И. Коряков

КРИПТОПОЛЕМИАДА

Предисловие

Автор преклоняется перед творчеством великого изобретателя Генриха Сауловича Альтшуллера, который попытался формализовать процесс создания изобретений. Его «Алгоритм изобретения» и соответствующие таблицы дают возможность при строгой формулировке исходной проблемы получить решение в виде полуабстрактной рекомендации, типа: «для снижения затрат энергии ледоколом требуется, чтобы от нижней кромки льда до верхней кромки льда поперечное сечение ледокола было нулевым».

Данная статья носит полемический характер и имеет своей целью рассмотрение ряда вопросов из области криптографии (симметричные блочные шифры) под иным, чем принято, углом зрения. Автор попытался применить методы Альтшуллера к криптографии (в последнее время ставшей очень заматематизированной и авторитарной наукой) и заранее приносит свои извинения специалистам, имеющим фундаментальное криптологическое образование, за использование инженерной лексики.

Плохие ключи

Недоверие к чистой случайности

В мировой криптографической литературе существует традиция: разделять ключи на «хорошие» и «плохие». Например, ключ, состоящий из одних только нулей – конечно же, плохой.

Представьте себе разовый блокнот, состоящий из одних нулей. Это же полная катастрофа – такой ключ недопустим!

Именно теорема 6 из статьи Клода Шеннона «Теория связи в секретных системах» определяет абсолютную стойкость совершенных секретных систем тем, что ключ представляет собой исключительно случайную последовательность.

Необходимое и достаточное условие совершенной секретности заключается в том, что условная вероятность $P_M(E)$ криптограммы E при передаче сообщения M должна быть равна вероятности $P(E)$ получения криптограммы E для всех M и E. Это условие определяет шифры с ключом, являющимся истинно случайной последовательностью с длиной, равной (или более) длине сообщения.

Может ли ключ типа разового блокнота принять значение «0» во всех разрядах? Да, с соответствующей вероятностью, если это совершенная секретная система по К.Шеннону. Все комбинации в последовательности ключа возможны и равновероятны. Можно ли какие-либо из участков последовательности разового блокнота выбрасывать, чтобы блокнот выглядел более случайным? Нет, нельзя. Иначе не будет справедлива теорема 6.

Как же побороть это противоречие? Предположим, мы шифруем секретную телеграмму особой важности о времени начала атаки наших войск (выбран разовый блокнот в качестве шифра для абсолютной секретности) и на текст этого времени начала атаки, например, «5:30» пришлись значения гаммы, равные нулю. Ноль по модулю 2 с любым значением равен этому значению и в

криптограмме, предназначенной для передачи по открытому каналу, фигурирует ничем не защищенный открытый текст «5:30»!

Вот вам и хваленые «совершенные секретные системы»! С довольно высокой вероятностью они пропускают открытый текст на выход! Более того, в любом шифртексте любого шифра половина бит совпадает с открытым текстом! Тождественна открытому тексту с вероятностью 0.5!

Да, это всё правда.

Но, тем не менее, теорема 6 продолжает работать, и никто, никогда, не зная ключа, не сможет восстановить открытый текст. «Даже когда чужаки из созвездия Андромеды приземлятся на нашей планете на своих огромных космических кораблях с компьютерами невообразимой мощности, и они не смогут прочесть сообщения советских агентов, зашифрованные с помощью одноразовых блокнотов» [1].

Далее мы попробуем взглянуть на проблему плохих ключей с позиции совершенных секретных систем и придем к выводу, что противоречия нет и, возможно, с ним не надо бороться.

Антропоморфизм в отношениях со случайностью

Википедия: «Парадокс закономерности – наблюдение, заключающееся в том, что большинство людей, увидев явную закономерность в результатах серии испытаний, будут склонны считать, что испытания не являются случайными, потому что появление определенной последовательности в случайных испытаниях представляется им маловероятным событием. Мозг человека специализирован на выявлении закономерностей в окружающем мире с целью их использования для увеличения шансов выжить, поэтому нет ничего удивительного в том, что идеальная случайная последовательность может казаться человеку неслучайной».

Автор должен признаться, что сам поскальзывался на человеческом отношении к случайности. Например, генерируются огромные массивы случайных последовательностей при помощи сертифицированного генератора и автор, открыв файл со случайными числами, с ужасом видит его начало:

1D BB BB C1 FD 93 05 27 B2 32 66 E9 99 0A 0D DA

Сразу бросается в глаза, что полубайт «B» повторяется 4 раза подряд, «1D» и «C1» зеркально переставлены и отличаются одним битом. «FD 93 05 27» – еще куда ни шло, а вот байты «B2 32» тоже разнятся только старшим битом, а дальше «66 E9 99» – две шестерки подряд и, почти впритык – три девятки подряд. А в конце уже просто издевательство: «0A 0D» отличаются только одним битом и в последнем байте повторяются как «DA». Кроме того, «0A 0D» – это символы «Перевод строки» и «Возврат каретки», которыми заканчивается каждая строка в текстовом файле! Разве такая последовательность может быть случайной?!

На поверку этот файл вместе со всеми его «малослучайными» фрагментами прекрасно прошел самые придирчивые тесты на случайность, как и все остальные файлы, порожденные данным генератором.

Второй пример касается неосознания того, что сила статистических законов является очень мощной, в каком-то смысле – непереборимой. В термодинамике эта сила подразумевается сама собой – мы хорошо понимаем, что температура воды в стакане, представляющая суммарную кинетическую энергию молекул, не может случайно измениться на 5 градусов, она будет очень стабильна. Наши измерители температуры будут иметь ошибку, неизмеримо бОльшую, чем дисперсия этой суммарной энергии.

Но когда рассматриваются статистические свойства реальных объектов, то человек легко может заблуждаться в оценках. Довольно неожиданный результат дает простейший пример с кусочком вещества из 10 молекул, случайно движущихся в одном направлении. Будем считать, что кусочек вещества летит, если все 10 молекул движутся в направлении, ограниченном сектором в 10 градусов. Число направлений движения каждой молекулы равно $(360/10)^2 = 1296$ и, если направления независимы, то вероятность совпадения направлений движения всех молекул равна $(1/1296)^9 = 10^{-29}$. Если направления меняются каждую наносекунду, то среднее время ожидания полета частицы составит $10^{29} / 10^9 / 3600 / 24 / 366$ – около 3000 миллиардов лет. Результат кажется неправдоподобным: всего 10 молекул, а такая статистическая мощность, которая не позволяет даже такой маленькой частице полетать!

Или, например, тот же автор, имея систему из десятков тысяч аппаратно реализованных автоматов, осуществляющих перебор 2^{27} состояний каждый, ломал себе голову над тем, как эффективнее использовать ресурс – ведь автомат может закончить перебор и на первом шаге и на последнем. Но на практике оказалось, что никакой оптимизации не требуется, автоматы заканчивают перебор практически одновременно: на интервале в 600 секунд отклонение времени окончания составляло доли миллисекунды!

Википедия: «Общий смысл закона больших чисел – совместное действие большого числа одинаковых и независимых случайных факторов приводит к результату, в пределе не зависящему от случая».

В чистой случайности заключена огромная мощь, пренебрегать которой неразумно.

Основная методологическая ошибка

Исторически сложилось так, что генераторы случайных последовательностей, используемые для формирования ключей, имели статистические характеристики генерируемых последовательностей, далекие от идеальных. Поэтому последовательности, ими формируемые, требовали тщательной отбраковки по строгим статистическим критериям. То есть, проверялась гипотеза: «на момент формирования данного отрезка последовательности генератор временно стал истинно случайным».

Более того, рассматривалась только последовательность с длиной, равной длине ключа, и критерии определялись из этой длины. Например, проверка на допустимое число единиц или нулей в последовательности заданной длины N определялась из доверительной вероятности β следующим образом [4]:

$$N\left(p-\arg\Phi*\left(\frac{1+\beta}{2}\right)\sqrt{\frac{pq}{N}}\right)<Np*<N\left(p+\arg\Phi*\left(\frac{1+\beta}{2}\right)\sqrt{\frac{pq}{N}}\right), \qquad (1)$$

где p и $p*$ - соответственно вероятность и ее оценка; $q = 1– p$.

Для $N = 256$ и $\beta = 0.99$ число единиц/нулей должно попадать в интервал от 112 до 144.

При этом ненадежность таких оценок заключается в том, что 1% истинно случайных последовательностей не пройдет проверку и, что самое ужасное, неизвестно какой процент неслучайных последовательностей ее пройдет.

Увеличивая доверительную вероятность β, мы «распахиваем ворота», допуская большее число неслучайных последовательностей, уменьшая – вводим более жесткую отбраковку, то есть фильтрацию последовательностей, что делает их всё более неслучайными. Во всяком случае, уменьшаем вероятность того, что совокупность таких последовательностей порождена генератором Истинно Случайных Последовательностей (ИСП).

В классических работах по генераторам случайных чисел [1,2,3] рассматривается проверка Нуль-гипотезы (H_0), которая заключатся в утверждении, что генератор случайных последовательностей генерирует истинно случайные последовательности.

Но правильнее было бы проверять генератор, а не последовательности.

Мы не имеем права подменять проверку Нуль-гипотезы по отношению к генератору ИСП аналогичной проверкой некоторого фрагмента последовательности, который мы хотим использовать в качестве ключа. При такой подмене будет происходить парадоксальный процесс: для некоторых фрагментов мы будем получать подтверждение Нуль-гипотезы, а для некоторых – нет. То есть, наш генератор то будет признаваться истинно случайным, то нет. Хотя, при проверке Нуль-гипотезы по отношению к генератору, она будет подтверждаться, пока генератор ИСП будет оставаться таковым.

В свете современных представлений [5] генератор ИСП должен рассматриваться как метрологическое оборудование, он ни чем не хуже осциллографа или генератора точного времени и частоты. Генератор ИСП, как прибор, должен быть сертифицирован и поверен, на протяжении жизненного цикла генератора должен действовать механизм подтверждения Нуль-гипотезы. В этой же работе приведено описание 7-и уровней тестирования генератора с целью подтверждения Нуль-гипотезы от этапа его проектирования до непосредственно процесса генерации ИСП.

При неподтверждении Нуль-гипотезы генератор ИСП должен немедленно выводиться из эксплуатации, а сгенерированные им последовательности по возможности не должны

использоваться.

Но, если генератор ИСП исправен и Нуль-гипотеза подтверждена, то мы обязаны использовать его выходные последовательности без какой либо фильтрации.

Это снимает описанное выше противоречие, возникающее при применении устаревших методик генерации случайных последовательностей, используемых для формирования ключей и случайных параметров криптоалгоритмов.

Вместе с тем проверки типа (1), конечно же, выполняются, но не с целью проверки последовательностей, а с целью формирования оценки вероятности подтверждения Нуль-гипотезы для генератора.

Примеры плохих ключей

Ключ, содержащий все нули или все единицы, безусловно, снижает стойкость алгоритма шифрования. Если мы считаем плохими такие ключи, то почему мы должны считать допустимыми ключи вида 01010101010... или 0000111100001111...? Список подозрительных ключей может быть очень большим и, все равно, мы обязательно что-то упустим. Что же делать?

Помимо ключей, содержащих одни нули или единицы, известны, например, ключи алгоритма ГОСТ 28147-89, у которых $X_0 = X_7$, $X_1 = X_6$, $X_2 = X_5$, $X_3 = X_4$. Множество таких ключей признано слабыми ключами, радикально снижающими стойкость алгоритма, причем оказывается, что таких ключей невероятно много: 2^{128} ($3*10^{38}$).

Посмотрим, как часто нам будут встречаться такие ключи. Пусть у нас имеется 100 миллиардов генераторов ключей, работающих со скоростью 100 миллиардов ключей в секунду каждый. Как часто мы будем встречать такие слабые ключи?

Ответ простой: $3*10^{38} / 10^{11} / 10^{11} / 3600 / 24 / 366 = 1076080100$. За это время (1 миллиард лет) мы с вероятностью 0.63 получим первый слабый ключ.

С ключами, содержащими одни нули или единицы, дело обстоит похуже – вероятность их появления равна $1/2^{255}$.

«Если вы выбираете ключ случайно, вероятность выбрать один из слабых ключей пренебрежимо мала. Если вы настоящий параноик, можете всегда проверять "на слабость" сгенерированный ключ. Некоторые думают, что нечего и беспокоиться на этот счет. Другие утверждают, что проверка очень легка, почему бы ее и не выполнить» [1].

О секретных параметрах

Что касается параметров криптографических преобразований, независимо от того, являются ли они секретными ключевыми элементами или нет, – задача решается всегда очень типично: вначале постулируется, что эти параметры или преобразования должны быть как можно ближе к случайным, а затем определяется формальный, далекий от случайного, порядок их формирования, накладывающий существенные ограничения на их возможные значения.

Авторы таких методик тут же попадают в плен парадокса – они хотят сделать что-то более случайным через ограничение случайности. А так как ничего четкого и доказуемого из этого не получается, они вводят некие субъективные параметры, выбор значений которых должен исправить положение и не лишать параметры доли случайности. Вот классический пример [7]:

«...применительно к подстановкам порядка n введем понятие случайной (квазислучайной) подстановки, под которой понимается подстановка, удовлетворяющая трем критериям случайности, включающим выполнение следующих свойств:

Свойство 1. Число инверсий η_n в подстановке степени n удовлетворяет условию

$$\left|\eta_n - \frac{n(n-1)}{4}\right| \le \alpha\sigma_\eta, \sigma_\eta = \frac{n^{3/2}}{6}.$$

Свойство 2. Число циклов ξ_n в подстановке степени n удовлетворяет условию

$$\left|\xi_n - \ln n\right| \le \alpha\sigma_\xi, \sigma_\xi = \sqrt{\ln n}.$$

Свойство 3. Число возрастаний θ_n в подстановке степени n удовлетворяет условию

$$\left|\theta_n - \frac{n}{2}\right| \le \alpha\sigma_\theta, \sigma_\theta = \sqrt{n/12}.$$

В этих соотношениях α – параметр, выбираемый в значительной степени из субъективных соображений (по крайней мере, из условия, что множество допустимых подстановок будет не чересчур уж ограниченным)».

Представьте себе, строгая математика плюс субъективные соображения! При том многие авторы сознаются, что попытка улучшения какого-либо показателя стойкости к одному определенному виду анализа, путем придания параметрам или функциям некоторых специфических свойств, обязательно приводят к ухудшению показателей стойкости к другим видам анализа.

Фильтрация ключей

Цитата из ГОСТ 28147-89:

«Ключи, определяющие заполнения КЗУ и таблиц блока подстановки К, являются секретными элементами и поставляются в установленном порядке».

Это означает, что и главные ключи 256 бит, и таблицы блока подстановки являются ключом, единственным требованием к которому является равновероятный выбор из множества ключей. И, если для главного ключа требование к вероятности значений всех бит, равной 0.5, и независимости является естественным и непререкаемым, то для таблиц блока подстановки наблюдается странный дуализм.

С одной стороны, содержимое этих таблиц должно быть близким к равновероятному и непредсказуемому, а с другой – предопределено в соответствии с некоторыми формальными правилами.

Обычно принимается за аксиому необходимость формирования узлов замены ГОСТ как перестановок значений 0...15 в 16-ти позициях (биективность). Однако, это требуется лишь в случае обратимости преобразования, которое для сети Фейстеля, основы алгоритма ГОСТ, не является обязательным.

Андрей Винокуров: «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 статье, однако в самом факте убедиться несложно. Для этого достаточно, используя демонстрационную программу шифрования файлов данных, прилагающуюся к настоящей статье, зашифровать а затем расшифровать файл данных, использовав для этой процедуры «неполноценную» таблицу замен, узлы которой содержат повторяющиеся значения».

Если мы сформируем таблицу блока подстановки без всяких ограничений, просто как случайные значения каждого элемента таблицы в пределах 0..15, то множество таблиц будет очень большим: 2^{512}. Это действительно, очень много. Какое бы ни существовало «слабое» содержимое этих таблиц, мы не сможем указать эффективный способ предсказания преобразования после 32-х раундов, так как их содержимое – случайно.

Наложение ограничения на биективность уменьшает мощность множества таблиц замены до 2^{354}, но это, все равно, дико невозможные вероятности. Пусть мы даже, для уменьшения некоего виртуального расстояния, потребуем, чтобы сами 8 таблиц были предельно далеки друг от друга, и, доказав, что это – строки латинского квадрата, попробуем оценить число возможных вариантов, то мы с удивлением обнаружим, что до порядка квадрата, равного 12, – это еще известно, а для 16-ти – нет, неизвестно. Мировая математика просто не знает, сколько этих вариантов. Что уже говорить о нас. Кстати, те же авторы [7] приходят к выводу, что устойчивость к различного рода атакам определяется, прежде всего, шириной и числом раундов шифра, а не особым содержимым таблиц подстановки, которое может быть просто случайным.

Еще раз повторюсь: какие бы мы ни придумывали критерии, сокращающие размер множества допустимых ключей, мощность этого множества в практических случаях настолько велика, что исключение или добавление к нему подмножества недопустимых ключей НЕ ВЛИЯЕТ на мощность множества. То есть, есть ли в этом множестве недопустимые ключи или нет – это все равно.

Например, возвращаясь к примеру «плохих» ключей ГОСТ, посчитаем на мощном калькуляторе:

всего ключей 2^{256} = 1.15792089237316195423570985008691065 2e+77

и оставшаяся часть без «плохих»: $2^{256} – 2^{128}$ = 1.15792089237316195423570985008691065 2e+77.

Как говорится, «найдите 10 отличий».

Выводы

Конечно, ограничения, превращающие случайную последовательность в неслучайную, никак недопустимы. Например, автору известна методика «выкусывания» из гаммы серии нулей с длиной, более определённой, для предотвращения попадания достаточно длинных последовательностей символов открытого текста в шифротекст. Но такая последовательность не проходит тесты NIST Special Publication 800-22 [9], она не может считаться случайной и неприменима в качестве гаммы.

Если пространство ключей однородно и размер его достаточен для исключения подбора ключа полным перебором, то разделять ключи на хорошие и плохие нет необходимости, законы статистики очень жестки и надежно защищают шифр без необходимости подбора «стойких» ключей. Выискивание среди ключей по определенным критериям только хороших – напрасная трата времени, в лучшем случае, а в худшем – работа на руку противника, уменьшение области поиска при вскрытии шифра, так как любое правило фильтрации случайной последовательности делает ее неслучайной.

Для генерации ключей должны использоваться исключительно генераторы истинно случайных последовательностей [5]. Фильтрация случайной последовательности, влияющая на ее статистику, должна быть запрещена, даже если нам кажется, что последовательность не совсем случайна или по каким-то критериям не подходит в качестве ключа.

Не будем забывать, что бабушки в Блечтли-парке доставали буквы из лотерейного барабана и записывали их в шифроблокнот (Криптономикон). Когда им «казалось», что буква не случайна, они возвращали букву в барабан и не записывали ее. Это сформировало дырку в случайности, которой враги сумели воспользоваться. Мировая фантастика, конечно [6]... Но.

Режимы шифрования

Единственный режим

В книге «Практическая криптография» [8] Фергюсон и Шнайер в параграфе «5.7 Какой режим выбрать» однозначно приходят к выводу о целесообразности применения всего одного режима шифрования – гаммирования (режим счётчика, CTR):

«На наш взгляд, преимуществ CTR вполне достаточно, чтобы отдать предпочтение именно этому режиму, кроме тех ситуаций, где вы не можете контролировать способ применения функции шифрования. С одной стороны, мы отмечали, что каждая часть системы должна сама обеспечивать свою безопасность и не зависеть от остальных ее частей. С другой стороны, рекомендуем использовать режим CTR, уникальность оказий которого обеспечивается другими частями системы.»

Напомним, что в режиме счётчика вначале шифруется блок, называемый вектором инициализации, синхропосылкой или оказией, результат зашифрования заносится в счётчик, который инкрементируется и шифруется для получения каждого последующего блока гаммы. Зашифрование и расшифрование выполняются одинаково: сложением по модулю 2 входного текста с гаммой.

Что хорошего в режиме CTR и каковы его ограничения?

Преимущества:

- Криптостойкость шифрования в режиме гаммирования строго определяется только стойкостью блочного шифра.
- Не требуется дополнения сообщений до полного блока.
- Режим CTR допускает параллельный подсчет любого количества блоков гаммы, в результате чего реализации CTR могут достигать более высоких скоростей, вплоть до одного такта на блок при аппаратном конвейере.

- Достаточно реализовать только функцию зашифрования блочного шифра, которая, к тому же может быть необратимой.
- Режим CTR обеспечивает неравенство открытого текста для каждой пары блоков шифрованного текста.
- При обнаружении коллизии утечка информации в режиме CTR меньше, чем в других режимах.

Особенности и ограничения:

- Необходимо обеспечивать уникальность каждого вектора инициализации при шифровании на одном и том же ключе.
- Парадокс задачи о днях рождения. Если размер блока равен 128 бит, ожидать первого повторения блока шифрованного текста можно, зашифровав около 2^{64} блоков. Практически на одном ключе можно безопасно зашифровать 2^{64} байт.

Необязательная оказия

Существуют ситуации, когда ключ является исключительно разовым, то есть, каждому сообщению соответствует свой ключ. В классических системах секретной связи это Циркулярная и Индивидуальная связь.

В этом случае можно использовать значение вектора инициализации, одинаковое для всех сообщений (например «12345...» или даже «0») и, таким образом, не передавать его в составе криптограммы.

Например, режим ECB, который не требует вектора инициализации, рекомендуется для шифрования ключевой информации, не обладающей выраженной статистикой. Но вместо него можно использовать режим CTR с фиксированным вектором инициализации, если весь массив ключей шифруется одним мастер ключом.

Итого, все режимы шифрования можно с успехом заменить на режим CTR.

Целостность

Атака модификации

Если противник знает структуру сообщения, потенциально он может модифицировать некоторый фрагмент сообщения, сложив шифротекст этого фрагмента с определённым кодом.

Борьба с атакой модификации может проводиться следующими методами.

1. Вычислением криптографически сильной имитовставки или хеш-функции от открытого или зашифрованного текста и передача этого значения в составе криптограммы.

2. Смещение текста: открытый текст циклически смещается на случайное число позиций, это число передаётся в зашифрованном заголовке сообщения.

3. Архивация со сжатием открытого сообщения. При этом контроль целостности осуществляется самой процедурой архивации. Несмотря на криптографически нестойкий контроль целостности при помощи CRC, защита от атаки модификации обеспечивается тем, что элементы сжатого текста определяются множеством элементов открытого текста и, чтобы модифицировать открытый текст путём операций над зашифрованным архивом, необходимо знать открытый либо сжатый текст. Но противник его, к счастью, не знает.

Архивация со сжатием

Вообще, процедуру архивации желательно сделать обязательной и автоматической при шифровании гаммированием. Это даёт:

- защиту от модификации;
- исключение статистики из шифруемого текста;
- сокращение объёма шифротекста.

Шифрование ключей

Классическая схема

Центр генерации ключей (ЦГК) обычно содержит генератор ИСП и средства записи ключей на носители ключей (НК). ЦГК для абонентов некоей секретной сети связи формирует наборы ключей, которые шифруются на мастер-ключе (МК).

МК заносится в каждый шифратор в составе сети и каждый шифратор генерирует свою случайную маску (СМ), которая складывается по модулю 2 с МК. Замаскированные МК безопасны и хранятся в шифраторах, а уникальные для каждого шифратора СМ хранятся на отдельном носителе, легко удаляемом в случае опасности компрометации и, возможно, уничтожаются.

Ключи используются следующим образом.

1. Считывается маска СМ.

2. СМ складывается по модулю 2 с замаскированным МК.

3. Каждый применяемый зашифрованный ключ расшифровывается на МК.

4. После применения все открытые ключи в памяти шифратора уничтожаются.

Несмотря на систему шифрования ключей, ЦГК является опасным объектом, который обращается с наборами незашифрованных ключей всей секретной сети и требует дорогих мероприятий, предотвращающих утечку ключевой информации.

Виртуальное шифрование ключей

Поскольку значения зашифрованных и незашифрованных разовых ключей неотличимы от случайной последовательности, то ключи необязательно сначала генерировать, потом шифровать на мастер-ключе, затем распределять и, затем расшифровывать на местах применения.

Ключи можно не зашифровывать, а, генерируя в ЦГК случайные последовательности, считать, что это уже зашифрованные ключи.

При применении эти условно (виртуально) зашифрованные ключи расшифровываются на мастер-ключе.

ЦГК становится относительно безопасным объектом, поскольку он манипулирует только с (виртуально) зашифрованными ключами. Открытые секретные ключи появляются только внутри шифратора и удаляются после применения.

Ещё одно интересное свойство: при компрометации мастер-ключа и его замены на резервный мастер-ключ во всех шифраторах никакой необходимости в перешифровании виртуально зашифрованных ключей нет. Мы просто кивнём головой и будем считать, что имеющиеся ключи были виртуально зашифрованы новым мастер-ключом.

Расширение ключа

Классическая схема

Процедуры расширения ключа со временем становятся всё более сложными и изощрёнными, они приближаются к собственно алгоритмам криптографических преобразований. Если в «Магме» (старое, до 1989 года, наименование алгоритма ГОСТ 28147-89) в качестве подключей выступают просто части ключа, применяемые в прямом и обратном порядке, то в «Калине» для генерации подключей уже применены примитивы самого алгоритма шифрования. Делается это для защиты от всё более сложных атак, которые в реальности всё менее вероятны, но не исключаются из рассмотрения.

Почему возникает задача расширения ключа? Потому что, как правило, число раундов шифра, умноженное на длину раундового ключа, существенно превышает необходимую длину ключа. Например, заявленная длина ключа – 256 бит, а в каждом из 10 раундов участвует по 128 бит ключевой информации. Итого: суммарная длина подключей равна 1280 бит.

Вторая причина – возможные слабости алгоритма шифрования в случае однократного применения

элементов ключа. Считается, что если элементы ключа применяются несколько раз в различных раундах и взаимодействует с разными элементами состояния шифра, то это – хорошо.

Предварительное расширение

Память в настоящее время – очень доступный ресурс, его можно не экономить. Какая нам разница, что хранить: ключ 256 бит или 1280 бит предварительно сгенерированных подключей.

Тем более, что существуют критические приложения, требующие частой и быстрой смены ключа. В таких приложениях всё более усложняющиеся процедуры расширения ключа могут существенно снижать быстродействие шифра.

Виртуальное расширение

Обращаемся к великому Альтшуллеру за советом: как наиболее эффективно решать задачу расширения ключа. И получаем простой ответ: её не надо решать.

Поскольку идеальное расширение подразумевает независимость подключей друг от друга и генерацию их из ключа, представленного случайной последовательностью, то все свойства совокупности подключей не должны отличаться от свойств случайной последовательности.

Так в чём же дело? Поскольку мы при предварительном расширении ключа отказались транспортировать и хранить сам ключ, а оперируем только с расширенными подключами, то никто не мешает нам не генерировать случайную последовательность, объявлять её ключом и генерировать из неё подключи, а непосредственно генерировать совокупность подключей как случайную последовательность. Назовём такое действие «виртуальным расширением ключа».

Если подключи будут такими же случайными, как и ключ, то мы можем условно говорить: «Да, эффективная длина ключа 256 бит и из него сгенерированы подключи длиной 1280 бит неким виртуальным алгоритмом, который невозможно вскрыть, что делает невозможными атаки на ключ по значениям подключей». Поскольку современные алгоритмы расширения не позволяют по значениям подключей восстановить исходный ключ, то без знания ключа будет невозможно определить, расширены ключи из исходного ключа процедурой расширения или это случайная последовательность – виртуально расширенные подключи.

Полная виртуализация

Теперь последний шаг: генерация виртуальных ключей с их виртуальным расширением.

Если ранее ЦГК формировал наборы ключей в виде случайных последовательностей по 256 бит и, затем, шифровал их на мастер-ключе и рассылал абонентам секретной сети, то теперь наш новый виртуализованный ЦГК формирует наборы виртуально зашифрованных и виртуально расширенных ключей в виде случайных последовательностей по 1280 бит и, не шифруя, рассылает абонентам.

Выгоды полные: ЦГК не работает с секретным представлением ключей, при применении ключей они не требуют расширения и, последнее, – это самые идеальные подключи, поскольку они получены не детерминированной процедурой расширения, а являют собой чистую случайность.

Нарушение закона больших чисел

Оценка вероятностей зависимости

Известна оценка степени близости некоей подстановки к случайной подстановке (СП) по параметрам строгого лавинного критерия (Strict Avalanche Criterion, SAC) [13], т.е. оценка вероятности $k_{SAC}(i,j)$ изменения выходного бита j при изменении входного бита i, определяющая степень зависимости изменения выходов подстановки от изменения её входов (вероятности зависимости).

Известен закон повторного логарифма (вариант закона больших чисел), который, упрощённо говоря, гласит, что отклонение суммы одинаково распределённых независимых случайных величин, делённое на n не превысит величины $\sqrt{(2\ln\ln n)/n}$.

Оценка вероятностей зависимости для истинной СП стремится к значению 0.5 по мере возрастания

числа опытов n. Однако, за счёт конечности СП, рано или поздно, скрытые закономерности начнут проявляться в виде смещения оценок вероятностей зависимости, нарушающего закон больших чисел. То есть, при некотором n отклонение оценки от 0.5 начнёт превышать значение $\sqrt{(2\ln\ln n)/n}$.

Если мы возьмём некоторую подстановку, пронормируем отклонение оценки от идеального значения к $\sqrt{(2\ln\ln n)/n}$ и построим график зависимости отклонения от n, то мы получим некие кривые не выходящие за пределы двух параллельных границ со значениями ± 1.

Метёлка оценок

На рисунке 1 приведён график оценок $k_{SAC}(i,j)$ для «чистой» СП 16 × 16 бит (ширина $B = 16$). На графике чёрные кривые представляют 32 оценки $k_{SAC}(i,j) - 0.5$ для всех i и всех j, нормированные к $\sqrt{(2\ln\ln n)/n}$. Соответствующие границы ± 1 обозначены зелёными линиями, а красные отметки показывают значения n в логарифмическом масштабе.

Как было определено экспериментально, при $n > 2^B$ оценки начинают расходиться и пересекают границу $\sqrt{(2\ln\ln n)/n}$. Получается некая «метёлка» оценок с быстро расширяющимся хвостом. Граница $n = 2^B$ (вертикальная зелёная линия) условно названа «границей неотличимости» подстановки: если наша подстановка даёт оценки, не выходящие за границы повторного логарифма и доходит до границы неотличимости, то мы считаем, что отличия подстановки от СП не обнаружены.

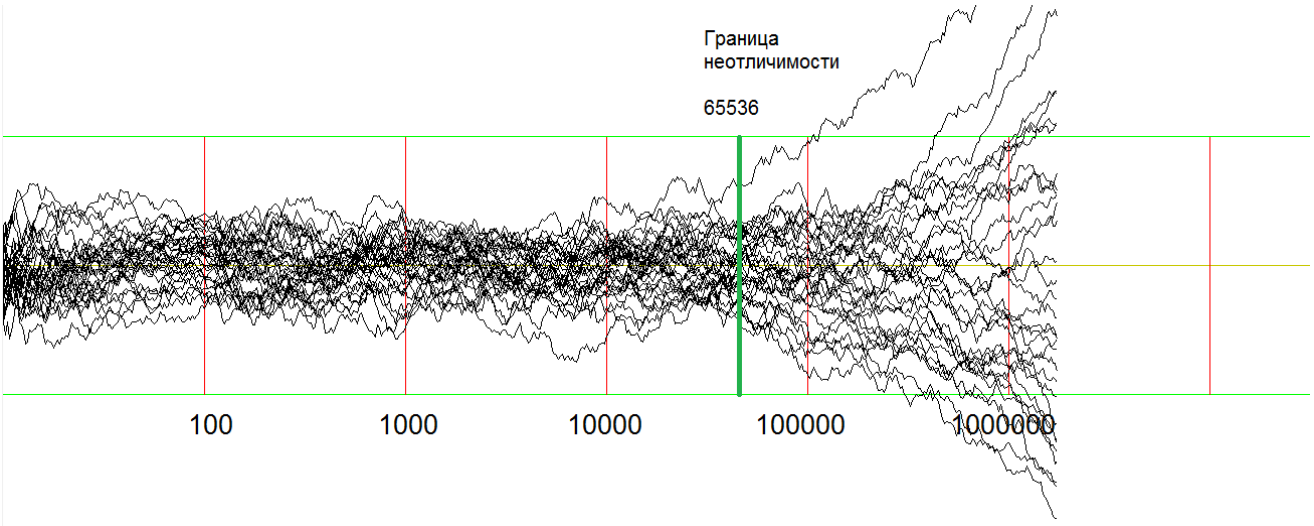


Рисунок 1 – Нормированные оценки $k_{SAC}(i,j)$ для «чистой» случайной подстановки 16 × 16

Декомпозиция случайной подстановки

Люби-Ракофф

Случайная подстановка достаточной ширины считается идеалом блочного шифра. Естественно она, например, даже для $B = 128$ никак не реализуема на практике. Случайную подстановку с сохранением криптографической стойкости можно заменить «сильной псевдослучайной перестановкой» (СПП) по Люби-Ракоффу [10], которая представляет собой 4-х раундовую сеть Фейстеля, выполняющую операции с левой и правой частями входного слова P длины $B/2$. В качестве криптостойких раундовых функций такой сети могут выступать 4 различные СП в два раза меньшей ширины $B/2$ по сравнению с исходной B. На рисунке 2*а* приведена схема такой перестановки.

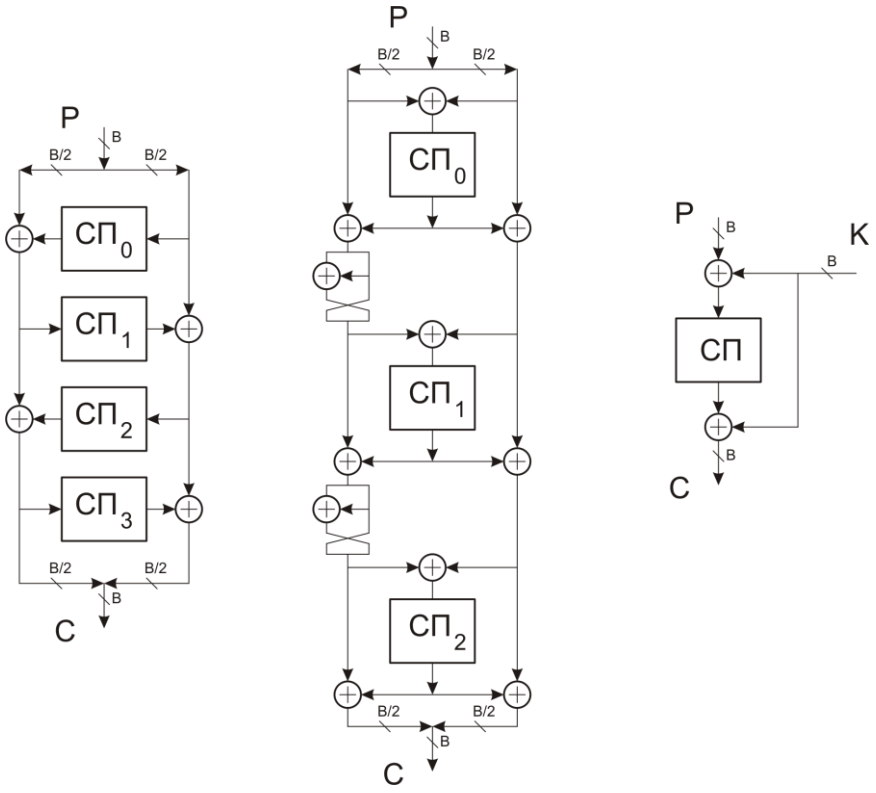


Рисунок 2 – Структуры Люби-Ракофф (*a*), Лей-Месси (*b*) и Ивен-Мансур (*c*)

Сильная псевдослучайная перестановка доказательно стойка к атакам с (адаптивно) подобранными входными и выходными текстами (практически все известные атаки сводятся к таковым).

Но зачем нам останавливаться? Каждую из этих 4-х СП мы тоже можем представить сильной псевдостучайной перестановкой с раундовыми функциями шириной уже *B*/4. Продолжая дальше, мы будем заменять каждую СП верхнего уровня 4-мя СП нижнего уровня с половинной шириной.

И когда мы увидим, что СП шириной $B = 128$ заменены на 256 СП шириной $B = 8$, то мы без трудностей можем сформировать 256 истинно случайных подстановок 8 × 8 бит (256 байт). Общий размер памяти таблиц составит всего 65536 байт.

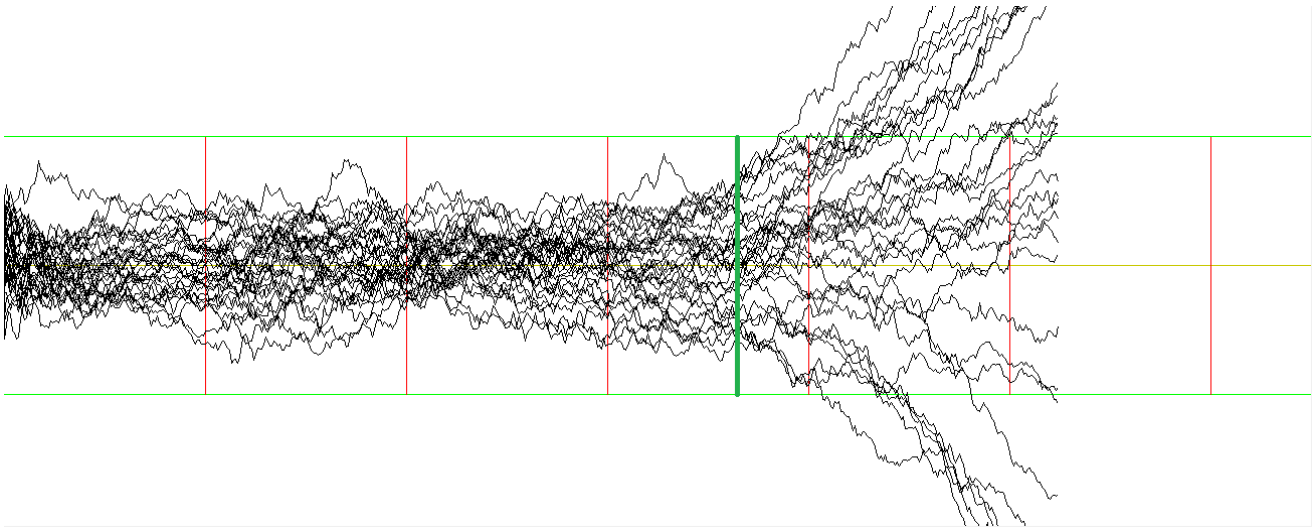
Можем остановиться на 64-х таблицах СП шириной 16 × 16 бит, правда тогда потребуется больше памяти – 8 Мбайт.

Лей-Месси

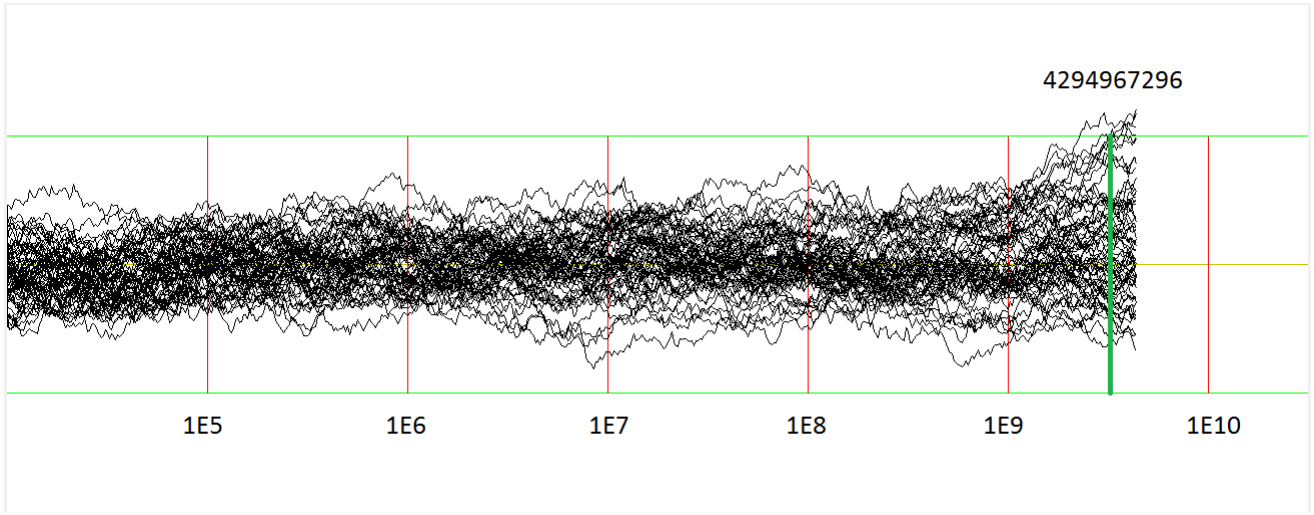
Существует структура сильной псевдослучайной перестановки Лей-Месси [11], приведённая на рисунке 2*b*, с такими же свойствами, как у схемы Люби-Ракофф, но содержащая только 3 различные криптостойкие раундовые функции. При $B = 128$ потребуется 81 истинно случайная подстановка 8 × 8 бит с общим размером памяти таблиц 20736 байт или 27 подстановок 16 × 16 бит с памятью около 3.5 Мбайт.

На рисунке 3*a* приведены графики оценок вероятностей зависимости для рекурсивной декомпозиции СП в виде структуры Люби-Ракофф из 4-х СП 8 × 8, каждая из которых представлена 4-мя СП 4 × 4 (всего 16 подстановок 4 × 4). Как видно, декомпозицию по заданному критерию можно считать неотличимой от СП.

На рисунке 3*b* показана аналогичная «метёлка» для декомпозиции СП 32 × 32 с 3-мя ступенями декомпозиции и 64-мя терминальными СП 4 × 4. Здесь граница неотличимости составляет уже около $2^{32} = 4294967296$.



a



b

Рисунок 3 – Декомпозиция СП 16 × 16 на основе 16-ти СП 4 × 4 (*a*) и СП 32 × 32 на основе 64-х СП 4 × 4 (*b*)

Здесь может возникнуть возражение в том смысле, что вложение СПП друг в друга даёт новую конструкцию, стойкость которой может быть неочевидной. Этот вопрос требует изучения.

Ивен-Мансур

Но если стойкость при вложении СПП сохраняется, что подтверждается многократной экспериментальной оценкой вероятностей зависимости, то видится путь построения большой псевдослучайной перестановки для шифра Ивен-Мансур [12], вариант которой представлен на рисунке 2*c*.

Этот шифр содержит единственную случайную или псевдослучайную подстановку шириной *B*, вход и выход которой складываются по модулю 2 с ключом K длиной *B* бит.

Подставляем в качестве СП нашу сильную псевдослучайную перестановку, полученную в результате декомпозиции СП требуемой ширины, и шифр с доказанной стойкостью готов!

На рисунке 4 приведена структура такого шифра.

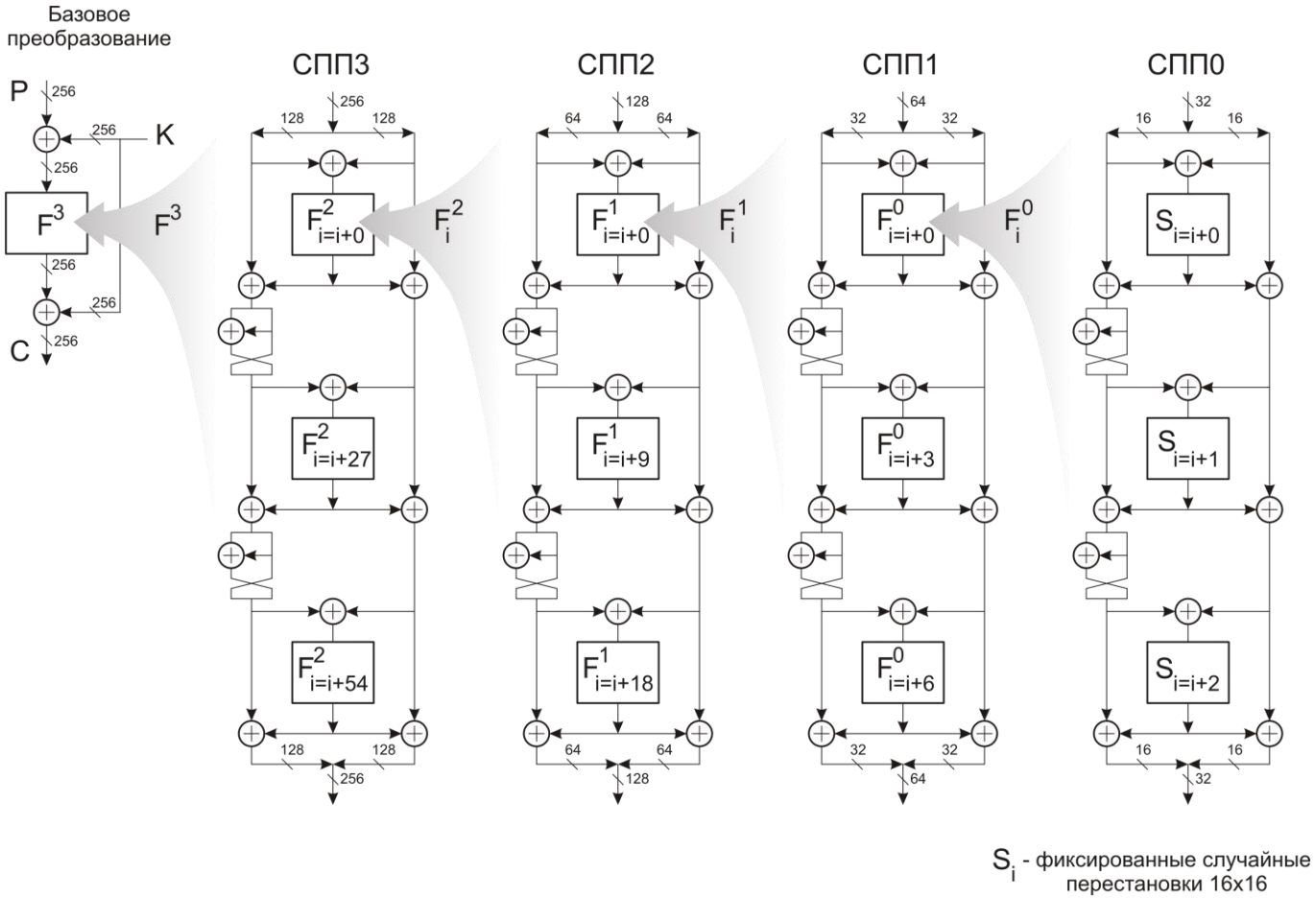


Рисунок 4 – Структура шифра Ивен-Мансура с декомпозицией СПП на основе схем Лей-Месси

Базовое преобразование шифрует блок открытого текста P размером 256 бит на ключе K длиной 256 бит и получает зашифрованный блок C размером 256 бит. Используемая в качестве СП 256 × 256 сильная псевдослучайная перестановка верхнего уровня (F^3) представлена схемой Лей-Месси СПП3 с 3-мя СПП2 (F^2) шириной 128 бит, каждая из которых состоит из 3-х СПП1 (F^1) шириной 64 бита, каждая из которых состоит из 3-х СПП0 (F^0) шириной 32 бита, каждая из которых содержит 3 истинно случайные подстановки (перестановки) S 16 × 16, заданные таблично. Индекс *i* в каждой СПП определяет номера подстановок S, входящих в СПП более верхних уровней. Общий объём таблиц составляет 10.6 Мбайт – довольно много. Можно перейти к следующему уровню вложений СПП: 243 СП 8 × 8 – это уже совсем немного памяти – 62208 байт.

Мы изобрели новый алгоритм шифрования? Нет, мы ничего не изобретали, просто показали, что симметричный блочный шифр необязательно должен быть итеративным и необязательно должен не иметь доказательства стойкости (для красоты всё же дадим этому шифру название – «Kpin» – по аналогии с фрактальной структурой укропа).

Широкие S-блоки

Для ускорения программной реализации шифров строят специальные таблицы, которые совмещают в себе узлы замены и последующие линейные операции перемешивания. Такие таблицы (например, для «Калины-128») представлены в виде 8-ми массивов по 256 64-х разрядных слов и различны для прямого и обратного преобразования. Выходы таблиц в определённом порядке суммируются по модулю 2. Вместо 4-х таблиц подстановки с общим объёмом 1 Кбайт получается два набора по 8 таблиц объёмом 16 Кбайт, зато быстродействие увеличивается в несколько раз.

А если попробовать заполнять таблицы случайными 64-х разрядными числами, а выходы просто складывать по модулю 2? Тогда мы получим необратимое преобразование (но мы необратимости не

боимся), при котором каждый выходной бит будет зависеть от всех входных и каждый входной будет влиять на все выходные. Значения лавинного и строгого лавинного критерия такой подстановки близки к показателям СП.

Умножители

В современном мире появляется ещё один, всё более доступный ресурс, – умножители, которые стали размножаться в связи с возрастающими объёмами задач цифровой обработки сигналов (DSP). И, если полвека назад целый полковник ехал в Зеленоград, чтобы через знакомства и связи «выбить» микросхему умножителя 12x12 по цене мотоцикла, то сейчас чуть ли не каждый микроконтроллер имеет аппаратное ядро DSP с несколькими умножителями. А средняя микросхема FPGA содержит сотни и даже тысячи блоков DSP, в каждом из которых лежит до 4-х умножителей.

Устаревшее представление о том, что умножение – это дорогая операция, укоренило традицию применения в криптографических примитивах простых операций типа табличной замены, сложения и сдвига.

Умножители же – это комбинаторные схемы, в которых массово реализованы таблицы, сложения и сдвиги, то есть, с технической точки зрения умножитель – мощная схема, обеспечивающая и конфузию и диффузию при преобразовании входных бит в выходные. Обычно целочисленный умножитель $n*n$ разрядов имеет выход $2n$ разрядов, типовое значение n – от 8-ми до 32-х.

Диффузионные свойства выходов умножителя неравномерны: наиболее хорошо перемешаны средние разряды выходного слова. Когда-то великий Кнут во втором томе «Искусства программирования» описывал генератор случайных чисел Дж. фон Неймана, называемый «середина квадрата» – число возводилось в квадрат, из него брались средние разряды, сдвигались вправо и снова возводились в квадрат. Мы можем при расширении разрядности после умножения поступать так же.

Теперь вопрос: что же на что умножать, чтобы достичь максимальной диффузии? Наиболее эффективным преобразованием представляется интегральное (типа Фурье), в котором каждый выход образуется взвешенной суммой всех входов. То есть, каждый вход умножается на некий коэффициент, все произведения суммируются и образуют один выход, для следующего выхода все входы суммируются с умножением на другие коэффициенты, и так далее.

Представим вход A нашего преобразования в виде конкатенации M частей A_m по n разрядов каждая (к примеру $M = 4$ и $n = 16$), тогда преобразование $A => B$ будет выглядеть как

$$B_m = \sum_{i=0}^{M-1} k_{m,i} A_i \ ,$$

где k – некие коэффициенты.

Что же использовать в качестве коэффициентов? Вывод напрашивается сам собой: конечно же подключи. Остаётся только вопрос: M входов по n бит порождают M выходов по $2n$ бит, куда девать лишние биты? Мы можем при расширении разрядности после умножения поступать по Дж. фон Нейману – выделять средние n бит. Такая структура кажется сложной, но для блоков DSP это типовая «бабочка» при вычислении БПФ. За счёт усечения старших и младших разрядов наше преобразование стало нелинейным и необратимым, но мы знаем, что это не страшно, зато значения лавинного и строгого лавинного критерия такого преобразования становятся близкими к показателям СП.

Умножители и широкие блоки в раундовых функциях

Построим раундовую функцию на основе вышеописанных структур. На рисунке 5 показан пример такой конструкции из «бабочки» с 16-ю перемножителями 16*16 и 4-мя 32-х разрядными сумматорами по модулю 2^{32} и 8-ю широкими S-блоками со случайными подстановками 8×64, выходы которых объединяются 64-х разрядным сумматором по модулю 2. На вход поступает 64х разрядное слово, разделяется на 4 16-ти разрядных слова, каждое из которых перемножается с 4-мя 16-ти разрядными подключами (всего 16 слов или 256 бит). Произведения в перегруппированном

порядке складываются на 4-х 32-х разрядных сумматорах. Каждая сумма усекается справа и слева на 8 разрядов, образуя 16-ти разрядные слова, конкатенация которых даёт 64-х разрядное слово, поступающее на 8-ми разрядные входы 8-ми S-блоков. 64-х разрядные выходы S-блоков суммируются по модулю 2, образуя выход раундовой функции (назовём её «MUL»).

Чем хороша такая функция? А тем, что она даёт значения лавинного критерия и строгого лавинного критерия, неотличимые от случайной подстановки. То есть это – настоящая криптостойкая раундовая функция.

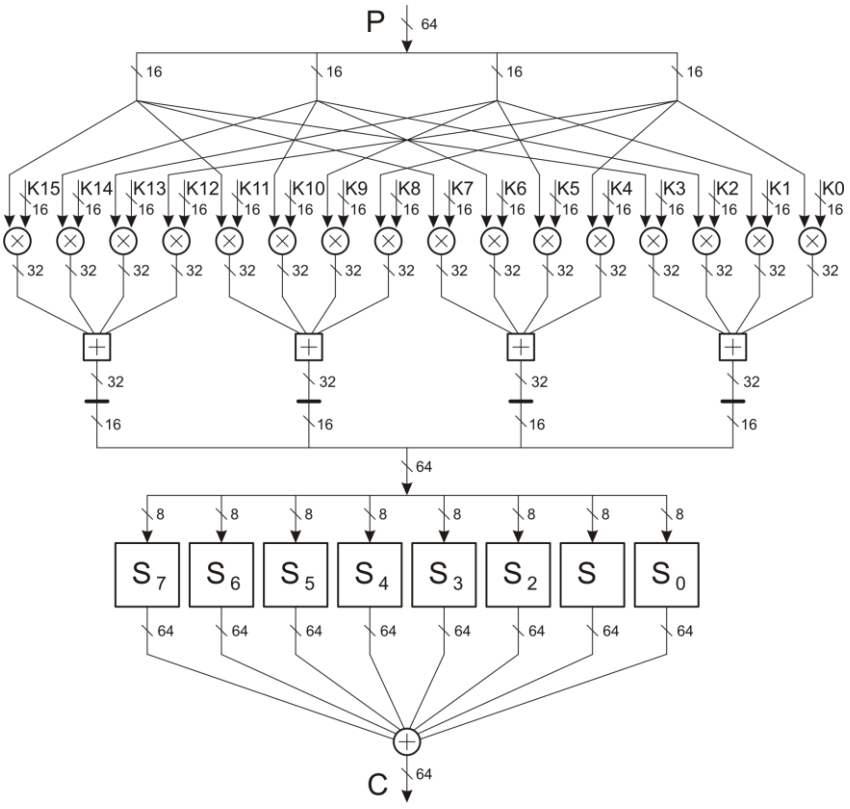


Рисунок 5 – 64-х разрядная раундовая функция MUL с «бабочкой» и широкими S-блоками

Уменьшенная версия раундовой функции miniMUL разбивает 16-ти разрядный вход на 4 слова по 4 бита, вычисляет «бабочку» с 16-ю умножителями на 16 4-х разрядных ключей и 4-мя 8-ми разрядными сумматорами. Средние 4 разряда сумматоров образуют 16-ти разрядный вход для 4-х узлов замены 4 × 16, выходы которых суммируются по модулю 2 и образуют выход функции.

На рисунке 6 приведён график оценок зависимостей для уменьшенной версии раундовой функции. Как видим, miniMUL неотличима от истинной СП 16 × 16.

Исходя из поведения 16-ти разрядной версии miniMUL, вполне логично полагать, что 64-х разрядная версия не хуже. Жаль, что в доквантовую эпоху проверить это затруднительно.

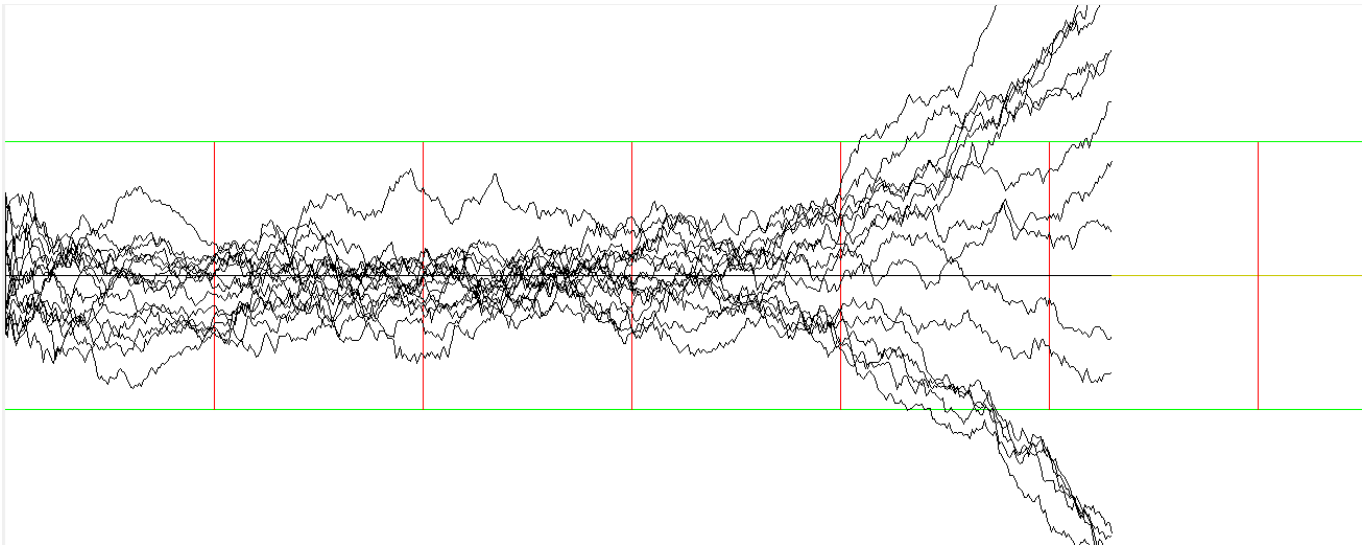


Рисунок 6 – Оценки вероятностей зависимости уменьшенной версии раундовой функции miniMUL

Для сравнения на рисунке 7 приведены графики оценок зависимостей для miniAES [14] после 3-го, 4-го и 5-го раундов. Заметно, что к оценкам истинной СП miniAES приходит довольно медленно.

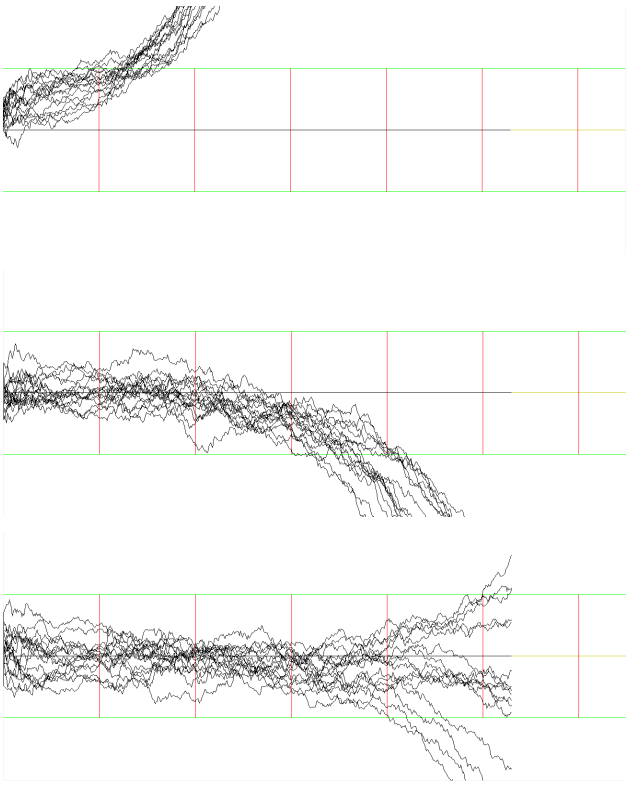


Рисунок 7 – Оценки вероятностей зависимости для 3-го, 4-го и 5-го раундов miniAES

Фейстель и SPN

Автор ни разу не вспомнил сеть SPN, на которой в последнее время строятся шифры, его тянет в сторону Фейстеля и к тому есть основания.

Проведём простейший вычислительный эксперимент: возьмём небольшую сеть Фейстеля и SPN (например миниГОСТ и miniAES) и, перебирая все входные векторы, подсчитаем оценки

вероятности единиц на всех выходах.

У Фейстеля эти вероятности окажутся равными строго 0.5, в то время как у SPN будут несколько отличаться от 0.5. Теперь применим запрещённый приём: создадим дефект в S-блоках – сместим вероятности в сторону нулей на 10 % (блоки потеряют свои свойства и станут необратимыми).

Снова подсчитаем оценки и увидим, что у Фейстеля они снова равны 0.5, а SPN отклонились в сторону нулей, почти все стали менее 0.5.

Таким образом, структура SPN склонна пропускать на выход внутренние дефекты, в то время, как сеть Фейстеля эти дефекты сглаживает и притягивает вероятности выходов к 0.5. Я бы поосторожнее вёл себя с SPN.

Более того, например, ни AES, ни «Кузнечик», как алгебраические шифры не содержат в структуре ни капли случайности. По глубокому убеждению автора приблизить шифр к идеальной СП может только обилие случайности, заключённой в S-блоках, различных для разных раундов. Если речь идёт о высокой степени секретности, то экономить ресурс ни к чему. Как же трудно, применив одну и ту же детерминированную подстановку сотни раз для шифрования одного блока, доказывать стойкость шифра к атакам, которые как раз и используют такое свойство многократного повторения операции от раунда к раунду.

А теперь шифр!

Криптография, во всяком случае, для симметричных блочных шифров, – наука больше инженерная, чем строго математическая, математика притягивается несколько насильно. Ведь общая картина за последние полвека такова: сонмища разработчиков шифров соревнуются, эмпирически создавая схемы из простейших примитивов, взятых на вооружение с лёгкой руки Клода Шеннона. Сложение с ключом, конфузия и диффузия, повторяемые неизменно и многократно в надежде, что результат будет очень стойким. Все пытаются сделать шифр простым, быстрым и стойким. Атакующая сторона тоже изощряется в изобретении атак на шифры. Атаки всё дальше удаляются от реальности, но позволяют сравнивать стойкость различных шифров.

Строгого доказательства стойкости пока ни один из известных блочных шифров не имеет. Стойкость только оценивается по устойчивости ко всем известным атакам и по числу лет существования шифра без компрометации.

Раз так, то мы тоже хотим потренироваться.

Возьмём полную 64-х разрядную функцию MUL, считая её криптостойкой раундовой функцией, подставим её в 128-ми разрядную структуру Люби-Ракофф или Лей-Месси и получаем шифр, тоже MUL, к тому же с доказуемой стойкостью. Поскольку сказано, что криптостойкие раундовые функции в сильной псевдослучайной перестановке должны быть различными, то применим функции с различными таблицами и, естественно, с различными подключами. Всего получается для структуры Лей-Месси необходимо 48 Кбайт таблиц и 768 бит (48 слов по 16 бит) подключей. При конвейерной реализации шифра на FPGA потребуется 48 умножителей 16*16, 24 блока памяти по 256 64-х разрядных слов (всего 48 Кбайт), 12 4-х входовых 32-х разрядных сумматоров и 11 сумматоров по модулю 2 (9 на 64 бита и 2 на 32 бита). Инициализация конвейера будет занимать (с учётом шифрования оказии) порядка 60 тактов, затем по одному такту на шифрование одного блока; при типовой тактовой частоте 200 МГц скорость шифрования составляет свыше 25 Гбит/с или 3.2 Гбайт/с. Хороший получился шифр MUL!

Википедия: «Преимущества мула заключаются в большой выносливости, силе и продолжительности жизни до 40 лет».

Заключение

Подведём краткий итог всему изложенному:

– Вместо проверки случайных последовательностей на случайность, требуется подтверждение гипотезы об истинно случайном генераторе и, если гипотеза подтверждается, то фильтровать и отбрасывать случайные последовательности – грубая ошибка.

- «Слабые ключи» отыскивать и отбрасывать не обязательно, законы статистики нас спасут.
- Брюс Шнайер рекомендует к применению единственный режим шифрования – гаммирование, тем более, что при определённых условиях он может заменить ряд других режимов и может не использовать вектор инициализации.
- Процедуру архивации желательно сделать обязательной и автоматической при шифровании гаммированием.
- В системе с шифрованием ключей их можно не зашифровывать, а, генерируя в Центре генерации случайные последовательности, считать, что это уже зашифрованные ключи. Такое виртуальное шифрование ключей снижает требования к безопасности Центра.
- Более того, вместо ключей можно генерировать виртуально расширенные подключи как случайные последовательности и считать, что это уже зашифрованные подключи.
- Закон повторного логарифма (вариант закона больших чисел) нарушается для любой подстановки, но не ранее чем при $n > 2^n$ для СП порядка n. Если подстановка удовлетворяет этому условию, значит с высокой вероятностью она неотличима от СП.
- Декомпозиция случайной подстановки при помощи сильной псевдослучайной перестановки, состоящей из случайных подстановок вдвое меньшей ширины, позволяет построить приближение к СП большого размера некоторым числом малых СП, реализуемых в виде таблиц.
- Такую большую сильную псевдослучайную перестановку можно вставить в шифр Ивен-Мансур, имеющий доказательство стойкости (шифр «Kpin»).
- «Широкие» S-блоки, заполненные случайными числами, выходы которых складываются по модулю 2 дают необратимое преобразование, при котором каждый выходной бит зависит от всех входных и каждый входной – влияет на все выходные. Значения лавинного и строгого лавинного критерия такой подстановки близки к показателям СП.
- Умножитель – мощная схема, обеспечивающая и конфузию и диффузию при преобразовании входных бит в выходные. Наиболее эффективным преобразованием представляется известная «бабочка» БПФ, в котором каждый выход образуется суммой всех входов, взвешенных подключами. Значения лавинного и строгого лавинного критерия такого преобразования близки к показателям СП.
- Раундовая функция MUL, построенная из «бабочки» и широких S-блоков даёт значения лавинного критерия и строгого лавинного критерия, неотличимые от случайной подстановки. То есть, это – настоящая криптостойкая раундовая функция. Уменьшенная версия раундовой функции по оценкам вероятностей зависимости не отличается от СП соответствующей ширины.
- С сетями SPN не всё хорошо, как кажется. Они транслируют на выходы возможные дефекты распределения вероятностей в S-блоках.
- Замечательный шифр получается трёхкратным применением раундовой функции MUL в сети Лей-Месси, имеющей доказательную стойкость к атакам с подобранными входными и выходными текстами. При конвейерной аппаратной реализации быстродействие шифра превышает 25 Гбит/с.

Конечно, многие соображения (если не все) из приведённых выше, могут представляться, мягко говоря, спорными. Да и ледокол Альтшуллера, если и получил авторское свидетельство на изобретение, то не нашёл практического применения из-за неточной постановки задачи: ведь ледоколу не требуется разрезать тонким лезвием лёд, а требуется своим весом его давить, образуя водный путь для идущего вслед за ним каравана судов.

Однако, пара добрых слов, типа: «Возможно, в этом что-то есть, надо подумать...» уже была бы достаточной наградой за старания автора.

Киев, 2018 г.

Литература

1. К. Шеннон «Работы по теории информации и кибернетике», М., ИЛ, 1963, с. 333-369

2. Брюс Шнайер. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. - М.: "Триумф", 2002 - 816 с.: ил. Тир. 3000 экз., ISBN 5-89392-055-4.

4. Вентцель Е.С. Теория вероятностей: Учеб. для вузов. – 6-е изд. стер. – М.: Высш. шк., 1999.– 576 с.

5. Коряков И.В. Проектирование генератора истинно случайных чисел для криптографических приложений. ООО «НВФ Криптон». Ресурс: http://www.crypton.ua/images/noiser_2012.pdf

6. Neal Stephenson. Cryptonomicon I-II. Avon Books, 1999.

7. Исследование криптографических свойств нелинейных узлов замены уменьшенных версий некоторых шифров, Долгов В.И., Кузнецов А.А., Лисицкая И.В., Сергиенко Р.В., Олешко О.И., Прикладная радиоэлектроника, 2009, Том 8, № 3.

8. Фергюсон, Нильс, Шнайер, Брюс. Практическая криптография. : Пер. с англ. – М. : Издательский дом “Вильямс”, 2004. – 432 с.

9. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Special Publication 800–22. Revision 1a. National Institute of Standards and Technology Gaithersburg, MD 0899–8930. Revised: April 2010.

10. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing, vol. 17, no. 2, pp. 373–386, April 1988.

11. X. Lai and J. L. Massey. A proposal for a new block encryption standard. Advances in Cryptology - EUROCRYPT’90, LNCS 473, pp. 389–404, Springer-Verlag, 1991.

12. Even, S., Mansour, Y.: A Construction of a Cipher From a Single Pseudorandom Permutation. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.): ASIACRYPT 1991. LNCS, vol. 739. Springer, Heidelberg (1993), pp. 210–224.

13. Pascale, S. The degrees of completeness, of avalanche effect, and of strict avalanche criterion for MARS, RC6, Rijndael, Serpent, and Twofish with reduced number of rounds. [Text] / S. Pascale // Siemens AG, ZT IK 3. April 3, 2000.

14. Raphael Chung-Wei Phan. Mini Advanced Encryption Standard (Mini-AES): A testbed for Cryptanalysis Students / Raphael Chung-Wei Phan // Cryptologia. – October 2002. – XXVI(4). – P. 283-306.