МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Защита информации и надежность информационных систем»

Тема: «Сравнительный анализ производительности и безопасности симметричных алгоритмов шифрования XXTEA и CAST»

Исполнитель

Студент 4 курса группы 7 Голубицкая Ю. В.

(Ф.И.О.)

Руководитель работы Сазонова Д. В. (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2024

Оглавление

[Введение 3](#_Toc180747005)

[1. Постановка задачи 4](#_Toc180747006)

[2. Описание методов 5](#_Toc180747007)

[3. Описание программного средства 10](#_Toc180747008)

[4. Тестирование программного средства 11](#_Toc180747009)

# Введение

С развитием цифровых технологий и увеличением объема передаваемой информации, вопросы безопасности данных становятся все более актуальными. Шифрование информации – один из ключевых методов защиты, позволяющий предотвратить несанкционированный доступ к конфиденциальным данным. Среди множества алгоритмов шифрования особое внимание уделяется симметричным алгоритмам, которые используют один и тот же ключ как для шифрования, так и для расшифрования данных.

В данной работе будет проведен сравнительный анализ производительности и безопасности двух симметричных алгоритмов шифрования – XXTEA и CAST. Выбор этих алгоритмов обусловлен их распространенностью и различиями в подходах к шифрованию. XXTEA, являющийся усовершенствованной версией алгоритма TEA, отличается высокой скоростью работы и простотой реализации. В то же время, алгоритм CAST, разработанный с акцентом на безопасность, обеспечивает надежную защиту данных благодаря использованию различных режимов шифрования и длины ключа.

Результатом данной курсовой работы является проект, содержащий в себе реализацию двух алгоритмов, и позволяющий сравнить их по некоторым признакам.

# Постановка задачи

В рамках данного курсового проекта необходимо провести сравнительный анализ производительности и безопасности симметричных алгоритмов шифрования XXTEA и CAST. Для достижения этой цели следует рассмотреть ряд задач:

Требование к средствам разработки:

* Язык программирования JavaScript.
* Средство разработки MS Visual Studio Code.

Приложение должно:

* Приложение должно шифровать и расшифровывать текстовые сообщения.
* Выводить время, затраченное на шифрование и расшифрование, что позволит оценить производительность каждого из алгоритмов.
* Реализовать удобный интерфейс для ввода данных и отображения результатов, включая возможность выбора алгоритма шифрования.

Кроме того, необходимо сравнить результаты тестирования по производительности и безопасности, а также сформулировать выводы о том, какой из алгоритмов лучше подходит для определенных условий и задач.

# Описание методов

Выбранные для сравнения алгоритмы шифрования относятся к классу блочно-симметричных. Блочно-симметричные шифры – это класс криптографических алгоритмов, которые шифруют данные блоками фиксированного размера, например, 128, 192 или 256 бит. Они используют симметричное шифрование, что означает, что для шифрования и дешифрования применяется один и тот же ключ.

Основной принцип работы блочно-симметричных шифров заключается в разбиении данных на блоки одинакового размера. Если размер данных не кратен размеру блока, используется механизм дополнения. На каждом раунде ключ преобразуется с помощью функции ключа, что обеспечивает дополнительную безопасность. Каждый блок проходит через несколько раундов, где применяются операции замены, перестановки и смешивания данных.

XXTEA – криптографический алгоритм, реализующий блочное симметричное шифрование и представляющий собой сеть Фейстеля. Является расширение алгоритма Block TEA. Разработан и опубликован Дэвидом Уилером и Роджером Нидхемом в 1998 году. Выполнен на простых и быстрых операциях: XOR, подстановка, сложение.

XXTEA, как и остальные шифры семейства TEA, обладает рядом отличительных особенностей по сравнению с аналогичными шифрами:

* высокая скорость работы;
* малое потребление памяти;
* простая программная реализация;
* относительно высокая надёжность.

Данный алгоритм работает по следующему принципу:

Исходный текст разбивается на слова по 32 бита каждый, из полученных слов формируется блок. Ключ также разбивают на 4 части, состоящие из слов по 32 бита каждый, и формируют массив ключей. В ходе одного раунда работы алгоритма шифруется одно слово из блока. После того, как были зашифрованы все слова, заканчивается цикл, и начинается новый. Количество циклов (раундов) зависит от количества слов и вычисляется по формуле:

R = 6 + 52/*n,*  (1.1)

где *n* – количество слов.

Шифрование одного слова состоит в следующем:

1. Над левым соседом выполняется операция битового сдвига влево на два, а над правым операция битового сдвига вправо на пять. Над полученными значениями выполняют операцию побитового сложения по модулю 2.
2. Над левым соседом выполняется операция битового сдвига вправо на три, а над правым операция битового сдвига влево на 4. Над полученными значениями выполняют операцию побитового сложения по модулю 2
3. Полученные числа складывают по модулю 232.
4. Константа *δ*, выведенная из Золотого сечения *δ* = (5 - 1) · 231 = 2654435769 = = 9E3779B9h, умножается на номер цикла (это было сделано для предотвращения простых атак, основанных на симметрии раундов).
5. Полученное в предыдущем пункте число складывают побитово по модулю 2 с правым соседом.
6. Полученное в 4 пункте число сдвигают побитово направо на 2, складывают побитово по модулю два с номером раунда и находят остаток от деления на 4. С помощью полученного числа выбирают ключ из массива ключей.
7. Выбранный в предыдущем раунде ключ складывают побитово по модулю 2 с левым соседом.
8. Числа, полученные в предыдущем и 4 пунктах, складывают по модулю 232.
9. Числа, полученные в предыдущем и 3 пунктах, складывают побитово по модулю 2, данную сумму складывают с шифруемым словом по модулю 232.

Схема алгоритма представлена на рисунке 1.1.

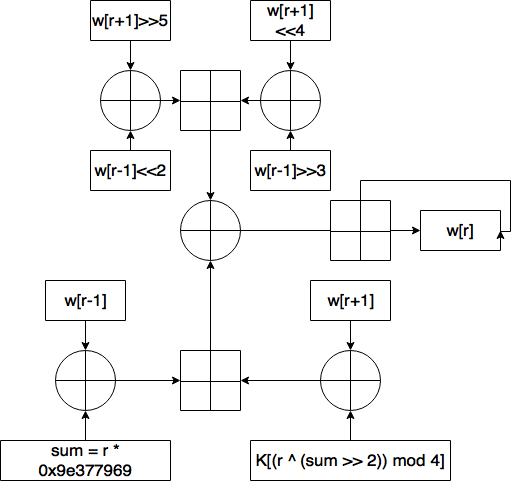


Рисунок 1.1 – Схема алгоритма XXTEA

Для расшифровки сообщений, зашифрованных с помощью XXTEA, необходимо взять зашифрованный текст и ключ, использованный при шифровании. Сначала зашифрованный текст обычно преобразуют из формата, например, Hex или Base64, в бинарный вид. Затем применяют алгоритм XXTEA, используя этот ключ, чтобы выполнить процесс расшифровки. В результате получается оригинальный текст. Важно, чтобы ключ был правильным, иначе расшифровка не будет успешной.

На данный момент для XXTEA существует атака на основе адаптивно подобранного открытого текста, опубликованная Элиас Яаррков в 2010 году. Существует два подхода, в которых используется уменьшение количества циклов за счет увеличения количества слов.

Первый подход дифференциального криптоанализа XXTEA заключается в шифровании нескольких слов из открытого текста, начиная с одного слова и добавляя к нему новое значение. Если после шифрования различия сохраняются только в одном слове, процесс продолжается. Элиас Яаррков обнаружил, что вероятность успешного прохождения пяти полных циклов колебалась между 2-109 и 2-110 для большинства ключей. Это означает, что, если пара текстов прошла 5 из 6 циклов, её можно считать верной; при этом возникновение различий в других словах указывает на необходимость повторного подбора. Схема первого подхода показана на рисунке 1.2.

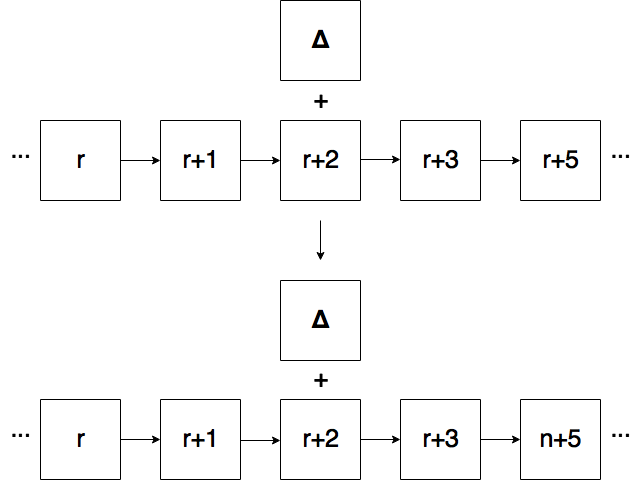


Рисунок 1.2 – Схема первого подхода криптоанализа XXTEA

Второй подход отличается тем, что после первого раунда разница должна перейти в текущее слово из следующего, изменив своё значение. Затем, после следующего раунда, разница возвращается к исходному слову, но со знаком, отличающимся от первоначального. Яаррков выяснил, что для успешного нахождения правильной пары достаточно 259 текстов с разницей в пределах от -32 до 32. Увеличение этого диапазона не улучшает результаты. Эта атака на XXTEA была успешной даже при уменьшении количества циклов до четырёх, где правильная пара была получена с помощью 235 пар текстов, что соответствует предыдущей оценке в 234,7 пар. Схему второго подхода можно увидеть на рисунке 1.3.

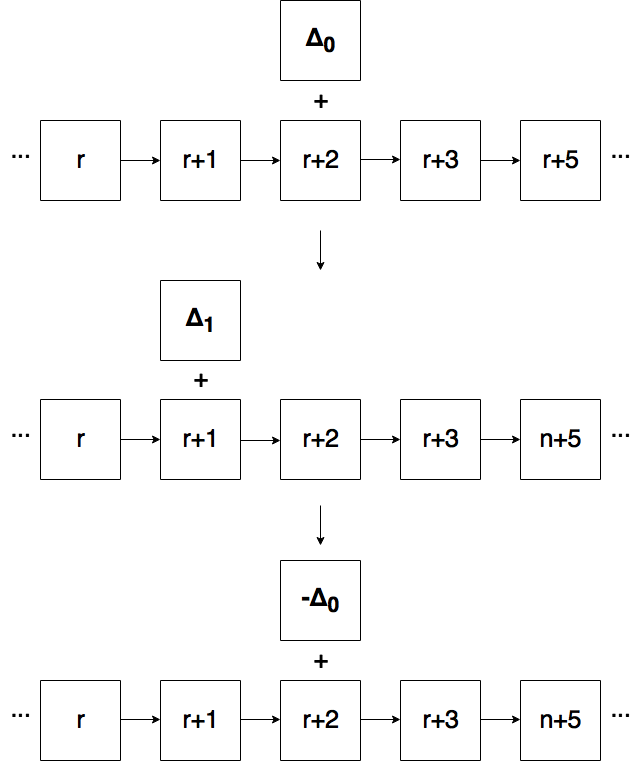


Рисунок 1.3 – Схема второго подхода криптоанализа XXTEA

CAST-128 (или CAST5) в криптографии – блочный алгоритм симметричного шифрования на основе сети Фейстеля, который используется в целом ряде продуктов криптографической защиты, в частности некоторых версиях PGP и GPG, и, кроме того, одобрен для использования Канадским правительством.

Алгоритм был создан в 1996 году Карлайлом Адамсом и Стаффордом Таваресом используя метод построения шифров CAST, который используется также и другим их алгоритмом CAST-256.

CAST-128 состоит из 12 или 16 раундов сети Фейстеля с размером блока 64 бита и длиной ключа от 40 до 128 бит (но только с инкрементацией по 8 бит). 16 раундов используются, когда размеры ключа превышают 80 бит. В алгоритме используются 8 на 16 S- блоки, основанные на бент-функции, операции XOR и модулярной арифметике. Есть три различных типа функций раундов, но они похожи по структуре и различаются только в выборе выполняемой операции в различных местах.

Хотя CAST-128 защищен патентом Entrust, его можно использовать во всём мире для коммерческих или некоммерческих целей бесплатно.

CAST-128 основан на сети Фейстеля. Полный алгоритм шифрования изложен в следующих четырех шагах:

ВХОД: текст *m1* ... *m64*, ключ *K* = *k1* ... *k128*.

ВЫХОД: зашифрованный текст *c1* ... *c64*.

1. Развертка ключа cоставляет 16 пар подключей {*Kmi, Kri*} полученных из *K*. CAST-128 использует пару подключей за раунд: 32-битные величины *Km* используется в качестве «маскировки» ключа и *Kr* используют как «перестановки» ключа, из которых используются только начальные 5-бит.
2. Разделяет текст на левую *L0* и правую *R0* 32-битные половины *L0* = *m1* ... *m32* и *R0* = *m33* ... *m64*).
3. 16 раундов вычисляется *Li* и *Ri* следующим образом: *Li* = *Ri-1*; *Ri* =   
   = *Li-1*F (*Ri-1*,*Kmi*,*Kri*), где *F* имеет тип 1, тип 2, тип 3 или, в зависимости от *i*. Три различных типов функции *F* используются в CAST-128. Типы выглядит следующим образом (где «*D*» является входными данными в функцию *F* и «*Ia*» – «*Id*» является наиболее значимый байт – наименее значимый байт *I*, соответственно). Нужно обратить внимание, что «+» и «-» сложение и вычитание по модулю 232, «^» является побитовое XOR и «<<<» является циклическим сдвигом влево. Таблица, которая показывает неидентичные раунды продемонстрирована под пунктом 1.

Таблица 1 – Неидентичные раунды

|  |  |
| --- | --- |
| Раунды 1,4,7,10,13,16 | *I = ((Kmi + Ri-1) <<< Kri)*  *F = ((S1[Ia] ^ S2[Ib]) – (S3[Ic]) )+ S4[Id]* |
| Раунды 2,5,8,11,14 | *I = ((Kmi ^ Ri-1) <<< Kri)*  *F = ((S1[Ia] - S2[Ib]) + (S3[Ic])) ^ S4[Id]* |
| Раунды 3,6,9,12,15 | *I = ((Kmi - Ri-1) <<< Kri)*  *F = ((S1[Ia] + S2[Ib]) ^ (S3[Ic]))- S4[Id]* |

CAST-128 использует восемь полей замены: поля *S1, S2, S3* и *S4* раундовые функции полей замены, *S5, S6, S7* и *S8* являются ключами развертки полей замены. Несмотря на то, что 8 полей замены требуют в общей сложности 8 Кбайт для хранения, только 4 Кбайта требуются во время фактического шифрования или дешифрование, так как генерация подключа обычно делается до любого ввода данных.

1. Меняем окончательные блоки местами *L16*, *R16* и объединяем, чтобы сформировать зашифрованный текст.

Расшифрование совпадает с алгоритмом шифрования, приведенным выше, кроме того, что раунды (и, следовательно, пары подключей), используются в обратном порядке, чтобы вычислить (*L0, R0*) из (*R16, L16*). Схема алгоритма представлена на рисунке 1.4.

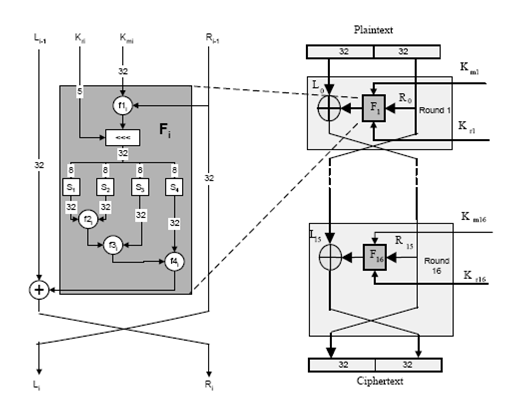


Рисунок 1.4 – Схема алгоритма CAST-128

Существует несколько подходов к криптоанализу CAST-128:

* Дифференциальный криптоанализ: Этот метод основан на анализе различий между парами зашифрованных текстов, чтобы выявить закономерности. В случае CAST-128 были проведены исследования, показывающие, что при определенных условиях можно уменьшить количество возможных ключей.
* Линейный криптоанализ: Этот метод использует линейные связи между открытым текстом, зашифрованным текстом и ключом. Исследования показывают, что с использованием большого количества пар текстов можно упростить задачу нахождения ключа.
* Атаки на основе статистического анализа: Криптоаналитики могут использовать частотный анализ и другие статистические методы, чтобы выявить слабые места в шифре.

Несмотря на некоторые уязвимости, CAST-128 считается достаточно безопасным для большинства приложений, особенно при использовании длинных ключей.

# Описание программного средства

В данной курсовой работе было реализовано веб-приложение, реализующее симметричное шифрование и дешифрование с использованием алгоритмов XXTEA и CAST.

Используются следующие языки и технологии:

* JavaScript/TypeScript: Основной язык программирования для серверной части приложения.
* Node.js: Среда выполнения для JavaScript на сервере.
* Express.js: Веб-фреймворк для Node.js, используемый для создания серверных маршрутов.
* EJS (Embedded JavaScript): Шаблонизатор для генерации HTML на сервере.

Помимо этого, были использованы и различные библиотеки:

* xxtea-node: Библиотека для работы с алгоритмом XXTEA.
* node-forge: Библиотека для работы с криптографией, в данном случае используется для кодирования в Base64.
* crypto: Стандартная библиотека Node.js, используемая для шифрования и дешифрования данных.

Разработанное веб-приложение позволяет выполнять следующие действия:

* Шифровать и расшифровывать текст с использованием двух различных алгоритмов:
* Проводить сравнительный анализ производительности между двумя алгоритмами. Для каждого алгоритма приложение измеряет время, затраченное на шифрование и расшифрование.
* Отображать результаты в удобном формате на веб-странице, где пользователи могут видеть: зашифрованный текст, расшифрованный текст и время, затраченное на каждую операцию.

Структура созданного приложения, следующая:

* app.ts: Основной файл, содержащий настройки сервера и маршруты.
* cast.ts: Модули для шифрования и дешифрования с использованием алгоритма CAST.
* xxtea.ts: Модули для шифрования и дешифрования с использованием алгоритма XXTEA.
* cp.ejs: HTML-шаблон, который предоставляет интерфейс для взаимодействия с пользователями.

Интерфейс включает в себя форму, где пользователи могут ввести оригинальное сообщение и ключ для шифрования. Результаты отображаются в виде таблицы с двумя колонками для каждого алгоритма, что позволяет легко сравнивать их производительность и результаты.

# Тестирование программного средства