Симметричные криптосистемы: поточные шифры

Лекция

Ревизия: 0.1

Последняя модификация: Vlad Kovtun
Дата последней модификации: 31.01.2011 21:23:00

История изменений

10.11.2009 – Версия 0.1. Первичный документ. Ковтун В.Ю.

Содержание

Лекция 3. Симметричные криптосистемы: поточные шифры. Часть 3

Вопросы

1. Введение. Симметричные поточные криптосистемы.
2. Описание симметричной поточной криптосистемы: SNOW.
3. Особенности реализации.

Введение. Симметричные поточные криптосистемы

Популярность поточных (потоковых) шифров (ПШ) стала очевидной после выхода в свет работы Клода Шеннона, посвященной анализу одноразовых гамм-блокнотов, изначально именовавшихся шифром Вернама.

ПШ преобразуют открытый текст в шифротекст по одному биту за операцию.

Одноразовый гамм-блокнот использует длинную управляющую последовательность (бегущий ключ), которая состоит из бит, выбираемых совершенно случайно. Управляющая последовательность побито накладывается на открытый текст. Данная последовательность имеет ту же самую длину, что и сообщение, и может использоваться только один единственный раз. Если открытый текст сообщения T записать как последовательность бит $T = t_0 t_1 \dots t_{n-1}$, а двоичную управляющую последовательность K той же самой длины как $K = k_0 k_1 \dots k_{n-1}$, то в самом общем виде процесс получения зашифрованного текста $C = c_0 c_1 \dots c_{n-1}$ определяется как $c_i = t_i \oplus k_i$, $0 \le i \le n-1$, где знак «$\oplus$» обозначает операцию побитового исключающего-или (XOR).

Шеннон доказал, что одноразовый гамма-блокнот является «невскрываемой» системой. Фактически, данная система была представлена как совершенная: даже противник, обладающий безграничным количеством вычислительной мощности, не в силах сделать ничего лучшего, как только предполагать значения бит сообщения, поскольку закрытый текст статистически не зависит от открытого текста.

ПШ, по своей сути, пытается имитировать концепцию одноразового гамма-блокнота, используя короткий ключ для генерации управляющей последовательности, которая была бы похожей на случайную. Такую управляющую последовательность называют псевдослучайной.

Безопасность системы полностью зависит от свойств генератора потока ключей. Если генератор потока ключей выдает бесконечную строку нулей, шифротекст будет совпадать с открытым текстом, и все операция будет бессмысленна. Если генератор потока ключей выплевывает повторяющийся 16-битовый шаблон, алгоритм будет являться простым XOR с пренебрежимо малой безопасностью. Если генератор потока ключей выдает бесконечный поток случайных (по настоящему, а не псевдослучайных) битов, получается одноразовый блокнот и идеальную безопасность .

На деле безопасность потокового шифра находится где-то между простым XOR и одноразовым блокнотом.

Генератор потока ключей создает битовый поток, который похож на случайный, но в действительности детерминирован и может быть безошибочно воспроизведен при дешифрировании. Чем ближе выход генератора потока ключей к случайному, тем больше времени потребуется криптоаналитику, чтобы взломать шифр.

Однако, если генератор потока ключей при каждом включении создает один и тот же битовый поток, то использующую его криптосистему взломать нетрудно. Покажем на примере, почему это так.

Если к Еве попал шифротекст и соответствующий открытый текст, то она, выполняя операцию XOR над открытым текстом и шифротекстом, раскрывает поток ключей. Или, если у нее есть два различных шифротекста, зашифрованных одинаковым ключом, она может выполнить над ними операцию XOR, получая два открытых текста сообщений, над которыми выполнена операция XOR. Это нетрудно взломать, и затем она может получить поток ключей, выполняя операцию XOR над одним из открытых текстов и шифротекстом.

Теперь, перехватив любое другое шифрованное сообщение, она сможет расшифровать его, используя полученный поток ключей. Кроме того, она может расшифровать и прочитать любое из ранее перехваченных сообщений. Когда Ева получит пару открытый текст/шифротекст, она сможет читать все.

Поэтому для всех потоковых шифров используются ключи. Выход генератора потока ключей является функцией ключа. Теперь, если Ева получит пару открытый текст/шифротекст, она сможет читать только те сообщения, которые зашифрованы тем же ключом. Измените ключ, и противнику придется начать все сначала. ПШ особенно полезны для шифрования бесконечных потоков коммуникационного трафика, например, канала T1, связывающего два коммуникационные узла.

Построение ПШ

Генератор потока ключей состоит из трех основных частей. Внутреннее состояние описывает текущее состояние генератора потока ключей. Два генератора потока ключей, с одинаковым ключом и одинаковым внутренним состоянием, выдают одинаковые потоки ключей. Функция выхода по внутреннему состоянию генерирует бит потока ключей. Функция следующего состояния по внутреннему состоянию генерирует новое внутреннее состояние.

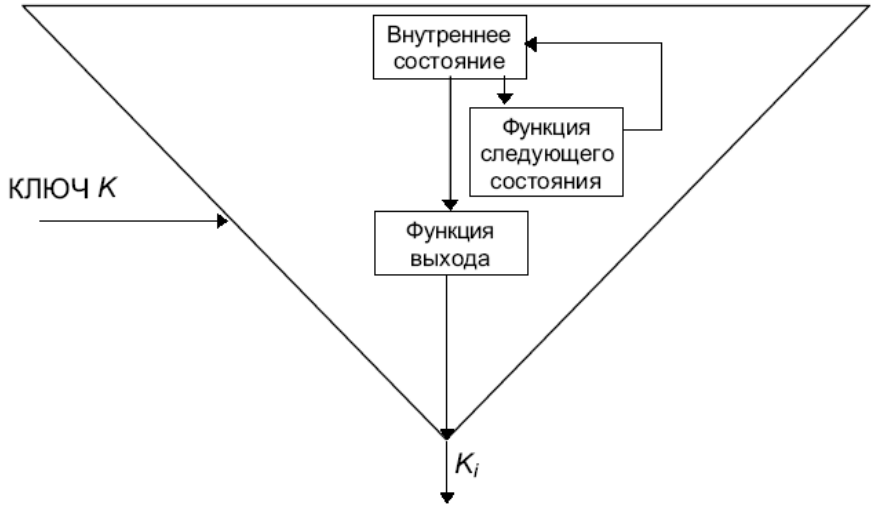

Рис. 1. Устройство генератора потока ключей (управляющей последовательности)

При построении ПШ, использующих в качестве базиса регистры сдвига с линейной обратной связью, имеется три альтернативных и достаточно очевидных метода генерирования управляющей последовательности:

- Первый метод подразумевает использование нескольких параллельно работающих регистров, выходы которых комбинируются каким-либо нелинейным узлом усложнения.
- Второй метод основан на нелинейной фильтрации выхода одного единственного регистра.
- Третий метод подразумевает управление движением основного регистра посредством управляющего регистра.

На основе этих методов могут быть построены синхронные и самосинхронизирующиеся системы, которые в дальнейшем реализуются как схемы с равномерным движением или неравномерным движением регистров, с управляющим регистром или с самоуправлением, с памятью или без.

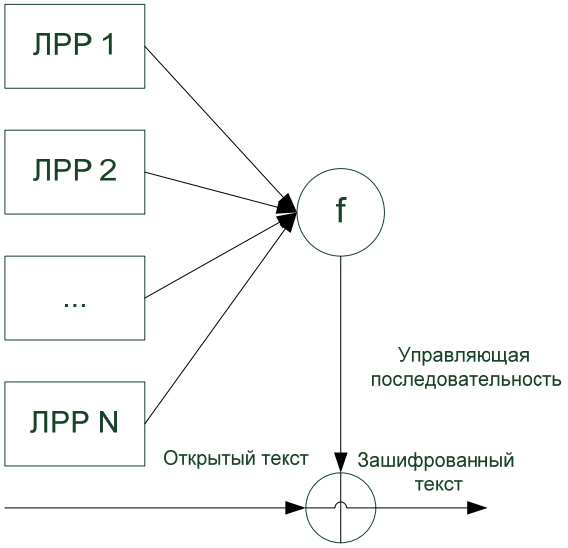

Рис. 2. Комбинирующий генератор

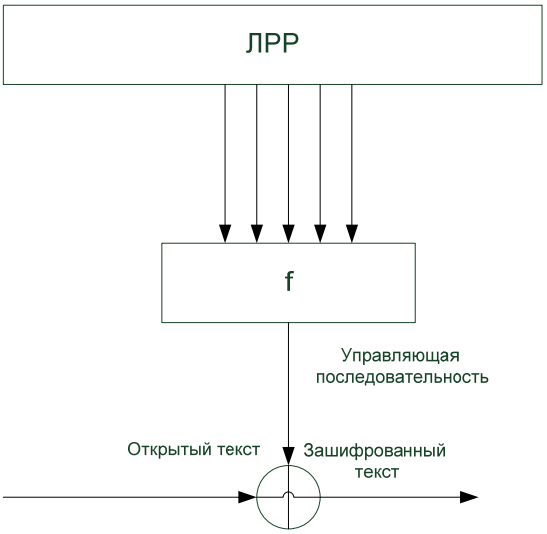

Рис. 3. Фильтр-генератор

Синхронный ПШ

В синхронном ПШ управляющая последовательность генерируется независимо от потока открытого текста и потока закрытого текста. Функционирование ПШ управляется двумя правилами:

$$s_{i+1} = \mathbf{F}(k, s_i),$$

$$z_i = f(k, s_i),$$

где s_i - внутреннее состояние в момент времени i, z_i — элемент управляющей последовательности, k - ключ.

В **синхронном ПШ** поток ключей генерируется независимо от потока сообщения. Военные называют этот шифр **ключевым автоключом** (Key Auto-Key, КАК). При шифровании генератор потока ключей один за другим выдает биты потока ключей. При дешифрировании другой генератор потока ключей один за другим выдает идентичные биты потока ключей. Это работает, если оба генератора синхронизированы. Если один из них пропускает один из циклов, или если бит шифротекста теряется при передаче, то после ошибки каждый символ шифротекста будет расшифрован неправильно.

Если такое случается, отправитель и получатель должны повторно синхронизировать свои генераторы потока ключей прежде, чем можно будет продолжить работу. Что еще хуже, они должны выполнить синхронизацию так, чтобы ни одна часть потока ключей не была повторена, поэтому очевидное решение перевести генератор в более раннее состояние не работает.

Положительная сторона синхронных фильтров - это отсутствие распространения ошибок. Если при передаче бит изменит свое значение, что намного вероятнее его потери, то только испорченный бит будет дешифрован неправильно. Все предшествующие и последующие биты не изменятся.

Генератор должен выдавать один и тот же поток ключей и для шифрования, и для дешифрирования, следовательно, выход генератора должен быть предопределен. Если он реализуется на конечном автомате (т.е., компьютере), последовательность со временем повторится. Такие генераторы потока ключей называются **периодическими.** За исключением одноразовых блокнотов все генераторы потока ключей являются периодическими.

Генератор потока ключей должен обладать длинным периодом, намного более длинным, чем количество битов, выдаваемых между сменой ключей. Если период меньше, чем размер открытого текста, то различные части открытого текста будут зашифрованы одинаковым образом, что сильно ослабляет безопасность системы. Если криптоаналитику известна часть открытого текста, он может раскрыть часть потока ключей и использовать ее для дальнейшего раскрытия открытого текста. Даже если у аналитика есть только шифротекст, он может выполнить XOR над разделами, шифрованными одинаковым потоком ключей, и получить XOR соответствующих

участков открытого текста. При этом используемый алгоритм превращается в простой алгоритм XOR с очень длинным ключом.

Конкретная длина периода зависит от приложения. Генератор потока ключей, шифрующий непрерывный канал T1, будет шифровать $2^?$ бит в день. Период генератора должен быть на несколько порядков больше этого значения, даже если ключ меняется ежедневно. Если период имеет достаточную длину, ключ можно будет менять раз в неделю или даже раз в месяц.

Синхронные ПШ также предохраняют от любых вставок и удалений шифротекста, так как они приводят к потере синхронизации и будут немедленно обнаружены. Однако, они не защищают полностью от битовых сбоев. Как и при блоковых шифрах в режиме CFB, Мэллори может изменить отдельные биты потока. Если ему известен открытый текст, он может изменить эти биты так, чтобы эти биты дешифрировались так, как ему надо. Дальнейшие биты при дешифрировании превратятся в чепуху (пока система не восстановится), но в определенных приложениях Мэллори может принести заметный ущерб.

Вскрытие вставкой

Синхронные ПШ чувствительны к **вскрытию вставкой**. Пусть Мэллори записал поток шифротекста, но не знает ни открытого текста, ни потока ключей, использованного для шифрования открытого текста:

Оригинальный открытый текст: $p_1 p_2 p_3 \dots p_i$. Оригинальный поток ключей: $k_1 k_2 k_3 \dots k_j \dots k_i$. Оригинальный шифротекст: $c_1 c_2 c_3 \dots c_i$.

Мэллори вставляет один известный ему бит, w', в открытый текст после p_1 и затем пытается получить модифицированный открытый текст, шифрованный тем же потоком ключей. Он записывает получившийся новый шифротекст:

Новый открытый текст: $p_1 p' p_2 p_3 \dots p_i$. Оригинальный поток: $k_1 k_2 k_3 \dots k_j \dots k_i$,

Обновленный шифротекст: $c_1 c'_z c'_2 c'_3 \dots c'_i$.

Так как он знает значение p', он может определить весь открытый текст после этого бита по оригинальному и новому шифротекстам:

$k_2 = c'_z \oplus p'$, затем $p_2 = c_2 \oplus k_2$, $k_3 = c'_3 \oplus p_2$, затем $p_3 = c_3 \oplus k_3$, $k_4 = c'_4 \oplus p_3$.

Мэллори даже не нужно знать точное положение вставленного бита, он может просто сравнить оригинальный и обновленный шифротексты, чтобы обнаружить, где они начинают отличаться. Для предотвращения такого вскрытия никогда не используйте один поток ключей для шифрования двух различных сообщений.

Режим обратной связи по выходу

Режим обратной связи по выходу (Output-feedback, OFB) представляет собой метод использования блочного шифра в качестве синхронного потокового шифра. Этот режим похож на CFB за исключением того, что n битов предыдущего выходного блока сдвигаются в крайние правые позиции очереди. Дешифрирование является обратным процессом. Такой режим называется n-битовым OFB. И при шифровании, и при дешифрировании блочный алгоритм работает в режиме шифрования. Это иногда называют **внутренней обратной связью,** потому что механизм обратной связи не зависит ни от потоков открытого текста, ни от потоков шифротекста.

К числу положительных свойств OFB относится то, что большая часть работы может быть выполнена автономно, даже до того, как появится открытый текст сообщения. Когда сообщение наконец появится, для получения шифротекста над сообщением и выходом алгоритма нужно будет выполнить операцию XOR.

В сдвиговый регистр OFB также сначала должен быть загружен IV. Он должен быть уникальным, но сохранять его в секрете не обязательно.

Распространение ошибки

В режиме OFB распространения ошибки не происходит. Неправильный бит шифротекста приводит к неправильному биту открытого текста. Это может быть полезно при цифровой передаче аналоговых величин, например оцифрованного звука или видеоизображения, когда случайный сбой бита допустим, но распространение ошибки нежелательно.

С другой стороны, потеря синхронизации смертельна. Если сдвиговые регистры при шифровании и при дешифрировании отличаются, то восстановленный открытый текст представляет собой бессмыслицу. Любая система, использующая режим OFB, должна включать механизм обнаружения потери синхронизации и механизм заполнения обоих сдвиговых регистров новым (или одинаковым) IV для восстановления синхронизации.

OFB и проблемы безопасности

Анализ режима OFB показывает, что OFB стоит использовать только, когда размер обратной связи совпадает с размером блока. Например, 64-битовый алгоритм нужно использовать только в 64-битовом режиме OFB. Несмотря на то, что правительство США разрешает для DES и другие размеры обратных связей DES, избегайте их.

Режим OFB выполняет XOR над потоком ключей и текстом. Этот поток ключей со временем повторяется. Важно, чтобы он не повторялся для того же ключа, в противном случае нарушается безопасность. Когда размер обратной связи равен размеру блока, блочный шифр переставляет w-битовые значения (где m - это размер блока), и средняя длина цикла составляет $2^m - 1$. При длине блока 64 бита это очень большое число. Когда размер обратной связи m меньше длины блока, средняя длина цикла падает до приблизительно 2'"*. Для 64-битного шифра это только * - что явно недостаточно.

Потоковые шифры в режиме OFB

Потоковые шифры также могут работать в режиме OFB. В этом случае ключ влияет на функцию следующего состояния. Функция выхода не зависит от ключа, очень часто она является чем-то простым, например, одним битом внутреннего состояния или результатом XOR нескольких битов внутреннего состояния. Криптографически сложной является функция следующего состояния, которая зависит от ключа. Этот метод также называется внутренней обратной связью, потому что механизм обратной связи является вложенным по отношению к алгоритму генерации ключей.

В одном из вариантов этого режима ключ определяет только начальное состояние генератора потока ключей. После того, как ключ определит внутреннее состояние генератора, генератор работает, не подвергаясь воздействиям извне.

Режим счетчика

Блочные шифры в **режиме счетчика** используют в качестве входов алгоритма последовательные номера. Для заполнения регистра используется счетчик, а не выход алгоритма шифрования. После шифрования каждого блока счетчик инкрементируется на определенную константу, обычно единицу. Для этого режима свойства синхронизации и распространения ошибки такие же, как и для OFB. Режим счетчика решает проблему n-битового выхода режима OFB, где n меньше длины блока.

К счетчику не предъявляется никаких особых требований, он не должен проходить по порядку все возможные значения. В качестве входа блочного алгоритма можно использовать генераторы случайных чисел, описанные в предыдущих лекциях, независимо от того, являются ли они криптографически безопасными или нет.

Потоковые шифры в режиме счетчика

У потоковых шифров в режиме счетчика простые функции следующего состояния и сложные функции выхода, зависящие от ключа. Функция следующего состояния может быть чем-то простым, например, счетчиком, добавляющим единицу к предыдущему состоянию.

Потоковый шифр в режиме счетчика может генерировать i-ый бит, k_i, без выдачи всех предшествующих ключевых битов. Просто установите счетчик вручную в i-ое внутреннее состояние и генерируйте бит. Это полезно для закрытия файлов данных с произвольным доступом, можно расшифровать конкретный блок данных не расшифровывая целый файл.

Самосинхронизирующийся ПШ

В самосинхронизирующемся ПШ управляющая последовательность генерируется в зависимости от потока зашифрованного текста. Функционирование ПШ можно описать как:

$$s_{i+1} = \mathbf{F}(y_{i-1}, y_{i-2}, \ldots, y_{i-N}),$$

$z_i = f(k, s_i)$,

где y_i, - элемент закрытого текста.

В самосинхронизирующихся ПШ каждый бит потока ключей является функцией фиксированного числа предыдущих битов шифротекста. Военные называют этот шифр **автоключом шифротекста** (ciphertext auto key, CTAK). Основная идея была запатентована в 1946.

Самосинхронизирующийся ПШ показан на рис. 2 и 3. Внутреннее состояние является функцией предыдущих n битов шифротекста. Криптографически сложной является выходная функция, которая использует внутреннее состояние для генерации бита потока ключей.

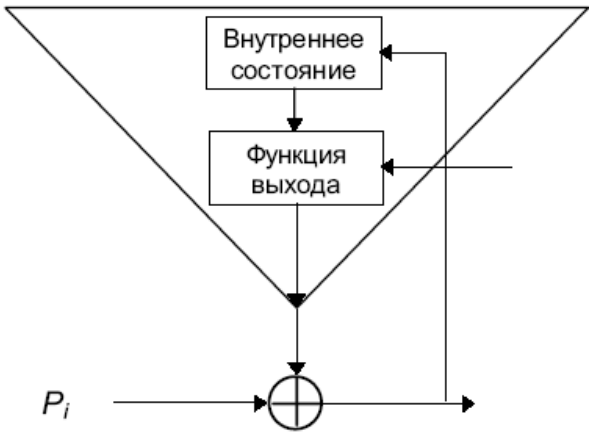

Рис. 4. Самосихронизирующийся ПШ (зашифровывание)

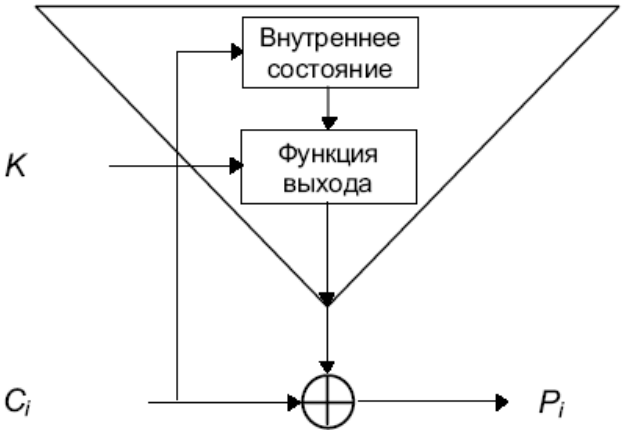

Рис. 5. Самосихронизирующийся ПШ (расшифровывание)

Так как внутреннее состояние полностью зависит от предыдущих n шифротекста, дешифрирующий генератор потока ключей автоматически синхронизируется с шифрующим генератором потока ключей, приняв n битов шифротекста.

В интеллектуальных реализациях этого режима каждое сообщение начинается случайным заголовком длиной n битов. Этот заголовок шифруется, передается и затем расшифровывается. Расшифровка будет неправильной, но после этих n битов оба генератора потока ключей будут синхронизированы.

Слабой стороной самосинхронизирующегося потокового шифра является распространение ошибки. Для каждого бита шифротекста, испорченного при передаче, дешифрирующий генератор потока ключей выдает n неправильных битов потока ключей. Следовательно, каждому неправильному биту шифротекста соответствуют n ошибок в открытом тексте, пока испорченный бит не перестанет влиять на внутреннее состояние.

Вопросы безопасности

Самосинхронизирующиеся потоковые шифры также чувствительны к вскрытию повторной передачей. Сначала Мэллори записывает несколько битов шифротекста. Затем, позднее, он вставляет эту запись в текущий трафик. После выдачи некоторой чепухи, пока принимающая сторона синхронизируется с вставленной записью, старый шифротекст будет расшифрован как нормальный. У принимающей стороны нет способа узнать, что полученные данные являются повторно передаваемой записью. Если не используются метки времени, Мэллори может убедить банк снова и снова зачислять деньги на его счет, повторно передавая одно и то же сообщение (конечно, при условии, что ключ не менялся). Другие слабые места этой схемы могут стать заметны при очень частой пересинхронизации.

Наиболее широкое применение получило использование схем с равномерным движением регистров без памяти, в силу сравнительной простоты реализации, высокой стойкости генерируемых последовательностей и высокого быстродействия. Данные схемы подразделяются на комбинирующие генераторы, см. рис. 1, и фильтр-генераторы, см. рис 2.

Режим обратной связи

Блочный шифр также может быть реализованы как самосинхронизирующийся потоковый шифр, такой режим называется режимом обратной связи по шифру (cipher-feedback, CFB). В режиме CBC шифрование не могло начаться, пока не получен целый блок данных. Это создает проблемы для некоторых сетевых приложений. Например, в безопасной сетевой среде терминал должен иметь возможность передавать главному компьютеру каждый символ сразу, как только он введен. Если данные нужно обрабатывать байтами, режим CBC также не работает.

В режиме CFB единица зашифрованных данных может быть меньше размера блока. В следующем примере каждый раз шифруется только один символ ASCII (это называется 8-битовым шифрованием), но в числе 8 нет ничего волшебного. Вы можете шифровать данные по одному биту с помощью 1-битового CFB, хотя использование для единственного бита полного шифрования блочным шифром потребует много ресурсов, потоковый шифр в этом случае был бы идеей получше. (Уменьшение количества циклов блочного фильтра для повышения скорости не рекомендуется) Можно также использовать 64-битовый CFB, или любой n-битовый CFB, где n больше или равно размеру блока.

Для инициализации процесса CFB в качестве входного блока алгоритма может использоваться вектор инициализации IV. Как и в режиме CBC IV не нужно хранить в секрете.

В режиме CFB ошибка в открытом тексте влияет на весь последующий шифротекст, но самоустраняется при дешифрировании. Гораздо интереснее ошибка в шифротексте. Первым эффектом сбоя бита шифротекста является сбой одного бита открытого текста. Затем ошибка попадает в сдвиговый регистр, и пока сбойный бит не выйдет из регистра, будет формироваться неправильный шифротекст. В 8-битовом режиме CFB из-за сбоя единственного бита портятся 9 байтов расшифрованного открытого текста. Потом система восстанавливается, и весь последующий шифротекст расшифровывается правильно. В общем случай в n-битовом режиме CFB одна ошибка шифротекста влияет на дешифрирование текущего и следующих $\frac{m}{n}-1$ блоков, где m - размер блока.

Более тонкой проблемой, связанной с такого рода распространением ошибки, является то, что если Мэллори знает открытый текст сообщения, он может поиграть битами данного блока, заставляя их расшифровываться в нужные ему данные. Следующий блок при дешифрировании превратится в чепуху, но вред уже будет причинен. К тому же, он может, оставаясь необнаруженным, менять последние биты сообщения .

CFB самовосстанавливается и после ошибок синхронизации. Ошибка попадает в сдвиговый регистр и, пока она находится там, портит 8 байтов данных. CFB представляет собой пример блочного шифра, который можно использовать как самосинхронизирующийся потоковый шифр (на уровне блоков).

Другие режимы блочных шифров

Режим сцепления блоков

Для использования блочного алгоритма в режиме **сцепления блоков** (block chaining, BC), просто выполните XOR входа блочного шифра и результата XOR всех предыдущих блоков шифротекста. Как и для CBC используется IV.

Как и CBC, обратная связь процесса BC приводит к распространению ошибки в открытом тексте. Главная проблема BC заключается в том, что из-за того, что дешифрирование блока шифротекста зависит от всех предыдущих блоков шифротекста, единственная ошибка шифротекста приведет к неправильной расшифровке всех последующих блоков шифротекста.

Режим распространяющегося сцепления блоков шифра

Режим **распространяющегося сцепления блоков шифра** (propagating cipher block chaining, PCBC) похож на режим CBC за исключением того, что и предыдущий блок открытого текста, и предыдущий блок шифротекста подвергаются операции XOR с текущим блоком открытого текста перед шифрованием (или после шифрования).

PCBC используется в Kerberos версии 4 для выполнения за один проход и шифрования, и проверки целостности. В режиме PCBC ошибка шифротекста приводит к неправильному дешифрированию всех последующих блоков. Это означает, что проверка стандартного блока в конце сообщения обеспечивает целостность всего сообщения.

К несчастью в этом режиме существует одна проблема. Перестановка двух блоков шифротекста приводит к неправильной расшифровке двух соответствующих блоков открытого текста, но из-за природы операции XOR над открытым текстом и шифротекстом, дальнейшие ошибки компенсируются. Поэтому, если при проверке целостности проверяются только несколько последних блоков расшифрованного открытого текста, можно получить частично испорченное сообщение. Хотя никто до сих пор не додумался, как воспользоваться этой слабостью, Kerberos версии 5 после обнаружения ошибки переключается в режим CBC.

Сцепление блоков шифра с контрольной суммой

Сцепление блоков шифра с контрольной суммой (cipher block chaining with checksum, CBCC) представляет собой вариант CBC. Сохраняйте значение XOR всех уже зашифрованных блоков открытого текста, выполняя для каждого текущего блока открытого текста перед его шифрованием XOR с сохраняемым значением. CBCC обеспечивает, что любое изменение любого блока шифротекста изменит результат дешифровки последнего блока. Если последний блок содержит какую-нибудь константу или служит для проверки целостности, то целостность расшифрованного открытого текста может быть проверена с минимальными дополнительными накладными расходами.

Обратная связь по выходу с нелинейной функцией

Обратная связь по выходу с нелинейной функцией (output feedback with a nonlinear function, OFBNLF) представляет собой вариант и OFB, и ECB, где ключ изменяется с каждым блоком.

Ошибка одного бита шифротекста распространяется только на один блок открытого текста. Однако, если бит теряется или добавляется, то ошибка распространяется до бесконечности. С блочным алгоритмом, использующим сложный алгоритм планирования ключей, этот режим работает медленно. Неизвестен метод криптоанализа шифротекста полученного в данном.

Прочие режимы

Возможны и другие режимы, хотя они используются нечасто. Сцепление блоков открытого текста (plaintext block chaining, PBC) похоже на CBC за исключением того, что операция XOR выполняется для с блока открытого текста и для предыдущего блока открытого текста, а не блока шифротекста. Обратная связь по открытому тексту (plaintext feedback, PFB) похожа на CFB за исключением того, что для обратной связи используется не шифротекст, а открытый текст. Существует также сцепление блоков шифротекста по различиям открытого текста (cipher block chaining of plaintext difference, CBCPD). Я уверен, что можно найти еще таинственнее.

Если у криптоаналитика есть машина для поиска ключей грубой силой, то он сможет раскрыть ключ, если угадает один из блоков открытого текста. Некоторые из упомянутых странных режимов, по сути, являются дополнительным шифрованием перед использованием алгоритма шифрования: например, XOR текста и фиксированной секретной строки или перестановка текста. Почти все отклонения от стандартов помешают подобному криптоанализу.

Выбор режима шифра

Если вашей основной заботой являются скорость и простота, то ECB является самым простым и самым быстрым способом использовать блочный шифр. Помимо уязвимости к вскрытию повтором, алгоритм в режиме ECB проще всего криптоанализировать. Не рекомендуется использовать ECB для шифрования сообщений.

ECB хорошо использовать для шифрования случайных данных, например, других ключей. Так как данные невелики по размеру и случайны, недостатки ECB не существенны для такого применения.

Для обычного открытого текста используйте CBC, CFB или OFB. Конкретный режим зависит от ваших требований.

Для шифрования файлов лучше всего подходит CBC. Значительно увеличивается безопасность, и при появлении ошибок в хранимых данных почти никогда не бывает сбоев синхронизации. Если ваше приложение -программное, то CBC почти всегда будет лучшим выбором.

Требования к современным ПШ

При определении эффективности функционирования ПШ целесообразно рассмотрение следующих основных групп показателей (рис.4).

Требуемый результат функционирования схемы R_{req} может быть представлен в виде:

$$R_{req} = \left\langle A_{\text{sec}}, A_{impl}, A_{ct} \right\rangle,$$

где A_{sec} – показатели стойкости схем ПШ, A_{impl} – программно- и аппаратно-реализационные показатели, A_{ct} - конструктивно-технологические показатели.

1. *Показатели стойкости ПШ*, A_{sec}, характеризуют стойкость схем к аналитическим методам анализа и являются основными показателями, поскольку целесообразность применения любых схем спецпреобразования информации основывается, в первую очередь, на оценке стойкости данных схем к известным на сегодняшний день методам (атакам) анализа, применимых к исследуемому классу схем.
2. *Программно- и аппаратно- реализационные показатели,* A_{impl}, характеризуют эффективность программной и аппаратной реализаций и являются показателями, характеризующими практичность схем спецпреобразований. Схема может обладать высокой стойкостью к различного рода методам анализа, но тем не менее иметь ограничения на практическое использование из-за высокой стоимости аппаратной реализации, низкого быстродействия и т.п. Одним из важнейших преимуществ методов поточного преобразования информации, благодаря которому они получили широкое распространение, является высокая скорость преобразования информации. Как следствие, схемы, имеющие высокие показатели эффективности реализации, с наибольшей вероятностью будут отобраны для практических приложений. Применимость схем часто рассматривается с позиций унификации (универсальности): например, одна и та же схема должна быть одинаково успешно реализуема и в программном обеспечении, и в аппаратном, на смарт-картах и т.п. Некоторые области приложений налагают очень высокие требования к скорости связи (гигабитные трафики), для других областей применения приложения приемлют минимальные аппаратные платформы и/или компактные аппаратные средства (сотовые телефоны, смарт-карты). В идеале рассматриваемые схемы должны быть гибкими, т.е. быть реализуемыми более чем на одной платформе. Таким образом, при определении эффективности реализации рассматриваемых схем целесообразно производить отбор показателей, отражающих эффективность функционирования схем в заданных областях приложений, поскольку, очевидно, одна и та же схема вряд ли может быть эффективно реализована на всех платформах и во всех приложениях.

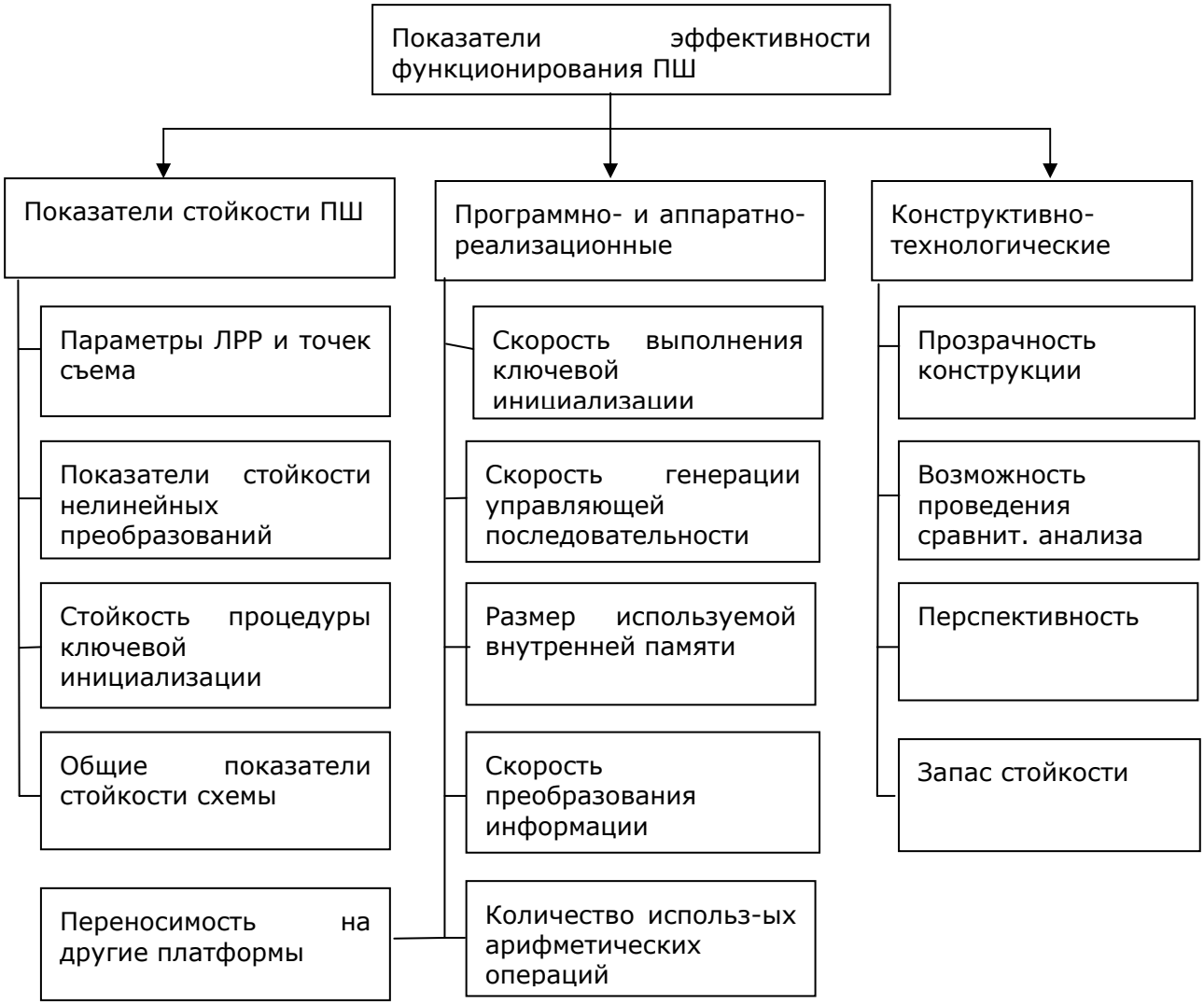

Рис. 6. Показатели эффективности функционирования ПШ

3. *Конструктивно-технологические показатели,* A_{ct}, характеризуют прозрачность конструкции, перспективность схем, запас их стойкости, пригодность известных методов анализа к оценке характеристик схем спецпреобразований, возможность проведения сравнительного анализа схем данного класса, и, как следствие, степень доверия пользователей.

Показатели стойкости ПШ

Общепринятым для ПШ является следующее множество показателей стойкости.

Показатели, характеризующие параметры регистров и точек съема для нелинейной функции и обратных связей, A_{LRR}.

- *Примитивность образующего полинома*. Примитивный многочлен степени n – это неприводимый многочлен, который делит x, но не делит $x^{2^{n-1}}+1$ для любого такого d, которое делит 2^n-1. Степень многочлена – это длина регистра сдвига. Только регистры, использующие в качестве образующего полинома примитивный полином, являются регистрами максимального периода (проходят через все возможные 2^n-1 состояний) и генерируют последовательности максимального периода, так называемые ml – последовательности. Критерием отбора является логическая переменная Да/Нет.
- *Взаимно простые степени образующих полиномов* (для комбинирующих генераторов). При выборе образующих полиномов, имеющих взаимно простые степени, линейная сложность (см. показатель линейная *сложность*) и период генерируемой последовательности z являются максимальными. Так,

$\wedge(z) = f(\wedge_1,\ldots,\wedge_N)$, где $\wedge_j$ - линейная сложность j-го генератора, $j = \overline{1,N}$. Критерием отбора является логическая переменная Да/Нет

- *Степень образующего полинома*. При правильно подобранном образующем полиноме степень полинома n определяет период генерируемой последовательности. На сегодняшний день для противостояния аналитическим атакам минимальная длина регистра должна составлять не менее 128 бит. Критерием отбора будем полагать $n \geq 128$ бит.
- *Плотность образующего полинома*. Для противостояния корреляционным атакам, основанным на низковесовых проверках четности, количество ненулевых коэффициентов k в образующем полиноме должно быть около $\frac{n}{2}$. Критерием отбора будем полагать $k \to \frac{n}{2}$.
- *Правильность определения количества точек съема для нелинейной функции* (для фильтр-генераторов). Для противостояния аналитическим атакам количество точек съема r должно быть $r' \leq \sqrt{2n}$. Критерием отбора является $r \to r'$.
- Соответствие множества точек обратных связей B полному множеству положительных разностей ΔB. Множество ΔB называют полным множеством положительных разностей, если все положительные попарные разности между его элементами различны. Требуемое соответствие обеспечивает противостояние аналитическим атакам. Критерием отбора является логическая переменная Да/Нет.
- *Соответствие множества точек съема для нелинейной функции* Γ *полному множеству положительных разностей* $\Delta\Gamma$ (для фильтр-генератов). Требуемое соответствие обеспечивает противостояние аналитическим атакам. Критерием отбора является логическая переменная Да/Нет.
- Наибольший общий делитель двух парных (соседних) положительных разностей должен быть равен 1. Критерием отбора является логическая переменная Да/Нет.

Показатели, характеризующие стойкость нелинейной функции, A_{NLF}

- *Сбалансированность выходной последовательности*. Функция f над V_n является *сбалансированной* функцией, если количество «0» и «1» в ее выходной последовательности равно: $\left|\{x \mid f(x) = 0\}\right| = \left|\{x \mid f(x) = 1\}\right| = 2^{n-1}$. Сбалансированность функции является показателем, отражающим стойкость управляющей последовательности к статистическим атакам. Критерием отбора является логическая переменная Да/Нет.
- *Нелинейность*. *Нелинейность* функции f - минимальное расстояние Хэмминга N_f между функцией f и всеми аффинными функциями над V_n: $N_f = \min\{d(f,\varphi)\}$, где φ - множество аффинных функций. Для сбалансированной функции f над V_n, $n \geq 3$, нелинейность N_f может достигать:

$$N_f \leq \begin{cases} 2^{n-1} - 2^{n/2-1} - 2, n = 2k \\ \left\lfloor\!\left\lfloor 2^{n-1} - 2^{n/2-1} \right\rfloor\!\right\rfloor, n = 2k+1' \end{cases}$$

где $\left\lfloor\!\left\lfloor x \right\rfloor\!\right\rfloor$ - максимальное четное целое, меньшее либо равное x.

Верхняя граница нелинейности для произвольной функции f над V_n может достигать:

$N_f \leq 2^{n-1} - 2^{\frac{n}{2}-1}$.

Нелинейность функции является важным показателем, поскольку несоблюдение данного показателя делает возможным проведение корреляционных атак, использующих корреляцию данной функции со множеством слабых функций. При построении стойких булевых функций необходимо обеспечить ее минимальную корреляцию со множеством всех слабых функций, то есть стремиться, чтобы нелинейность данной функции стремилась к верхней границе нелинейности.

Критерием отбора является $N_f = \max_{N_j}\{N_1, \ldots, N_r\}$, где N_j – нелинейность j–ой функции; $j = \overline{1,r}$.

- *Алгебраическая степень*. Алгебраическая степень $\deg(f)$ является степенью самого длинного слагаемого функции, представленной в алгебраической нормальной форме. Алгебраической нормальной формой называется выражение вида:

$$f(x_1,\ldots,x_n) = a_0 \oplus \bigoplus_{i=1}^{n} a_i x_i \oplus \bigoplus_{1\le i<j\le 1} a_{ij} x_i x_j \oplus \ldots \oplus a_{12\ldots n} x_1 x_2 \cdots x_n .$$

Высокая алгебраическая степень позволяет противостоять различным аналитическим атакам, призванным свести данную функцию к слабой (линейной).
Критерием отбора является $\deg(f) = \max_{\deg(f_j)}\{\deg(f_1), \ldots, \deg(f_r)\}$, где $\deg(f_j)$ – нелинейность j –ой функции; $j = \overline{1, r}$.

Показатели, характеризующие стойкость процедуры ключевой инициализации, A_{KI}

- *Нелинейность операций ключевой загрузки*. Нелинейность операций повышает стойкость к алгоритмам, основанных на использовании ключевой загрузки. Критерием отбора является логическая переменная Да/Нет.
- *Каждый бит инициализированного регистра является результатом нелинейных преобразований всех бит ключа*. Желательной является реализация, при которой каждый бит начального состояния регистра является функцией от нелинейного преобразования всех бит ключа, в таком случае:

$$I\left(K_{out}^{n}, K_{in}^{n-k}\right) \to 0 ,$$

где n - длина ключа; k - количество бит инициализированного ключа, введенных нелинейными преобразованиями, $k \to n$; K_{in}^{n-k} и K_{out}^{n} вводимый и инициализированный ключи соответственно. Таким образом, знание вводимого ключа исключает знание инициализированного ключа (при гарантированной стойкости нелинейных преобразований). Критерием отбора является логическая переменная Да/Нет.

Общие показатели, A_{comm}

- *Период управляющей последовательности*. Периодом последовательности *T* именуется количество бит до того момента, как последовательность начнет повторяться. Период — одна из наиболее важных характеристик при изучении свойств управляющей последовательности. Если период управляющей последовательности окажется слишком коротким, то различные части открытого текста окажутся преобразованными идентичным образом, что составляет серьезную слабость схемы. Зная фрагмент открытого текста, аналитик может восстановить соответствующий фрагмент управляющей последовательности. Тот факт, что данный участок управляющей последовательности появляется и во множестве других мест управляющей последовательности, позволяет аналитику успешно вскрывать другие закрытые тексты. Кроме того, если имеются только преобразованные тексты, закрытые одной и той же управляющей последовательностью, то их можно попарно складывать друг с другом для получения последовательности, которая равноценна сумме двух открытых сообщений и не зависит от управляющей последовательности. Имея необходимую статистику открытого текста, на этой основе можно восстановить как открытые тексты сообщений, так и саму управляющую последовательность. Поскольку ЛРР длиной n бит может находиться в одном из $2^n - 1$ внутренних состояний, то теоретически он может иметь период $2^n - 1$. Вопрос о том, насколько большим должен быть период управляющей последовательности, в открытой литературе является дискуссионным, и решение его зависит от конкретной области применения. Для ПШ, работающей на скорости 1 Мбайт/сек, последовательность с периодом 2^{32} повторит сама себя всего через 2^9 секунд или 8.5 минут, поэтому средства закрытия информации с подобной длиной периода не могут считаться адекватной защитой.

На проходящем в настоящее время открытом европейском криптографическом конкурсе NESSIE, целью которого является определение европейских стандартов специального преобразования данных, для ПШ определено два уровня стойкости:

- «высокий», при котором схема должна обладать длиной ключа $l_{key} \geq 256$ бит и иметь внутреннюю память $M_h \geq 256$ бит;
- «нормальный», при котором схема должна обладать длиной ключа $l_{key} \geq 128$ бит и иметь внутреннюю память $M_l \geq 128$ бит.

Таким образом, согласно европейского стандарта, длины регистров для обеспечения требуемой стойкости должны составлять по меньшей мере 256 и 128 бит соответственно. Как следствие, можем записать, что длины периодов T_h для схем с «высоким» уровнем стойкости и T_l для схем с «нормальным» уровнем стойкости должны составлять: $T_h \geq 2^{256}$ бит; $T_l \geq 2^{128}$ бит.

Тщательная оценка периода управляющей последовательности, порождаемой схемой, совершенно необходима при разработке любой ПШ. С практической точки зрения, управляющая последовательность должна быть достаточно длинной для того, чтобы в подавляющем большинстве случаев было маловероятным повторное использование одного и того же фрагмента в процессе закрытия информации.

Критерием отбора является $T_f = \max_{T_j}\{T_1, \ldots, T_r\}$, где T_j – период T_j – ой схемы; $j = \overline{1, r}$.

- *Линейная сложность последовательности*. Линейная сложность $\wedge(s')$ последовательности $s' = s_0, s_1, \ldots, s_{l-1}$ - длина L самого короткого регистра сдвига, порождающего заданную периодическую последовательность s', когда первые L цифр последовательности s' являются начальным заполнением регистра.

Данный критерий является основополагающим критерием. Любая последовательность, которую можно сгенерировать конечным автоматом (линейным или нелинейным) над конечным полем, имеет конечную линейную сложность. Следовательно, возможно построение алгоритмов, определяющих линейную сложность любой последовательности вне зависимости от способа ее генерации. Так, существует эффективный алгоритм Берлекампа-Месси, который быстро находит такой кратчайший регистр после изучения всего $2L$ бит управляющей последовательности, сгенерированной ЛРР.

По своей сути линейная сложность является мерой сложности сгенерированной последовательности. Рассматривая некоторый отрезок последовательности, аналитик пытается на основе имеющейся последовательности сконструировать свою собственную последовательность. Определение линейной сложности рассматриваемой последовательности дает возможность на основе найденного линейного эквивалента построить схему, которая бы генерировала последовательность аналогичную рассматриваемой, при этом знания о структуре схемы, сгенерировавшей заданную последовательность, являются излишними. Таким образом, большая линейная сложность управляющей последовательности - необходимое, но не достаточное условие практической стойкости ПШ.

Критерием отбора является $\wedge_f = \max_{\wedge_j}\{\wedge_1, \ldots, \wedge_r\}$, где $\wedge_j$ – линейная сложность последовательности j-ой схемы, $j = \overline{1, r}$.

- *Длина используемых ключей*. Длина ключа определяет стойкость к силовым атакам и составляет $k_h \geq 256$ бит для схем с «высоким» уровнем стойкости и $k_l \geq 128$ бит для схем с «нормальным» уровнем стойкости.

Критерием отбора является $k_f = \max_{k_j}\{k_1, \ldots, k_r\}$, где k_j – длина ключа j-ой схемы с заданным уровнем стойкости, $j = \overline{1, r}$.

- *Внутреннее состояние генератора (внутренняя память)* M должно быть, не менее, $2k$ бит, при длине ключа k бит. Данный показатель отражает стойкость к атакам «время-память». Критерием отбора является логическая переменная Да/Нет.
- *Длина генерируемой последовательности* l_{seq} *не должна превышать некоторого порогового значения* $N_{\max}$, $l_{seq} < N_{\max}$. Данный показатель отражает стойкость к корреляционным атакам, позволяющим по имеющемуся фрагменту управляющей

последовательности восстановить начальное заполнение ЛРР. Целью введения $N_{\max}$ является недопущение генерации последовательности, длина которой гарантировала бы использование корреляции последовательности с последовательностью ЛРР. Критерием отбора является логическая переменная Да/Нет.

- *Показатель, характеризующий стойкость схемы к аналитическим методам анализа*, $П_{АН}$. В целом, именно стойкость схемы к методам анализа определяет эффективность функционирования ПШ. Критерием отбора является логическая переменная Да/Нет.

Показатель, характеризующий стойкость схемы к статистическим методам анализа, A_{stat}

Статистические свойства управляющей последовательности являются одной из составляющих, определяющей стойкость схемы преобразования. Стойкость схемы зависит от того, насколько близко она аппроксимирует генератор случайных чисел, то есть насколько управляющая последовательность будет вычислительно непредсказуемой и неотличимой от действительно случайной последовательности. Общепризнанной методикой оценки статистических свойств спецпреобразований на сегодняшний день является методика, предложенная NIST. В данной методике по выборке строится статистический портрет источника, который содержит значения вероятности $P_{ij} \in [0,1]$ - значения вероятности, полученной в результате тестирования i-той последовательности j-ым тестом, $i = \overline{1,m}$, $j = \overline{1,q}$. Рекомендуемые значения: $m = 100$ последовательностей по 10^6 бит (генеральная совокупность составляет 10^8 бит), $q = 189$. Затем при анализе статистического портрета применяются такие критерии. Первым критерием отбора полагается коэффициент прохождения тестов последовательностями:

$$r_j = \frac{\#\{P_{ij} \geq \alpha \mid i = 1,2,...,m\}}{m},$$

попавший в доверительный интервал $[r_{\max}, r_{\min}]$, $r_{\max(\min)} = \hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}$,

где $\hat{p} = 1-\alpha$, α – уровень значимости, равный 0,01.

Второй критерий строится на основе расчета значения χ^2: $\chi_j^2 = \sum_{k=1}^{10} \frac{\left(F_k - \frac{m}{10}\right)^2}{\frac{m}{10}}$.

Считается, что схема прошла статистическое тестирование, если значения коэффициентов r_j для $\forall\ j = \overline{1,q}$ находятся внутри доверительного интервала $[r_{\max}, r_{\min}]$ и соблюдается условие $\chi_j^2 > 0{,}0001$ для всех j = 1, …, q.

Таким образом, комплексный показатель стойкости $П_{СТ}$ ПШ имеет вид:
$П_{СТ} = (П_{ЛРР}, П_{НЛФ}, П_{КИ}, П_{О})$.

Показатели программной и аппаратной реализации

Используют следующее множество показателей эффективности реализации A_{impl}

- *Скорость ключевой инициализации*, A_{KI}, циклов/байт – отображает скорость выполнения инициализации схемы – количество загруженных байт n_{db} (download bytes) ключа за один полный цикл сдвига регистра.

Критерием отбора является $n_{db} = \max_{n_j}\{n_1, \ldots, n_r\}$, где n_j - количество загруженных байт n_{db} ключа за один полный цикл сдвига регистра j –ой схемы; $j = \overline{1,r}$.

- *Скорость генерации выходной последовательности*, A_{os} (out sequence), циклов/байт – отображает скорость генерации выходной последовательности схемы - количество сгенерированных байт m_{ob} (out bytes) выходной последовательности за один полный цикл сдвига регистра.

Критерием отбора является $m_{ob} = \max_{m_j}\{m_1, \ldots, m_r\}$, где m_j - количество сгенерированных байт m_{ob} выходной последовательности за один полный цикл сдвига регистра j–ой схемы, $j = \overline{1, r}$.

- *Размер используемой памяти*, A_{mem}, Кб. Отображает размер используемой памяти, ОЗУ и ПЗУ.

Критерием отбора является $V_{mem} = \max_{V_j}\{V_1, \ldots, V_r\}$, где V_j - размер памяти, используемый j–ой схемой, $j = \overline{1, r}$.

- *Скорость зашифровывания/расшифровывания информации*, A_{encs} (encryption speed), Мб/сек.

Критерием отбора является $s = \max_{s_j}\{s_1, \ldots, s_r\}$, где s_j - скорость закрытия/ открытия информации j–ой схемой, $j = \overline{1, r}$.

- *Количество используемых различных арифметических операций*, A_{dao} (different arithmetic operations).

Критерием отбора является $op = \max_{op_j}\{op_1, \ldots, op_r\}$, где op_j-количество используемых различных арифметических операций j–ой схемой, $j = \overline{1, r}$.

- *Показатель переносимости на другие платформы*, A_{port} (portability). Критерием отбора является логическая переменная Да/Нет.

При аппаратной реализации в качестве дополнительных показателей могут рассматриваться: габариты изделия, стоимость изделия, возможность сопряжения с другими типами аппаратуры, возможность реализации на различных микросхемах.

Таким образом, комплексный показатель эффективности программной и аппаратной реализации A_{impl} схем ПШ имеет вид: $A_{impl} = \langle A_{KI}, A_{os}, A_{mem}, A_{encs}, A_{aop}, A_{port} \rangle$.

Конструктивно-технологические показатели

Определим следующее множество показателей эффективности реализации A_{ct}:

- *Прозрачность конструкции*, A_{tr} (transparency). Критерием отбора является логическая переменная Да/Нет.
- *Возможность проведения сравнительного анализа*, A_{ca} (contrastive analysis). Критерием отбора является логическая переменная Да/Нет.
- *Перспективность*, A_a (availability). Критерием отбора является логическая переменная Да/Нет.
- *Запас стойкости*, A_{sr} (security reserve). Критерием отбора является логическая переменная Да/Нет.

На сегодняшний день для оценки конструктивно-технологических показателей можно использовать методы экспертных оценок.

ПШ, удовлетворяющие данным показателям, обеспечивают безопасность информации (конфиденциальность информации).

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