ЛЕКЦИЯ 4

ГЕНЕРАТОРЫ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДАТЕЛЬНОСТЕЙ. ПОТОКОВЫЕ ШИФРЫ

На прошлой лекции были рассмотрены блочные шифры, которые применяются сразу к блокам данных размером от 64 бит и выше. Им присуща надежность и относительная быстрота: с их помощью них можно шифровать информацию со скоростью порядка десяти мегабайт в секунду.

Но помимо упомянутых качеств всегда остаётся актуальным вопрос надежности. Чем меньше элементов в схеме, тем надёжнее эта схема. Обычно стремятся создать такие шифры, которые являются быстрыми, которые просто реализовать, используя самую примитивную радиотехническую элементную базу, но которые тем не менее являются достаточно надёжными.

1. Генераторы случайных последательностей

Генерация случайных чисел имеет широкое применение — от игр (например, «Тетрис») до криптографии (криптографические протоколы, генерация сеансовых ключей).

В пример можно привести шифр блокнота: система с таким шифром абсолютно надёжна, пока собеседники используют ключ один раз. Чем дольше он используется, тем менее надёжным он становится. Вместо того, чтобы использовать общий секретный ключ, которые знают оба собеседника, можно генерировать с его помощью «сеансовый» ключ и периодически его менять. Чтобы сгенерировать надёжный сеансовый ключ, который злоумышленник не может предсказать, нужен генератор случайных чисел.

Генератор случайных чисел — это алгоритм (или физический процесс), который воспроизводит случайные числа. Этот процесс стараются сделать качественным, быстрым и дешёвым. Под качеством понимают способность алгоритма генерировать случайные числа с одинаковой вероятностью. Можно сказать, что если генератор случайных

чисел является источником случайной величины X, то энтропия этой случайной величины должна быть максимальна, а это возможно, только если значения каждого события равновероятны.

Выполнение требования по скорости означает, что в секунду выдается достаточное количество событий, чтобы зашифровать нужный поток данных. Например, если речь идёт о шифровании гигабита данных в секунду, то генератор случайных чисел должен давать миллиарды случайных событий в секунду. Никакая механическая машина, подбрасывающая монеты и записывающая результаты, не подходит для такой цели.

Можно предложить другие способы генерации истинно случайных событий. Например, в качестве случайного процесса можно взять шум, записываемый со входа микрофона. Тепловой шум тоже можно рассматривать как случайный процесс. Но эти процессы дают слишком малое количество данных. Если же попробовать использовать радиоактивный распад, то это даст порядка десяти событий в секунду (если использовать миллионы, то находиться рядом с таким генератором будет опасно). Также для вышеперечисленных способов не будет выполняться правило дешевизны.

В настоящее время для целей криптографии пока что нет дешёвого, быстрого и качественного генератора. Поэтому приходится использовать так называемые **псевдослучайные генераторы** (генераторы псевдослучайной последовательности). Они отличаются от генератора случайной последовательности тем, что являются алгоритмом, то есть можно записать математический закон получения следующего случайного числа на основании предыдущего случайного числа и некоторого состояния вычислительного комплекса.

Все существующие алгоритмы генерации псевдослучайных последовательностей можно реализовать на машине Тьюринга, то есть это полноценные алгоритмы. Они детерминированы, то есть ничего случайного в этих алгоритмах нет. Тем не менее выход этих алгоритмов обладает свойствами случайной последовательности:

1. **Примерное совпадение количества сгенерированных чисел** разных типов (нулей и единиц).

2. **Наличие конечного числа состояний.** Любой алгоритм использует определенный набор памяти. В этом объёме памяти находится его состояние. Количество состояний конечно, потому что объём используемой памяти конечен. Если он будет бесконечен, то это будет плохой алгоритм, потому что его нельзя будет использовать в постоянном режиме.

3. **Наличие периода**, что следует из принципа ящиков Дирихле, суть которого состоит в следующем: если имеется набор из 9-ти ящиков, и там сидят 10 кроликов, то хотя бы в одном ящике сидят 2 кролика. Значит, если внутреннее состояние состоит из 20-ти бит, то в это внутреннее состояние можно поместить максимум 2^{20} бит.

2. Линейный конгруэтный генератор

Это генератор, который описывается следующей формулой:

$$X_{n+1} = (aX_n + b) \mod m,$$

где X_n — состояние генератора.

При этом в качестве выхода генератора можно использовать как число X, так и какой-то набор битов этого числа. Оказывается, что это довольно хороший со статистической точки зрения генератор.

Этот генератор используется в большом количестве приложений. Период этого генератора можно сделать максимальным, а именно $2^m - 1$. Для этого необходимо выполнить следующие правила:

1. Число b должно быть взаимно простым с m.

2. Число a должно быть кратным всем простым делителям числа m.

3. $(a-1)$ должно быть кратно четырём, если $(m-1)$ кратно четырём.

В качестве числа m можно взять 2 в некоторой степени, например, 2^{32} или 2^{64}. И, таким образом, вообще избавиться от операции взятия по модулю. Вместо этого будет выполняться умножение с отбрасыванием переполнения и сложение с отбрасыванием переполнения.

Линейный конгруэтный генератор используется во многих языках программирования: как в старых, так и в недавних (например, Java, C/C++). Иногда используются генераторы с разными коэффициентами.

Этот генератор довольно хорош для целого спектра приложений, но не для криптографии. По четырём числам, т. е. по четырём выходам, этого генератора можно восстановить все коэффициенты: a, b и m.

3. Статистические свойства генераторов

Выше рассматривалось свойство алгоритмов генерировать числа с одинаковой вероятностью. Если брать в качестве выхода нули и единицы, то количество этих нулей и единиц должно быть примерно одинаково.

Есть и куда более сложные характеристики, например: будет ли хорошим генератор, указанный ниже?

0101010101...

Число нулей и единиц у него одинаково. Но очевидно, что никакой случайности данный генератор не несёт.

Нужно добавлять какие-нибудь условия. Например, можно сказать, что должно быть одинаково не только число нулей и единиц, но и более крупных блоков. Например, рассматривать по 2 бита и проверять, чтобы количество разных блоков 00, 01, 10 и 11 было примерно одинаково.

Также существует большое количество других тестов. Например, можно брать последовательность и рассматривать в ней последовательные 8 бит как некие координаты. Отложив эти координаты, проверим, сколько точек вошло в единичную окружность, а сколько не вошло. Очевидно, что для хорошего генератора можно посчитать площадь круга. Нужно поделить площадь квадрата на площадь круга и получить нужное соотношение.

Таких тестов существует сотни, и некоторые из этих тестов были отобраны, в том

числе Национальным Институтом Стандартизации и Технологии, то есть тем же самым институтом, который организовал в своё время конкурс AES (Advanced Encryption Standard), в большой массив тестов НИСТ для генераторов случайных чисел. Этот массив тестов проверяет, что генератор случайной последовательности обладает хорошей статистикой.

Но это ничего не говорит о применимости данного генератора для целей криптографии. Это говорит лишь о том, что выход генератора похож на выход случайной последовательности. В криптографии нужен не просто генератор, который обладает случайностью или похож на случайный, а такой генератор, чтобы криптоаналитик не мог предсказать его выход, основываясь на всей возможной информации. То есть даже если злоумышленник перехватил большое количество информации о предыдущем выходе генератора, ему должно быть сложно предсказать, что будет следующим выходом.

Было показано, что следующее определение соответствует понятию **надёжного с криптографической точки зрения генератора**: генератор считается надёжным, если по k бит перехваченной информации никаким полиноминальным по времени и памяти тестам нельзя предсказать, какой будет следующий бит с вероятностью больше, чем в 50%.

Также было показано, что надёжные генераторы случайных чисел можно построить в том случае, если можно построить надёжные однонаправленные функции, которые нельзя инвертировать за полиномиальное время.

На предыдущей лекции были рассмотрены режимы связи, в том числе два особенных режима связи: режим связи по счётчику и Output Feedback Mode, когда результат шифрования используется как вход шифрования для следующего блока.

Вектор инициализации шифруется неким ключом, потом результат шифрования передаётся на следующий этап шифрования. Тем же самым ключом шифруется и второй вектор инициализации, и так далее. В этом состоит суть Output Feedback Mode.

Особенность этого режима состоит в том, что можно отделить процедуру создания γ от сложения с открытым текстом.

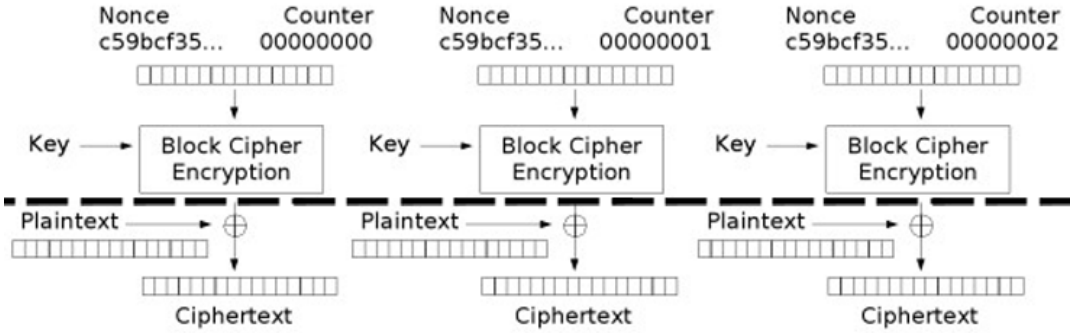

Рис. 4.1

Имеется некая процедура, которая формирует γ, и потом γ складывается с текстом как в этом режиме, так и в режиме счётчика. И в этом смысле функцию шифрования можно рассматривать как генератор псевдослучайной последовательности. Очевидно, что если функция шифрования является надёжной, то подобный генератор будет являться надёжным с криптографической точки зрения. Потому что, даже если зло-

умышленник знает предыдущий выход, но не знает ключ, предсказать следующий выход у него не получится. Это касается и режима счётчика, и режима OFB.

У этого генератора есть проблема: он медленный. На рисунке один квадратик соответствует одной ячейке Фейстеля или 14-ти раундам шифра AES, которые являются нетривиальными математическими операциями.

Генерация γ считается быстрой, когда на генерацию одного бита уходит максимум 10 тактов процессора.

4. Регистр сдвига с обратной связью

Рассмотрим генераторы, которые основаны на регистре сдвига с обратной связью. Вычисление случайных чисел с помощью такого генератора можно представить в виде операции в поле Галуа. Когда есть неприводимый многочлен, описываемый связями этого генератора, то можно говорить о наличии такой связи, если есть единица в соответствующем многочлене над полем GF(2^m), где m — количество ячеек регистра с обратной связью.

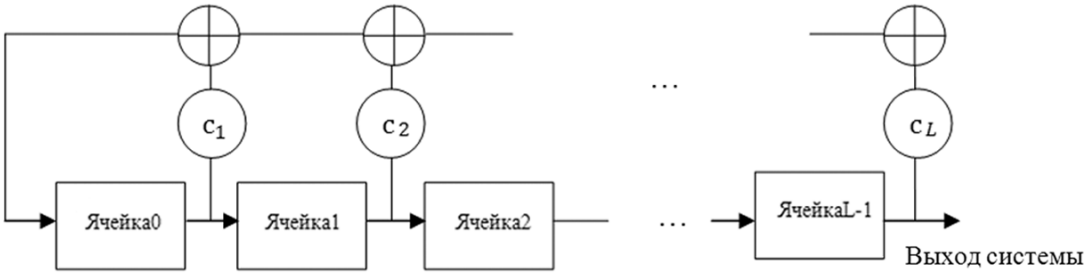

Рис. 4.2

Если характеристический многочлен данного регистра является примитивным, а m является чётным, тогда данный регистр имеет максимальный период, который равен (2^m–1), то есть все возможные значения регистра, за исключением нулевого, потому что если там будут нули, то это тривиальное значение просто зацикливается само на себя.

Однако с точки зрения криптографии данный генератор является совершенно ненадёжным. Для того чтобы восстановить внутреннее состояние многочлена и все его связи, достаточно 2^m бит на выходе чтобы восстанавить и многочлен, и всю схему. Поэтому в тривиальном виде, когда есть только регистр сдвига, данный способ не используется. Тем не менее, регистр сдвига с линейной обратной связью можно объединять в сложные конструкции. Например, использовать некоторые нелинейные комбинации. Есть множество регистров, выходы из которых каким-то образом комбинируются по каким-то нелинейным правилам (т. е. когда нельзя представить 3 регистра как один большой регистр). И в этом случае максимальный период увеличивается, и вместе с этим растёт криптографическая сложность взлома данного регистра сдвига, то есть подобные комбинации уже можно начинать использовать как криптографически стойкие генераторы.

Могут быть более сложные варианты.

В качестве двух регистров сдвига можно взять один большой регистр сдвига, состоящий из более мелких. Вход вспомогательных регистров будет зависеть от предыдущего.

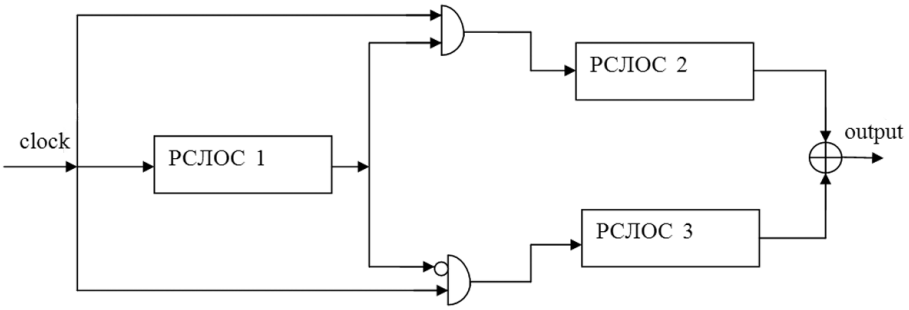

Рис. 4.3

Можно проделать следующие модификации: заставить вход зависеть от предыдущего; можно вот два регистра сдвига тактировать, т. е. осуществлять сдвиг только в том случае, если у одного регистра на выходе «единица». Также можно в этом случае сдвигать один из регистров, а другой — нет. На подобном подходе построен **шифр A5**.

5. Шифр A5

Он используется в современной сотовой связи в Европе и США. Шифр A5 состоит из трёх регистров сдвига.

Во многих случаях поточный шифр — это криптографически стойкий генератор псевдослучайной последовательности (КСГПСЧ).

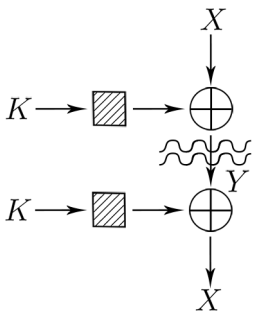

Рис. 4.4

Генератор можно представить как некий чёрный ящик, который на выходе даёт γ, т. е. набор нулей и единиц. Отправитель берёт открытый текст и складывает его по модулю 2 с γ, которую дал чёрный ящик, причём в самом начале этот чёрный ящик инициализируется ключом. После того, как шифротекст проходит по каналу связи, получатель использует тот же самый ключ для задания того же самого состояния чёрного ящика и получает на выходе чёрного ящика тот же набор γ, который можно сложить по модулю 2 и получить открытый текст.

Причём, если в результате радиопередачи часть битов портится, то на выходе будут испорчены только эти биты. Выходит, если инициализировать ключом надёжный генератор псевдослучайной последовательности, то получится надёжный поточный шифр.

При рассмотрении поточных шифров анализируют способность чёрного ящика генерировать случайные числа. Для этого берут этот поточный шифр, тестриуют выход

этого шифра в тестах НИСТ и смотрят, верно ли, что γ является случайной. Если там обнаруживаются какие-то статистические отклонения, то поточный шифр считается плохим. Далее пытаются провести какой-то анализ шифра: смотрят насколько он сложный точки зрения линейного криптоанализа, дифференциального криптоанализа и т. д.

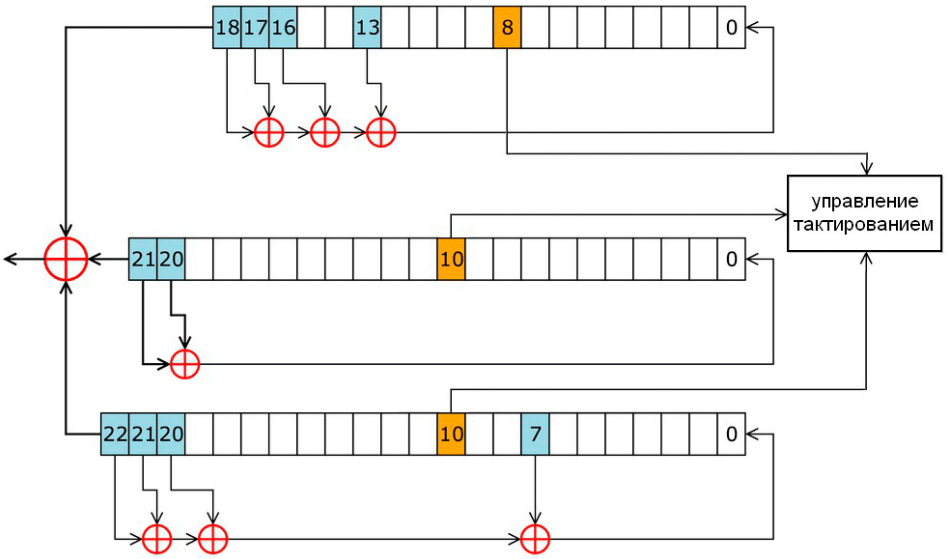

Рис. 4.5

Генератор A5 состоит из трёх регистров сдвига с обратной связью, причём разной длины. Предположительно, длина периода равна перемножению периодов каждого из трёх генераторов, причём здесь используется нетривиальная схема тактирования. Выделяется 3 бита, а затем вычисляется мажоритарная функция, и дальше сдвигаются те регистры, у которых данный бит равен значению мажоритарной функции (мажоритарная функция даёт на выходе 0, если в качестве её аргументов выступают нули; если это не так, то на выходе 1). То есть сдвигаются те регистры, у которых в данных ячейках совпадающие биты.

Оказалось, что у такой сложной структуры реально в среднем число периодов составляет 2^{23} (не очень большой период для такой сложной схемы). Более того, к данному шрифту возникали претензии: по поводу генерации пароля, по поводу инициализации и т. д. Сложность атаки для данного шифра составляет 2^{40}. Это означает, что сейчас взломать данный шифр может любой персональный компьютер.

Однако всё это касается шифра A5/1, который используется в Европе и США. В развивающихся странах используется другая модификация — **A5/2**. Специально для усиления стойкости шифра был введён дополнительный регистр, который управляет тактированием. В результате этого усиления реальный период равен 2^{17}. По всем документам сказано, что шифр усилили, но все, кто его разрабатывал, понимали, что ни о каком усилении специально для развивающихся стран речи быть не может.

6. Шифр RC4

Этот шифр является примером надёжного поточного шифра. RC4 в качестве внутреннего состояния использует 256 ячеек по одному байту. Они обозначаются буквой S (State): от S_0 до S_{255}. Кроме того, к состоянию относятся два числа от 0 до 255, условно называемым i и j. При выполнении итерации шифра с помощью этих двух чисел вычисляются следующие два числа i и j:

$$i := i + 1 \mod 256,$$

$$j := j + 1 \mod 256.$$

Потом ячейки с индексами i и j и меняются местами:

$$S_{\mathrm{i}} \leftrightarrow S_{\mathrm{j}}.$$

После этого с помощью значения из этих ячеек вычисляется число t:

$$t = S_{\mathrm{i}} + S_{\mathrm{j}} \mod 256,$$

и выходом поточного шифра является значение в ячейке $K = S_{\mathrm{t}}$. По сравнению с шифром AES, в котором было 14 раундов и операции в полях Галуа, это очень быстрый шифр. Компания RSA, которая разработала этот шифр, утверждает, что он является устойчивым к дифференциальному и линейному криптоанализу, а количество его внутренних состояний, как легко посчитать, это

$$256! \times 256^2 \approx 2^{1700},$$

где 256! — это количество перестановок в этих ячейках; 256^2 — это из-за значений i и j. Это, конечно, меньше, чем период шифра AES или шифра GOST, если использовать их в режиме счётчика. Тем не менее, для поточного шифра это очень неплохой результат. Каких-либо уязвимостей в данном шифре, которые приводят к практической криптографической атаке, в настоящий момент не найдено.

Подводя итог, можно сказать, что поточные шифры, которые основываются на криптографически стойких генераторах псевдослучайной последовательности, представляют собой быстрые алгоритмы, в которых генерация следующего бита составляет всего несколько тактов, в отличие от блочных шифров. Построить надёжный поточный шифр намного сложнее, потому что необходимо помнить о том, чтобы этот шифр было просто реализовать на примитивной радиоэлементной базе, или же чтобы он быстро работал на самых простых и медленных процессорах, включая, например, процессоры от смарт-карт. Тем не менее, надёжные поточные шифры существуют, и RC4 — пример такого шифра.