Применение алгоритма SEAL в целях защиты речевой информации

В. А. СТЕПАНЦОВ, Д.С. ЛУНИС

Введение

Проблема защиты информации, в частности речевой, путем ее преобразования, исключающего несанкционированный доступ к ней посторонних лиц, в значительной степени решается использованием криптографии. Без использования криптографических методов и алгоритмов невозможно сегодня представить осуществление таких задач обеспечения безопасности информации, как конфиденциальность, целостность и аутентификация.

В общем случае современные криптосистемы делятся на симметричные и асимметричные.

Симметричные криптосистемы наряду с недостатками, к которым относятся организационные сложности управления и обмена ключами в большой сети, имеют по сравнению с ассиметричными криптосистемами ряд несомненных достоинств:

- скорость передачи сообщений;
- простота реализации за счет более простых операций;
- меньшая требуемая длина ключа при сопоставимой стойкости;
- изученность за счет более длительной практики применения.

С практической точки зрения значительный интерес представляет оптимизированный под программную реализацию поточный шифр SEAL.

SEAL – Software-Optimized Encryption Algorithm

SEAL позиционируется как увеличивающая длину псевдослучайная функция, используемая для реализации поточного шифрования. Для работы ему требуются восемь 32-битовых регистров и память объемом несколько Кбайт. На предварительном этапе SEAL преобразует ключевую информацию в набор таблиц, суммарный объем которых составляет всего 3 Кбайта. Эти таблицы используются для быстрого шифрования информации, затрачивая около пяти машинных инструкций на байт данных.

Получая на входе 160-битовый секретный ключ k и 32-битовый параметр n (индекс), SEAL формирует последовательность $SEAL_k$(n длиной L, где L не превосходит 64 Кбайт - **SEAL (k, n, L)**. Наиболее приемлемыми для реальных нужд считаются величины от 512 до 4096 байт. SEAL порождает последовательность переменной длины. В случае, когда ключ k случаен и неизвестен, последовательность $SEAL_k$(n) псевдослучайна, и определить использованную функцию не представляется возможным.

Шаг 1. Заполнение таблиц

Алгоритм SEAL использует зависимые от ключа таблицы R,S,T. Заполнение таблиц выполняется с помощью функции G, которая является функцией сжатия алгоритма хеширования SHA.

Функция G_k (i) для 160-битовой последовательности k и 32-битового целого числа i ($0 < i < 2^{32}$) может быть представлена следующим образом:

Последовательность k разбивается на пять 32-битовых слов $k = H_0\ H_1\ H_2\ H_3\ H_4$. $a=H_0$, $b=H_1$, $c=H_2$, $d=H_3$, $e=H_4$ – вспомогательные 32-битовые регистры.

Строится 512-битовая последовательность Y по правилу i || 0^{480} (конкатенация 32-битового числа i и 480-битовая последовательность нулей).

Блок Y рассматривается как массив (w_0 ,….,w_{15}) из 16 (32-битовых) слов, так что

$w_0 = i$, $w_1 = w_2 =\ldots= w_{15} = 0^{32}$. Блок Y расширяется до 80 слов по 32 разряда в каждом. Расширение происходит следующим образом. Пусть (w_0 ,….,w_{15}) – исходный блок, (W_0 ,….,W_{79}) – расширенный блок, при этом

$$W_t = \begin{cases} w_t & t = 0\ldots15; \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1, & t = 16\ldots79. \end{cases}$$

Выполняются 80 раундов алгоритма (t от 0 до 79, f_t – раундовая функция, K_t – раундовая константа), на каждом из которых происходит выполнение следующих операций:

temp=(a<<<5)+f_t(b,c,d)+e+W_t+K_t; e=d; d=c; c=b<<<30; b=a; a=temp. Далее определяются функция f_t(x,y,z) и шестнадцатеричная константа K_t.

Шаг 1. Заполнение таблиц

$$f_t(x,y,z) = \begin{cases} (x \wedge y) \vee (\neg x \wedge z), & t = 0 \ldots 19; \\ x \oplus y \oplus z, & t = 20 \ldots 39, 60 \ldots 79; \\ (x \wedge y) \vee (x \wedge z) \vee (y \wedge z), & t = 40 \ldots 59. \end{cases}$$

$$K_t = \begin{cases} 5A827999, & t = 0 \ldots 19; \\ 6ED9EBA1, & t = 20 \ldots 39; \\ 8F1BBCDC, & t = 40 \ldots 59; \\ CA62C1D6, & t = 60 \ldots 79. \end{cases}$$

Выполняется сложение по модулю 2^{32} полученных значений a, b, c, d, e соответственно с $H_0\ H_1\ H_2\ H_3\ H_4$:

$H_0 = H_0 + a$; $H_1 = H_1 + b$; $H_2 = H_2 + c$; $H_3 = H_3 + d$; $H_4 = H_4 + e$.

После выполнения функции G_k (i) получим 160-битовое значение $H_0\ H_1\ H_2\ H_3\ H_4$.

Далее строится функция Г с выходным значением длиной 32 бита, путем переиндексирования функции G, Γ_k (i) = $H^i_{i \bmod 5}$.

Таблицы R, S, T определяются следующим образом:

$T[i] = \Gamma_k(i)$, $0 \le i < 512$;

$S[j] = \Gamma_k(0x1000+j)$, $0 \le j < 256$;

$R[q] = \Gamma_k(0x2000+q)$, $0 \le q < 4[(L-1)/8192]$;

Шаг 2. Генерация ключевой последовательности

Имея число L, таблицы R, S и T, заданные ключом k, и параметр n, представленный ниже алгоритм растягивает n в L-битную последовательность y.

```
function SEAL(k;n;L)
{
  y = ∅;
  for l = 0 to ∞ do {
    Initialize(n;l;A;B;C;D;n1;n2;n3;n4);
    for i = 1 to 64 do {
      P = A ∧ 0x7fc; B = B + T[P/4]; A = A >>> 9; B = B ⊕ A;
      Q = B ∧ 0x7fc; C = C ⊕ T[Q/4]; B = B >>> 9; C = C + B;
      P = (P + C) ∧ 0x7fc; D = D + T[P/4]; C = C >>> 9; D = D ⊕ C
      Q = (Q + D) ∧ 0x7fc; A = A ⊕ T[Q/4]; D = D >>> 9; A = A + D
      P = (P + A) ∧ 0x7fc; B = B ⊕ T[P/4]; A = A >>> 9;
      Q = (Q + B) ∧ 0x7fc; C = C + T[Q/4]; B = B >>> 9;
      P = (P + C) ∧ 0x7fc; D = D ⊕ T[P/4]; C = C >>> 9;
      Q = (Q + D) ∧ 0x7fc; A = A + T[Q/4]; D = D >>> 9;
      y = y || B + S[4i − 4] || C + S[4i − 3] || D + S[4i − 2] || A + S[4i − 1];
      if |y| ≥ L then return(y0 y1 ... yL−1);
      if odd(i) then (A;B;C;D) = (A + n1; B + n2; C ⊕ n1; D ⊕ n2)
                else (A;B;C;D) = (A + n3; B + n4; C ⊕ n3; D ⊕ n4);
    }
  }
}
```

Алгоритм использует подпрограмму Initialize для отображения n и l в значения внутренних переменных и регистров (A, B, C, D, n_1,n_2 ,n_3 ,n_4).

```
procedure Initialize(A,B,C,D,n1,n2,n3,n4)
  A = n ⊕ R[4l];
  B = (n >>> 8) ⊕ R[4l + 1];
  C = (n >>> 16) ⊕ R[4l + 2];
D = (n >>> 24) ⊕ R[4l + 3];
  for j = 1 to 2 do {
    P = A ∧ 0x7fc; B = B + T[P/4]; A = A >>> 9;
    P = B ∧ 0x7fc; C = C + T[P/4]; B = B >>> 9;
    P = C ∧ 0x7fc; D = D + T[P/4]; C = C >>> 9;
    P = D ∧ 0x7fc; A = A + T[P/4]; D = D >>> 9;
  }
  (n1,n2,n3,n4) = (D;B;A;C);
  P = A ∧ 0x7fc; B = B + T[P/4]; A = A >>> 9;
  P = B ∧ 0x7fc; C = C + T[P/4]; B = B >>> 9;
  P = C ∧ 0x7fc; D = D + T[P/4]; C = C >>> 9;
  P = D ∧ 0x7fc; A = A + T[P/4]; D = D >>> 9;
}
```

Программная реализация алгоритма SEAL

Главным преимуществом алгоритма SEAL наряду с высоким быстродействием является возможность его использования для шифрования звуковых файлов, что делает данный алгоритм незаменимым в целях защиты речевой информации.

Прототип программного обеспечения, реализующего алгоритм SEAL для шифрования звуковых файлов, разработан на языке Jawa.

На первом этапе работы программы генерируется ключ, который обеспечивает создание уникальных таблиц, необходимых для шифрования.

```
int[] key = generateKey();
Seal seal1 = new Seal(key,  n: 234567);

private static int[] generateKey() {
    int key[] =  {
                0x10000000, 0x02000000, 0x00300000, 0x00040000, 0x00005000, 0x00000600,
                0x00000070, 0x00000008, 0x90000000, 0x0a000000, 0x00b00000, 0x000c0000,
                0x0000d000, 0x00000e00, 0x000000f0, 0x10000000, 0x01100000, 0x00120000,
                0x00013000, 0x00001400
    };
    return key;
}
```

Программная реализация алгоритма SEAL

Далее создаются файловые потоки. Было создано три файловых потока: для входящего, шифрованного и дешифрованного файла

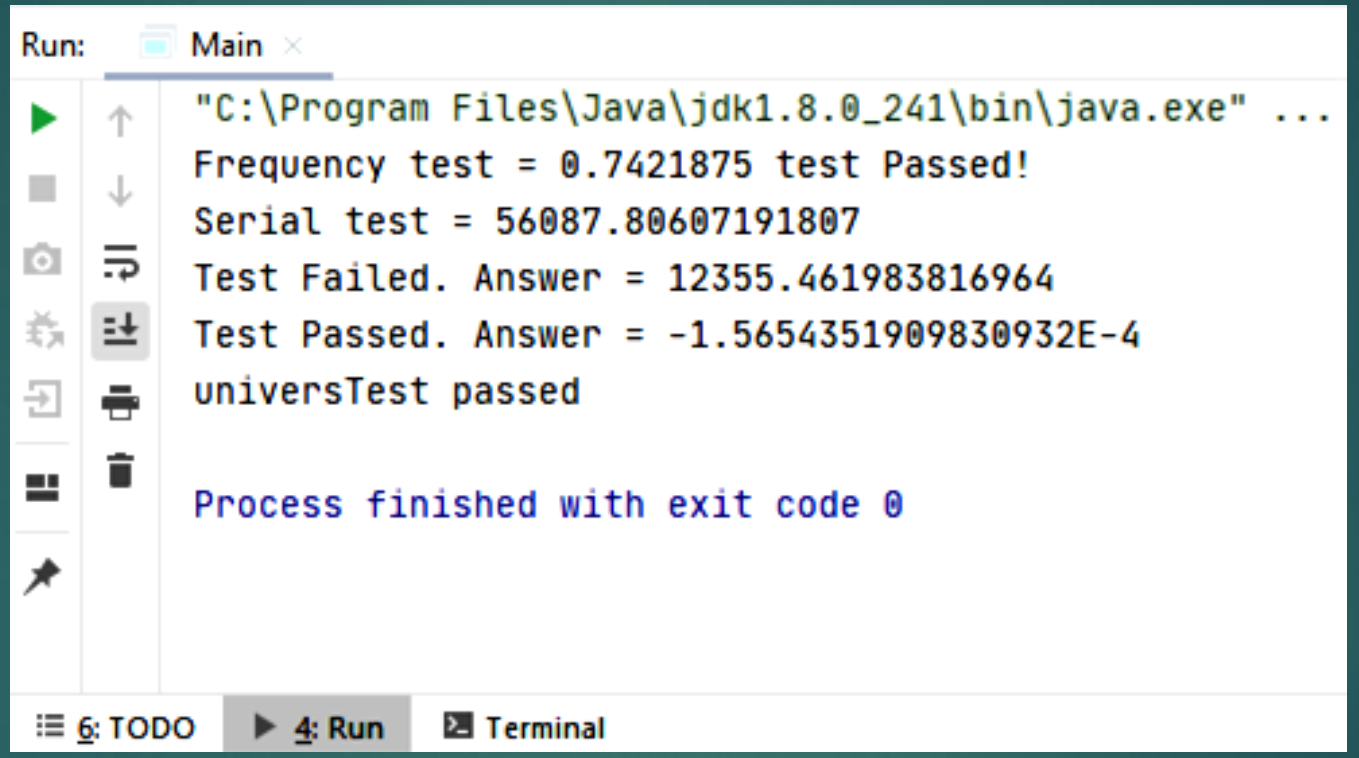
```
FileInputStream in = new FileInputStream(new File(pathInputFile));

FileOutputStream encryptOut = new FileOutputStream(new File(pathEncryptFile));

FileOutputStream decryptedOut = new FileOutputStream(new File(pathOutputFile));
```

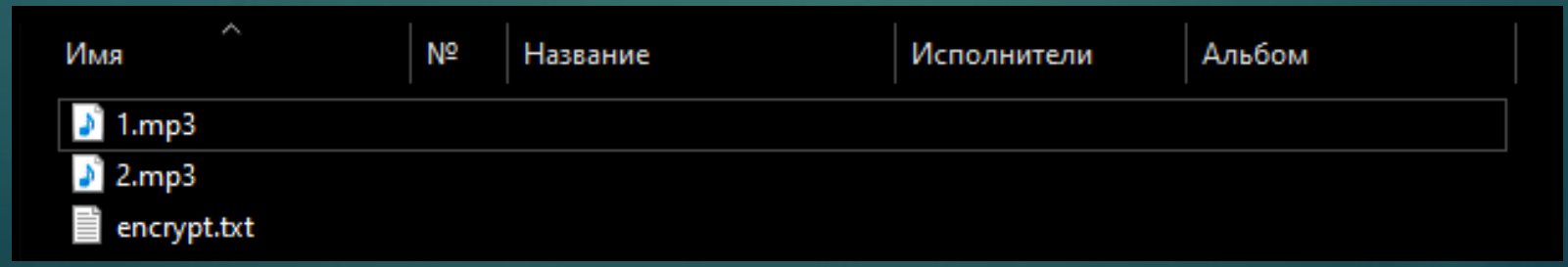
Аудио файл разбивается на блоки. Запускается цикл, который будет работать до конца входящего потока. Каждый блок будешь шифроваться и дешифроваться при помощи таблицы сгенерированной в алгоритме SEAL

```
}
int[] encrypted = seal1.encrypt(block);
for (int i = 0; i < counter; i++) {
    encryptOut.write(encrypted[i]);
}
int[] decrypted = seal1.decrypt(encrypted);
for (int i = 0; i < counter; i++) {
    decryptedOut.write(decrypted[i]);
}
```

Пример работы программы

Была проведена шифрация аудиофайла 1.mp3. Дешифрованный файл 2.mp3 был сохранен в той же папке, что и исходный. Формат шифрованного файла encrypt.txt был выбран из соображений безопасности.





Заключение

Рассмотрена возможность применения поточного алгоритма SEAL в целях защиты речевой информации. Разработан прототип программного обеспечения, реализующего шифрацию и дешифрацию аудиофайлов. Использованием данного алгоритма не исчерпывается проблема защиты аудиоданных. Проведенное исследование показало целесообразность применения рассмотренного алгоритма в системах информационной безопасности.

Список литературы

1. Асосков А. В., Иванов М. А., Мирский А. А., Рузин А. В., Сланин А. В., Тютвин А. Н. Поточные шифры. – М.: КУДИЦ-ОБРАЗ, 2003. – 336 с.