Применение алгоритма SEAL в целях защиты речевой информации

В. А. Степанцов, email: mrstep@yandex.ru, Д. С. Лунис

Воронежский государственный университет

Аннотация. *Рассмотрена возможность применения поточного криптографического алгоритма SEAL в целях защиты речевой информации.*

Ключевые слова: *защита речевой информации, алгоритм SEAL.*

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.

Безопасность современных алгоритмов шифрования полностью основана на ключах, а не на деталях алгоритмов. Это значит, что алгоритм может быть опубликован и проанализирован. Продукты, использующие некоторый алгоритм, могут широко тиражироваться. Не имеет значения, что злоумышленнику известен алгоритм, если ему не известен конкретный ключ, то он не сможет прочесть сообщения.

В общем случае современные криптосистемы делятся на симметричные и асимметричные. В симметричных криптосистемах и для шифрования, и для дешифрования применяется один и тот же ключ. Он является секретным и передается отправителем получателю по каналу связи. Алгоритм шифрования выбирается сторонами до начала обмена сообщениями. В асимметричных для шифрования и дешифрования используются разные ключи.

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.

Среди симметричных криптографических систем следует отдельно выделить поточные шифры. Поточный шифр – это симметричный шифр, в котором каждый символ открытого текста преобразуется в символ шифрованного текста в зависимости не только от используемого ключа, но и от его расположения в потоке открытого текста.

С практической точки зрения значительный интерес представляет оптимизированный под программную реализацию поточный шифр SEAL [1].

1. 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. Такой шифр обозначим $SEAL(k, n, L)$. 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. Поточный шифр на основе семейства псевдослучайных функций зависит не только от ключа k и сообщения p, но и от позиции n данного сообщения в потоке данных. Шифрование с позиции n сообщения p задается выражением $(n, p \oplus SEAL_k(n))$, при этом используется L бит выхода $SEAL_k(n)$. После формирования L^* бит, где L^* – наименьшее кратное 128, большее или равное L алгоритм прекращает генерацию.

Использование псевдослучайных функций дает возможность прямого доступ к любому фрагменту выходной последовательности.

Алгоритм состоит из нескольких шагов. Обозначим:

– $y <<< t$ – циклический сдвиг слова y на t разрядов влево;

– $y >>> t$ – циклический сдвиг слова y на t разрядов вправо;

– $\wedge, \vee, \oplus, \neg$ – поразрядные операции AND, OR, XOR, NOT;

– символ $\|$ обозначает операцию конкатенации;

– $odd()$ – предикат истинный тогда и только тогда, когда его аргумент – четное число;

– $y + t$ – сумма целых чисел y и t по модулю 2^{32}.

Этап 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 $k = H0H1H2H3H4$, где $a = H0, b = H1, c = H2, d = H3, e = H4$ 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 – массив $(w_0, \ldots, w_{15})$ из 16 (32-битовых) слов, так что $w_0 = i, w_1 = \ldots = w_{15} = 0^{32}$. Далее он расширяется до 80 слов по 32 разряда в каждом. Пусть $(w_0, \ldots, w_{15})$ – исходный блок, $(W_0, \ldots, W_{79})$ – расширенный блок, при этом

$$W_t = \begin{cases} w_t & t = 0 \ldots 15; \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1, & t = 16 \ldots 79. \end{cases}$$

Выполняются 80 раундов алгоритма (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 константа K_t.

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

$$K_t = \begin{cases} 5A827999, & t = 0\ldots19; \\ 6ED9EBA1, & t = 20\ldots39; \\ 8F1BBCDC, & t = 40\ldots59; \\ CA62C1D6, & t = 60\ldots79. \end{cases}$$

Выполняется сложение по модулю 2^{32} полученных значений a, b, c, d, e соответственно с $H0H1H2H3H4$:

$H0 = H0 + a; H1 = H1 + b; H2 = H2 + c; H3 = H3 + d; H4 = H4 + e;$

После выполнения функции $G_k(i)$ получим 160-битовое значение $H0H1H2H3H4$. Далее строится функция Γ с выходным значением длиной 32 бита, путем переиндексирования функции G $\Gamma_k(i) = H^{j}_{i \bmod 5}$.

Таблицы R, S, T определяются следующим образом: $T[i] = \Gamma_k(i), (0 \le i < 512); \; S[j] = \Gamma_k(0x1000 + j), (0 \le j < 256);$

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

Этап 2. Генерация ключевой последовательности.

Перед генерацией псевдослучайной функции $SEAL(k, n, L)$ необходимо подготовить служебные 32-битовые регистры A, B, C, D и четыре 32-битовых слова n_1, n_2, n_3, n_4. Их значения определяются из таблиц R и T, 32-битового числа n и некоторого числа l следующим образом:

$procedure \;\; Initialize(A,B,C,D,n_1,n_2,n_3,n_4)$

$A = n \oplus R[4l];$

$B = (n >>> 8) \oplus R[4l+1];$ $\qquad C = (n >>> 16) \oplus R[4l+2];$

$D = (n >>> 24) \oplus R[4l+3];$

for $j = 1$ to 2 do {

$P = A \wedge 0x7fc; \; B = B + T[P/4]; \; A = A >>> 9;$

$P = B \wedge 0x7fc; \; C = C + T[P/4]; \; B = B >>> 9;$

$P = C \wedge 0x7fc; \; D = D + T[P/4]; \; C = C >>> 9;$

$P = D \wedge 0x7fc; \; A = A + T[P/4]; \; D = D >>> 9;$

}

$(n_1, n_2, n_3, n_4) = (D; B; A; C);$

$P = A \wedge 0x7fc; \; B = B + T[P/4]; \; A = A >>> 9;$

$P = B \wedge 0x7fc; \; C = C + T[P/4]; \; B = B >>> 9;$

$P = C \wedge 0x7fc; \; D = D + T[P/4]; \; C = C >>> 9;$

$P = D \wedge 0x7fc; \; A = A + T[P/4]; \; D = D >>> 9;$

}

Построение функции $SEAL(k,n,L)$, выполняющей растяжение n в L - битную последовательность y, реализуется с использованием числа L, параметра n и таблиц R, S, T, заданных ключом k. При этом используется отображение n и l в значения внутренних переменных и регистров $A, B, C, D, n_1, n_2, n_3, n_4$.

function SEAL(k;n;L)
{
y = ∅;
for $l = 0$ to ∞ do {
Initialize($n;l;A;B;C;D;n_1;n_2;n_3;n_4$);
for $i = 1$ to 64 do {
$P = A \wedge 0x7fc;\ B = B + T[P/4];\ A = A >>> 9;\ B = B \oplus A;$
$Q = B \wedge 0x7fc;\ C = C \oplus T[Q/4];\ B = B >>> 9;\ C = C + B;$
$P = (P + C) \wedge 0x7fc;\ D = D + T[P/4];\ C = C >>> 9;\ D = D \oplus C;$
$Q = (Q + D) \wedge 0x7fc;\ A = A \oplus T[Q/4];\ D = D >>> 9;\ A = A + D;$
$P = (P + A) \wedge 0x7fc;\ B = B \oplus T[P/4];\ A = A >>> 9;$
$Q = (Q + B) \wedge 0x7fc;\ C = C + T[Q/4];\ B = B >>> 9;$
$P = (P + C) \wedge 0x7fc;\ D = D \oplus T[P/4];\ C = C >>> 9;$
$Q = (Q + D) \wedge 0x7fc;\ A = A + T[Q/4];\ D = D >>> 9;$
$y = y \,||\, B + S[4i-4] \,||\, C + S[4i-3] \,||\, D + S[4i-2] \,||\, A + S[4i-1];$
if $|y| \geq L$ then return $(y_0 y_1 \ldots y_{L-1})$;
if $odd(i)$ then $(A;B;C;D) = (A + n_1; B + n_2; C \oplus n_1; D \oplus n_2)$
else $(A;B;C;D) = (A + n_3; B + n_4; C \oplus n_3; D \oplus n_4);$
}
}
}

2. Программная реализация алгоритма SEAL

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

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

На первом этапе работы программы генерируется ключ, который обеспечивает создание уникальных таблиц, необходимых для шифрования. На рис. 1 приведен скриншот фрагмента кода генерации ключа.

```
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;
}
```

Рис. 1. Скриншот фрагмента кода генерации ключа

Далее создаются файловые потоки. Было создано три файловых потока: для входящего, шифрованного и дешифрованного файла. Скриншот фрагмента кода создания файловых потоков представлен на рис. 2.

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

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

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

Рис. 2. Скриншот фрагмента кода создания файловых потоков

Аудио файл разбивается на блоки. Запускается цикл, который будет работать до конца входящего потока. Каждый блок шифруется и дешифруется при помощи таблицы сгенерированной в алгоритме SEAL. Скриншот фрагмента шифрации и дешифрации блоков представлен на рис. 3.

```
}
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]);
}
```

Рис. 3. Скриншот фрагмента кода шифрации и дешифрации

На рис. 4 приведен скриншот фрагмента кода выполнения программы. В качестве примера была проведена шифрация аудиофайла

1.mp3. Дешифрованный файл 2.mp3 был сохранен в той же папке, что и исходный (рис. 5).

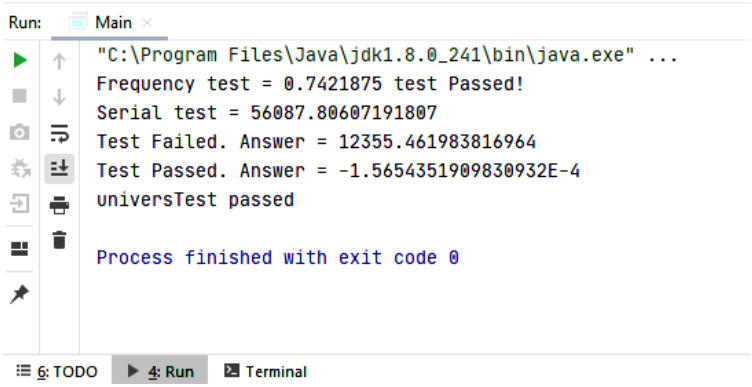

Рис. 4. Скриншот фрагмента кода выполнения программы

Рис. 5. Исходный, дешифрованный и шифрованный файлы

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. Поточные шифры / А. В. Асосков, М. А. Иванов [и др.]. – М.: КУДИЦ-ОБРАЗ, 2003. – 336 с.