**Федеральное государственное автономное образовательное учреждение высшего образования**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»**

**кафедра «Инфокоммуникационных технологий»**

**КУРСОВАЯ РАБОТА**

**по «Информационная безопастность»**

**тема: Криптографический алгоритм KASUMI**

Студент, ФИО: Мангушев Ильдар Надирович

Группа: БИСТ-21-1

Преподаватель: Бахаров Леонид Ефимович

**Москва** **2023г.**

**Введение:**

Оглавление

[Историческая справка 2](#_Toc152085985)

[Криптоанализ 3](#_Toc152085986)

[Описание алгоритма 4](#_Toc152085987)

[Рассмотрение блочных схем 10](#_Toc152085988)

[Структура кода 13](#_Toc152085989)

[Вывод 17](#_Toc152085990)

[Список литературы 17](#_Toc152085991)

# **Историческая справка**

Этот разработанный относительно недавно алгоритм уже имеет достаточно интересную историю и, видимо, будет широко применяться в будущем.   
В основе алгоритма KASUMI лежит алгоритм MISTY1 —   
один из алгоритмов семейства MISTY, разработанного в 1996–1997 гг. японской корпорацией Mitsubishi Electric. Впоследствии алгоритм MISTY1   
стал одним из победителей европейского конкурса криптоалгоритмов   
NESSIE. В декабре 1998 г. был образован консорциум 3GPP (3rd Generation Partnership Project), целью которого являлась разработка стандартов мобильной связи третьего поколения. Одно из направлений стандартизации— разработка.   
протоколов и алгоритмов аутентификации и защиты данных в сетях сотовой   
связи.

Рабочая группа консорциума 3GPP — SAGE (Security Algorithms Group of Experts, группа экспертов по алгоритмам безопасности) — проанализировала ряд известных на тот момент алгоритмов шифрования, и в качестве кандидата будущего стандарта шифрования данных в сетях сотовой связи был выбран алгоритм блочного симметричного шифрования MISTY1, шифрующий 64-битные блоки данных 128-битным ключом. Поскольку к тому времени Описание алгоритмов   
были известны некоторые теоретические атаки на MISTY1, алгоритм был   
доработан с целью его адаптации для аппаратной реализации (в том числе   
в условиях ограниченных вычислительных ресурсов), а также усиления против некоторых видов криптоаналитических атак, в том числе:

* Атак «встреча посередине»;
* Дифференциального криптоанализа и ряда его вариантов;
* Атак, использующих наличие слабых ключей или классов ключей;
* Линейного криптоанализа.

В результате доработки MISTY1 был получен алгоритм KASUMI, незначительно отличающийся по своей структуре от MISTY1. Подробная спецификация KASUMI появилась в конце декабря 1999 г. Слово «kasumi» —   
это перевод на японский английского слова «misty» («туманный»).   
Следует учесть, что для использования алгоритма KASUMI необходимо по-  
лучение лицензии у корпорации Mitsubishi Electric.

# **Криптоанализ**

В 2001 году Кюн (2001) представил невозможную дифференциальную атаку в шести раундах КАСУМИ.

В 2003 году Элад Баркан, Эли Бихам и Натан Келлер продемонстрировали атаки типа "человек посередине" на протокол GSM, которые позволяют обойти шифр A5/3 и, таким образом, нарушить протокол. Однако этот подход не атакует шифр A5/3. Полная версия их статьи была опубликована позже в 2006 году.

В 2005 году израильские исследователи Эли Бихам, Орр Дункельман и Натан Келлер опубликовали атаку КАСУМИ на прямоугольник с соответствующим ключом (бумеранг), которая может пройти все 8 раундов быстрее, чем исчерпывающий поиск. Для атаки требуется 254,6 выбранных открытых текста, каждый из которых был зашифрован одним из четырех связанных ключей, и имеет временную сложность, эквивалентную 276,1 шифрованиям KASUMI. Хотя это не практическая атака, она сводит на нет некоторые доказательства безопасности протоколов 3GPP, которые основывались на предполагаемой силе KASUMI.

В 2010 году Данклеман, Келлер и Шамир опубликовали новую атаку, которая позволяет восстановить полный ключ A5/3 с помощью атаки с использованием связанного ключа. Временные и пространственные сложности атаки достаточно малы, чтобы авторы провели атаку в течение двух часов на скромном настольном компьютере, даже используя неоптимизированную эталонную реализацию KASUMI. Авторы отмечают, что эта атака может быть неприменима к способу использования A5/3 в системах 3G; их основной целью было дискредитировать заверения 3GPP в том, что их изменения в MISTY существенно не повлияют на безопасность алгоритма.

# **Описание алгоритма**

Алгоритм KASUMI указан в технической спецификации 3GPP.  KASUMI — это блочный шифр со 128-битным ключом и 64-битным вводом и выводом. Ядром KASUMI является восьмиуровневая сеть Фейстела. Функции round в основной сети Фейстела представляют собой необратимые сетевые преобразования, подобные Файстелю. В каждом раунде функция round использует ключ round, который состоит из восьми 16-битных вспомогательных ключей, полученных из исходного 128-битного ключа с использованием фиксированного расписания ключей.

Расписание ключей 128-битный ключ K разделен на восемь 16-битных вспомогательных ключей Ki:

Дополнительно используется модифицированный ключ K', аналогично разделенный на 16-битные подключи K'i. Модифицированный ключ получен из исходного ключа путем XORing с помощью [0x123456789ABCDEFFEDCBA9876543210](https://cryptography.fandom.com/wiki/Nothing_up_my_sleeve_number).

Круглые ключи получаются либо из вспомогательных ключей путем побитового поворота влево на заданную величину, либо из модифицированных вспомогательных ключей (без изменений).

Круглые ключи, следующие:

Добавление индекса подключа происходит циклически, так что, если i + j больше 8, нужно вычесть 8 из результата, чтобы получить фактический индекс подключа.

Алгоритм КАСУМИ обрабатывает 64-битное слово в двух 32-битных частях, левой (Li) и правой (Ri). Входное слово - это объединение левой и правой половин первого раунда:

input=L0||R0.

В каждом раунде правая половина преобразуется с помощью выходных данных функции round, после чего половины меняются местами:

Li=Fi(KLi,KOi,KIi,Li-1)⊕Ri-1

Ri=Li-1

где KLi, KOi, KIi - круглые ключи для i-го раунда.

Функции раундов для четных и нечетных раундов немного отличаются. В каждом случае функция round представляет собой комбинацию двух функций FLi и FOi. Для нечетного раунда

и для ровного раунда

Результат — это объединение результатов последнего раунда.

output=R8||L8.

Обе функции FL и FO разделяют 32-битные входные данные на две 16-битные половины. Функция FL представляет собой необратимую манипуляцию битами, в то время как функция FO представляет собой необратимую сеть, подобную Файстелю, состоящую из трех раундов.

Функция FL

32-битный ввод x из FL(KL,x) разделен на две 16-битные половины x=l||r.  Сначала левая половина входных данных , обрабатывается побитово круглой клавишей KLi,1 и поворачивается влево на один бит. Результат этого преобразуется в XOR к правой половине входных данных, r чтобы получить правую половину выходных данных r′.

Затем правая половина выходных данных r′ преобразуется побитово с помощью круглой клавиши KLi,2 и поворачивается влево на один бит. Результат этого преобразуется в левую половину входных данных,  чтобы получить левую половину выходных данных l′.

Результатом работы функции является объединение левой и правой половин x′=l′||r′.

Функция FO

32-битный ввод x из FO(KO, KI, x) разделен на две 16-битные половины x=l0||r0.

В каждом из трех раундов (с индексом j, который принимает значения 1, 2 и 3) левая половина модифицируется, чтобы получить новую правую половину, а правая половина становится левой половиной следующего раунда.

Функция FI представляет собой нерегулярную сеть, подобную Фейстелю.

Функция FI

16-битный ввод  x функции FI(KI,x)  разделен на две половины, x=l0||r0 из которых r0 имеет ширину 9 бит и l0 имеет ширину 7 бит.

Биты в левой половине l0 сначала перетасовываются 9-битным S-box S9, а результат преобразуется в XOR с расширенной до нуля правой половиной r0, чтобы получить новую 9-битную правую половину r1.

Биты правой половины r0 перетасовываются 7-битным S-box S7, и результат преобразуется в XOR с семью младшими значащими битами (LS7) новой правой половины, r1 чтобы получить новую 7-битную левую половину l1

Промежуточное слово x1=l1||r1 преобразуется в XOR с помощью круглого ключа KI, для получения которого x2=l2||r2 используется  7 бит в ширину, а другой - 9 бит в ширину.

X2 =KI⊕x1

Биты в правой половине r2 затем перетасовываются 9-битным S-box S9, и результат преобразуется в XOR с увеличенной до нуля левой половиной l2, чтобы получить новую 9-битную правую половину выходных данных r3.

Наконец, биты левой половины l2  перетасовываются 7-битным S-box S7, и результат преобразуется в XOR с семью младшими значащими битами (LS7) правой половины выходных данных, r3 чтобы получить 7-битную левую половину l3 выходных данных.

Результатом является объединение последней левой и правой половин x′=l3||r3.

S-боксы

S-боксы S7 и S9 определяются как побитовыми выражениями, так и выражениями-XOR и справочными таблицами в спецификации. Побитовые выражения предназначены для аппаратной реализации, но в настоящее время принято использовать справочные таблицы даже в дизайне HW.

# **Рассмотрение блочных схем**

Изображение выглядит как текст, диаграмма, Шрифт, линия

Автоматически созданное описание

Структура алгоритма KASUMI

Изображение выглядит как текст, диаграмма, снимок экрана, План

Автоматически созданное описание

Операция FL

Изображение выглядит как текст, чек, диаграмма

Автоматически созданное описание

Операция FO

Изображение выглядит как текст, диаграмма, чек, линия

Автоматически созданное описание

Операция FI

# **Структура кода**

***Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание***

Структура реализации алгоритма

***Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание***

функция для работы с ключом

Эта функция строит подключи, используемые в основном алгоритме. Она принимает 128-битный ключ и строит подключи KLi1, KLi2, KOi1, KOi2, KOi3, KIi1, KIi2 и KIi3 с использованием нескольких битовых операций и циклических сдвигов.

***Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание***

Функция для разделения функции на правые и левые части

Эта функция представляет собой основной цикл алгоритма KASUMI, который применяет операции FO и FL в цикле для обработки входных 64 бит. Внутри цикла данные разделяются на левую и правую части, затем применяются операции FO и FL. Цикл повторяется четыре раза.

***Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание***

Функция для работы с левой частью

Эти функции реализуют операции FO и FL соответственно.. FL также принимает 32-битное значение и индекс подключа, выполняет линейное преобразование с использованием битовых операций и возвращает 32-битный результат.

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Функция FI

Функция FI реализует операции FI из алгоритма KASUMI, включая использование таблиц замен S7 и S9. Функция принимает 16-битное входное значение in и 16-битный подключ subkey, выполняет несколько операций, таких как битовые операции, XOR и обращения к таблицам S7 и S9, и возвращает 16-битный результат.

***Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание***

Функция трех раундов

Эти функции реализуют операции FO и FL соответственно. FO принимает 32-битное входное значение in и индекс подключа, затем выполняет операции XOR, вызывает FI и возвращает 32-битный результат.

Далее рассмотрим функции для декодирования:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Функция дешифрования

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Функция обратная FO

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Функция обратная FL

Результаты выполнения программы:

На вход программы даны цифры:

//1: fifteen, fourteen, thirteen, twelve, eleven, ten, zero, nine.

//2: eight, seven, six, five, four, three, two, one.

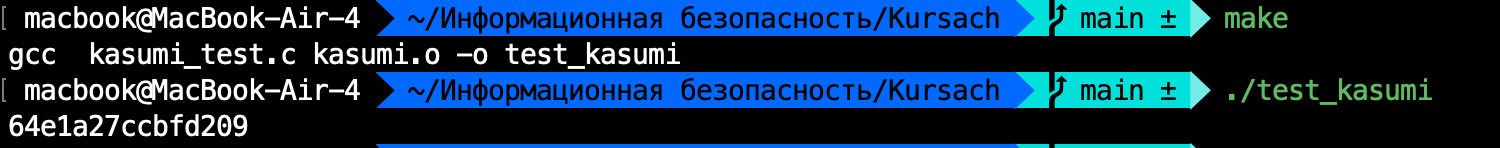
И представлены они в 32 битном формате:

//0xFEDCBA09, 0x87654321,

Ключ задали, как:

//0x9900, 0xAABB, 0xCCDD, 0xEEFF, 0x1122, 0x3344, 0x5566, 0x7788

На выход мы получаем зашифрованное значение:



При расшифровании сообщения получим 2 32-битных значения

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

# **Вывод**

В заключении курсовой работы по алгоритму KASUMI можно подвести итоги и выделить несколько ключевых точек.

Рассмотрено происхождение алгоритма KASUMI, начиная от его предшественника MISTY1 и приведено его внедрение в стандарты мобильной связи третьего поколения (3G).

Отмечено, что KASUMI прошел доработку по сравнению с MISTY1, направленную на улучшение аппаратной реализации и усиление защиты от различных криптоаналитических атак, таких как "встреча посередине", дифференциальный криптоанализ и другие.

Обсуждены различные атаки на KASUMI, представленные исследователями в разные периоды времени. Отмечено, что некоторые из этих атак могут подрывать доказательства безопасности протоколов 3GPP, основанные на предполагаемой стойкости KASUMI.

Подчеркнуто, что несмотря на выявленные уязвимости и проведенные атаки, применение KASUMI остается актуальным в некоторых сценариях, и в реальных условиях проведение некоторых атак может быть непрактичным.

Помимо теоретических аспектов, обозначено, что не все выявленные уязвимости могут быть успешно эксплуатированы в реальных сетевых сценариях, что подчеркивает относительную стойкость алгоритма в контексте применения.

Отмечено, что для использования KASUMI требуется получение лицензии у корпорации Mitsubishi Electric, что является дополнительным аспектом в его применимости.

В целом, курсовая работа позволяет лучше понять историю, структуру и уязвимости алгоритма KASUMI, предоставляя основание для дальнейших исследований в области криптографии и сетевой безопасности.

## **Список литературы**

1. KASUMI (block cipher): [**https://cryptography.fandom.com/wiki/KASUMI\_(block\_cipher)**](https://cryptography.fandom.com/wiki/KASUMI_(block_cipher))
2. АЛГОРИТМЫ ШИФРОВАНИЯ специальный справочник. Сергей Панасенко. 2009 г. Стр. 270.