В.А. Козлов, А.Б. Чернышев

Самонастраивающийся обучающий программно-алгоритмический комплекс симметричных криптографических преобразований

Современные криптосистемы классифицируют следующим образом: асимметричные системы с открытым ключом и симметричные с секретным ключом. Для алгоритмов асимметричного шифрования характерно применение нелинейных аналитических преобразований шифруемого текста и двух ключей – общедоступного и личного (секретного). Процесс шифрации осуществляется первым ключом (общедоступным), а дешифрации – личным и по другому алгоритму. Вот почему эти алгоритмы получили название асимметричных.

Симметричные криптосистемы построены на использовании одного и того же ключа двумя партнерами как для процедур шифрации, так и для процедур расшифрования. Поэтому их и называют симметричными системами криптографических преобразований.

Быстродействие симметричных криптосистем гораздо выше асимметричных. Основной проблемой применения симметричных криптосистем являются определенные неудобства, связанные с некоторой сложностью передачи одной и той же секретной ключевой комбинации обоим сторонам конфиденциальной переписки. Эта проблема легко решить путем использования асимметричной криптосистемы, с помощью которой передавать достаточно короткую ключевую комбинацию друг другу. С этой задачей асимметричные системы, не очень быстрые по своей сути, успешно справляются.

Алгоритмы симметричных криптографических преобразований ориентированы, как правило, на работу с файлами больших и очень больших размеров, поскольку определяющим преимуществом этих алгоритмов является их быстродействие, так как асимметричные алгоритмы криптографических преобразований существенно уступают им в быстродействии [1].

Вот почему проблемы повышения быстродействия симметричных криптографических систем (СКС) являются актуальными и в наше время. СКС состоят, как правило, из целого ряда последовательно исполняемых модулей элементарных криптографических преобразований. Быстрота реализации каждого модуля элементарных криптографических преобразований определяется целым рядом факторов, связанных зачастую не только с алгоритмами, но и с особенностями программной их реализации [2].

Так, современные объектно-ориентированные языки программирования при работе с битовой информацией существенно теряют в скорости реализации по сравнению с ассемблерами, у которых имеется целый арсенал команд, реализующих битовые операции [3].

Вторым, не менее важным аспектом является применение исключительно реляционных баз данных, работающих с табличными массивами, столбцы и строки которых строго фиксированной длины, что позволяет в десятки раз повысить скорость их обработки.

В данном случае адрес любой ячейки таблицы определяется элементарной расчетной формулой:

$$\mathrm{A}_{\mathrm{n,m}} = B_t + (n-1)\cdot D_n + (m-1)\cdot D_m \ , \qquad (1)$$

где $A_{n,m}$ – базовый адрес таблицы; n – номер строки; m – номер столбца; D_n – длина строки; D_m – длина ячейки.

Как известно, самым эффективным способом повышения криптостойкости является разработка, так называемого, составного алгоритма, представляющего собой ряд последовательно выполняемых модулей, каждый из которых реализует всего одну элементарную операцию симметричных криптографических преобразований: рассеивания, перемешивания или гаммирования. При этом последовательность выполнения этих модулей в алгоритмах шифрации и дешифрации прямо противоположная. Так, например, если модуль M_1, в алгоритме шифрации стоит на первом месте, то в алгоритме дешифрации – на последнем. Криптостойкость такой системы в целом зависит от криптостойкости каждого модуля в отдельности и числа, последовательно исполняемых элементарных модулей, входящих в состав системы. Эту зависимость можно представить в виде следующего аналитического выражения [1]:

$$\mathrm{Q}_{\mathrm{s}} = \prod_{i=1}^{N} Q_i \ , \qquad (2)$$

где Q_{s} – криптостойкость системы в целом; Q_{i} – криптостойкость i-ой элементарной криптографической операции типа рассеивания, перемешивания или гаммирования.

Формула (2) действительна лишь в том случае, если включает в себя в обязательном порядке и алгоритмы подстановки, и алгоритмы рассеивания, и алгоритмы гаммирования.

Не менее важной особенностью компьютерной криптографии является то, что она работает исключительно с электронными документами. Это не только текстовые документы, но и программы, чертежи, рисунки, видео и звуковые файлы. Другими словами, алгоритм криптографических преобразований должен быть ориентирован на работу с любым

электронным документом – с файлом, имеющим любое расширение. Таким образом, любой «исходный текст», это всегда компьютерный файл, содержащий конфиденциальную информацию, который необходимо передать партнеру по открытым каналам связи, а процесс шифрации должен заканчиваться созданием шифрограммы в виде электронного документа (файла) пересылаемого по открытым каналам связи.

Так как у нас речь идет о защите именно компьютерной информации, то в качестве алфавита должна использоваться вся таблица ASCII-символов, т. е. множество всевозможных целых неотрицательных чисел от 0 до 255 с определенными на нем операциями сложения и вычитания по модулю 256. Такую конструкцию в математике еще называют кольцом вычетов по модулю 256 и обозначают как Z/256. Вот почему в качестве алфавита во всех процедурах шифрации и дешифрации электронных документов необходимо использовать алфавит из $2^8 = 256$ значений, т.е. в наш алфавит входят 256 однобайтовых значений (в десятичном исчислении это числа 0,1,2,…,255).

Алгоритмы СКС базируются на применении в различных сочетаниях всего трех принципов: *рассеивания, перемешивания и гаммирования. Рассеивание* представляет собой распространение влияния каждого символа открытого текста на каждый символ шифртекста, что позволяет скрыть статистические свойства открытого текста. *Перемешивание* предполагает использование таких шифрующих преобразований, которые усложняют восстановление взаимосвязи статистических свойств. *Гаммирование* – это механизм, позволяющий «растянуть» секретную ключевую комбинацию до размеров шифруемого сообщения с целью как можно большего приближения по степени криптостойкости к уровню криптографической стойкости одноразового шифровального блокнота, т.е. к уровню абсолютной криптостойкости.

Алгоритмы СКС состоят из целого ряда последовательно выполняемых модулей, каждый из которых реализует всего одну элементарную операцию симметричных криптографических преобразований: гаммирования, подстановки или перестановки. При этом последовательность выполнения этих модулей в алгоритмах шифрации и дешифрации прямо противоположная. Так, например, если модуль M_1, в алгоритме шифрации стоит на первом месте, то в алгоритме дешифрации он должен стоять последним.

Другими словами, криптографически стойкий шифр с хорошим рассеиванием и перемешиванием получается при многократном чередовании простых *перестановок* и *замен*, в сочетании с алгоритмами гаммирования, управляемых достаточно длинным секретным ключом.

Именно такой подход используется практически во всех современных алгоритмах симметричного шифрования. Рассмотрим один из возможных вариантов подбора модулей элементарных криптографических преобразований, каждый из которых обладает одновременно двумя очень важными характеристиками: криптостойкостью и быстродействием. Подбор таких алгоритмов необходимо производить исходя из того факта, что наша система ориентирована на работу с большими и очень большими массивами исходной информации, оформленными в виде электронных документов – файлов с любым возможным расширением: вордовские; загрузочные; изображения; звуковые и т.д.

Вашему вниманию предлагается симметричная криптосистема, в основу которой положены нелинейные комбинированные р-алгоритмы симметричных криптографических преобразований, отличающаяся существенным быстродействием и высокой криптостойкостью. Система криптографических преобразований состоит из трех этапов.

На первом, *предварительном этапе* формируются специальные рабочие массивы R_1 и R_2 и модуль элементарной предварительной подстановки, далее информация исходника (файл с исходной конфиденциальной информацией) разбивается на блоки одинаковой длины (в нашем случае длина блока составляет 256 байт) и перемешивается. На втором этапе реализованы алгоритмы внутри блочного рассеивания, перемешивания и гаммирования.

Первый этап. Объявляем одномерный массив (M_1) однобайтных элементов из 256 строк со значениями от 0 до 255. Перемешав его элементы, получаем рабочие массивы – R_1 и R_2, с помощью которых реализованы нелинейные р-алгоритмы табличного перемешивания и табличной подстановки.

Модуль создания рабочих массивов R_1 и R_2. Алгоритм формирования рабочего массива R_1 следующий. Значения строк N_i и N_j меняются местами. Номера строк N_i и N_j определяются по следующим формулам.

В первом цикле:

$$N_1 = (ЧЧ + MM + SS + K_1) \bmod 256; \quad N_2 = (N_1 + K_2) \bmod 256. \quad (3)$$

Во втором цикле:

$$N_3 = (N_2 + K_3) \bmod 256; \; N_4 = (N_3 + K_4) \bmod 256 \; \ldots\ldots\ldots\ldots \; (4)$$

$$N_{12} = (N_{12} + K_{12}) \bmod 256; \; N_{13} = (N_{12} + K_1) \bmod 256 \text{ и т.д. } \ldots\ldots\ldots\ldots \; (5)$$

В последнем цикле:

$$N_{255} = (N_{254} + K_3) \bmod 256; \; N_{256} = (N_{255} + K_4) \bmod 256 . \; (6)$$

Здесь – ЧЧ; ММ; SS – маркант (текущее время), который не только

используется в алгоритме шифрования, но и запоминается как неконфиденциальная информация для использования в алгоритме дешифрации;

– K_i – i-ый символ 12-символьной ключевой комбинации (если количество символов в ключевой комбинации меньше 12, то недостающие символы дополняются из последовательности – ЧЧ+K1, ММ+K2, SS+K3,K4 и т.д).

Таким образом, содержимое текущей пары строк меняется в 128 циклах. Такая процедура в математике называется прямым π-преобразованием. Использование марканта позволят реализовать весьма эффективную процедуру получения различного шифртекста при многократной шифрации одного и того же исходного текста одним и тем же секретным ключом.

Алгоритм формирования рабочего массива R_2 следующий: в качестве исходного массива берем массив R_1 и реализуем парное перемешивание по тем же формулам в 128 циклах с той лишь разницей, что во всех формулах вместо K_i используются G_i, где G_i – некоторая случайная последовательность, полученная с помощью стандартного ГПСЧ (CryptGenRandomи) при взаимодействии с 12-символьной секретной ключевой комбинацией K_i.

Массивы R_1 и R_2 типа char размерностью *256* байт каждый – это массивы, в которых некоторым образом перемешаны все возможные значения из множества типа *char*. Нуль может быть в нем на *35* месте, *1* – на *147* и т.д. Но в нем обязательно должны быть и *0*, н *1*, и вся остальная таблица *ASCII*-символов.

Такого рода процедура перемешивания называется прямой подстановкой и обозначается как π. А обратная подстановка – это когда в этой таблице адреса и значения меняются ролями: что было адресом – становится значением, а значение – адресом. И обозначается обратная подстановка как π^{-1}.

С точки зрения программной реализации такого рода преобразования называются π –нелинейными и являются весьма эффективными с точки зрения их нелинейности и скоростными. Если взять, к примеру, одну страницу текста на русском языке (*2000–2200* символов), то на ней будет встречаться почти каждая буква русского алфавита и всячески выпячивать свою индивидуальность – частоту появления на каждой странице текста. То же самое происходит и в компьютерном файле. Электронные документы, как известно, предопределяют наличие в них значительных по своему размеру фрагментов заполненных одинаковыми символами, так называемыми символами-заполнителями.

Вторым не менее неприятным моментом является существенный

разброс частоты появления одних символов по сравнению с другими. Возьмите, к примеру, файл, содержащий текст, набранный в редакторе *Microsoft Word*, и подсчитайте в нем количество пробелов, имеющих явное преимущество перед «серой» массой остальных символов. Например, в набранном в *Microsoft Word* текстовом файле блоки из одних нулей могут быть достаточно велики. Для шифрования таких блоков можно применить на предварительном этапе простейший алгоритм предварительной подстановки, добавляя при зашифровывании к каждому символу в блоке исходного текста его порядковый номер в блоке.

Модуль разбиения исходника (шифртекста) на блоки. Нашей отправной точкой является тот факт, что мы должны ориентироваться на работу с большими и очень большими массивами информации (именно в такой ситуации и проявляется главное преимущество симметричных алгоритмов криптографических преобразований – их быстродействие). Поэтому разработка алгоритма перемешивания охватывающего весь исходник одновременно это, конечно же, очень «круто», но весьма не рационально, так как приведет с существенным потерям в быстродействии системы в целом, при не высокой ее криптостойкости. Куда более целесообразно разбить исходник на небольшие, одинаковой размерности блоки, дополнив самый последний блок до стандартной размерности символами-заполнителями. Самым оптимальным вариантом размерности такого блока с точки зрения быстродействия и простоты обработки будут блоки размерностью 256 символов каждый, т.е. FFH в шестнадцатеричной системе счисления.

Модули второго этапа и блока самонастройки программного комплекса на конкретные варианты реализации будут рассмотрены в последующих публикациях.

Библиографический список

1. Шнайер Б. Прикладная криптография. М.: Изд-во «ТРИУМФ», 2002. 780 с.
2. Масленников М.Е. Практическая криптография. СПб.: БХВ-Петербург, 2003. 464 с.