3.7. Алгоритмы создания случайных чисел

Большинство алгоритмов сетевой безопасности, основанных на криптографии, используют случайные числа. Например:

1. Схемы взаимной аутентификации.
2. Ключ сессии, созданный KDC или кем-либо из участников.

Двумя основными требованиями к последовательности случайных чисел являются *случайность* и *непредсказуемость*.

Шифрующие приложения используют для создания случайных чисел специальные алгоритмы. Эти алгоритмы детерминированы и, следовательно, создают последовательность чисел, которая не является статистически случайной. Тем не менее, если алгоритм хороший, полученная последовательность будет проходить много тестов на случайность. Такие числа часто называют *псевдослучайными числами*.

Генераторы псевдослучайных чисел

Первой широко используемой технологией создания случайного числа был алгоритм, предложенный Лехмером, который известен как метод линейного конгруента. Этот алгоритм параметризуется четырьмя числами следующим образом:

M	Модуль (основание системы)	$m > 0$
A	Множитель	$0 \le a < m$
C	Приращение	$0 \le c < m$
X_0	Начальное значение или зерно (seed)	$0 \le X_0 < m$

Последовательность случайных чисел $\{X_n\}$ получается с помощью следующего итерационного равенства:

1. $X_{n+1} = (a X_n + c) \bmod m$

Если m, a и c являются целыми, то создается последовательность целых чисел в диапазоне $0 \le X_n < m$.

Существует три критерия, используемые при выборе генератора случайных чисел:

1. Функция должна создавать полный период, т.е. все числа между 0 и m до того, как создаваемые числа начнут повторяться.
2. Создаваемая последовательность должна появляться случайно. Последовательность не является случайной, так как она создается детерминированно, но различные статистические тесты, которые могут применяться, должны показывать, что последовательность случайна.
3. Функция должна эффективно реализовываться на 32-битных процессорах.

Значения a, c и m должны быть выбраны таким образом, чтобы эти три критерия выполнялись. В соответствии с первым критерием можно показать, что если m является простым и c = 0, то при определенном значении a период, создаваемый функцией, будет равен m-1. Для 32-битной арифметики

соответствующее простое значение $m = 2^{31} - 1$. Таким образом, функция создания *псевдослучайных чисел* имеет вид:

$X_{n+1} = (a\ X_n) \bmod (2^{31} - 1)$

Сила алгоритма линейного конгруента в том, что если сомножитель и модуль (основание) соответствующим образом подобраны, то результирующая последовательность чисел будет статистически неотличима от последовательности, являющейся случайной из набора 1, 2, ..., m-1. Но не может быть случайности в последовательности, полученной с использованием алгоритма, независимо от выбора начального значения X_0. Если значение выбрано, то оставшиеся числа в последовательности будут предопределены. Это всегда учитывается при криптоанализе.

Если противник знает, что используется алгоритм линейного конгруента, и если известны его параметры ($a = 7^5$, $c = 0$, $m = 2^{31} - 1$), то, если раскрыто одно число, вся последовательность чисел становится известна. Даже если противник знает только, что используется алгоритм линейного конгруента, знания небольшой части последовательности достаточно для определения параметров алгоритма и всех последующих чисел. Предположим, что противник может определить значения X_0, X_1, X_2, X_3. Тогда :

$X_1 = (a\ X_0 + c\ ) \bmod m$

$X_2 = (a\ X_1 + c\ ) \bmod m$

$X_3 = (a\ X_2 + c\ ) \bmod m$

Эти равенства позволяют найти a, c и m.

Таким образом, хотя алгоритм и является хорошим генератором *псевдослучайной последовательности чисел*, желательно, чтобы реально используемая последовательность была непредсказуемой, поскольку в этом случае знание части последовательности не позволит определить будущие ее элементы. Эта цель может быть достигнута несколькими способами. Например, использование внутренних системных часов для модификации потока случайных чисел. Один из способов применения часов состоит в перезапуске последовательности после N чисел, используя текущее значение часов по модулю m в качестве нового начального значения. Другой способ состоит в простом добавлении значения текущего времени к каждому случайному числу по модулю m.

Криптографически созданные случайные числа

В криптографических приложениях целесообразно шифровать получающиеся случайные числа. Чаще всего используется три способа.

Циклическое шифрование

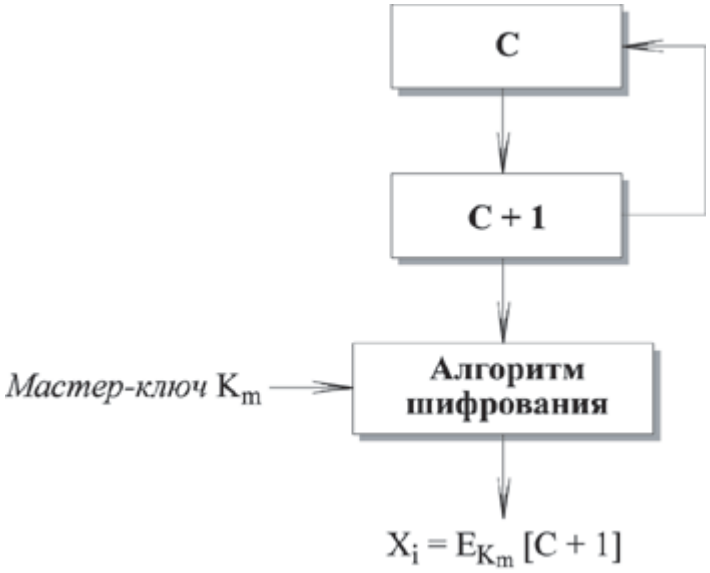

Рисунок 3.12 – Циклическое шифрование

В данном случае применяется способ создания ключа сессии из мастер-ключа. Счетчик с периодом N используется в качестве входа в шифрующее устройство. Например, в случае использования 56-битного ключа DES может применяться счетчик с периодом 2^{56}. После каждого созданного ключа значение счетчика увеличивается на 1. Таким образом, псевдослучайная последовательность, полученная по данной схеме, имеет полный период: каждое выходное значение X_0, X_1,...X_{N-1} основано на различных значениях счетчика и, следовательно, $X_0 \neq X_1 \neq X_{N-1}$. Так как мастер-ключ защищен, легко показать, что любой секретный ключ не зависит от знания одного или более предыдущих секретных ключей.

Режим Output Feedback DES

Режим *OFB* DES может применяться для генерации ключа, аналогично тому, как он используется для потокового шифрования. Заметим, что выходом каждой стадии шифрования является 64-битное значение, из которого только левые j битов подаются обратно для шифрования. 64-битные выходы составляют последовательность *псевдослучайных чисел* с хорошими статистическими свойствами.

Генератор псевдослучайных чисел ANSI X9.17

Один из наиболее сильных генераторов *псевдослучайных чисел* описан в ANSI X9.17. В число приложений, использующих эту технологию, входят приложения финансовой безопасности и PGP.

Алгоритмом шифрования является тройной DES. Генератор ANSI X9.17 состоит из следующих частей:

1. **Вход:** генератором управляют два псевдослучайных входа. Один является 64-битным представлением текущих даты и времени, которые изменяются каждый раз при создании числа. Другой является 64-битным начальным значением; оно инициализируется некоторым

произвольным значением и изменяется в ходе генерации последовательности *псевдослучайных чисел*.

2. **Ключи:** генератор использует три модуля тройного DES. Все три используют одну и ту же пару 56-битных ключей, которая должна держаться в секрете и применяться только для генерации *псевдослучайного числа*.
3. **Выход:** выход состоит из 64-битного *псевдослучайного числа* и 64-битного значения, которое будет использоваться в качестве начального значения при создании следующего числа.

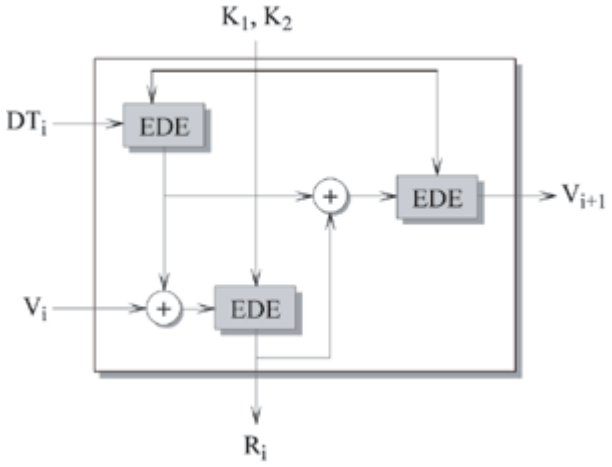

Рисунок 3.13 – Генератор псевдослучайных чисел ANSI X9.17

DT_i - значение даты и времени на начало i-ой стадии генерации.

V_i - начальное значение для i-ой стадии генерации.

R_i - *псевдослучайное число*, созданное на i-ой стадии генерации.

K_1, K_2 - ключи, используемые на каждой стадии.

Тогда:

$R_i = EDE_{K1,K2} [EDE_{K1,K2} [DT_i] \oplus V_i]$

$V_{i+1} = EDE_{K1,K2} [EDE_{K1,K2} [DT_i] R_i]$

Схема включает использование 112-битного ключа и трех EDE-шифрований. На вход подаются два *псевдослучайных значения*: значение даты и времени и начальное значение очередной итерации, на выходе создаются начальное значение для следующей итерации и очередное *псевдослучайное значение*. Даже если *псевдослучайное число* R_i будет скомпрометировано, вычислить V_{i+1} из R_i невозможно, и, следовательно, следующее *псевдослучайное значение* R_{i+1}, так как для получения V_{i+1} дополнительно выполняются три операции EDE.