^0, \Omega_i^1), |\Omega_i^0| = |\Omega_i^1| = 32, \Omega_{i+1}^0 = \hat{f}(\Omega_i^0), \Omega_{i+1}^1 = \widetilde{f}(\Omega_i^1)$, фактически, существуют два независимых РГПЧ для старшей и младшей частей блока.
- рекуррентные соотношения для старшей и младшей частей следующие:

$\Omega_{i+1}^0 = (\Omega_i^0 + C_1) \bmod 2^{32}$, где $\mathbf{C}_1 = 1010101_{16}$;

$\Omega_{i+1}^1 = (\Omega_i^1 + C_2 - 1) \bmod (2^{32} - 1) + 1$, где $\mathbf{C}_2 = 1010104_{16}$;

Нижний индекс в записи числа означает его систему счисления, таким образом, константы, используемые на данном шаге, записаны в 16-ричной системе счисления.

Второе выражение нуждается в комментариях, так как в тексте ГОСТа приведено нечто другое: $\Omega_{i+1}^1 = (\Omega_i^1 + C_2) \bmod (2^{32} - 1)$, с тем же значением константы $\mathbf{C}_1$. Но далее в тексте стандарта дается комментарий, что, оказывается, под операцией взятия остатка по модулю $(2^{32} - 1)$ там понимается не то же самое, что и в математике. Отличие заключается в том, что согласно ГОСТу $(2^{32} - 1) \bmod (2^{32} - 1) = (2^{32} - 1)$, а не 0. На самом деле, это упрощает реализацию формулы, а математически корректное выражение для нее приведено выше.

- период повторения последовательности для младшей части составляет 2^{32}, для старшей части $(2^{32} - 1)$, для всей последовательности период составляет $2^{32} \cdot (2^{32} - 1)$, доказательство этого факта, весьма несложное, получите сами. Первая формула из двух реализуется за одну команду, вторая, несмотря на ее кажущуюся громоздкость, за две команды на всех современных 32-разрядных процессорах.

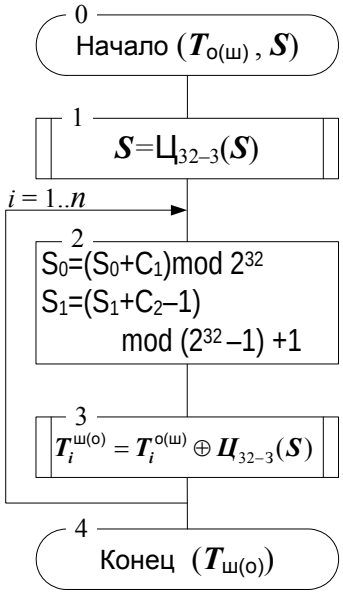


Рис. 4. Алгоритм зашифровывания (расшифровывания) данных в режиме гаммирования

Схема алгоритма шифрования в режиме гаммирования приведена на рисунке 4, ниже изложены пояснения к схеме:

Шаг 0.Определяет исходные данные для основного шага криптопреобразования:

- $T_{о(ш)}$ – массив открытых (зашифрованных) данных произвольного размера, подвергаемый процедуре зашифровывания (расшифровывания), по ходу процедуры массив подвергается преобразованию порциями по 64 бита;
- $\mathbf{S}$ – ***синхропосылка***, 64-битный элемент данных, необходимый для инициализации генератора гаммы;

Шаг 1. Начальное преобразование синхропосылки, выполняемое для ее «рандомизации», то есть для устранения статистических закономерностей, присутствующих в ней, результат используется как начальное заполнение РГПЧ;

Шаг 2. Один шаг работы РГПЧ, реализующий его рекуррентный алгоритм. В ходе данного шага старшая $(\mathbf{S}_1)$ и младшая $(\mathbf{S}_0)$ части последовательности данных вырабатываются независимо друг от друга;

Шаг 3. Гаммирование. Очередной 64-битный элемент, выработанный РГПЧ, подвергается процедуре зашифровывания по циклу 32–3, результат используется как элемент гаммы для зашифровывания (расшифровывания) очередного блока открытых (зашифрованных) данных того же размера.

Шаг 4.Результат работы алгоритма – зашифрованный (расшифрованный) массив данных.

Ниже перечислены особенности гаммирования как режима шифрования.

1. Одинаковые блоки в открытом массиве данных дадут при зашифровывании различные блоки шифротекста, что позволит скрыть факт их идентичности.
2. Поскольку наложение гаммы выполняется побитно, шифрование неполного блока данных легко выполнимо как шифрование битов этого неполного блока, для чего используется соответствующие биты блока гаммы. Так, для зашифровывания неполного блока в 1 бит можно использовать любой бит из блока гаммы.
3. Синхропосылка, использованная при зашифровывании, каким-то образом должна быть передана для использования при расшифровывании. Это может быть достигнуто следующими путями:

- хранить или передавать синхропосылку вместе с зашифрованным массивом данных, что приведет к увеличению размера массива данных при зашифровывании на размер синхропосылки, то есть на 8 байт;

- использовать предопределенное значение синхропосылки или вырабатывать ее синхронно источником и приемником по определенному закону, в этом случае изменение размера передаваемого или хранимого массива данных отсутствует;

Оба способа дополняют друг друга, и в тех редких случаях, где не работает первый, наиболее употребительный из них, может быть использован второй, более экзотический. Второй способ имеет гораздо меньшее применение, поскольку сделать синхропосылку предопределенной можно только в том случае, если на данном комплекте ключевой информации шифруется заведомо не более одного массива данных, что бывает в редких случаях. Генерировать синхропосылку синхронно у источника и получателя массива данных также не всегда представляется возможным, поскольку требует жесткой привязки к чему-либо в системе. Так, здравая на первый взгляд идея использовать в качестве синхропосылки в системе передачи зашифрованных сообщений номер передаваемого сообщения не подходит, поскольку сообщение может потеряться и не дойти до адресата, в этом случае произойдет десинхронизация систем шифрования источника и приемника. Поэтому в рассмотренном случае нет альтернативы передаче синхропосылки вместе с зашифрованным сообщением.

С другой стороны, можно привести и обратный пример. Допустим, шифрование данных используется для защиты информации на диске, и реализовано оно на низком уровне, для обеспечения независимого доступа данные шифруются по секторам. В этом случае невозможно хранить синхропосылку вместе с зашифрованными данными, поскольку размер сектора нельзя изменить, однако ее можно вычислять как некоторую функцию от номера считывающей головки диска, номера дорожки (цилиндра) и номера сектора на дорожке. В этом случае синхропосылка привязывается к положению сектора на диске, которое вряд ли может измениться без переформатирования диска, то есть без уничтожения данных на нем.

Режим гаммирования имеет еще одну интересную особенность. В этом режиме биты массива данных шифруются независимо друг от друга. Таким образом, каждый бит шифротекста зависит от соответствующего бита открытого текста и, естественно, порядкового номера бита в массиве: $t_i^{\text{о}} = t_i^{\text{ш}} \oplus \gamma_i = f\left(t_i^{\text{о}}, i\right)$. Из этого вытекает, что изменение бита шифротекста на противоположное значение приведет к аналогичному изменению бита открытого текста на противоположный:

$$\bar{t}_i^{\text{ш}} = t_i^{\text{ш}} \oplus 1 = \left(t_i^{\text{о}} \oplus \gamma_i\right) \oplus 1 = \left(t_i^{\text{о}} \oplus 1\right) \oplus \gamma_i = \bar{t}_i^{\text{о}} \oplus \gamma_i ,$$

где $\bar{t}$ обозначает инвертированное по отношению к t значение бита $\left(\overline{0} = 1, \overline{1} = 0\right)$.

Данное свойство дает злоумышленнику возможность воздействуя на биты шифротекста вносить предсказуемые и даже целенаправленные изменения в соответствующий открытый текст, получаемый после его расшифровывания, не обладая при этом секретным ключом. Это иллюстрирует хорошо известный в криптологии факт, что ***«секретность и аутентичность суть различные свойства шифров»***. Иными словами, свойства шифров обеспечивать защиту от несанкционированного ознакомления с содержимым сообщения и от несанкционированного внесения изменений в сообщение являются независимыми и лишь в отдельных случаях могут пересекаться. Сказанное означает, что существуют криптографические алгоритмы, обеспечивающие определенную секретность зашифрованных данных и при этом никак не защищающие от внесения изменений и наоборот, обеспечивающие аутентичность данных и никак не ограничивающие возможность ознакомления с ними. По этой причине рассматриваемое свойство режима гаммирования не должно рассматриваться как его недостаток.

Гаммирование с обратной связью

Данный режим очень похож на режим гаммирования и отличается от него только способом выработки элементов гаммы – очередной элемент гаммы вырабатывается как результат преобразования по циклу 32-3 предыдущего блока зашифровыванных данных, а для зашифровывания первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки. Этим достигается зацепление блоков – каждый блок шифротекста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста. Поэтому данный режим иногда называется ***гаммированием с зацеплением блоков***. На стойкость шифра факт зацепления блоков не оказывает никакого влияния.

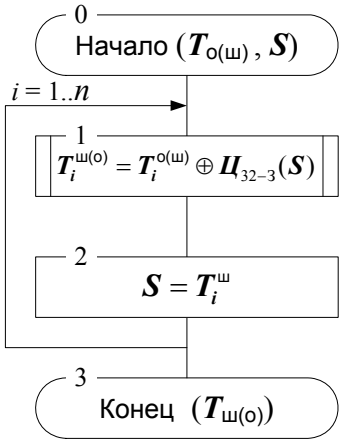


Рис. 5. Алгоритм зашифровывания (расшифровывания) данных в режиме гаммирования с обратной связью

Схема алгоритмов за- и расшифровывания в режиме гаммирования с обратной связью приведена на рисунке 5 и ввиду своей простоты в комментариях не нуждается.

Шифрование в режиме гаммирования с обратной связью обладает теми же особенностями, что и шифрование в режиме обычного гаммирования, за исключением влияния искажений шифротекста на соответствующий открытый текст. Для сравнения запишем функции расшифровывания блока для обоих упомянутых режимов:

- $T_i^{\circ} = T_i^{ш} \oplus \Gamma_i$, гаммирование;
- $T_i^{\circ} = T_i^{ш} \oplus Ц_{32-3}\left(T_{i-1}^{ш}\right)$, гаммирование с обратной связью;

Если в режиме обычного гаммирования изменения в определенных битах шифротекста влияют только на соответствующие биты открытого текста, то в режиме гаммирования с обратной связью картина несколько сложнее. Как видно из соответствующего уравнения, при расшифровывании блока данных в режиме гаммирования с обратной связью, блок открытых данных зависит от соответствующего и предыдущего блоков зашифрованных данных. Поэтому, если внести искажения в зашифровыванный блок, то после расшифровывания искаженными окажутся два блока открытых данных – соответствующий и следующий за ним, причем искажения в первом случае будут носить тот же характер, что и в режиме гаммирования, а во втором случае – как в режиме простой замены. Другими словами, в соответствующем блоке открытых данных искаженными окажутся те же самые биты, что и в блоке шифрованных данных, а в следующем блоке открытых данных все биты независимо друг от друга с вероятностью 1/2 изменят свои значения.

Выработка имитовставки к массиву данных

В предыдущих разделах мы обсудили влияние искажения шифрованных данных на соответствующие открытые данные. Мы установили, что при расшифровывании в режиме простой замены соответствующий блок открытых данных оказывается искаженным непредсказуемым образом, а при расшифровывании блока в режиме гаммирования изменения предсказуемы. В режиме гаммирования с обратной связью искаженными оказываются два блока, один предсказуемым, а другой непредсказуемым образом. Значит ли это, что с точки зрения защиты от навязывания ложных данных режим гаммирования является плохим, а режимы простой замены и гаммирования с обратной связью хорошими? Ни в коем случае. При анализе данной ситуации необходимо учесть то, что непредсказуемые изменения в расшифровыванном блоке данных могут быть обнаружены только в случае избыточности этих самых данных, причем чем больше степень избыточности, тем вероятнее обнаружение искажения. Очень большая избыточность имеет место, например, для текстов на естественных и искусственных языках, в этом случае факт искажения обнаруживается практически неизбежно. Однако в других случаях, например, при искажении сжатых звуковых образов, мы получим просто другой образ, который сможет воспринять наше ухо. Искажение в этом случае останется необнаруженным, если, конечно, нет априорной информации о характере звука. Вывод здесь такой: поскольку способность некоторых режимов шифрования обнаруживать искажения, внесенные в шифрованные данные, существенным образом опирается на наличие и степень избыточности шифруемых

данных, эта способность не является имманентным свойством соответствующих режимов и не может рассматриваться как их достоинство.

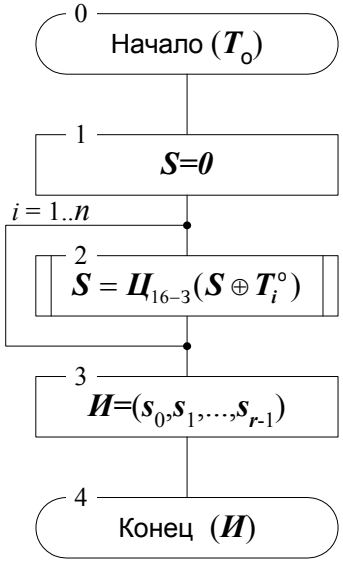


Рис. 6. Алгоритм выработки имитовставки для массива данных

Для решения задачи обнаружения искажений в зашифровыванном массиве данных с заданной вероятностью в ГОСТе предусмотрен дополнительный режим криптографического преобразования – выработка имитовставки. Имитовставка – это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации. Целью использования имитовставки является обнаружение всех случайных или преднамеренных изменений в массиве информации. Проблема, изложенная в предыдущем пункте, может быть успешно решена с помощью добавления к шифрованным данным имитовставки. Для потенциального злоумышленника две следующие задачи практически неразрешимы, если он не владеет ключевой информацией:

- вычисление имитовставки для заданного открытого массива информации;
- подбор открытых данных под заданную имитовставку;

Схема алгоритма выработки имитовставки приведена на рисунке 6. В качестве имитовставки берется часть блока, полученного на выходе, обычно 32 его младших бита. При выборе размера имитовставки надо принимать во внимание, что вероятность успешного навязывания ложных данных равна величине $2^{-|И|}$ на одну попытку подбора. При использовании имитовставки размером 32 бита эта вероятность равна $2^{-32} \approx 0.23 \cdot 10^{-9}$.

Обсуждение криптографических алгоритмов ГОСТа

Криптографическая стойкость ГОСТа

При выборе криптографического алгоритма для использования в конкретной разработке одним из определяющих факторов является его стойкость, то есть устойчивость к попыткам противоположной стороны его раскрыть. Вопрос о стойкости шифра при ближайшем рассмотрении сводится к двум взаимосвязанным вопросам:

- можно ли вообще раскрыть данный шифр;
- если да, то насколько это трудно сделать практически;

Шифры, которые вообще невозможно раскрыть, называются абсолютно или теоретически стойкими. Существование подобных шифров доказывается теоремой Шеннона, однако ценой этой стойкости является необходимость использования для шифрования каждого сообщения ключа, не меньшего по размеру самого сообщения. Во всех случаях за исключением ряда особых эта цена чрезмерна, поэтому на практике в основном используются шифры, не обладающие абсолютной стойкостью. Таким образом, наиболее употребительные схемы шифрования могут быть раскрыты за конечное время или, что точнее, за конечное число шагов, каждый из которых является некоторой операцией над числами. Для них наиважнейшее значение имеет понятие

практической стойкости, выражающее практическую трудность их раскрытия. Количественной мерой этой трудности может служить число элементарных арифметических и логических операций, которые необходимо выполнить, чтобы раскрыть шифр, то есть чтобы для заданного шифротекста с вероятностью, не меньшей заданной величины, определить соответствующий открытый текст. При этом в дополнении к дешифруемому массиву данных криптоаналитик может располагать блоками открытых данных и соответствующих им зашифрованных данных или даже возможностью получить для любых выбранных им открытых данных соответствующие зашифрованные данные – в зависимости от перечисленных и многих других неуказанных условий различают отдельные виды криптоанализа.

Все современные криптосистемы построены по принципу Кирхгоффа, то есть секретность зашифрованных сообщений определяется секретностью ключа. Это значит, что даже если сам алгоритм шифрования известен криптоаналитику, тот тем не менее не в состоянии расшифровать сообщение, если не располагает соответствующим ключом. Все классические блочные шифры, в том числе DES и ГОСТ, соответствуют этому принципу и спроектированы таким образом, чтобы не было пути вскрыть их более эффективным способом, чем полным перебором по всему ключевому пространству, т.е. по всем возможным значениям ключа. Ясно, что стойкость таких шифров определяется размером используемого в них ключа.

В шифре ГОСТ используется 256-битовый ключ и объем ключевого пространства составляет 2^{256}. Ни на одной из существующих в настоящее время или предполагаемых к реализации в недалеком будущем ЭВМ общего применения нельзя подобрать ключ за время, меньшее многих сотен лет. Советский стандарт проектировался с большим запасом и по стойкости на много порядков превосходит американский стандарт DES с его реальным размером ключа в 56 бит и объемом ключевого пространства всего 2^{56}. В свете прогресса современных вычислительных средств этого явно недостаточно. В этой связи DES может представлять скорее исследовательский или научный, чем практический интерес. Как ожидается, в 1998 году он перестанет быть стандартом США на шифрование.

Замечания по архитектуре ГОСТа

Общеизвестно, что шифр ГОСТ 28147-89 является представителем целого семейства шифров, построенных на одних и тех же принципах. Самым известным его «родственником» является американский стандарт шифрования, алгоритм DES. Все эти шифры, подобно ГОСТу, содержат алгоритмы трех уровней. В основе всегда лежит некий «основной шаг», на базе которого сходным образом строятся «базовые циклы», и уже на их основе построены практические процедуры шифрования и выработки имитовставки. Таким образом, специфика каждого из шифров этого семейства заключена именно в его основном шаге, точнее даже в его части. Хотя архитектура классических блочных шифров, к которым относится ГОСТ, лежит далеко за пределами темы настоящей лекции, все же стоит сказать несколько слов по ее поводу.

Алгоритмы «основных шагов криптопреобразования» для шифров, подобных ГОСТу, построены идентичным образом. Их общая схема приведена на рисунке 7. На вход основного шага подается блок четного размера, старшая и младшая половины которого обрабатываются отдельно друг от друга. В ходе преобразования младшая половина блока помещается на место старшей, а старшая, скомбинированная с помощью операции побитного ***исключающего или*** с результатом вычисления некоторой функции, на место младшей. Эта функция, принимающая в качестве аргумента младшую половину блока и некоторый элемент ключевой информации (X), является содержательной частью шифра и называется его ***функцией шифрования***. Соображения стойкости шифра требуют, чтобы размеры всех перечисленных элементов блоков были равны: $|N_1| = |N_2| = |X|$, в ГОСТе и DESe они равны 32 битам.

Если применить сказанное к схеме основного шага алгоритма ГОСТ, станет очевидным, что блоки 1,2,3 алгоритма определяют вычисление его функции шифрования, а блоки 4 и 5 задают формирование выходного блока основного шага исходя из содержимого входного блока и значения функции шифрования.

Рис. 7. Содержание основного шага криптопреобразования для блочных шифров, подобных ГОСТу

В предыдущем разделе мы уже сравнили DES и ГОСТ по стойкости, теперь мы сравним их по функциональному содержанию и удобству реализации. В циклах шифрования ГОСТа основной шаг повторяется 32 раза, для DESa эта величина равна 16. Однако сама функция шифрования ГОСТа существенно проще аналогичной функции DESa, в которой присутствует множество перекодировок по таблицам с изменением размера перекодируемых элементов. Кроме того, между основными шагами в циклах шифрования DESa необходимо выполнять битовые перестановки в блоках данных. Все эти операции чрезвычайно неэффективно реализуются на современных неспециализированных процессорах. ГОСТ не содержит подобных операций, поэтому он значительно удобней для программной реализации. Ни одна из рассмотренных автором реализаций DESa для платформы Intel x86 не достигает даже половины производительности предложенной вашему вниманию в настоящей лекции реализации ГОСТа, несмотря на вдвое более короткий цикл. Все сказанное выше свидетельствует о том, что разработчики ГОСТа учли как положительные, так и отрицательные стороны DESa, а также более реально оценили текущие и перспективные возможности криптоанализа.

Требования к качеству ключевой информации и источники ключей

Не все ключи и таблицы замен обеспечивают максимальную стойкость шифра. Для каждого алгоритма шифрования существуют свои критерии оценки ключевой информации. Так, для алгоритма DES известно существование так называемых «***слабых ключей***», при использовании которых связь между открытыми и зашифрованными данными не маскируется достаточным образом, и шифр сравнительно просто вскрывается.

Исчерпывающий ответ на вопрос о критериях качества ключей и таблиц замен ГОСТа если и можно вообще где-либо получить, то только у разработчиков алгоритма. Соответствующие данные не были опубликованы в открытой печати. Однако согласно установленному порядку, для шифрования информации, имеющей гриф, должны быть использованы ключевые данные, полученные от уполномоченной организации. Косвенным образом это может свидетельствовать о наличии методик проверки ключевых данных на «вшивость». Сам факт существования слабых ключевых данных в Российском стандарте шифрования не вызывает сомнения. Очевидно, нулевой ключ и тривиальная таблица замен, по которой любое значение заменяется, но него самого, являются слабыми, при использовании хотя бы одного из них шифр достаточно просто взламывается, каков бы ни был второй ключевой элемент.

Как уже было отмечено выше, критерии оценки ключевой информации недоступны, однако на их счет все же можно высказать некоторые соображения:

- ***Ключ* должен являться массивом статистически независимых битов, принимающих с равной вероятностью значения 0 и 1.** При этом некоторые конкретные значения ключа могут оказаться «слабыми», то есть шифр может не обеспечивать заданный уровень стойкости в случае их использования. Однако,

предположительно, доля таких значений в общей массе всех возможных ключей ничтожно мала. Поэтому ключи, выработанные с помощью некоторого датчика истинно случайных чисел, будут качественными с вероятностью, отличающейся от единицы на ничтожно малую величину. Если же ключи вырабатываются с помощью генератора псевдослучайных чисел, то используемый генератор должен обеспечивать указанные выше статистические характеристики, и, кроме того, обладать высокой криптостойкостью, не меньшей, чем у самого ГОСТа. Иными словами, задача определения отсутствующих членов вырабатываемой генератором последовательности элементов не должна быть проще, чем задача вскрытия шифра. Кроме того, для отбраковки ключей с плохими статистическими характеристиками могут быть использованы различные статистические критерии. На практике обычно хватает двух критериев, – для проверки равновероятного распределения битов ключа между значениями 0 и 1 обычно используется критерий Пирсона (χ^2), а для проверки независимости битов ключа – критерий серий.

- ***Таблица замен*** является долговременным ключевым элементом, то есть действует в течение гораздо более длительного срока, чем отдельный ключ. Предполагается, что она является общей для всех узлов шифрования в рамках одной системы криптографической защиты. Даже при нарушении конфиденциальности таблицы замен стойкость шифра остается чрезвычайно высокой и не снижается ниже допустимого предела. К качеству отдельных узлов замен можно предъявить приведенное ниже требование. **Каждый узел замен может быть описан четверкой логических функций, каждая из которых имеет четыре логических аргумента. Необходимо, чтобы эти функции были достаточно сложными. Это требование сложности невозможно выразить формально, однако в качестве необходимого условия можно потребовать, чтобы соответствующие логические функции, записанные в минимальной форме (т.е. с минимально возможной длиной выражения) с использованием основных логических операций, не были короче некоторого необходимого минимума.** В первом и очень грубом приближении это условие может сойти за достаточное. Кроме того, отдельные функции в пределах всей таблицы замен должны отличаться друг от друга в достаточной степени. На практике бывает достаточно получить узлы замен как независимые случайные перестановки чисел от 0 до 15, это может быть практически реализовано, например, с помощью перемешивания колоды из шестнадцати карт, за каждой из которых закреплено одно из значений указанного диапазона.

Необходимо отметить еще один интересный факт относительно таблицы замен. Для обратимости циклов шифрования 32–З и 32–Р не требуется, чтобы узлы замен были перестановками чисел от 0 до 15. Все работает даже в том случае, если в узле замен есть повторяющиеся элементы, и замена, определяемая таким узлом, необратима, однако в этом случае снижается стойкость шифра. Почему это именно так, не рассматривается в настоящей лекции, однако в самом факте убедиться несложно. Для этого достаточно, используя демонстрационную программу шифрования файлов данных, прилагающуюся к настоящей лекции, зашифровать, а затем расшифровать файл данных, использовав для этой процедуры «неполноценную» таблицу замен, узлы которой содержат повторяющиеся значения.

Если вы разрабатываете программы, использующие криптографические алгоритмы, вам необходимо позаботиться об утилитах, вырабатывающих ключевую информацию, а для таких утилит необходим источник случайных чисел (СЧ) высокого статистического качества и криптостойкости. Наилучшим подходом здесь было бы использование аппаратных датчиков СЧ, однако это не всегда приемлемо по экономическим соображениям. В качестве разумной альтернативы возможно (и очень широко распространено) использование различных программных датчиков СЧ. При генерации небольшого по объему массива ключевой информации широко применяется метод «электронной рулетки», когда очередная получаемая с такого датчика порция случайных битов зависит от момента времени нажатия оператором некоторой клавиши на клавиатуре компьютера.

Три шага оптимизации

Целью описываемой в настоящей лекции реализации ГОСТа было не создание максимального по эффективности кода любой ценой, целью было создание близкого к оптимальному по быстродействию, но при этом компактного и легкого для восприятия программистом кода.

Программирование алгоритмов ГОСТа «в лоб» даже на языке ассемблера дает результат, очень далекий от возможного оптимума. Задача создания эффективной реализации указанных алгоритмов для 16-разрядных процессоров Intel 8088–80286 представляет собой хотя и не сверхсложную, но все же и не вполне тривиальную задачу, и требует очень хорошего знания архитектуры и системы команд упомянутого семейства процессоров. Трудность здесь заключается в том, что для достижения максимальной производительности программная реализация должна как можно большую часть операций с данными выполнять в регистрах и как можно реже обращаться к памяти, а регистров у рассматриваемых процессоров не так много и все они 16-разрядные. С 32-разрядными процессорами этой же линии (Intel 80386 и старше) все намного проще именно в силу их 32-разрядности, здесь не будет трудностей даже у новичка.

В реализации алгоритмов были использованы изложенные ниже подходы, позволившие достигнуть максимальной производительности. Первые два из них достаточно очевидны, настолько, что встречаются практически в каждой реализации ГОСТа.

1. Базовые циклы ГОСТа содержат вложенные циклы (звучит коряво, но по-другому не скажешь), причем во внутреннем цикле порядок использования восьми 32-битных элементов ключа может быть прямой или обратный. Существенно упростить реализацию и повысить эффективность базовых циклов можно, если избежать использования вложенных циклов и просматривать последовательность элементов ключа только один раз. Для этого необходимо предварительно сформировать последовательность элементов ключа в том порядке, в котором они используются в соответствующем базовом цикле.
2. В основном шаге криптопреобразования 8 раз выполняется подстановка 4-битных групп данных. Целевой процессор реализации не имеет команды замены 4-битных групп, однако имеет удобную команду байтовой замены (**xlat**). Ее использование дает следующие выгоды:
    - за одну команду выполняются сразу две замены;
    - исчезает необходимость выделять полубайты из двойных слов для выполнения замены, а затем из 4-битовых результатов замен вновь формировать двойное слово.

Этим достигается значительное увеличение быстродействия кода, однако мир устроен так, что за все приходится платить, и в данном случае платой является необходимость преобразования таблицы замен. Каждая из четырех пар 4-разрядных узлов замен заменяется одним 8-разрядным узлом, который, говоря языком математики, представляет собой прямое произведение узлов, входящих в пару. Пара 4-разрядных узлов требует для своего представления 16 байтов, один 8-разрядный – 256 байтов. Таким образом, размер таблицы замен, которая должна храниться в памяти компьютера, увеличивается до $4 \cdot 256 = 1024$ байтов, или до одного килобайта. Конечно, такая плата за существенное увеличение эффективности реализации вполне приемлема.

3. После выполнения подстановок кода по таблице замен основной шаг криптопреобразования предполагает циклический сдвиг двойного слова влево на 11 бит. В силу 16-разрядной архитектуры рассматриваемых процессоров вращение 32-разрядного блока даже на 1 бит невозможно реализовать менее, чем за три ассемблерные команды, а вращение на большее число разрядов только как последовательность отдельных вращений на 1 разряд. К счастью, вращение на 11 бит влево можно представить как вращение на 8 бит, а затем еще на 3 бита влево. Думаю, для всех очевидно, что первое вращение реализуется тремя командами обмена байтовых регистров (**xchg**). Но секрет третьей оптимизации даже не в этом. Замена одного байта по таблице замен осуществляется командой **xlat**, которая выполняет операцию над аргументом в регистре **AL**, для того, чтобы заменить все байты двойного слова, их надо последовательно помещать в этот регистр. Секрет третьей оптимизации заключается в том, что эти перестановки можно организовать

так, что в результате двойное слово окажется повернутым на 8 бит влево, то есть в совмещении замены по таблице и во вращении на байт влево. Еще один момент, на который стоит обратить внимание, это оптимальное кодирование трех последовательных вращений на 1 бит, это может быть реализовано по-разному и важно было выбрать оптимальный способ, который оказался вовсе не очевидным, поскольку потребовал выхода за пределы логики битовых сдвигов и использования команды суммирования с битами переноса (**adc**), то есть бит помещается на свою позицию не командой сдвига, а командой суммирования!

Литература

1. Криптографическая защита информации в АСУ СН. Курс лекций. В.И. Долгов. ХВУ. 1998.

2. Криптографическая защита информации в информационных системах. Курс лекций. И.Д. Горбенко. ХНУРЭ. 2002.

3. Теория информации. Курс лекций. В.И. Долгов. ХВУ. 1998.

4. Брюс Шнайер. Прикладная криптография. 2-ое издание. Протоколы, алгоритмы и исходные тексты на языке С. Доступно: http://nrjetix.com/r-and-d/lectures

5. Андрей Винокуров. Алгоритм шифрования ГОСТ 28147-89, его использование и реализация для компьютеров платформы Intel x86.

6. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.