УГАТУ

Лекция №5

ГЕНЕРАТОРЫ СЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ И ПОТОКОВЫЕ ШИФРЫ

Дисциплина: Криптографическая защита информации
Преподаватель: Миронов Константин Валерьевич
Поток: БПС-3
Учебный год: 2020/21

УГАТУ

Уфимский государственный авиационный технический университет

Содержание лекции

- **Датчики истинно случайных чисел**
- Генераторы псевдослучайных последовательностей
- Генераторы псевдослучайных последовательностей с добавлением истинной случайности
- Потоковые алгоритмы шифрования

УГАТУ

Уфимский государственный авиационный технический университет

Датчики случайных чисел

В лекции рассмотрена задача генерации двоичных случайных данных

? как сгенерировать небинарные случайные данные («число от 0 до 99»)

Вариант 1. Сгенерировать 7 случайных бит (число от 0 до 127) и вычислить значение по модулю 99. **Плохой вариант:** вероятность выпадения 10 вдвое больше, чем вероятность выпадения 90

Вариант 2 (предпочтительный). генерировать 7 случайных бит, пока не выпадет число в промежутке от 0 до 99.

УГАТУ

Уфимский государственный авиационный технический университет

Датчики истинно случайных чисел

- Повторить сгенерированную последовательность невозможно
- Наиболее распространенное применение – генерация ключей
- Недостаток – зачастую невозможно сгенерировать быстро много случайных чисел
- Существуют тесты на случайность последовательности:
  - Наиболее известные наборы – NIST и DIEHARD
  - Если последовательность можно сжать без потери данных – она неслучайна
- Примеры источников случайных данных:
  - Специализированные электронные элементы - шумовые диоды
  - Временные интервалы между нажатиями на клавиатуры (могут быть неслучайными из-за правил обработки нажатий)
  - Пиксели зашумленного изображения, напр. фото лунной поверхности
  - Результаты измерения некоторых физических величин
  - Иррациональные математические константы (π,e,$\sqrt{2}$ и т.д.)
- Из получаемых чисел берутся биты на несколько позиций младше младших значащих

УГАТУ

Уфимский государственный авиационный технический университет

Содержание лекции

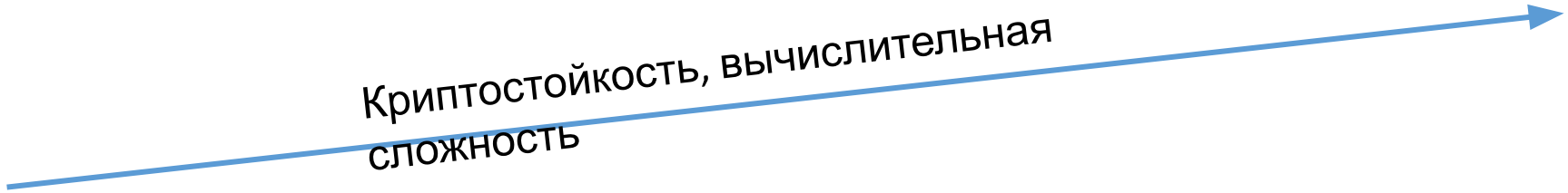
- Датчики истинно случайных чисел
- **Генераторы псевдослучайных последовательностей**
  - **Конгруэнтные генераторы**
  - **Сдвиговые регистры с обратной связью**
  - **Генератор Блюм-Блюм-Шуба**
- Генераторы псевдослучайных последовательностей с добавлением истинной случайности
- Потоковые алгоритмы шифрования

УГАТУ

Уфимский государственный авиационный технический университет

Генераторы ПСП

- Вся последовательность (гамма) определяется некоторым начальным блоком
- Каждый следующий блок вычисляется на основе предыдущего
- Гамма имеет конечную длину – **период**; после достижения периода гамма зацикливается
- Криптографический ГПСП – по сути «хеш-функция наоборот»
  - Аргумент - двоичный код фиксированной длины (секретный ключ); значение - двоичный код произвольной длины
  - По любому фрагменту ПСП нельзя определить ни аргумент, ни остальные части ПСП



Конгруэнтные генераторы	Сдвиговые регистры с обратной связью	Поточные шифры	Блочные шифры в режиме гаммирования	Генератор Блюм-Блюм-Шуба

УГАТУ

Уфимский государственный авиационный технический университет

Линейные конгруэнтные генераторы

Следующий блок данных вычисляется на основе предыдущего по формуле:

$$X_n = (a * X_{n-1} + b) \, mod \, p$$

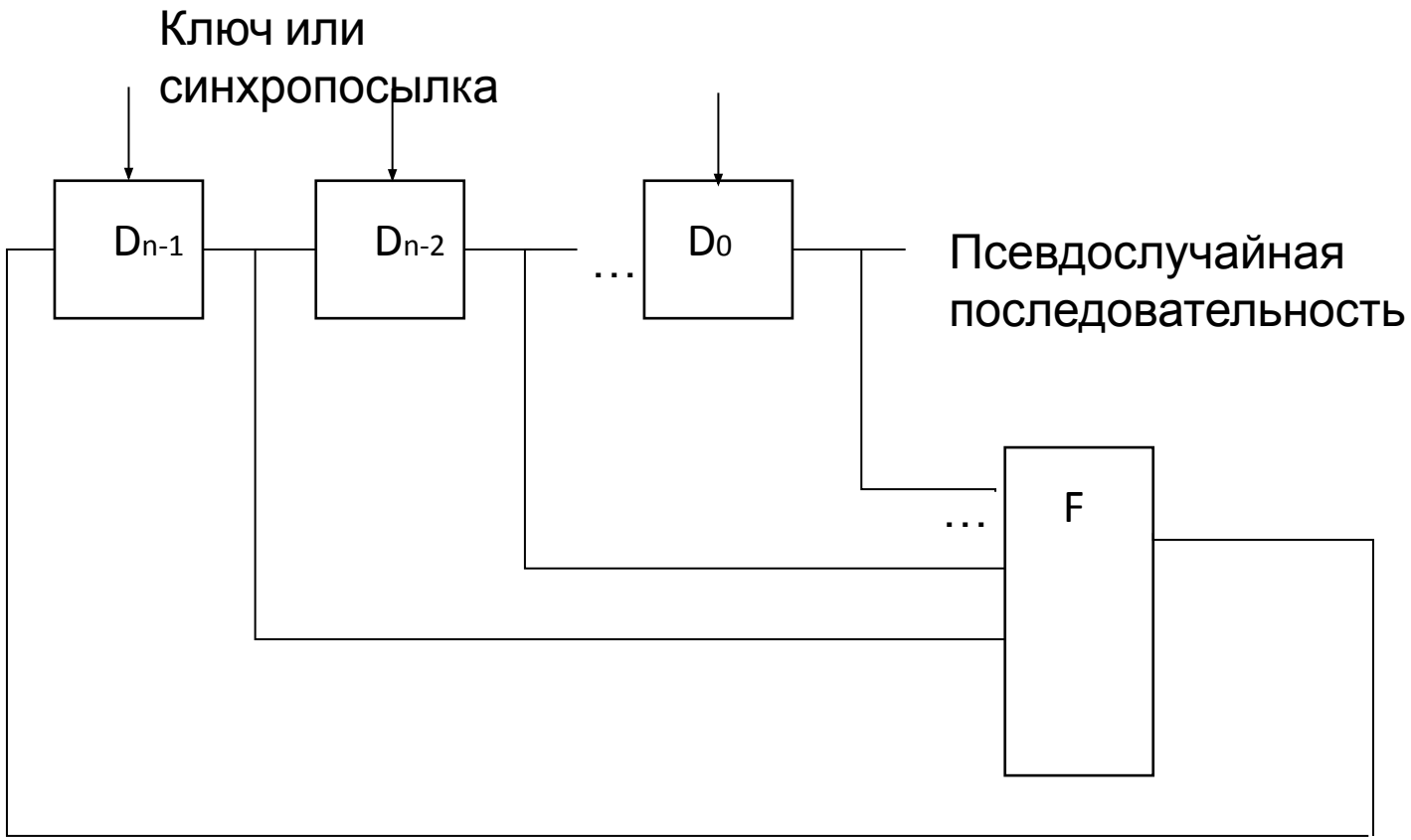
a – множитель, b – инкремент, p – модуль

- Максимальный период генератора - p блоков
- Период максимален, если:
  - модуль и инкремент взаимно просты
  - a-1 делится на любой простой делитель модуля, а также на 4, если модуль делится на 4
- Чтобы последовательность выглядела случайной, множитель и инкремент должны иметь тот же двоичный порядок, что и модуль
- В качестве модуля часто используется степень 2
- Такие генераторы не являются криптостойкими; вместе с тем, они используются для некоторых частных задач, например, генерации констант

УГАТУ

Уфимский государственный авиационный технический университет

Сдвиговые регистры с обратной связью



УГАТУ

Уфимский государственный авиационный технический университет

Сдвиговые регистры с обратной связью

Каждый новый бит вычисляется как логическая функция от предыдущих:

$$D_{n-1}(t) = F(D_0(t-1), D_1(t-1), \dots, D_{n-1}(t-1))$$

F – функция обратной связи

- Применялись еще в 1930-е
- Сами по себе ненадежны; используются как составные части более сложных схем
- Большинство таких схем тоже обычно оказываются уязвимыми
- Пример удачного применения – алгоритм A5

УГАТУ

Уфимский государственный авиационный технический университет

Сдвиговые регистры с обратной связью

Регистр сдвига с линейной обратной связью (РСЛОС), Регистр Фибоначчи, `Linear Feedback Shift Register (LFSR)`

$$D_{n-1}(t) = XOR(D_a(t-1), D_b(t-1), D_c(t-1), \dots)$$

a,b,c,… - **отводная последовательность**

- Максимальный период 2^n−1 бит
- **М-последовательность** – отводная последовательность, дающая максимальный период
  - обязательно включает младший бит
  - Число элементов нечетное (кроме первого примера)
  - Дает максимальный период при сдвиге по регистру как вправо, так и влево

(1, 0)	(7, 3, 0)	(14, 5, 3, 1, 0)	(18, 5, 2, 1, 0)
(2, 1, 0)	(8, 4, 3, 2, 0)	(15, 1, 0)	(19, 5, 2, 1, 0)
(3, 1, 0)	(9, 4, 0)	(16, 5, 3.2, 0)	(20, 3, 0)
(4, 1, 0)	(10, 3, 0)	(17, 3, 0)	(21, 2, 0)
(5, 2, 0)	(11, 2, 0)	(17, 5, 0)	(22, 1, 0)
(6, 1, 0)	(12, 6, 4, 1, 0)	(17, 6, 0)	(23, 5, 0)
(7, 1, 0)	(13, 4, 3, 1, 0)	(18, 7, 0)	(24, 4, 3, 1, 0)

УГАТУ

Уфимский государственный авиационный технический университет

Генератор Блюм-Блюм-Шуба

ПСП определяется на основе последовательности $X_1, X_2, \ldots, X_n$, которая не псевдослучайна

$$X_i = X_{i-1}^2 \bmod n$$

- При этом должны выполняться условия:
  - n = pq, где p и q – простые числа, причем p mod 4 = q mod 4 = 3
  - n и X_1 взаимно просты
  - Наименьший общий делитель $\varphi(p-1)$ и $\varphi(q-1)$ д.б. невелик (чем меньше, тем больше период)
    - здесь $\varphi(x)$ – функция Эйлера, равна количеству натуральных чисел, меньших чем x и взаимно простых с x
- i-й элемент ПСП равен младшим $\log_2 m$ битам X_i, где m – длина X_i в битах
- Генератор непредсказуем влево и вправо

УГАТУ

Уфимский государственный авиационный технический университет

Содержание лекции

- Датчики истинно случайных чисел
- Генераторы псевдослучайных последовательностей
- **Генераторы псевдослучайных последовательностей с добавлением истинной случайности**
  - **FORTUNA**
- Потоковые алгоритмы шифрования

УГАТУ

Уфимский государственный авиационный технический университет

ГПСП с добавлением энтропии

- Применяется, если нужно быстро сгенерировать много непредсказуемых случайных данных
  - Пример: чтобы создать пару ключей алгоритма RSA, нужно сгенерировать порядка мегабайта случайных данных
- Данные с датчика истинно случайных чисел используются для перезапуска ГПСП
- Если злоумышленник получит временный доступ к внутреннему состоянию ГПСП, он сможет восстановить только часть ПСП
- ПСП непредсказуема и потому не может применяться для гаммирования

*Когда речь идет о таких генераторах, понятие «энтропия» обозначает истинно случайные данные.

В других областях информатики у этого понятия более сложное определение.

УГАТУ

Уфимский государственный авиационный технический университет

FORTUNA

- Нильс Фергюссон и Брюс Шнайер, 2003
- Три программных модуля:
  - Генератор ПСП
  - Аккумулятор энтропии
  - Установщик начального значения
    - В ходе работы системы периодически сохраняет истинно случайные данные в файл
    - При загрузке системы данные из этого файла становятся начальным числом генератора

FORTUNA

Генератор ПСП

- Может использоваться любой блочный шифр со 128-битным блоком и 256-битным ключом
- Параметры, задающие внутреннее состояние генератора:

  K – ключ, 256 бит

  c – счетчик блоков, 128 бит

  c* - нонс, генерируемый на основе счетчика, 128 бит
- Инициализация генератора при первом запуске

  K=0; c=0;
- Обновление ключа при получении энтропии

  \\ S – энтропия (истинно случайные данные)
  K = SHA2-256(SHA2-256(K||S));
  c = c + 1;
- Вычисление нонса

  c* = NONCE(c); \\ порядок байт в c меняется на противоположный

УГАТУ

Уфимский государственный авиационный технический университет

FORTUNA

Генератор ПСП

- Генерация n блоков псевдослучайных данных

```
r = NULL;
K0 = NULL;
для i от 1 до n+2
        c* = NONCE(c);
        BLOCK = Ш(c*,K);
        c = c + 1;
        если i < n+1
                то r= r||BLOCK
                иначе  K0=K0||BLOCK
конец
K = K0
```

УГАТУ

Уфимский государственный авиационный технический университет

FORTUNA

Аккумулятор энтропии

- Энтропия поступает из разных источников числом до 256
- Каждый источник записывает энтропию в один из 32-х пулов (P[0]…P[31])
  - (Источник – программа, отслеживающая случайный процесс, а пул - область памяти, куда записываются случайные данные)
- Запись энтропии от одного события производится только в один пул
- Каждое новое событие источник записывает в новый по порядку пул
- Генератор обновляется, когда объем энтропии, накопленной пулом P[0] достигнет 128 бит, но не раньше чем через 100 миллисекунд после прошлого обновления
- Запись полученной энтропии в пул
  - Поскольку используются не сами случайные данные, а хэш-функция от них, в памяти пула достаточно хранить хэш-функцию от заполненных блоков и последний недозаполненный блок

УГАТУ

Уфимский государственный авиационный технический университет

FORTUNA

Аккумулятор энтропии

- Инициализация пулов

  для i от 0 до 31 P[i]=NULL;
  j=0; \\ количество произведенных записей энтропии в генератор

- Обновление генератора истинно случайными данными

  // t – время, прошедшее с прошлого обновления, мс

  ОБРАБОТКА СОБЫТИЯ (LENGTH(P0)>127 И t>10) S = NULL; j = j + 1;

  для i от 0 до 31

  если j mod 2^i != 0 то BREAK;

  S = S || SHA2-256 (SHA2-256(P[i]));

  P[i] = NULL

  конец

УГАТУ

Уфимский государственный авиационный технический университет

Содержание лекции

- Датчики истинно случайных чисел
- Генераторы псевдослучайных последовательностей
- Генераторы псевдослучайных последовательностей с добавлением истинной случайности
- **Потоковые алгоритмы шифрования**
  - **A5**
  - **RC4**
  - **Salsa20**

A5

- Используется в GSM для шифрования трафика между телефоном и базовой станцией
- A5/0 – шифрование не применяется
- A5/1, A5/2 – поточные алгоритмы с длиной ключа 64 бита
  - С современной точки зрения эти алгоритмы не криптостойки, однако представляют интерес как наиболее защищенные схемы на LFSR; теоретически по аналогии с этими схемами можно создать LFSR-шифры, криптостойкие в современном понимании
- A5/3 – применяется блочный алгоритм KASUMI

A5/1

- 3 LFSR, выход алгоритма равен XOR их выходов
  - 19 бит, отводная последовательность {5,2,1,0}, 10-й тактовый бит
  - 22 бита, отводная последовательность {1,0}, 11-й тактовый бит
  - 23 бита, отводная последовательность {15,2,1,0}, 12-й тактовый бит
- Сигнал синхронизации
  - f = MA(t1,t2,t3) – мажоритарная функция тактовых битов соответствующих регистров
  - если ti = f, то i-й регистр сдвигается
  - на каждом такте сдвигаются либо 2, либо 3 регистра

A5/1

Режим работы

- Данные шифруются по кадрам
- В начале обработки кадра все регистры обнулены

этап 1. 64 такта - помимо отводной последовательности выход каждого регистра складывается с ключом

этап 2. 22 такта - помимо отводной последовательности выход каждого регистра складывается с номером кадра

этап 3. 100 холостых тактов

этап 4. 114 тактов - передача 114 бит от Алисы Бобу

этап 5. 114 тактов – передача 114 бит от Боба Алисе

A5/2

Отличия от A5/1

- По другому определяются синхроимпульсы
  - Дополнительный LFSR – 17 бит, отводная последовательность {5,0}
  - t1,t2,t3 – это 10-й, 3-й и 7-й биты данного LFSR
- Выход алгоритма определяется как XOR
  - Трех основных регистров
  - Мажоритарной функции от 12-го, 14-го и 15-го битов первого регистра
  - Мажоритарной функции от 9-го, 13-го и 16-го битов второго регистра
  - Мажоритарной функции от 13-го, 16-го и 18-го битов второго регистра
- На первом холостом такте в 3-й, 7-й и 10-й биты дополнительного регистра записываются единицы

RC4

- Разработан в 1987
- До 1994 был секретен, затем опубликован анонимно
- Длина ключа от 1 до 255 байт
- Самый простой по структуре из получивших широкое применение современных симметричных шифров
- Первые 3072 элемента гаммы рекомендуется генерировать вхолостую
- В алгоритме не задан параметр IV
  - Каждый ключ можно использовать только 1 раз
  - Либо IV конкатенируется с ключом перед инициализацией таблицы

RC4

- Принцип работы алгоритма:
  - Используется динамическая таблица элементов гаммы, записываемая в байтовый массив S[0],…,S[255]; в ней содержатся все возможные значения, которые может принимать 1 байт
  - На этапе инициализации производится перестановка элементов таблицы, зависящая от ключа
  - При генерации гаммы, каждый новый элемент берется из таблицы, после чего производится перестановка таблицы

RC4

Инициализация

```
\\ K0[0]…K0[k-1] – секретный ключ длиной k байт, K[0]…K[255] – байтовый массив
n = 0; j=0;
ДЛЯ i = 0:255
    S[i] = i;
    K[i] = K0[n]; n = (n + 1) mod k; \\ в K записывается ключ, при необходимости повторяясь
КОНЕЦ
ДЛЯ i = 0:255
    j = (j + S[i] + K[i]) mod 256;
    S = S[i]; S[i] = S[j]; S[j] = S; \\ S[i] и S[j] меняются местами
КОНЕЦ
```

RC4

Генерация одного байта гаммы

\\ i и j – переменные, в которые перед началом генерации записываются нули

i = (i + 1) mod 256;

j = (j + S[i]) mod 256;

S = S[i]; S[i] = S[j]; S[j] = S; \\ S[i] и S[j] меняются местами

t = (S[i] + S[j]) mod 256;

Г[m] = S[t]; \\ m – порядковый номер элемента гаммы

Salsa20

- Даниэль Бернштейн, 2004
- Длина ключа – 128 или 256 бит
- Длина IV – 64 бита
- Каждый 512-битный блок гаммы вычисляется по формуле $\Gamma_i = Salsa20\,(\,IV||\mathrm{i}^*, \mathrm{K}\,)$,

  где i* - номер блока, записанный в 64 бита в формате littleendian (байты числа записаны в обратном порядке)

УГАТУ

Уфимский государственный авиационный технический университет

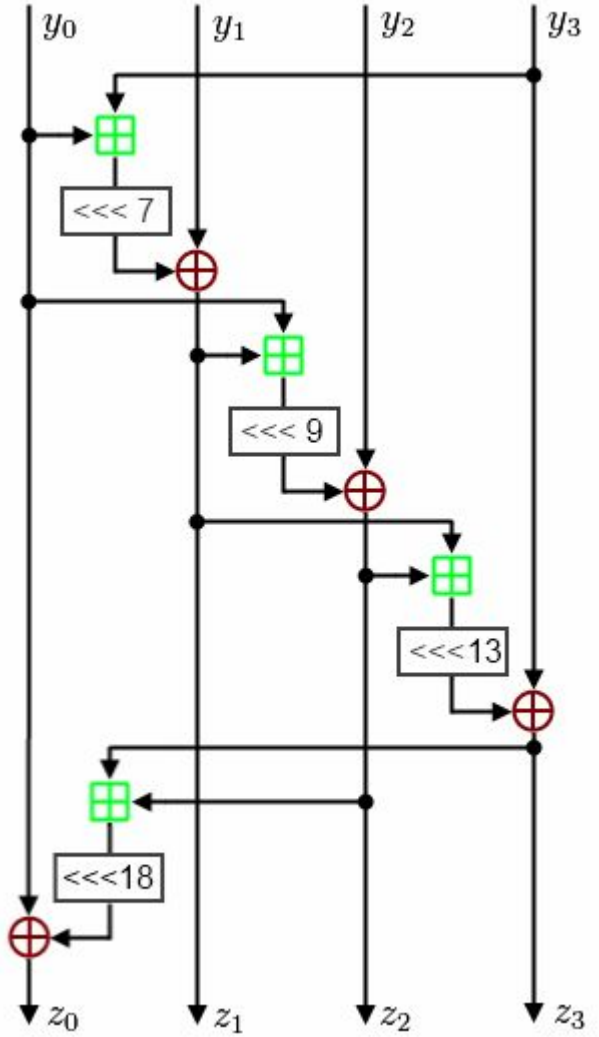
Salsa20

- Базовая операция **quarterround** QR(0,1,2,3) выполняется над четырьмя 32-битными словами $y_{0\ldots3}$
- Каждый раунд алгоритма состоит из 4 операций QR охватывающих все 16 32-битных слов 512-битного блока $y_{0\ldots15}$
  - Нечетный раунд:

```
QR( 0,  4,  8, 12)
QR( 5,  9, 13,  1)
QR(10, 14,  2,  6)
QR(15,  3,  7, 11)
```

  - Четный раунд:

```
QR( 0,  1,  2,  3)
QR( 5,  6,  7,  4)
QR(10, 11,  8,  9)
QR(15, 12, 13, 14)
```



Salsa20

- Основная операция **Salsa20**(y):

```
y1 = y
ДЛЯ i от 1 до 10 \\ на каждой итерации выполняется 2 раунда, соответственно всего раундов 20
        y1 = НЕЧЕТНЫЙ_РАУНД(y1)
        y1 = ЧЕТНЫЙ_РАУНД(y1)
Salsa20 = xor (y, y1)
```

- Зашифрование в версии со 128-битным ключом

$$\Gamma_i = Salsa20(\tau_0 \,||\, K \,||\, \tau_1 \,||\, IV \,||\, i^* \,||\, \tau_2 \,||\, K_2 \,||\, \tau_3)$$

$$\tau_0 = (101; 120; 112; 97),\ \tau_1 = (110; 100; 32; 49),\ \tau_2 = (54; 45; 98; 121),\ \tau_3 = (116; 101; 32; 107)$$

- Зашифрование в версии с 256-битным ключом

$$\Gamma_i = Salsa20(\,\sigma_0 \,||\, K_1 \,||\, \sigma_1 \,||\, IV \,||\, i^* \,||\, \sigma_2 \,||\, K_2 \,||\, \sigma_3)$$

$$\sigma_0 = (101, 120, 112, 97),\ \sigma_1 = (110, 100, 32, 51),\ \sigma_2 = (50, 45, 98, 121),\ \sigma_3 = (116, 101, 32, 107)$$