Защита информации

Тема 10. Управление ключами

Управление ключами

Под ***ключевой информацией*** понимается совокупность всех действующих в ИС ключей.

Управление ключами - информационный процесс, включа-ющий в себя три элемента:

1) ***генерацию ключей***;
2) ***накопление ключей***;
3) ***распределение ключей***.

Генерация случайных чисел и ключей

Генерируемые числа должны быть *случайны* или, по крайней мере, *непредсказуемы*.

Генерируемая последовательность чисел должна удовлетворять 2-ум критериям:

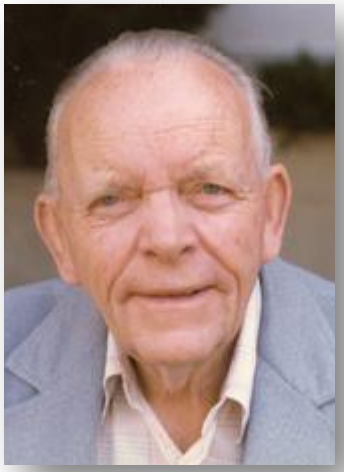
1. ***Однородное распределение*** чисел в последовательности; это означает, что частота появления каждого числа должна быть приблизительно одинаковой.
2. ***Независимость***. Ни одно значение не должно статистически зависеть от других, что должны показать многочисленные статистические тесты, определяющие зависимость.

Не стоит использовать неслучайные ключи с целью легкости их запоминания. В серьезных ИС используются специальные аппаратные и программные методы генерации случайных ключей.

Чтобы получить линейные последовательности элементов гаммы, длина которых превышает размер шифруемых данных, используются ***датчики ПСЧ***. На основе теории групп было разработано несколько типов таких датчиков.

Конгруэнтные датчики

В настоящее время наиболее доступными и эффективными являются конгруэнтные генераторы ПСЧ, впервые предложенные американским математиком ***Лéмером***. Для этого класса генераторов можно сделать математически строгое заключение о том, какими свойствами обладают выходные сигналы этих генераторов с точки зрения периодичности и случайности.

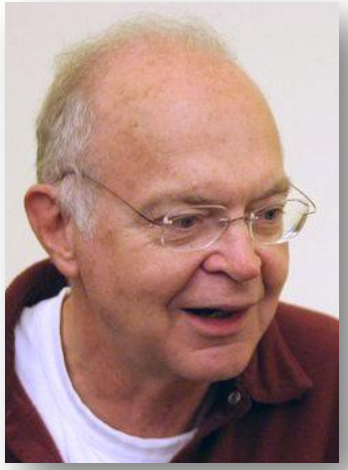


Деррик Лéмер
(Derrick Lehmer)

Одним из хороших конгруэнтных генераторов является ***линейный конгруэнтный датчик ПСЧ***. Он вырабатывает

$$T_{i+1} = (a\,T_i + c) \bmod m,$$

где a и c - константы, T_0 - исходная величина, выбранная в качестве порождающего числа. Такой датчик ПСЧ генерирует псевдослучайные числа с определенным периодом повторения, зависящим от выбранных значений a и c. Значение m часто устанавливается равным 2^n, где n - длина машинного слова в битах. Как показано американским программистом ***Дональдом Кнутом***, линейный конгруэнтный датчик ПСЧ имеет максимальную длину M тогда и только тогда, когда:



Дональд Кнут
(Donald Knuth)

1) $gcd(c, m) = 1$, то есть c и m взаимно просты.
2) $b = a-1$ кратно p для всех простых p – делителей m.

Конгруэнтные датчики. Примеры

Например. В семействе компьютеров IBM 360 использовался датчик:

$$T_{i+1} = (7^5 \cdot T_i) \bmod (2^{31}-1)$$

$m = 2^{31}-1 = 2\ 147\ 483\ 648 - 1 = 2\ 147\ 483\ 647 =$

$111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2$

1) gcd(c, m) = gcd(0, 2147483647) = 1

2) $b = a - 1 = 7^5 - 1 = 16807 - 1 = 16806$

3) m не кратно 4.

Еще один рекомендуемый датчик:

$$T_{i+1} = (1664525 \cdot T_i + 1013904223) \bmod 2^{32}$$

$m = 2^{32} = 4294967296 = 1\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2$

1) gcd(c, m) = gcd(1013904223, 4294967296) = 1

2) $b = a - 1 = 7^5 - 1 = 1664525 - 1 = 1664524$

3) b mod 4 = 0

Циклическое шифрование

Для генерации гаммы шифра часто используют шифрование случайных чисел, полученных с помощью генераторов. Например, режим Stream Mode алгоритма DES.

$$X_{i+1} = E_K(T_{i+1})$$

Иногда в качестве генератора используется просто счетчик.

$$X_{i+1} = E_K(Cnt{+}{+})$$

Генератор псевдослучайных чисел ANSI X9.17

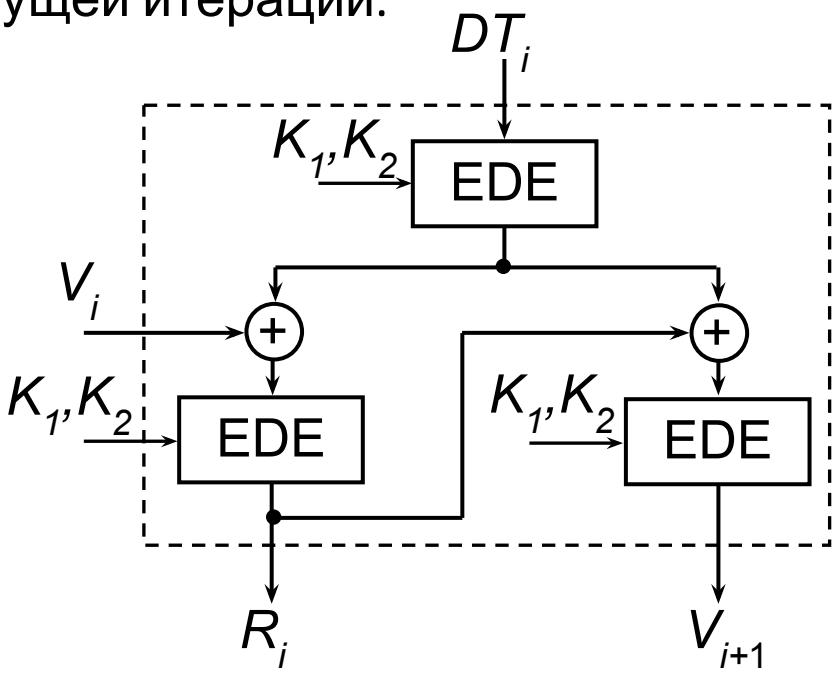
Этим генератором пользуются приложения финансовой безопасности и ***PGP***.

Алгоритмом шифрования является тройной ***DES*** (***EDE***) с 2-мя 56-битными ключами K_1 и K_2.

На входе алгоритма два 64-битных числа. Одно из них это текущее значение даты и времени на момент *i*-ой итерации (DT_i), а второе – начальное значение (V_i), которое сначала берется произвольное, а затем берется вычисленное на предыдущей итерации.

$$R_i = EDE_{K1,K2}\left(EDE_{K1,K2}\left(DT_i \right) \oplus V_i \right)$$

$$V_{i+1} = EDE_{K1,K2}\left(EDE_{K1,K2}\left(DT_i \right) \oplus R_i \right)$$



Датчики М-последовательностей

М-последовательности популярны, благодаря относительной легкости их реализации. Они представляют собой ***линейные рекуррентные последовательности максимального периода***, формируемые k-разрядными генераторами на основе ***регистров сдвига***.

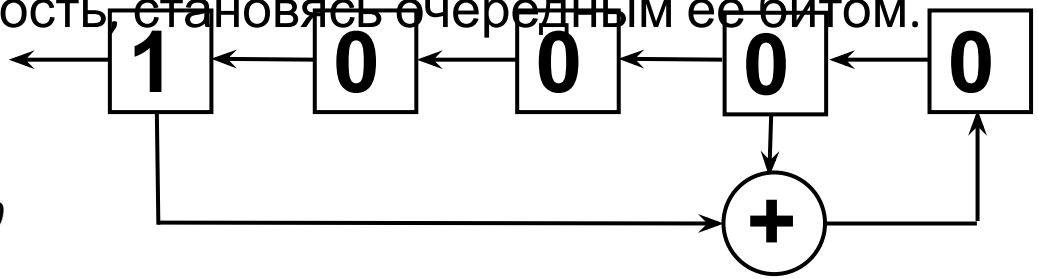
Из текущего набора бит выбираются значения определенных разрядов и складываются по XOR между собой. Все разряды сдвигаются на 1 бит, а только что полученное значение ("0" или "1") помещается в освободившийся самый младший разряд. Значение, находившееся в самом старшем разряде до сдвига, добавляется в кодирующую последовательность, становясь очередным ее битом.

$$x_i := r_{k-1}$$

$$r := a_0 r_0 \oplus a_1 r_1 \oplus \ldots \oplus a_{k-1} r_{k-1}$$

$$r_{k-1} := r_{k-2} \quad \ldots \quad r_2 := r_1 \quad r_1 := r_0$$

$$r_0 := r$$



Здесь $r_0\ r_1 \ldots\ r_{k-1}$ - k однобитных регистров, $a_0\ a_1 \ldots\ a_{k-1}$ - коэффициенты неприводимо-го двоичного полинома степени k-1. x_i - i-е значение выходной битовой гаммы.

Для представленной схемы a_0=0, a_1=1, a_2=0, a_3=0, a_4=1. Для: r_0=0, r_1=0, r_2=0, r_3=0, r_4=1, получим последовательность

Примеры

Для генератора М-последовательности любой разрядности ***k*** всегда существует такой выбор охватываемых обратной связью разрядов, что генерируемая ими последовательность бит будет иметь период, равный 2^k-1 битам. Так, например, в 8-битном скремблере, при охвате 0-го, 1-го, 6-го и 7-го разрядов действительно за время генерации 255 бит последовательно проходят все числа от 1 до 255, не повторяясь ни разу.

М-последовательности неразрывно связаны с математической теорией ***неприводимых полиномов***. Достаточно чтобы полином степени ***k*** не был представим по модулю ***2*** в виде произведения никаких других полиномов, для того, чтобы генератор, построенный на его основе, создавал ***ПНД***. Например, единственным неприводимым полиномом степени 3 является x^3+x+1, в двоичном виде он записывается как 1011_2. Скремблеры на основе неприводимых полиномов образуются отбрасыванием самого старшего разряда (он всегда присутствует, а следовательно, несет информацию только о степени полинома), так на основе указанного полинома, мы можем создать скремблер 011_2 с периодом зацикливания $7(=2^3-1)$. Естественно, что на практике применяются полиномы значительно более высоких порядков. А таблицы неприводимых полиномов любых порядков можно всегда найти в специализированных математических справочниках.

Пример

Например, для 32-битового сдвигового регистра имеется последовательность 32, 7, 5, 3, 2, 1. Это означает, что для генерации нового бита необходимо с помощью функции XOR просуммировать 32-й, 7-й, 5-й, 3-й, 2-й и 1-й биты.

Код для такого генератора на языке Си следующий:

```
int LFSR (void){
  static unsigned long SR = 1;
  SR = ((((SR >> 31)
    ^ (SR >> 6)
    ^ (SR >> 4)
    ^ (SR >> 2)
    ^ (SR >> 1)
    ^ SR) & 0x00000001) << 31)
    | (SR >> 1);
  return SR & 0x00000001;
}
```

Программные реализации таких генераторов медленно работают. Одним из решений является использование параллельно ***32-х генераторов***.

В такой схеме используется массив слов, размер которого равен ***k***, а каждый бит слова массива относится к своему генератору. При условии, что используются одинаковые номера отводных последовательностей, то это может дать заметный выигрыш в производительности.

Идеальные генераторы

Идеальным генераторами являются устройства на основе «натураль-ных» случайных процессов. Например, появились образцы генерации ключей на основе ***белого радиошума***, на основе газоразрядных трубок, на основе ***детекторов событий ионизирующей радиации*** и др. генераторов шумов. Другим случайным математическим объектом являются ***десятичные знаки иррациональных чисел***, например ***π***, которые вычисляются с помощью стандартных математических методов.

В ИС со средними требованиями защищенности вполне приемлемы программные генераторы ключей, которые вычисляют ***ПСЧ*** как сложную функцию от ***текущего времени*** и (или) числа, введенного пользователем.

Накопление ключей

Под накоплением ключей понимается организация их хранения, учета и удаления.

Секретные ключи никогда не должны записываться в явном виде на носителе, который может быть считан или скопирован.

В достаточно сложной ИС один пользователь может работать с большим объемом ключевой информации, и иногда даже возникает необходимость организации мини-баз данных по ключевой информации. Такие базы данных отвечают за принятие, хранение, учет и удаление используемых ключей.

Итак, каждая информация об используемых ключах должна храниться в зашифрованном виде. Ключи, зашифровывающие ключевую информацию называются ***мастер-ключами***. Мастер-ключи каждый пользователь должен помнить, и не хранить их на каких-либо материальных носителях.

Очень важным условием безопасности информации является периодическое обновление ключевой информации в ИС. При этом переназначаться должны как обычные ключи, так и мастер-ключи. В особо ответственных ИС ключи обновляются ежедневно.

Вопрос обновления ключевой информации связан с распределением ключей.

Распределение ключей

Распределение ключей - самый ответственный процесс в управлении ключами. К нему предъявляются требования: оперативности, точности и скрытности.

В последнее время заметен сдвиг в сторону использования криптосистем с открытым ключом, в которых проблема распределения ключей отпадает.

Распределение ключей между пользователями реализуются двумя разными подходами:

1. Путем создания одного ли нескольких ***центров распределения ключей***. Недостаток такого подхода состоит в том, что в центре распределения известно, кому и какие ключи назначены и это позволяет читать все сообщения, циркулирующие в ИС.

2. Прямой ***обмен ключами между пользователями*** информационной системы. В этом случае проблема состоит в том, чтобы надежно удостоверить подлинность субъектов.

Гарантия подлинности сеанса

В обоих случаях должна быть гарантирована подлинность сеанса связи. Это можно обеспечить двумя способами:

1. *Механизм **запроса-ответа.*** Если пользователь ***A*** желает быть уверенным, что сообщения которые он получает от ***B***, не являются ложными, он включает в посылаемое для ***B*** сообщение непредсказуемый элемент (запрос). При ответе пользователь ***B*** должен выполнить некоторую операцию над этим элементом (например, добавить 1). Это невозможно осуществить заранее, так как неизвестно, какое случайное число придет в запросе. После получения ответа с результатами действий пользователь ***A*** может быть уверен, что сеанс является подлинным. Недостатком этого метода является возможность установления закономерности между запросом и ответом.

2. *Механизм **отметки времени** («временной штемпель»)*. Он подразумевает фиксацию времени для каждого сообщения. В этом случае каждый пользователь ИС может знать, насколько «старым» является пришедшее сообщение.

В обоих случаях следует использовать шифрование, чтобы быть уверенным, что ответ послан не злоумышленником и штемпель отметки времени не изменен.

Допустимое запаздывание «штемпеля»

При использовании отметок времени встает проблема допустимого временного интервала задержки для подтверждения подлинности сеанса. Ведь сообщение с «временным штемпелем» не может быть передано мгновенно. Кроме этого компьютерные часы получателя и отправителя не могут быть абсолютно синхронизированы.

В системах оплаты кредитных карточек используется механизм ***отметки времени*** для установления подлинности и защиты от подделок. Используемый интервал составляет от одной до нескольких минут. Большое число известных способов кражи электронных денег, основано на «вклинивании» в этот промежуток подложных запросов на снятие денег.

Для обмена ключами можно использовать криптосистемы с открытым ключом, используя тот же алгоритм RSA.

Но весьма эффективным оказался алгоритм ***Диффи-Хеллмана***, позволяющий двум пользователям без посредников обменяться ключом, который может быть использован затем для симметричного шифрования.

Алгоритм обмена ключами Диффи-Хеллмана

Алгоритм предложили в 1976 году американцы Уитфилд Диффи и Мартин Хеллман.

Обоим абонентам известны некоторые два числа *v* и *n*, причем n – простое.

1. A: *Xa*
2. B: *Xb*
3. A → B: $(v^{Xa})\ mod\ n$,
4. B → A: $(v^{Xb})\ mod\ n$,
5. A: $k_{ab} = (((v^{Xb})\ \text{mod}\ n)^{Xa})\ \text{mod}\ n$,
6. B: $k_{ba} = (((v^{Xa})\ \text{mod}\ n)^{Xb})\ \text{mod}\ n$.

Пример.

v = 17; n = 23;

Xa = 3;

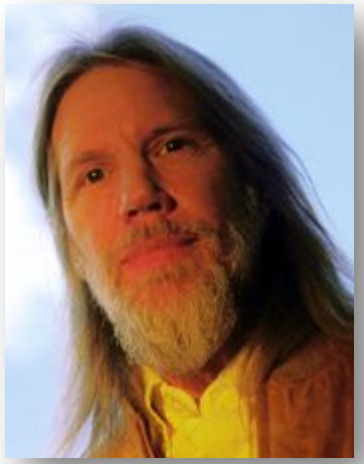
$v^{Xa} = 17^3 \text{ mod } 23 = 4\,913 \text{ mod } 23 = 14$;

Xb = 5;

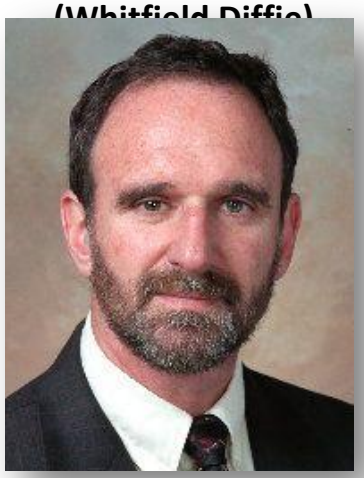
$v^{Xb} = 17^5 \text{ mod } 23 = 1\,419\,857 \text{ mod } 23 = 21$;

$k_{ab} = (v^{Xb})^{Xa} = 21^3 = 9\,261 \text{ mod } 23 = 15$;

$k_{ba} = (v^{Xa})^{Xb} = 14^5 = 537\,824 \text{ mod } 23 = 15$.



Уитфилд Диффи (Whitfield Diffie)



Мартин Хеллман (Martin Hellman)

Аналог алгоритма Диффи-Хеллмана с использованием эллиптических кривых

Сначала выбираются параметры эллиптической кривой $Ep(a,b)$ (p ≈ 2^{180}) и генерирующая точка G, принадлежащая этой кривой. При выборе G важно, чтобы наименьшее значение q, при котором $[q]G=O$, оказалось очень большим простым числом. Эти параметры известны всем абонентам системы. Используется операция композиции.

1. A: выбирает целое число $n_A < q$; вычисляет точку $P_A = [n_A]G$;
2. B: выбирает целое число $n_B < q$; вычисляет точку $P_B = [n_B]G$;
3. A $\longrightarrow$ B: P_A;
4. B $\longrightarrow$ A: P_B;
5. A: $K = [n_A]P_B = [n_A \cdot n_B]G$;
6. B: $K = [n_B]P_A = [n_B \cdot n_A]G$.

Алгоритм Шамира

Ади Шамир
(Adi Shamir)

Данный алгоритм для передачи коротких сообщений (например, ключей) предложил израильский криптоаналитик Ади Шамир.

Пусть абонент ***A*** хочет передать сообщение ***m*** абоненту ***B*** в зашифрованном виде.

1. A → B: случайное большое число p = 11
3. A: выбирает два числа *Ca* и *Da*, такие что
   Ca·*Da* mod (p-1) = 1. 3*7=1 (mod 10)
4. B: выбирает два числа *Cb* и *Db*, такие что
   Cb·*Db* mod (p-1) = 1. 9*9=1 (mod 10)
5. A → B: $x_1 = m^{Ca} \bmod p$. 5^3=4 (mod 11)
6. B → A: $x_2 = x_1^{Cb} \bmod p$. 4^9=3 (mod 11)
7. A → B: $x_3 = x_2^{Da} \bmod p$. 3^7=9 (mod 11)
8. B: $m = x_3^{Db} \bmod p$. 9^9=5 (mod 11)