УДК 681.322

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПОТОКОВОЙ СИСТЕМЫ ШИФРОВАНИЯ A5

Керножицкий А.В., Беть Е.А.
Научный руководитель – к.т.н., доцент Несенчук А.А.

Потоковая система шифрования A5 представляет собой поточный шифр, используемый для обеспечения конфиденциальности передаваемых данных между телефоном и базовой станцией в европейской системе мобильной цифровой связи GSM (Group Special Mobile).

В этом алгоритме каждому символу открытого текста соответствует символ шифротекста [1]. Текст не делится на блоки (как в блочном шифровании)[1]и не изменяется в размере. используются логические функции, сложение по модулю 2 (XOR) и сдвиг регистра.

Поскольку A5 широко применяется в сетях GSM, его можно найти в большом количестве мобильных устройств по всему миру. Это делает его удобным и совместимым стандартом. Хотя A5 не является идеальным шифром, он обеспечивает базовый уровень защиты от прослушивания, что делает его сложным для нелегального доступа к разговорам. A5 предназначен только для шифрования данных, но не предоставляет механизмы аутентификации. Это означает, что, несмотря на то, что данные могут быть зашифрованы, не существует гарантии, что они отправлены именно ожидаемым отправителем.

Функционирование алгоритма разделяется на два основных этапа:

1). Генерация гаммы шифра (ключевой последовательности);

2). Непосредственно реализация процедуры шифрования. исходного текста.

1. Генерация гаммы шифра в системе A5

Для генерации гаммы шифра (ключевой последовательности или ключаK(Key)) используется генератор в форме регистра сдвига с линейными обратными связями (РСЛОС) [1,2].

1.1. Регистр сдвига с линейными обратными связями и его функционирование

Ключ генерируется в форме бинарного ключевого потока (ключевой последовательности, т.е. последовательности битов ключа):$k_i \epsilon K$.

Математической моделью генератора ключа в форме РСЛОС является полиномиальная модель, а именно модель в форме полинома заданной степени n, который в общем виде представим формулой

$$a_0 x^n \oplus a_1 x^{n-1} \oplus a_2 x^{n-2} \oplus \ldots \oplus a_{n-1} x \oplus a_n = p(x) \qquad (1)$$

где a_j – коэффициенты полинома,$a_j \in \{0,1\}$,$j = 1, 2, ..., n$; $\oplus$ – операция XOR.

Разрядность РСЛОС должна соответствовать степени nего полиномиальной математической модели (1), а моделирующий полином (1) должен быть примитивным по модулю 2[1]. Это делается с целью повышения криптостойкости шифра, поскольку генерируемая с использованием примитивного полинома последовательность, называемая М-последовательностью, всегда имеет максимальный период$T=2^n–1$. Например примитивный полином при $n = 8$ имеет вид

$$1 \oplus x \oplus x^5 \oplus x^6 \oplus x^8 = p(x). \quad (2)$$

Каждое слагаемое математической модели регистра (1) соответствует разряду регистра, номер которого определяется степенью неизвестного данного слагаемого.

В соответствии с математической моделью (1) разряды регистра (их содержимое), номера которых определяются степенями неизвестного моделирующего полинома, должны складываться по модулю 2. Назовем эти разряды *отводными разрядами* регистра. С целью сложения содержимого этих разрядов от них выполняются отводы, формирующие отводную последовательность (ОП) РСЛОС.

8-ми разрядный РСЛОС, моделируемый полиномом (2), показан на рисунке 1.

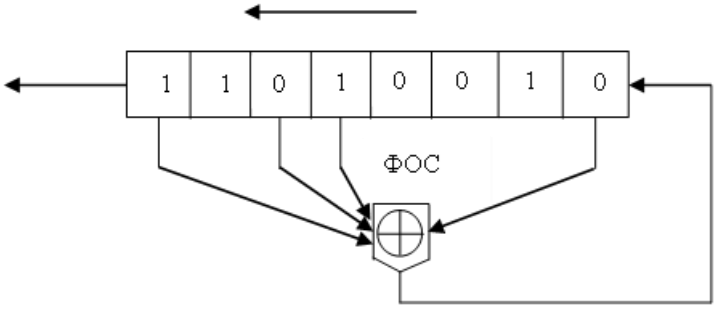

Рис. 1. 8-ми разрядный сдвиговый регистр РСЛОС

В нижней части рисунка четырьмя линиями со стрелками показана отводная последовательность регистра. Поскольку моделью регистра, показанного на рисунке 1, является примитивный полином (2), ОП определяется от 1, 5, 6 и 8 разрядов регистра. Содержимое соответствующих разрядов подвергается обработке с помощью функции, называемой функцией обратной связи (ФОС), которая в данном случае представляет собой логическую функцию XOR (сложение по модулю 2) (рисунок 1).

Результат сложения содержимого отводных разрядов регистра по линии обратной связи (ОС) поступает в первый разряд регистра и в этот момент происходит сдвиг содержимого регистра на один разряд влево (рисунок 1). Таким образом очередной бит выходит из регистра, формируя очередной бит гаммы шифра (ключевой последовательности). Количество генерируемых бит гаммы шифра (длина ключевой последовательности)

должно равняться длине в битах потока исходного текста, подлежащего шифрованию.

Следует отметить, что перед началом работы РСЛОС необходимо проинициализировать произвольным ненулевым начальным вектором*IV*. На рисунке 1 это вектор*IV*: 11010010.

1.2. Функционирование генератора A5

Практическое применение имеют системы регистров переменного тактирования, с различными длинами и функциями обратной связи, реализуемые в A5. Генератор A5 состоит из двух основных частей: трех РСЛОС и блока управления движением.

Три регистра (*R*1, *R*2, *R*3) имеют соответственно длины 19, 22 и 23 бита. Многочлены обратных связей для *R*1, *R*2, *R*3 соответственно равны:

$$f_1(x) = x^{19} + x^5 + x^2 + x + 1$$
$$f_2(x) = x^{22} + x + 1$$
$$f_3(x) = x^{23} + x^{15} + x^2 + x + 1$$

Структурная схема потоковой криптосистемы A5, изображена на рисунке 2.

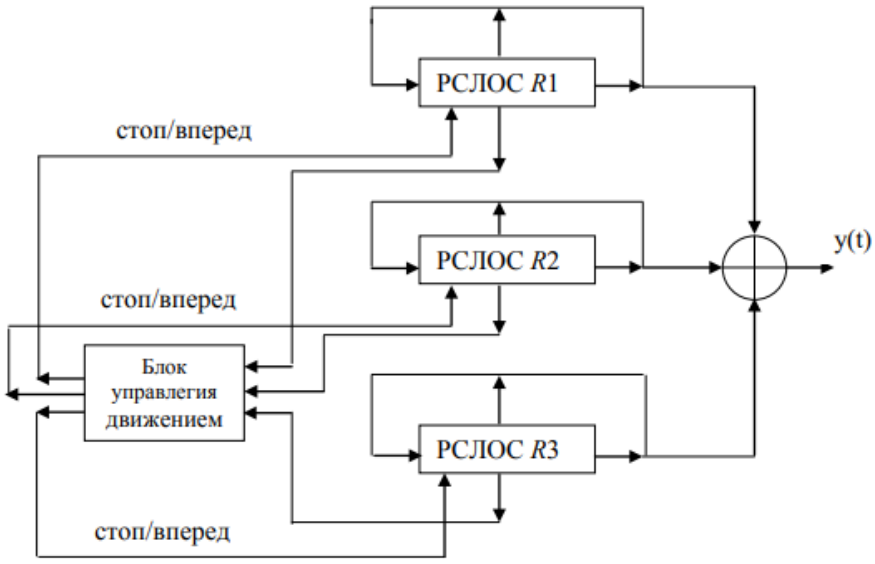

Рис. 2. Генерация ключа в криптосистеме A5

Управление тактированием осуществляется специальным механизмом в блоке управления движением (рисунок 2). В каждом регистре есть биты синхронизации: τ_1 = 10 (*R*1), τ_2 = 11 (*R*2), τ_3 = 12 (*R*3). Вычисляется логическая функция

$$F(\tau_1,\tau_2,\tau_3) = \tau_1 \& \tau_2 | \tau_1 \& \tau_3 | \tau^2 \& \tau_3, \quad (3)$$

где & – булево AND, | – булево OR, а τ_1,τ_2 и τ_3 – биты синхронизации R1, R2 и R3 соответственно.

Сдвигаются только те регистры, у которых бит синхронизации τ_i(синхробит) равен значению функции (3).

Фактически, сдвигаются только те регистры, синхробит которых принадлежит большинству. Выходной бит системы $y(t)$ – результат операции XOR над выходными битами регистров.

2. Реализация процесса шифрования/ дешифрования

2.1. Преобразование «инволюция»

Потоковое шифрование основано на использовании преобразования, называемого «инволюция» [1,2]. Инволюция – это преобразование (функция), которое является обратным самому себе. Функция $f(x)$является инволюцией, если

$$f\big(f(x)\big) = x.$$

Пример 1:

$$f(x) = -x,$$
$$f\big(f(x)\big) = -(-x) = x.$$

Пример 2:

$$f(x) = x \oplus b,$$
$$f\big(f(x)\big) = (x \oplus b) \oplus b = x.$$

2.2. Процесс шифрования / дешифрования

Потоковое шифрование и дешифрование основано на использовании преобразования «инволюция» (см. подраздел 2.1). Синхронный потоковый шифратор имеет структуру, включающую начальное состояние генераторов ключа и функционирует синхронно. Каждый символ шифротекста вычисляется как функция XOR от соответствующих символов исходного текста и ключа (рисунок 3).

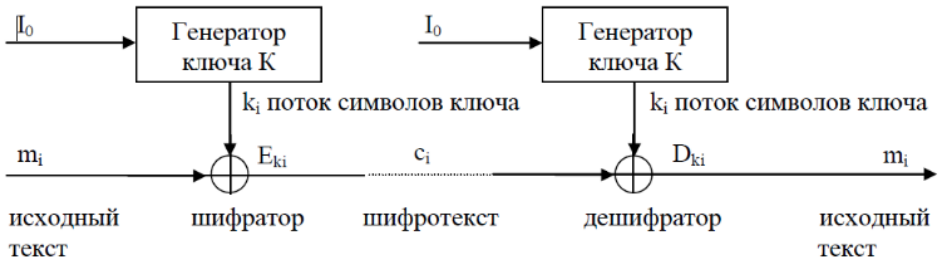

Рис. 3. Структура синхронного потока шифратора/ дешифратора

Алгоритм шифрования в потоковой системе описывается следующей формулой шифрования:

$$C = P \oplus K,$$

где C – бинарный поток шифротекста, P– бинарный поток исходного текста.

Благодаря тому, что функция XOR является инволюцией, алгоритм дешифрования в потоковой системе описывается формулой дешифрования, которая аналогична формуле шифрования, т.е. формулой

$$P = C \oplus K.$$

2.3. Аппаратная и программная реализация потокового шифра

Потоковые шифры могут быть реализованы в аппаратном или программном варианте.

Аппаратная реализация потокового шифра обеспечивает высокую криптостойкость именно по той причине, что при аппаратной реализации формируется истинно случайная бинарная последовательность. Однако недостатком является высокая стоимость реализации.

К достоинствам программной реализации потоковых систем шифрования относятся низкая стоимость реализации в сравнении с аппаратной и высокая гибкость. Однако программно можно генерировать только псевдослучайную бинарную последовательность, которая обеспечивает значительно более низкую криптостойкость шифра в сравнении с истинно случайной.

3. Программная реализация потоковой системы A5

3.1. Программа «Потоковая система шифрования A5»

В данной статье представлена программа, реализующая шифросистему A5 и разработанная на языке C#.Использована среда разработки VisualStudio 2022. Основным компонентом программы является класс A5Cipher, который реализует алгоритм потокового шифрования шифра A5. Для генерации ключевой последовательности (гаммы) в программе используются генераторы на основе сдвиговых регистров с линейными обратными связями.

3.2. Реализация интерфейса пользователя

С использованием WindowsForms, программа предоставляет удобный интерфейс для пользователей (рисунок 4), позволяющий задать исходный текст и сгенерировать ключ, а затем производить шифрование и дешифрование с использованием системы A5.

Интерфейс разделен на 2 блока: стороны отправителя (шифрования) в левой и получателя (дешифрования) в правой половине интерфейса.

В окне «Исходный текст» вводится текст для шифрования, под которым для каждого регистра можно задать степени полиномов и синхробиты τ_i. В полях IV_i задаются случайным образом начальные состояния регистров. Отводы регистров задаются в полях $Taps_l$. Для генерации ключа требуется нажать кнопку «Генерировать ключ» на стороне отправителя.

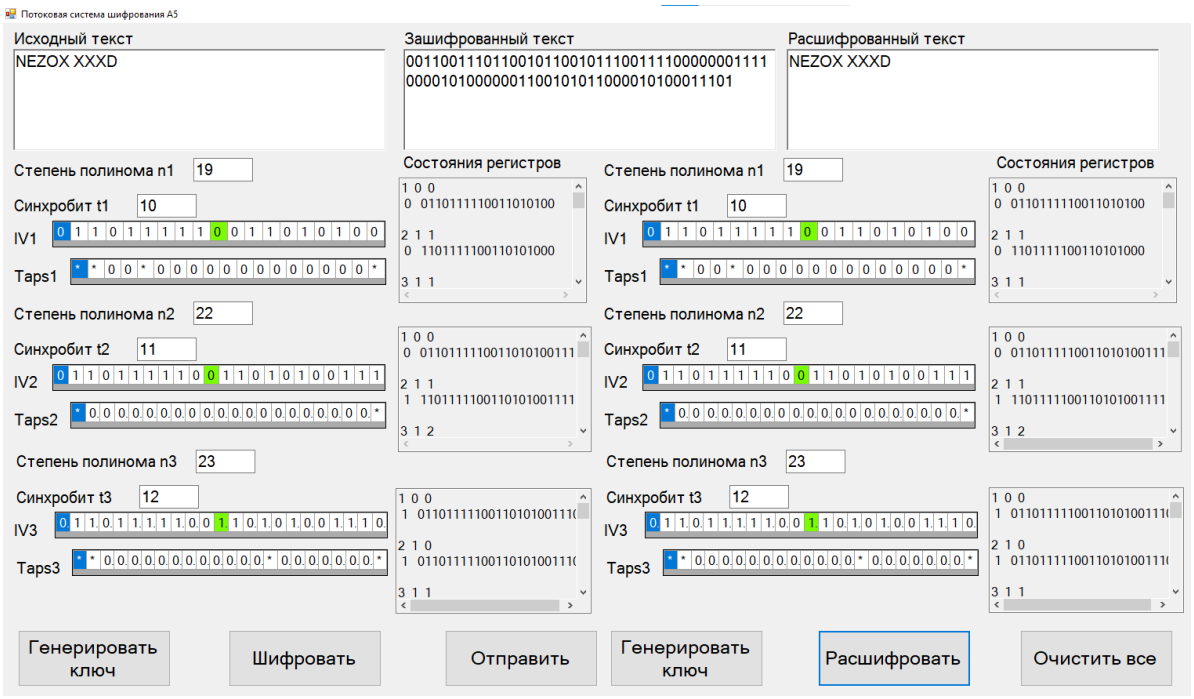

Рис. 4. Интерфейс программы потокового шифрования системв A5

В полях «Состояния регистров» выводятся состояния регистров в процессе функционирования при генерации ключевой последовательности. Первое число – номер итерации, далее идет значение логической функции $F(\tau_1,\tau_2,\tau_3)$, после него состояние регистра на данном шаге и затем количество сдвигов регистра до текущего шага. По нажатии кнопки «Шифровать» в окне «Зашифрованный текст» выводится шифротекст.

По нажатии кнопки «Отправить» на стороне получателя происходит заполнение всех полей данными, необходимыми для генерации ключа дешифрования, которые полностью аналогичны данным шифрования. Дешифрование шифротекста производится аналогично шифрованию.

В результате выполнения работы реализован программный генератор псевдослучайной бинарной последовательности, шифратор и дешифратор в структуре потоковой системы A5. разработан удобный пользовательский интерфейс системы A5 с возможностью визуализации состояния генераторов в процессе функционирования, генерируемой ключевой последовательности и шифротекста. Это позволяет следить за состоянием генераторов и процессом формирования гаммы шифра, делает процесс шифрования более удобным и понятным для пользователей.

Литература

1. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных сетях. М.: Радио и связь, 2001.

2. Чмора А.Л. Современная прикладная криптография. М.: Гелиос АРВ, 2002.

УДК 004.652.4+004.6

ОПТИМИЗАЦИЯ ХРАНЕНИЯ И ДОСТУПА К УЧЕБНЫМ ДАННЫМ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ В ОБРАЗОВАНИИ

Бондаревич Б.Б. Дульский М.С.
Научный руководитель – Воронич Л.В. ассистент

В современном образовательном процессе информационные технологии играют ключевую роль, предоставляя учреждениям образования эффективные инструменты для хранения, управления и доступа к учебным данным. Реляционные базы данных являются одним из наиболее распространенных инструментов для организации информации в образовательных учреждениях. Оптимизация процесса хранения и доступа к учебным данным в реляционных базах данных имеет решающее значение для повышения эффективности образовательного процесса.

Анализ существующей проблемы

В образовательных учреждениях объемы учебных данных постоянно растут, включая информацию о студентах, учебных планах, оценках, учебных материалах и прочее. С увеличением объемов данных возникают проблемы с производительностью и эффективностью доступа к ним. Стандартные методы хранения и доступа могут стать неэффективными при масштабировании системы.

Предлагаемое решение

Для оптимизации хранения и доступа к учебным данным в реляционных базах данных предлагается следующий подход:

1. **Нормализация данных**: Эффективное использование нормализации позволяет уменьшить дублирование данных и снизить объем хранимой информации. Нормализация помогает избежать аномалий данных и обеспечивает целостность информации.[2]
2. **Индексирование**: Создание индексов на ключевых полях ускоряет процесс поиска и извлечения данных. Индексы позволяют снизить время выполнения запросов к базе данных.[2]
3. **Оптимизация запросов**: Анализ и оптимизация SQL-запросов позволяет снизить нагрузку на базу данных и ускорить выполнение запросов.[3]
4. **Использование кэширования**: Введение кэширования данных позволяет снизить время доступа к часто используемым данным, уменьшая нагрузку на базу данных.[1]