Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ЭВОЛЮЦИОННЫХ МЕТОДОВ ДЕКОМПОЗИЦИИ ЗАДАЧ ОБРАЩЕНИЯ КРИПТОГРАФИЧЕСКИХ ФУНКЦИЙ С ИСПОЛЬЗОВАНИЕМ СТАТИСТИЧЕСКИХ ТЕСТОВ И ИНКРЕМЕНТАЛЬНЫХ SAT-РЕШАТЕЛЕЙ

Автор: Павленко Артём Леонидович _____

Направление подготовки: 01.04.02 Прикладная математика и информатика

Квалификация: Магистр

Руководитель ВКР: Ульянцев В.И., к.т.н. _____

Санкт-Петербург, 2020 г.

Обучающийся Павленко Артём Леонидович
Группа М42391 Факультет ИТиП

Направленность (профиль), специализация
Технологии разработки программного обеспечения

ВКР принята «____» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты «____» _____ 20___ г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

УТВЕРЖДАЮ
Руководитель ОП
проф., д.т.н. Парфенов В.Г. _____
«____» _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся Павленко Артём Леонидович
Группа M42391 **Факультет** ИТиП
Квалификация: Магистр
Направление подготовки: 01.04.02 Прикладная математика и информатика
Направленность (профиль) образовательной программы: Технологии разработки программного обеспечения
Тема ВКР: Разработка эволюционных методов декомпозиции задач обращения криптографических функций с использованием статистических тестов и инкрементальных SAT-решателей
Руководитель Ульянцев В.И., к.т.н.,
2 Срок сдачи студентом законченной работы до: «____» _____ 20___ г.
3 Техническое задание и исходные данные к работе

Требуется разработать методы декомпозиции задач обращения криптографических функций с использованием статистических тестов и инкрементальных SAT-решателей. Применить разработанные методы для построения guess-and-determine атак на ряд криптографических алгоритмов.

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

а) Обзор предметной области. Обзор существующих эволюционных методов в криптоанализе. Обзор существующих автоматизированных методов декомпозиции задач обращения криптографических функций.
б) Описание разрабатываемого метода с использованием статистических тестов. Описание разрабатываемого метода с использованием инкрементальных SAT-решателей.
в) Экспериментальное исследование разработанных методов декомпозиции задач обращения криптографических функций. Сравнение различных статистических тестов. Сравнение построенных декомпозиционных множеств.

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

1 On cryptographic attacks using backdoors for SAT / A. Semenov [et al.] // Thirty-Second AAAI Conference on Artificial Intelligence. — 2018.

2 *Semenov A.*, *Zaikin O.* Algorithm for finding partitionings of hard variants of boolean satisfiability problem with application to inversion of some cryptographic functions // SpringerPlus. — 2016. — Vol. 5, no. 1. — P. 554.

7 Дата выдачи задания «_____» _____ 20___ г.

Руководитель ВКР _____

Задание принял к исполнению _____ «_____» _____ 20___ г.

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся: Павленко Артём Леонидович

Наименование темы ВКР: Разработка эволюционных методов декомпозиции задач обращения криптографических функций с использованием статистических тестов и инкрементальных SAT-решателей

Наименование организации, в которой выполнена ВКР: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Разработка и реализация новых методов декомпозиции задач обращения криптографических функций

2 Задачи, решаемые в ВКР:

а) Разработка эволюционных методов декомпозиции задач обращения криптографических функций с использованием статистических тестов;

б) Разработка эволюционных методов декомпозиции задач обращения криптографических функций с использованием инкрементальных SAT-решателей;

3 Число источников, использованных при составлении обзора: 0

4 Полное число источников, использованных в работе: 36

5 В том числе источников по годам:

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	8	6	22

6 Использование информационных ресурсов Internet: нет

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	**Раздел работы**
Язык программирования Python 3	2, 3
Библиотеки для Python 3: numpy, python-sat, scipy	2, 3
Среда разработки PyCharm	2, 3
Система компьютерной верстки LaTeX	1, 2, 3

8 Краткая характеристика полученных результатов

Разработан и реализован эволюционный метод декомпозиции задач обращения криптографических функций с использованием статистических тестов. Разработан и реализован эволюционный метод декомпозиции задач обращения криптографических функций с использованием инкрементальных SAT-решателей. Проведено экспериментальное исследование разработанных методов для ряда криптографических алгоритмов.

9 Гранты, полученные при выполнении работы

Разработка эволюционных стратегий поиска декомпозиций трудных вариантов задачи о булевой выполнимости с применением к обращению криптографических функций (РНФ, проект 18-71-00150, исполнитель)

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы

1 *Павленко А.*, *Ульянцев В.* Применимость статистических тестов к эволюционным методам декомпозиции экземпляров задачи о булевой выполнимости для криптоанализа генераторов ключевого потока. — 2019. — VIII Конгресс молодых ученых.

2 *Павленко А.*, *Ульянцев В.* Эволюционные алгоритмы построения декомпозиционных множеств для трудных вариантов задач о булевой выполнимости, позволяющих достичь сверхлинейного ускорения при решении. — 2020. — IX Конгресс молодых ученых.

3 Parallel Framework for Evolutionary Black-box optimization with Application to Algebraic Cryptanalysis / A. Pavlenko [et al.] // 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). — IEEE. 2019. — P. 1144–1149.

4 *Pavlenko A.*, *Buzdalov M.*, *Ulyantsev V.* Fitness comparison by statistical testing in construction of SAT-based guess-and-determine cryptographic attacks // Proceedings of the Genetic and Evolutionary Computation Conference. — 2019. — P. 312–320.

Обучающийся Павленко А.Л. _____

Руководитель ВКР Ульянцев В.И. _____

«____» _____ 20___ г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наше время сложно представить себе жизнь без современных технологий. В основе многих из них используются методы информационной безопасности. Для безопасной передачи и хранения информации используются различные алгоритмы шифрования. Для обоснования криптографической стойкости используемых алгоритмов шифрования, строятся различные криптографические атаки.

Процесс аналитического построения таких атак является трудоемкой задачей. Это обуславливается тем, что подходы, которые успешно показали себя на одних алгоритмах, оказываются совершенно бесполезными для других. Главным недостатком аналитического метода является необходимость в индивидуальном рассмотрении особенностей каждого алгоритма, что требует больших затрат времени и ручного труда.

Альтернативой являются автоматизированные методы декомпозиции задач обращения криптографических функций, которые позволяют строить guess-and-determine атаки для многих известных криптографических алгоритмов. В их основе лежат современные алгоритмы решения задачи булевой выполнимости (SAT), зарекомендовавшие себя в криптоанализе.

На данный момент существует не так много автоматизированных методов, а криптографические атаки, в большинстве своем, строятся аналитическими методами. Однако автоматизированные методы уже позволяют строить guess-and-determine атаки, в некоторых случаях, не хуже, а иногда даже и лучше известных ранее. Как минимум, эти методы могут быть использованы для первоначальной оценки исследуемого криптографического алгоритма, а иногда этой оценки может быть достаточно для обоснования недостаточной криптографической стойкости.

Целью данной работы является разработка и реализация новых методов декомпозиции задач обращения криптографических функций. Для достижения данной цели предлагается решить следующие задачи:

а) Применить эволюционные алгоритмы для построения декомпозиционных множеств.
б) Применить статистические тесты к вычислению функции приспособленности в процессе работы эволюционного алгоритма.

в) Использовать инкрементальное решение задач в процессе вычисления функции приспособленности.

ГЛАВА 1. АВТОМАТИЗИРОВАННЫЕ МЕТОДЫ ДЕКОМПОЗИЦИИ ЗАДАЧ ОБРАЩЕНИЯ КРИПТОГРАФИЧЕСКИХ ФУНКЦИЙ

В данной главе приводится обзор автоматизированных методов построения декомпозиционных множеств для криптографических алгоритмов. В главе 1.1 описаны криптографические алгоритмы исследуемые в данной работе. В разделе 1.2 даётся определение задачи о булевой выполнимости. В разделе 1.3 формулируется задача криптоанализа и приводятся основные его методы. В разделе 1.4 описывается алгоритм работы современных SAT-решателей. В разделе 1.5 рассматриваются различные метаэвристические алгоритмы, затронутые в данной работе. В разделе 1.6 сделан обзор применения эволюционных алгоритмов в криптоанализе. В разделе 1.7 приводится обзор известных автоматизированных методов декомпозиции задач обращения криптографических функций. А в разделе 1.8 даётся определение статистическим тестам.

1.1. Криптографические алгоритмы

В данной работе будут рассмотрены алгоритмы симметричного потокового шифрования. Шифрование называется *симметричным*, если для процесса шифрования и расшифровывания используется один и тот же секретный ключ (или, криптографический ключ). А шифрование с использованием синхронных потоковых алгоритмов выглядит следующим образом:

а) На вход подается поток бит открытого текста $m_1, m_2, ..., m_n$.

б) Алгоритм генерирует ключевой поток бит $k_1, k_2, ..., k_n$.

в) Биты открытого текста шифруются с помощью битов ключевого потока, посредством их побитового сложения по модулю 2 (битовая операция $\oplus$).

г) На выходе получаются биты шифротекста $c_1, c_2, ..., c_n$, где $c_i = m_i \oplus k_i$

Схема описанного процесса представлена на рисунке 1.

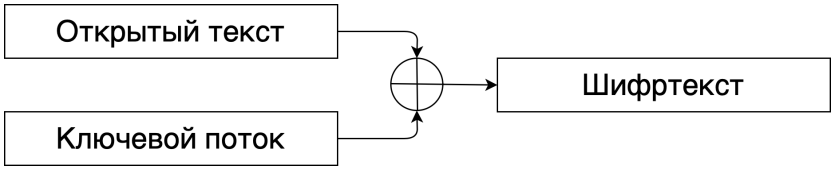

Рисунок 1 – Процесс потокового шифрования

Важным криптографическим примитивом для потокового шифрования является *регистр сдвига с линейной обратной связью* (РСЛОС), пример которого приведен на рисунке 2. РСЛОС используется для генерации псевдослучайной последовательности бит и состоит из сдвигового регистра и функции

обратной связи. Регистр состоит из ячеек памяти, которые заполняются битами, а функция обратной связи представляет из себя линейную булеву функцию от значений всех или некоторых битов регистра, результат выполнения которой помещается в первую ячейку регистра во время сдвига. Сдвиги происходят каждый такт, когда значение $clock$ бита равно 1. На его основе были построены некоторые из рассматриваемых в этой работе криптографических алгоритмов, а именно: A5/1 и Alternating Step Generator. Алгоритмы из семейства Trivium тоже используются в своей структуре сдвиговые регистры, однако уже с нелинейной комбинацией прямой и обратной связи.

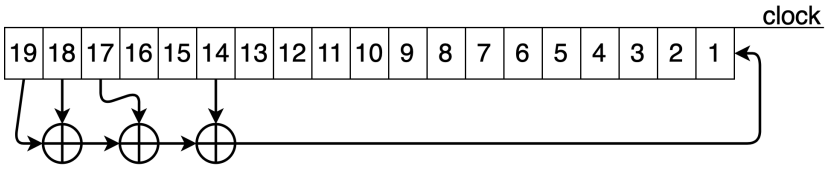

Рисунок 2 – РСЛОС с длиной 19 бит и функцией обратной связи $x^{19} \oplus x^{18} \oplus x^{17} \oplus x^{14} \oplus 1$ (первый из трёх регистров алгоритма A5/1)

Далее будут описаны структура и принцип работы исследуемых в работе криптографических алгоритмов, а именно: алгоритм A5/1, алгоритм Trivium [12, 13] и его ослабленные версии Trivium 64, Trivium 96 [26], Bivium [12], генератор ключевого потока Alternating Step Generator (ASG) [19].

1.1.1. Алгоритм A5/1

A5/1 – симметричный алгоритм синхронного потокового шифрования, который использовался для шифрования передаваемых данных в стандарте мобильной сотовой связи GSM. Структура алгоритма являет собой систему из трёх РСЛОС (R_1, R_2 и R_3) с длинами 19, 22 и 23 бита соответственно и представлена на рисунке 3. В качестве функций обратной связи выступают $r_1 = x^{19} \oplus x^{18} \oplus x^{17} \oplus x^{14} \oplus 1$ для регистра R_1, $r_2 = x^{41} \oplus x^{40} \oplus 1$ для R_2 и $r_3 = x^{64} \oplus x^{63} \oplus x^{62} \oplus x^{49} \oplus 1$ для R_3. А во время такта происходит следующее:

— вычисляется значение выходного бита $z = x^{19} \oplus x^{41} \oplus x^{64}$
— если $b_j \equiv \langle b_1, b_2, b_3 \rangle$, то вычисляется значение r_j и сдвигаем регистр R_j, где $b_1 = x^9$, $b_2 = x^{30}$ и $b_3 = x^{52}$ биты синхронизации
— в первую ячейку сдвинутого регистра j помещается бит r_j

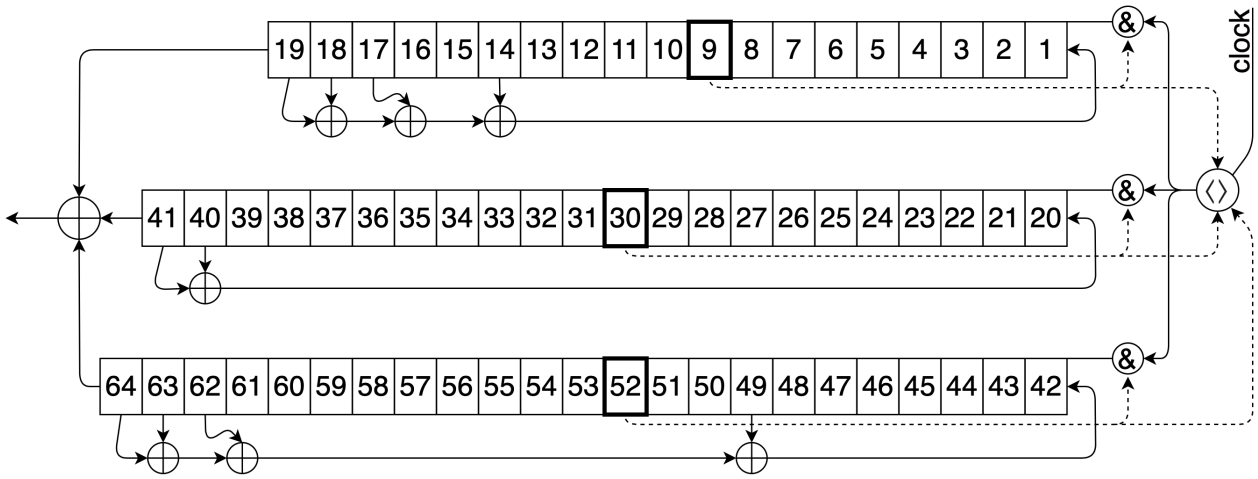

Рисунок 3 – Структура алгоритма A5/1

1.1.2. Семейство Trivium

Trivium [12, 13] – симметричный алгоритм синхронного потокового шифрования. Состоит из трёх сдвиговых регистров (T, P и Q) с длинами 93, 85 и 110 бит соответственно и представлен на рисунке А.1 в приложении А. Во время такта происходят изменения битов в регистрах сдвига путём нелинейной комбинации прямой и обратной связи, которые описываются функциями $t = x^{288} \oplus (x^{287} \wedge x^{286}) \oplus x^{243} \oplus x^{69}$ для регистра T, $p = x^{93} \oplus (x^{92} \wedge x^{91}) \oplus x^{66} \oplus x^{171}$ для P и $q = x^{177} \oplus (x^{176} \wedge x^{175}) \oplus x^{162} \oplus x^{264}$ для Q. А во время такта происходит следующее:

— вычисляется значение выходного бита $z = t_z \oplus p_z \oplus q_z$, где $t_z = x^{66} \oplus x^{93}$, $p_z = x^{162} \oplus x^{177}$ и $q_z = x^{243} \oplus x^{288}$
— вычисляются значения t, p и q
— сдвигаются все регистры, а в первые ячейки помещаются соответствующие биты t, p и q

Bivium [12] – симметричный алгоритм синхронного потокового шифрования, состоящий только из двух регистров (T и P) оригинального алгоритма Trivium. Функции обратной связи выглядят следующим образом: $t = x^{177} \oplus (x^{176} \wedge x^{175}) \oplus x^{162} \oplus x^{69}$ для регистра T и $p = x^{93} \oplus (x^{92} \wedge x^{91}) \oplus x^{66} \oplus x^{171}$ для P. А значение выходного бита $z = t_z \oplus p_z$.

В статье [26] описан способ сокращения длин регистров для шифров Trivium и Bivium, который позволяет существенно снизить их вычислительную сложность без потери алгебраических свойств. Используя этот способ были построены упрощенные версии алгоритма Trivium: Trivium 64 и Trivium 96 (или Trivium-toy). Они отличаются от исходного алгоритма тем, что имеют ре-

гистры меньшей длины и другие функции сдвига. Их структура представлена на рисунках А.2 и А.3 приложения А соответственно.

1.1.3. Alternating Step Generator

Alternating Step Generator (ASG) [19] является общей конструкцией генераторов ключевого потока, который можно использовать для создания потоковых криптографических алгоритмов с различной длиной секретного ключа. Его главные особенности – простота реализации и высокая скорость генерации битов ключевого потока.

Алгоритм состоит из трёх РСЛОС (K, M_1 и M_0), которые могут иметь различные длины и функции обратной связи в зависимости от реализации. Регистр K является управляющим, и в зависимости от значения выходного бита сдвигается либо регистр M_1, либо регистр M_0. В данной работе будут рассмотрены ASG 72, ASG 96 и ASG 192.

Длины регистров алгоритма ASG 72 составляют 23, 24 и 25 бит для K, M_1 и M_0 соответственно, а его структура приведена на рисунке А.4 приложения А. Функции обратной связи выглядят следующим образом: $k = x^{23} \oplus x^{22} \oplus x^{20} \oplus x^{18}$ для K, $m_1 = x^{47} \oplus x^{46} \oplus x^{45} \oplus x^{40}$ для M_1 и $m_0 = x^{72} \oplus x^{71} \oplus x^{70} \oplus x^{69}$ для M_0. А во время такта происходит следующее:

— запоминаем значение $k_o = x^{23}$ и вычисляем значение k
— сдвигаем регистр K и в его первую ячейку помещаем бит k
— вычисляется значение выходного бита $z = x^{72} \oplus x^{47}$
— вычисляем значение m_{k_o}, сдвигаем регистр M_{k_o} и помещаем вычисленное значение в его первую ячейку

Принцип работы работы алгоритмов ASG 96 и ASG 192 аналогичен, а их структуры представлены рисунках А.5 и А.6 приложения А соответственно.

1.2. Задача о булевой выполнимости

Задача о булевой выполнимости (SAT) – алгоритмическая задача в теории вычислительной сложности, которая задается с помощью булевой формулы. Любая формула может быть задача с помощью имен переменных, логических операторов и скобок. Основными логическими операциями являются $\wedge$ (конъюнкция), $\vee$ (дизъюнкция) и $\neg$ (отрицание). Остальные операции, такие как, например, уже упоминавшееся $\oplus$ (исключающее или), могут быть выражены с использованием основных.

Задача же заключается в том, чтобы найти такой набор значений для всех булевых переменных в формуле, при котором данная формула станет истиной либо доказать, что такого набора значений не существует.

Поскольку одна и та же SAT-формула может быть записана разными способами, то существуют *нормальные формы*, в которых любая формула задается единственным образом в простейшем или каноническом виде. Например, конъюнктивная нормальная форма (КНФ) – нормальная форма, в которой формула имеет вид конъюнкции дизъюнкции литералов (переменных или их отрицаний). Главное её преимущество заключается в том, что для решения формул в КНФ существуют специальные программные средства SAT-решатели (SAT-solvers), о которых будет подробней рассказано в одном из последующих разделах.

1.3. Алгебраический криптоанализ

Криптоанализ – наука, изучающая методы дешифровки зашифрованной информации без использования предназначенного для этого секретного ключа. Представим криптографический алгоритм потокового шифрования в виде функции $f_a : \{0,1\}^n \to \{0,1\}^*$, где n – длина секретного ключа. Зададим длину m генерируемого ключевого потока, тогда функция f_a примет вид:

$$f_a : \{0,1\}^n \to \{0,1\}^m$$

Задача криптоанализа состоит в том, чтобы по фрагменту ключевого потока $y = f(x)$ восстановить секретный ключ x и оценить время необходимое на его восстановление. То есть решить задачу обращения криптографической функции $f_a^{-1}(y) = x$ и оценить её криптографическую стойкость.

Алгебраический криптоанализ [5] предполагает использование систем полиномиальных уравнений (обычно над полем *GF(2)*) для построения криптографических атак. В процессе построения атаки исследуемый криптографический алгоритм транслируется в систему уравнений. В качестве набора переменных обычно выступают входные (секретный ключ) и выходные (ключевой поток) биты. Решить систему означает: для конкретных значений выходных переменных получить значения входных переменных. Далее, полученная система анализируется и выбирается эффективный способ её решения.

Упоминания заслуживает метод *eXtended Sparse Linearization* (*XSL-атака*) [10]. Данный метод предполагает приведения задач криптоанализа к многомерному уравнению над полем *GF(2)*. Однако отдельная проблема этого метода состоит в решении полученной системы уравнений. Другим важным направлением алгебраического криптоанализа является SAT-based криптоанализ. Его идея заключается в том, чтобы сводить задачу обращения криптографических функций к SAT. Для сведения могут быть использованы специальные программные средства – автоматические трансляторы. В данной работе для трансляции криптографических алгоритмов к SAT будет использоваться транслятор Transalg [14]. А для решения получаемых в процессе трансляции задач используются SAT-решатели.

Далее будут описаны некоторые, наиболее популярные и значимые для этой работы, методы криптоанализа.

1.3.1. Метод грубой силы

Самый простой и универсальный метод. Метод грубой силы (*Brute force*) осуществляется путем полного перебора всех возможных решений, пока не будет найдено правильное решение. Его главный недостаток заключается в том, что для криптоанализа большинства современных криптографических алгоритмов потребуется колоссальное количество вычислительных ресурсов. С помощью этого метода можно получить первоначальную верхнюю оценку времени, за которое исследуемая криптографическая функция точно будет взломана, и он в основном используется в сравнении с другими более интеллектуальными методами криптоанализа.

1.3.2. Guess-and-determine атака

Пусть B – некоторое непустое множество переменных из множества всех переменных V исходной задачи такое, что $|B| \ll |V|$. И если суммарное время необходимое на решение всех подзадач, полученных в процессе декомпозиции исходной задачи множеством B, много меньше времени необходимого на решение исходной задачи Brute force методом, то говорят о нетривиальной *Guess-and-determine* атаке. А такое множество B называют *guessed bits* множеством. В подавляющем большинстве известных guess-and-determine атак множество B состоит только из входных переменных рассматриваемой криптографической задачи.

Хорошим примером применения этой атаки в криптоанализе является атака Андерсона [3] на алгоритм A5/1. В качестве *guessed bits* выступает множество из 53 переменных. А оставшиеся значения 11 переменных могут быть получены используя технику *Unit Propogation*. Еще одним примером является guess-and-determine атака на алгоритм A5/1 состоящая из 31 переменной и представленной в статье [29]. Примечательно, что для её осуществления авторами были использованы SAT-решатели.

1.4. SAT-решатели

В основе всех современных SAT-решателей лежит алгоритм *CDCL* [7, 22] (*Conflict-Driven Clause Learning*), который в свою очередь основывается на алгоритме *DPLL*. DPLL – полный алгоритм поиска с возвратом для решения SAT-задач в КНФ. Процесс поиска решения можно представить в виде графа: одной переменной назначается конкретное значение *истина* или *ложь*, после чего формула упрощается с использованием техники *Unit Propagation*. В случае возникновения конфликта при подстановке алгоритм возвращается к предыдущей переменной, иначе переходит к следующей. Если в результате конфликтов процесс решения вернулся в корень, то задача считается невыполнимой (*UNSAT*), иначе будет найдено решение удовлетворяющее задаче (*SAT*). Алгоритм DPLL подразумевает еще использование дополнительных правил в процессе решения:

— если дизъюнкт (clause) содержит только одну переменную, то ее значение может быть сразу определено
— если переменная входит во все дизъюнкты с одной полярность (либо только без отрицания, либо только с отрицанием), то такую переменную называют *чистой* (pure), а все такие дизъюнкты можно удалить из формулы.

Ко всему этому алгоритм CDCL предполагает еще использование нехронологического возврата и запоминание дизъюнктов в ходе анализа конфликта. Благодаря этим особенностям была значительно повышена эффективность при решении задач. Для решения SAT-задач в данной работе используются современные SAT-решатели на основе алгоритма CDCL, а именно: RoKK [34] и Glucose 3 [4].

Однако SAT-задачи, получаемые в результате трансляции современных криптографических функций, являются достаточно сложными для того, что-

бы напрямую решать их с помощью SAT-решателей. Поэтому предлагается строить guess-and-determine атаки посредством декомпозиции исходных SAT-формул с использованием метаэвристических алгоритмов описанных в следующем разделе.

1.5. Эвристические и метаэвристические алгоритмы

Эвристический алгоритм – это практический алгоритм решения задачи, который не гарантирует точность и оптимальность, но являющийся достаточным для решения поставленной задачи. Такие алгоритмы используются для поиска решений в тех случаях когда о задаче мало что известно: нет представления о том, как выглядит оптимальное решение и как его искать. Они применяются в том случае, когда возможность полного перебора решений не возможна или занимает колоссальное количество времени, то есть пространство поиска слишком велико. Например, для исследуемых в данной работе задач пространство поиска экспоненциально от размера входных данных. Использование же эвристических алгоритмов позволяет находить близкие к оптимальным решения за сравнительно небольшое время. А для их применения достаточно уметь проверять на правильность или оценивать качество потенциального решения.

Метаэвристические алгоритмы отличаются от эвристических лишь тем, что используют более интеллектуальные эвристики, которые добавляют им возможность обходить проблемные участки в пространстве поиска, например, такие как локальный минимум. В данной работе метаэвристические алгоритмы используются для минимизации значения целевой функции. Более точно, производится стохастическая оптимизация целевой функции посредством метаэвристических алгоритмов.

1.5.1. Локальный поиск

Локальный поиск является обобщением для эвристических алгоритмов, процесс оптимизации для которых представляет из себя итеративный процесс. На каждой итерации некоторым образом выбирается новая точка и сравнивается с текущей точкой, посредством вычисления для них значений целевой функции. При этом алгоритм не предполагает запоминания предыдущих состояний, а поиск ведется на основании текущего состояния.

Один из самых простых алгоритмов локального поиска является поиск восхождением к вершине (Hill climbing). На каждой итерации перебираются

все возможные соседние к текущей точки пока не будет улучшено значение целевой функции. Если такая точка найдена, то она заменяет текущую и алгоритм переходит к следующей итерации. В противном случае алгоритм завершает свою работу и возвращает текущую точку в качестве решения. Однако найденное решение скорее всего окажется локальным экстремумом, поскольку алгоритм не является оптимальным и не гарантирует нахождения глобального экстремума.

1.5.2. Поиск с запретами

Поиск с запретами (Tabu Search) – метаэвристический алгоритм локального поиска, в процессе работы которого поддерживается список запрещенных к посещению точек: *список запретов* (tabu list). Данный список включает в себя уже пройденные точки пространства поиска, возврат к которым невозможен в течение некоторого времени, что вынуждает алгоритм продолжать поиски новых решений.

1.5.3. Алгоритм имитации отжига

Алгоритм имитации отжига (Simulated annealing) – метаэвристический алгоритм локального поиска, основывающийся на имитации физических процессов, которые протекают в веществах в процессе их кристаллизации. В процессе кристаллизации вещество охлаждается и атомы образуют кристаллическую решетку, свойственную твердому состоянию вещества. Поскольку этот процесс не мгновенный и занимает определенной количество времени, некоторые атомы могут переходить из одной ячейки решетки в другую. Число таких переходов прямо пропорционально зависит от *температуры* и уменьшается в процессе охлаждения вещества. В алгоритме имитации отжига эти особенности отражаются на этапе принятия решения о переходе из текущей точки к следующей. Алгоритм начинает свою работу с большим значением температуры T, которая постепенно снижается в процессе работы алгоритма. При принятии решения о переходе от точки x_i к точке x_{i+1} алгоритм следует следующим соображениями:

— если $\Phi(x_i) > \Phi(x_{i+1})$, то переход осуществляет всегда

— иначе переход осуществляется с вероятность $\exp\left(\frac{\Phi(x_i)-\Phi(x_{i+1})}{T}\right)$

Таким образом видно, что поведение алгоритма на первых итерациях напоминает случайные блуждания в пространстве поиска, а при снижении температуры T до 0 превращается в обычный локальный поиск. Эти особенности

позволяют ему выбираться из локальных минимумов в процессе оптимизации при этом гарантируя оптимальное, или близкое к оптимальному, решение в итоге.

1.5.4. Эволюционные алгоритмы

Эволюционные алгоритмы – метаэвристические алгоритмы локального поиска, основывающиеся на эволюционных процессах протекающих в живой природе, тем самым моделируя процессы естественного отбора. В процессе естественного отбора, как и в живой природе, на каждой итерации алгоритма происходит изменение *популяции* состоящей из отдельных *особей*. Под особью подразумевается определенная точка в пространстве поиска решения и соответствующее ей значение приспособленности. От значения приспособленности особи зависит вероятность её участия в формировании следующего поколения. Процесс формирования популяции для следующего поколения разнится от типа используемого эволюционного алгоритма, но всегда протекает с использованием всех или некоторых из следующих генетических операторов:

— мутация – унарный оператор, в результате применения которого изменяется один или несколько генов родительской особи. Под геном понимается некоторая часть представления особи.
— скрещивание – бинарный оператор, в результате применения которого родительские особи обмениваются генами. Таким образом новые особи наследуют гены обоих родителей.
— селекция – оператор, который применяет ко всей популяции в целом и выбирает особи, которые будут участвовать в формировании следующей популяции.

Традиционно эволюционные алгоритмы подразделяются на *эволюционные стратегии* [8] и *генетические алгоритмы* [25].

Выделяют два основных вида эволюционных стратегий: (μ, λ) и $(\mu + \lambda)$. Для обеих стратегий размер популяции равен μ, и в процессе формирования следующего поколения генерируется λ новых особей, используя только оператор мутации. Отличие заключается в том, что в стратегии $(\mu + \lambda)$ учитываются особи из предыдущей популяции, а в (μ, λ) популяция состоит только из новых особей. В генетических же алгоритмах применяется оператор скрещивания.

1.6. Эволюционные алгоритмы в криптоанализе

Известны применения эволюционных алгоритмов в криптоанализе. Однако в большинстве работ эволюционные алгоритмы применялись непосредственно в процессе проведения атаки. Так, например, в работах [15, 18, 23] были предложены методы дешифровки зашифрованного английского текста с применением генетических алгоритмов. В работах [15, 23] были рассмотрены примеры простых перестановочных алгоритмов шифрования, а функция приспособленности оценивала корректность текущей перестановки. В работе [18] объектом исследования уже был упрощенный алгоритм шифрования S-DES (Simplified Data Encryption Standard) [31]. Однако функция приспособленности все еще была завязана на символах английского алфавита. В работе [11] была представлена атака на 4-х раундовый DES (Data Encryption Standard). В отличии от предыдущих работ, предложенный метод рассматривал шифротекст как последовательность битов, а не символов английского алфавита.

В данной работе предлагается использовать эволюционные алгоритмы для декомпозиции задачи обращения криптографических функции, то есть для построения guess-and-determine атак, а не их проведения. Впоследствии, используя построенное *guessed bits* множество, можно совершить атаку с применением SAT-решателей.

1.7. Автоматизированные методы в алгебраическом криптоанализе

В данном разделе приводится обзор известных автоматизированных методов декомпозиции задач обращения криптографических функций. В их основе лежат различные метаэвристические алгоритмы, а целью является построение guess-and-determine атак.

В работе [32] для автоматизированного построения декомпозиционных (guessed bits) множеств использовались метаэвристические алгоритмы поиск с запретами (Tabu search) и имитация отжига (Simulated annealing). Описанный в работе [32] метод позволяет строить SAT-разбиения для трудных вариантов задач о булевой выполнимости и был применен для построения криптографических атак на ряд известных алгоритмов, таких как: A5/1, Bivium и Grain. В качестве значения целевой функции Φ используется оценка времени, которое потребуется на решение декомпозиции исходной задачи для соответствующего декомпозиционного множества B. Для вычисления оценки времени решения применяется метод Монте-Карло [24], суть которого заключается в

проведении вероятностного эксперимента, состоящего из N независимых наблюдений за случайной величиной ξ. В качестве значения ξ выступает время $T_A[C(\gamma,\beta)]$ затраченное SAT-решателем A для решения формулы $C(\gamma,\beta)$, где $C(\gamma)$ – задача обращения криптографической функции над фрагментом ключевого потока γ, и β – набор значений для декомпозиционного множества B, а $C(\gamma,\beta)$ – одна из задач декомпозиции исходной задачи $C(\gamma)$ для заданного декомпозиционного множества B. Число таких задач экспоненциально от размера s декомпозиционного множества B и равно $N_B = 2^s$, где $s = |B|$. Таким образом для каждого из N наблюдений независимо выбирается случайная задача $C(\gamma,\beta_j)$ из всего множества задач декомпозиции исходной задачи $C(\gamma)$ с соответствующей набором значений β_j для декомпозиционного множества B, где $j \in \{1,...,2^s\}$. Для каждого наблюдения i определяется значение величины $\xi_i = T_A[C(\gamma,\beta_j^i)]$, которое используются для вычисления целевой функции по формуле 1.

$$\Phi_B = 2^s \cdot \frac{\sum_{i=1}^{N} \xi_i}{N} \tag{1}$$

Следующим шагом в этом направлении можно считать работу [27], в которой был введен новый тип декомпозиционных множеств – *Inverse Backdoor Set* (*IBS*), а для их построения использовался метаэвристический алгоритм поиск с запретами (*Tabu search*). Описанный в работе [27] метод позволяет строить guess-and-determine атаки основанные на IBS множествах (*G-a-D atacks, based on IBS*) и был применен к криптографическим алгоритмам Trivium, AES 128, Magma (ГОСТ 28147-89). Для оценки трудоемкости обращения соответствующей криптографической функции также применяется метод Монте-Карло [24] для N независимых наблюдений за случайной величиной ξ. Для проведения наблюдения i генерируется правильная комбинация секретного ключа α_i и ключевого потока γ_i. Затем формируется задача $C(\gamma_i,\beta_i)$, где $\beta_i = B(\alpha_i)$ фрагмент секретного ключа α_i для декомпозиционного множества B, и решается SAT-решателем A с ограничением времени t. Если задача была успешно решена за время $T_A[C(\gamma_i,\beta_i)] < t$, то $\xi_i = 1$, иначе $\xi_i = 0$. Утверждается, что с вероятностью 0.95 guess-and-determine атака с декомпозиционным множеством B будет завершена за время, которое можно вычислить по формуле 2.

$$\Phi_B = 2^s \cdot t \cdot \frac{3N}{\sum_{j=1}^{N} \xi_B^i} \tag{2}$$

В описанных выше работах для оптимизации целевой функции используются метаэвристические алгоритмы поиск с запретами и имитация отжига. В данной работе предлагается использовать эволюционные алгоритмы для этой цели.

1.8. Статистические тесты

Статистические тесты (статистические критерии) применяются к выборке данных для подтверждения или отвержения *статистической гипотезы*. Статистическая гипотеза – предположение о виде распределения и свойствах случайной величины. Статистические критерии, которые напрямую применяются к значениям из выборки, являются параметрическими и требуют нормального распределения случайной величины. Непараметрические тесты, наоборот, не требуют нормальности распределения, оперируя рангами или частотами, а не значениями напрямую. В данной работе будут рассмотрены следующие статистические тесты:

— U-критерий Манна Уитни (Mann-Whitney U test) [21], также известный как критерий суммы рангов Уилкоксона (Wilcoxon rank sum test) [33]

— тест Барнарда (Barnard's test) [6], являющийся улучшенной версией точного теста Фишера (Fisher's exact test) [16]

Статистические тесты и ранее находили свое применение в области эволюционных вычислений. Так, например, в работах [1, 2] статистические тесты применялись для исследования поведения эволюционных алгоритмов в процессе оптимизации. Авторы данных работ анализировали результаты, опубликованные в рамках специальной сессии CEC'2005 по оптимизации реальных параметров (CEC'2005 Special Session on Real Parameter Optimization). А в работе [17] статистические тесты применялись для настройки параметров эволюционных алгоритмов.

Все эти примеры объединяет то, что статистические тесты применяются постфактум. В данной же работе предполагается их использовать непосредственно в процессе работы алгоритма, а именно в процессе вычисления функции приспособленности.

Выводы по главе 1

а) Метаэвристические алгоритмы успешно применяются для автоматизации процесса криптоанализа задач обращения криптографических функций.

б) Эволюционные алгоритмы и ранее применялись в криптоанализе. Однако, в ходе проведения обзора не было обнаружено их применения для автоматического построения guess-and-determine атак в алгебраическом криптоанализе.

в) Статистические тесты применялись в области эволюционных вычислений для проведения анализа полученных в процессе оптимизации результатов. Но не применялись для вычисления функции приспособленности непосредственно в процессе работе алгоритма.

ГЛАВА 2. ЭВОЛЮЦИОННЫЕ МЕТОДЫ ДЕКОМПОЗИЦИИ ЗАДАЧ ОБРАЩЕНИЯ КРИПТОГРАФИЧЕСКИХ ФУНКЦИЙ

В данной главе приводится описание разработанных автором эволюционных методов построения декомпозиционных множеств для криптографических алгоритмов. В разделе 2.1 приводится краткое описание предыдущих исследований автора в данном направлении. А в разделах 2.2 и 2.3 будут описаны разработанные в настоящей работе эволюционные методы построения декомпозиционных множеств для криптографических алгоритмов с применением статистических тестов и инкрементальных SAT-решателей.

2.1. Эволюционный метод с адаптивной стратегией

В своей бакалаврской работе автор предложил использовать эволюционные алгоритмы для построения декомпозиционных множеств типа *IBS*, используя формулу 2 в качестве функции приспособленности. Также в работе было предложено использовать адаптивную стратегию изменения объема выборки N для метода Монте-Карло в процессе работы алгоритма. Суть предложенной стратегии заключалось в том, чтобы менять размер выборки в зависимости от текущего значения функции приспособленности. Зависимость размера выборки от значения функции приспособленности была получена экспериментальным путем. Но поскольку объектом исследования являются криптографические алгоритмы, сложность которых разнится, то эту зависимость было необходимо получать для каждого криптографического алгоритма отдельно. После получения данной зависимости для исследуемых криптографических алгоритмов, а именно: A5/1, Bivium и Trivium 64, в рамках бакалаврской работы было проведено экспериментальное исследование предложенной адаптивной стратегии.

Впоследствии, разработанный эволюционный метод и полученные результаты были опубликованы в статье [30]. А сравнение результатов, полученных с использованием адаптивной стратегии, с новыми результатами приведено в главе 3.1.4.1.

2.1.1. Проблемы адаптивной стратегии

Предложенная эвристика адаптивного изменения объема выборки N позволяла экономить вычислительные ресурсы, пока значение функции приспособленности велико. Однако на рисунке 4 видно, что значении функции

приспособленности резко уменьшается, и, примерно, после 1 часа работы эволюционного алгоритма эффективность адаптивной стратегии падает. Такое поведение в процессе оптимизации присуще не только для алгоритма Bivium (рис. 4), но и для других криптографических алгоритмов. Для алгоритмов Trivium 64, Trivium 96, A5/1 и ASG 72 можно наблюдать подобную картину на рисунках Б.1, Б.2, Б.3 и Б.4 приложения Б соответственно. Таким образом у предложенной стратегии есть ряд недостатков, таких как:

— необходимость заранее подстраивать эвристику под каждый криптографический алгоритм отдельно
— снижение эффективности по мере уменьшения значения функции приспособленности (а большую часть времени алгоритм стагнирует, что видно на рисунке 4)

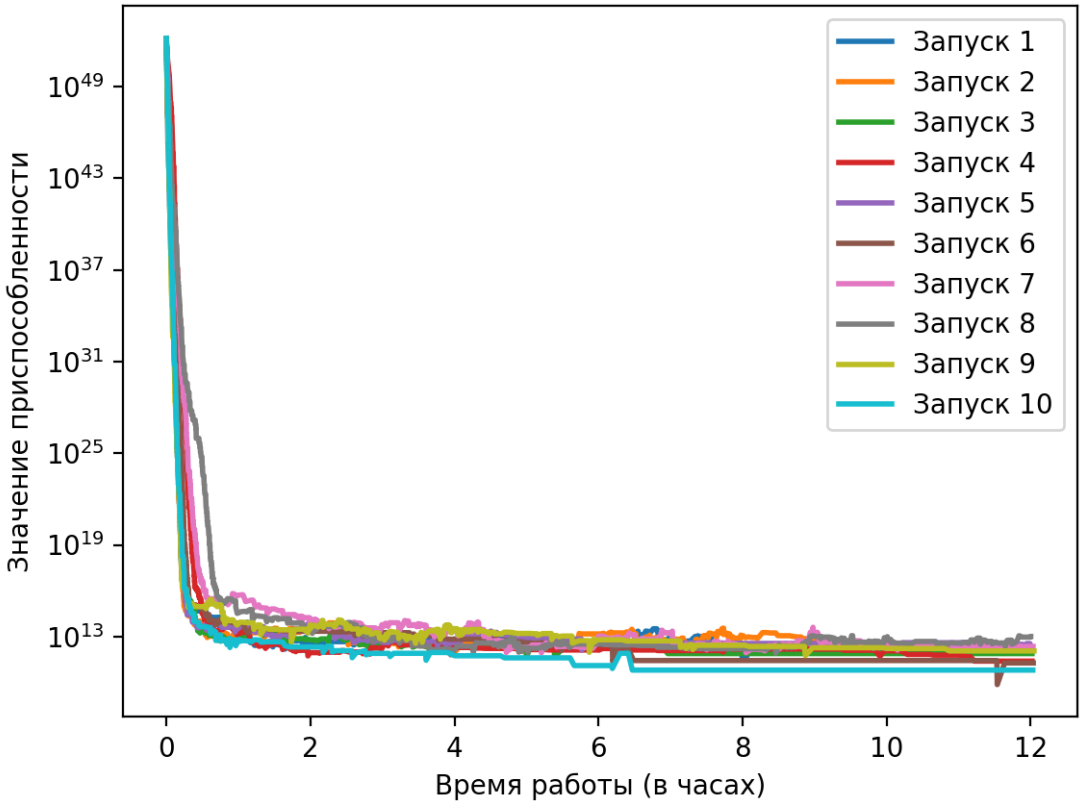

Рисунок 4 – Пример процесса оптимизации для алгоритма Bivium

В настоящей работе предлагается продолжить работу над эволюционными методами декомпозиции задач обращения криптографических функций и разработать новые методы сокращения временных затрат на вычисление функции приспособленности, которые не зависят от исследуемого криптографического алгоритма и не требуют затрат времени на предварительные вычис-

ления. Также их эффективность не зависит от значения функции приспособленности и не понижается в процессе работы алгоритма.

2.2. Эволюционный метод с использованием статистических тестов

В процессе вычисления значения функция приспособленности 2, которая используется для оценки декомпозиционных множеств типа *IBS*, всю последовательность событий S можно разделить на два типа:

— задача i была успешно решена за время t_i и удалось восстановить значения входных переменных, используя SAT-решатель A с ограничением на время работы T, то есть $t_i < T$

— задача i не была решена за время T с использованием SAT-решателя A

Предположим, что последовательность S из N событий состоит из N_+ событий первого типа, то есть N_+ задач было успешно решено SAT-решателем A с ограничением на время работы T. Обозначим время решения этих задач $t_1, t_2, ..., t_{N_+}$. Тогда оставшиеся $N - N_+$ событий второго типа соответствуют задачам, которые не были решены за время T с использованием SAT-решателя A. Исходя из этого, можно посчитать приблизительное время восстановления значений входных переменных для тех задач, которые были успешно решены на первом раунде:

$$E'(S) = \frac{\sum_{i=1}^{N_+} t_i}{N_+} + \frac{N - N_+}{N_+} \cdot T$$

Однако процесс проведения настоящей атаки отличается от процесса вычисления функции приспособленности для декомпозиционного множества B. Помимо того, что для каждого множества значений выходных переменных y^i необходимо восстановить значения входных переменных α^i по значениям переменных декомпозиционного множества $\beta(\alpha^i)$, но и проверить все оставшиеся $2^s - 1$ неправильных подстановок β^j в худшем случае. Тогда для событий второго типа, когда задача не была решена за время T, необходимо будет учесть еще время для решения всех неправильных подстановок β^j, а именно $(2^s - 1) \cdot T$. А для событий первого типа, учесть время $2^s/2 \cdot T$, которое в среднем будет дополнительно затрачено на перебор неправильных подстановок β^j пока не будет достигнута подстановка $\beta(\alpha^i)$, при решении которой будет осуществлен переход на следующий раунд.

С учетом этого оценка времени проведения атаки выглядит следующим образом:

$$E_B(S) = \frac{\sum_{i=1}^{N_+} t_i}{N_+} + \left(\frac{N}{N_+} - \frac{1}{2}\right) \cdot (2^s \cdot T) \tag{3}$$

Обратим внимание на то, что первое слагаемое много меньше второго, поскольку любое t_i не превышает T. Однако от этого оно не становится менее важным в виду того, что в процессе работы эволюционного алгоритма особи с одинаковым значением N_+ будут сравниваться исходя из величины этого первого слагаемого. Это позволяет направлять процесс оптимизации также в сторону уменьшения значений t_i при равном N_+.

Недостаток же такого подхода заключается в том, что N должно быть достаточно большим, чтобы погрешность вычисляемой оценки по формуле 3 была в разумных пределах, и получаемые значения приспособленности особей можно было сравнивать между собой. С другой стороны некоторые движения в пространстве поиска точно приводят к ухудшению особи. Однако для них также требуется вычислять значение функции приспособленности, что требует значительных затрат вычислительных ресурсов.

В работе [27] была впервые введена формула 2 для вычисления которой предполагалось использовать фиксированное значение N. В своей бакалаврской работе автор предложил стратегию адаптивного изменения значения N, которая позволяла экономить вычислительные ресурсы в процессе работы алгоритма. Однако у неё есть ряд недостатков, которые были описаны в подразделе 2.1.1.

В данной работе предлагается использовать статистические тесты для управления размером выборки N в процессе сравнения особей. От значения N линейно зависит время вычисления функции приспособленности. Изначально для каждой особи выборка состоит из минимального числа задач. Этих задач может быть уже достаточно, чтобы обнаружить статистически значимые различия между двумя особями и перейти к следующей особи или итерации. В противном случае, выборки расширяются пока особи не будут статистически различаться. Таким образом, статистические тесты используются в качестве функции приспособленности, подбирая оптимальный размер выборки N для каждой особи в процессе их сравнения. Предложенный метод позволит экономить вычислительные ресурсы для особей, которые заведомо лучше или хуже

чем текущая, обнаруживая статистически значимые различия на ранних стадиях.

2.2.1. Ограничения накладываемые на статистические тесты

В процессе оценки приспособленности особи, могут встретиться задачи двух типов описанных выше. Еще раз рассмотрим ситуацию, когда последовательность S из N событий состоит из N_+ событий первого типа (успешно решенные задачи с ограничением времени работы T SAT-решателя A) и $N - N_+$ событий второго типа. Тогда для оцениваемой особи множество измерений будет выглядеть следующим образом: $\{t_1, t_2, ..., t_{N_+}, T, T, ..., T\}$, где $N - N_+$ последних значений равно T. Во первых получаемые измерения ограничены справа величиной T, то есть невозможно различить измерения, значения которых выше T. Во вторых измерения $t_1, t_2, ..., t_{N_+}$ являются замером времени работы SAT-решателя A на булевой формуле $C(\gamma, \beta)$, и могут зависеть только от взаимодействия эвристик используемого SAT-решателя со структурой решаемой булевой формулы. Это говорит о том, что входное множество измерений не соответствует требованию о нормальном распределении, что является частым критерием для применения статистических тестов [20]. Исходя из этого был выбран U-критерий Манна Уитни [21], который является непараметрическим тестом и удовлетворяет накладываемым ограничениям. Также для сравнения был выбран еще один статистический тест – тест Барнарда [6]. В отличии от U-критерий Манна Уитни, этот тест используется для сравнения двух бинарных случайных величин, поэтому значения времени для успешно решенных задач будет проигнорировано, разделяя тестовую выборку лишь по типу события.

2.2.2. Сравнение особей

При сравнении особей B_1 и B_2 с помощью статистических тестов важным фактором является размер декомпозиционных множеств $|B_1|$ и $|B_2|$ для соответствующих особей. Если $|B_1| = |B_2|$, то можно сразу применять статистические тесты для соответствующих последовательностей событий S_1 и S_2.

В случае когда $|B_1| < |B_2|$ необходимо выполнить дополнительные преобразования над множеством измерений для особи B_1 или B_2. Исходя из формулы 3 оценки для этих особей будут различаться в $k = 2^{|B_2|-|B_1|}$ раз при прочих равных. Таким образом нам необходимо домножить или разделить все измерения для одной особи. Однако, в случае деления всех измерений, не будут

учитываться те измерения, которые в новых условиях могли бы стать положительными. Другими словами, задачи, которые не были решены за время T, могли быть решены за время $T \cdot k$, тем самым улучшив оценку. Но без затрат дополнительных вычислительных ресурсов невозможно определить такие задачи и корректно сформировать новую последовательность событий S_1' для B_1. Поэтому правильным решением будет домножить измерения для особи B_2. Тогда для сравнения особей B_1 и B_2 сформируем новую последовательность событий S_2' следующим образом:

— задача i была успешно решена за время t_i, где $t_i \cdot k < T$
— задача i была успешно решена за время t_i, где $t_i \cdot k \geqslant T$ или не была решена за время T

Теперь можно корректно сравнить особи B_1 и B_2 используя последовательностей событий S_1 и S_2'. А для случая $|B_1| > |B_2|$ порядок действий будет аналогичным.

Также при сравнении родительской B_p и дочерней B_c особей предполагается, что число измерений N_p для B_p было не меньше чем число измерений N_c для B_c. Это делается для того, чтобы свести погрешность метода Монте-Карло к минимуму.

2.2.3. Реализация на примере эволюционной стратегии (1+1)

Основная идея разрабатываемого эволюционного метода декомпозиции задач обращения криптографических функций с использованием статистических тестов представлена в листинге 1 на примере эволюционной стратеги (1+1). Алгоритм представлен в общей форме без уточнения используемых оператора мутации Mutate, статистического теста S и функции приспособленности F. А функция GetNewPoint используется для выбора начальной точки в пространстве поиска. Важной константой является N_{max}, которая контролирует максимальное число измерений для одной особи и используется для предотвращения ситуации, когда две очень похожие особи никогда не будут различаться статистическим тестом S (или потребуется слишком много времени для того, чтобы их различить). Также определена константа Q, которая отвечает за число производимых измерений за один раз. При запуске алгоритма в однопоточном режиме идеальным выбором будет использовать $Q = 1$, однако для запуска в многопоточном режиме, а тем более в распределенных вычислительных системах, потребуется выбрать такое Q, чтобы максимально сократить

Листинг 1 – Алгоритм (1+1)-EA с использованием статистических тестов

```
– f : {0; 1}^n → ℝ ∪ {∞}
– S : [ℝ ∪ {∞}] × [ℝ ∪ {∞}] → {−1; 0; 1}
x ← GetNewPoint()
f_x ← [f(x) for Q times]
while limit is not exhausted do
    y ← Mutate(x)
    f_y ← [f(y) for Q times]
    δ ← S(f_x, f_y)
    while δ = 0 do
        if |f_y| < |f_y| then
            f_y ← f_y ∪ [f(y) for Q times]
        else if |f_x| < N_max then
            f_x ← f_x ∪ [f(x) for Q times]
        else
            break
        end if
        δ ← S(f_x, f_y)
    end while
    if δ > 0 or (δ = 0 and F(f_x) > F(f_y)) then
        x ← y
        f_x ← f_y
    end if
end while
```

время простоя. Алгоритм работает пока не будет достигнут заданный $limit$ использованных вычислительных ресурсов, например по времени работы. Однако также может быть прерван при нахождении удовлетворяющего решения x.

В процессе оценки популяции особей алгоритм пытается выполнить как можно меньше измерений для того, чтобы понять является ли потенциальное решение y лучше ($\delta = 1$) или хуже ($\delta = -1$) текущего x. Заведомо плохие особи будут быстро отсеяны, а заведомо хорошие быстро займут место текущей. А для похожих особей, алгоритм будет увеличивать число измерений, пока статистический тест S не сможет их различить, или пока не будет достигнут предел для числа измерений N_{max}. В случае, если максимального числа измерений N_{max} недостаточно для достижения порога статистической значимости p-value, предлагается вычислить значение функции приспособленности F для принятия решения о замене текущей особи x на y. При замене особи

$x \leftarrow y$ также сохраняются все измерения которые были ранее для нее вычислены $f_x \leftarrow f_y$.

2.3. Эволюционные методы с использованием инкрементальных SAT-решателей

В данной работе декомпозиция задачи обращения криптографических функций представляет из себя guess-and-determine атаку. Проведение атаки на криптографический алгоритм, используя построенное декомпозиционное (guessed bits) множество B, осуществляется посредством перебора всех возможных подстановок значений переменных для этого самого декомпозиционного множества, пока не будет найден удовлетворяющий набор значений β_{true} соответствующий заданному ключевому потоку γ. После нахождения такого удовлетворяющего набора значений β_{true} можно восстановить секретный ключ α, фрагментом которого он и является.

Рассматривая процесс атаки в рамках SAT-based криптоанализа, каждая такая подстановка i значений переменных β_i из всех возможных будет соответствовать определенной SAT-задаче $C(\gamma, \beta_i)$. Все эти задачи очень похожи. Во первых, КНФ формула C задает криптографический алгоритм и может быть использована как для генерации бит ключевого потока $C(\alpha)$ (для этого необходимо подставить значения бит секретного ключа), так и для формирования задачи обращения $C(\gamma)$ (для этого необходимо подставить значения бит ключевого потока). Во вторых, целью проведения атаки является восстановить секретный ключ α, который использовался для генерации соответствующего ключевого потока γ, а значит ключевой поток не меняется в процессе проведения атаки. В третьих, декомпозиционное множество, с помощью которого производится атака, не меняется в процессе проведения атаки, а разнятся лишь значения переменных этого множества. Эти особенности позволяют использовать инкрементальное решения задач в процессе проведения атаки, используя технику допущений (assumptions). Для этого инициализируем SAT-решатель формулой $C(\gamma)$, которая не будет меняться на протяжении всей guess-and-determine атаки. И для каждой задачи i будем запускать решение в инкрементальном режиме, используя в качестве допущения l_i множество значений β_i для соответствующего декомпозиционного множества B. Таким образом, инкрементальный SAT-решатель будет сохранять дизъюнкты конфликтов выведенных в одном запуске, чтобы использовать их для решения последу-

ющих задач. Это должно сократить затраты вычислительных ресурсов и, как следствие, уменьшить время на проведение атаки.

2.3.1. Инкрементальная функция приспособленности

В данной работе предлагается использовать инкрементальное решение задач в процессе вычисления значения приспособленности. Функция приспособленности в данной работе позволяет получить оценку времени, которое потребуется на совершение атаки для оцениваемого декомпозиционного множества. Для вычисления оценки выбираются N случайных задач вида $C(\gamma, \beta)$ из всего множества задач оцениваемой декомпозиции. Как уже было описано выше, все эти задачи очень похожи, что позволяет использовать инкрементальный режим в процессе их решения.

Небольшое отличие будет для формулы 2, в процессе вычисления которой для каждой задачи i генерируется правильная уникальная комбинация секретного ключа α_i и ключевого потока γ_i. Таким образом SAT-решатель будет инициализирован формулой C, а в качестве допущений l_i будет выступать множество значений $\gamma_i \cap \beta_i$, где $\beta_i = B(\alpha_i)$ фрагмент секретного ключа α_i для декомпозиционного множества B.

Процесс вычисления функции приспособленности выглядит следующим образом:

— формируется выборка из N задач обращения криптографической функции ослабленных оцениваемой декомпозицией B
— определяется общая для этих N задач КНФ формула C', в зависимости от используемой функции ($C(\gamma)$ для формулы 1, C для формул 2 и 3)
— для каждой задачи i определяется множество допущений l_i, также в зависимости от используемой функции (β_i для формулы 1, $\gamma_i \cap \beta_i$ для формул 2 и 3)
— для каждого потока j (в случае использования многопоточности) инициализируется SAT-решатель A_j формулой C'
— для каждой задачи i на потоке j запускается SAT-решатель A_j на множестве допущений l_i и сохраняется результат его работы
— когда все N задач будут решены, в зависимости от используемой функции, вычисляется значение приспособленности для оцениваемого декомпозиционного множества B

Выводы по главе 2

а) Предложен новый эволюционный метод декомпозиции задач обращения криптографических функций с использованием статистических тестов. Статистические тесты применяют в процессе сравнения особей, выступая в качестве функции приспособленности и управляя размеров выборки N.

б) Предложена инкрементальная функция приспособленности для оценки декомпозиционных множеств.

ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ РАЗРАБОТАННЫХ МЕТОДОВ ДЕКОМПОЗИЦИИ ЗАДАЧ ОБРАЩЕНИЯ КРИПТОГРАФИЧЕСКИХ ФУНКЦИЙ

В данной главе приводятся результаты экспериментального исследования эффективности методов декомпозиции, разработанных в данной работе. В разделе 3.1 представлено описание и результаты применения статистических тестов к декомпозиции задачи обращения исследуемых криптографических функций. В разделе 3.2 представлено описание и результаты применения инкрементальных SAT-решателей к декомпозиции задачи обращения исследуемых криптографических функций.

Для экспериментов были использован кластер с возможностью проведения распределенных вычислений. Каждый вычислительный узел кластера содержит два 18-ядерных процессора Intel®Xeon®E5-2695 v4 с частотой 2.1 ГГц. Далее все полученные оценки и измерения времени указаны из расчета на одно ядро процессора Intel®Xeon®E5-2695 v4.

3.1. Исследование применимости статистических тестов

Для экспериментов из этого раздела используются две различные конфигурации экспериментальной среды.

Конфигурация 1. Генетический алгоритм с моделью развития популяции Элитизм. Размер популяции был равен 10. Популяция формируется следующим образом:

а) Выбираются две лучшие особи из предыдущей популяции (элита) для образования следующей популяции.

б) Используя селекцию по принципу рулетки, выбираются четыре особи из предыдущей популяции, к которым применяется оператор мутации. Получившиеся особи добавляются к следующей популяции.

в) Используя селекцию по принципу рулетки, выбираются две пары особей из предыдущей популяции, к которым применяется оператор скрещивания. Пары получившихся особей добавляются к следующей популяции.

Для мутации использовался стандартный битовый оператор мутации с вероятностью обратно пропорциональной размеру особи $p_{mutate} = \frac{1}{l}$. Для скрещивания – равномерный оператор скрещивания с вероятностью $p_{cross} = 0{,}2$. Размер выборки Монте-Карло $N_{max} = 500$, а размер одного блока $Q = 100$ (выборка может состоять максимум из 5 блоков размера Q). Для статистического теста

уровень значимости p-value $= 0.05$. В качестве SAT-решателя использовался ROKK [34] с ограничением на время работы $tl = 10$ секунд. Каждый запуск алгоритма производился на 5 узлах вычислительного кластера с выделенным бюджетом времени в 12 часов. И для каждого исследуемого криптографического алгоритма с использованием этой конфигурации было произведено 5 независимых запусков.

Конфигурация 2. Эволюционная стратегия (1+1). Для мутации использовался такой же стандартный битовый оператор мутации. Размер выборки Монте-Карло $N_{max} = 500$, а размер одного блока $Q = 50$. Для статистического теста уровень значимости p-value $= 0.05$. В качестве SAT-решателя также использовался ROKK [34] с ограничением на время работы $tl = 10$ секунд. Каждый запуск алгоритма производился на одном узле вычислительного кластера с выделенным бюджетом времени в 12 часов. И для каждого исследуемого криптографического алгоритма с использованием этой конфигурации было произведено 10 независимых запусков.

Все приведенные далее оценки времени guess-and-determine атак на криптографические алгоритмы, используя соответствующие декомпозиционные множества B, были вычислены на выборке из $N = 10000$ задач. А полученные значения времени были вычислены из расчета на одно ядро процессора Intel®Xeon®E5-2695 v4.

3.1.1. Сравнение U-критерия Манна-Уитни и теста Барнарда

В ходе проведения экспериментального исследования использовался не только U-критерия Манна-Уитни для сравнения особей. Также был произведен запуск теста Барнарда на тех же входных данных. Сравнение статистических тестов осуществлялось с использованием первой конфигурации экспериментальной среды. А для его проведения был сохранен каждый результат сравнения w_i двух особей в ходе работы эволюционного алгоритма с использование U-критерия Манна-Уитни для каждого запуска на криптографических алгоритмах A5/1, Bivium, Trivium 64 и Trivum 96. Далее, для каждого такого сохраненного результата w_i было также проведено сравнение b_i тех же двух особей уже с использованием теста Барнарда. Для каждой такой пары сравнений двух особей (w_i, b_i) могли получиться следующие исходы:

— w_i различает сравниваемые особи, а b_i – нет

— b_i различает сравниваемые особи, а w_i – нет

— оба теста различают сравниваемые особи

— никто не различает сравниваемые особи

Таблица 1 – Число исходов при сравнении статистических тестов, полученных на одних и тех же входных данных

Алгоритм	U-критерий Манна-Уитни (s_w)	Тест Барнарда (s_b)	Оба (s_{both})	Никто (s_{none})
A5/1	215	146	1182	5812
Bivium	3786	946	9381	3974
Trivium 64	1943	560	5951	8476
Trivium 96	738	318	3322	8092

Результаты данного эксперимента представлены в таблице 1. Из полученных данных видно, что предложенные к сравнению статистические тесты, в большинстве случаев, согласуются в своих решениях при проверке нулевой гипотезы. Для вычисления процента согласованности используем формулу:

$$\frac{(s_{both} + s_{none}) \cdot 100\%}{s_{both} + s_{none} + s_w + s_b}$$

Например, для криптографического алгоритма A5/1 процент согласованности равен 95.1%, для алгоритма Bivium – 73.8%, для алгоритма Trivium 64 – 85.2% и для алгоритма Trivium 96 – 91.5%.

Для сравнения статистических тестов посчитаем их разрешающую способность. Под разрешающей способностью понимается процент таких исходов, когда нулевая гипотеза была верно отвергнута до достижения лимита на размер выборки N_{max}. То есть, такой процент сравнений особей, когда статистический тест различил особи на выборке меньшей чем N_{max}, тем самым сэкономив вычислительные ресурсы. Для вычисления разрешающей способности воспользуемся следующей формулой ($s = s_w$ для U-критерий Манна-Уитни и $s = s_b$ для теста Барнарда):

$$\frac{(s + s_{both}) \cdot 100\%}{s_{both} + s_{none} + s_w + s_b}$$

Из результатов, представленных в таблице 2 видно, что разрешающая способность U-критерий Манна-Уитни больше чем у теста Барнарда.

Таблица 2 – Разрешающая способность используемых статистических тестов

Алгоритм	U-критерий Манна-Уитни, %	Тест Барнарда, %
A5/1	18.99	18.06
Bivium	72.80	57.10
Trivium 64	46.63	38.46
Trivium 96	32.56	29.19

Таким образом, проведенный эксперимент оправдывает использование U-критерия Манна-Уитни для эволюционных методов декомпозиции задач обращения криптографических функций.

3.1.2. Сравнение числа просмотренных точек

Алгоритм просматривает много потенциальных решений в процессе оптимизации. От числа просмотренных точек в пространстве поиска зависит качество найденного решения. Поэтому было проведено сравнение числа просмотренных точек для метода с использование статистических тестов и метода с адаптивной стратегией. Результаты эксперимента приведены в таблице 3.

Таблица 3 – Сравнение числа просмотренных точек в пространстве поиска

Алгоритм	Статистические тесты	Адаптивная стратегия
A5/1	1471	341
Bivium	3616	2439
Trivium 64	3398	1323
Trivium 96	2494	1299

Видно, что для всех криптографических алгоритмов было просмотрено точек в несколько раз больше. Для алгоритма A5/1 в 4.31 раза больше, для алгоритма Bivium – 1.48, для алгоритма Trivium 64 – 2.57 и для алгоритма Trivium 96 – 1.92. Далее (в разделе 3.1.4.1) будет приведено сравнение качества построенных декомпозиционных множеств, которое докажет важность данного показателя.

3.1.3. Построение декомпозиционных множеств

В результате проведенных экспериментов с использованием первой конфигурации для криптографических алгоритмов A5/1, Bivium, Trivium 64 и

Trivum 96 были построены декомпозиционные множества. Параметры лучших построенных декомпозиционных множеств приведены в таблице 4. А сами множества приведены в таблице B.1 приложения B.

Таблица 4 – Параметры лучших декомпозиционных множеств построенных генетическим алгоритмом

Алгоритм	$\|B\|$	Отсечка по времени tl, сек.	Оценка, сек.
A5/1	32	0.278	$2.19 \cdot 10^{12}$
Bivium	28	2.715	$1.15 \cdot 10^{12}$
Trivium 64	21	2.373	$3.23 \cdot 10^{7}$
Trivium 96	25	2.485	$1.24 \cdot 10^{12}$

Также были проведены эксперименты с использованием второй конфигурации для криптографических алгоритмов A5/1, Bivium, Trivium, Trivium 64, Trivum 96, ASG 72, ASG 96, ASG 192. Параметры для лучших построенных декомпозиционных множеств приведены в таблице 5. А сами множества приведены в таблице B.2 приложения B.

Таблица 5 – Параметры лучших декомпозиционных множеств построенных эволюционной стратегией (1+1)

Алгоритм	$\|B\|$	Отсечка по времени T, сек.	Оценка, сек.
A5/1	26	9.91	$3.63 \cdot 10^{11}$
Bivium	27	9.93	$7.40 \cdot 10^{11}$
Trivium	130	8.34	$4.87 \cdot 10^{43}$
Trivium 64	17	9.64	$2.03 \cdot 10^{7}$
Trivium 96	28	9.79	$1.46 \cdot 10^{12}$
ASG 72	10	0.57	4794.55
ASG 96	19	0.88	$1.56 \cdot 10^{6}$
ASG 192	42	9.75	$1.74 \cdot 10^{16}$
Grain v0	61	9.80	$3.76 \cdot 10^{22}$
Grain v1	87	1.59	$7.37 \cdot 10^{30}$
Mickey	153	9.03	$2.81 \cdot 10^{50}$

Все оценки указанные в обеих таблицах означают оценку времени, за которое можно совершить guess-and-determine атаку на соответствующий криптографический алгоритм, используя построенное декомпозиционное множество. Для этого потребуется решить порядка $2^{|B|}$ задач для соответствующей

декомпозиции. Следует отметить, что каждая такая задача может быть решена независимо от остальных, что позволяет проводить guess-and-determine атаки в параллельных и распределенных системах. Даже с учетом этого, не для всех алгоритмов это можно сделать за разумное время. Однако в разделе 3.3 будет продемонстрирована атака на алгоритм ASG 72.

3.1.4. Сравнение эффективности построенных guess-and-determine атак

В данном разделе будет приведено сравнение эффективности атак, которые могут быть реализованы с использованием построенных декомпозиционных множеств.

3.1.4.1. Сравнение с адаптивной стратегий

Было проведено сравнение с методом на основе адаптивной стратегии, которая был предложен автором в своей бакалаврской работе. Для сравнения рассмотрим следующие криптографические алгоритмы:

— в работе [30] были опубликованы результаты для алгоритмов Bivium, Trivium 64, Trivum 96, ASG 72, ASG 96 и ASG 192

— в работе [28] были опубликованы результаты для алгоритмов Trivium, Mickey, Grain v0 и Grain v1

Для всех этих алгоритмов декомпозиционные множества строились с использованием адаптивной стратегией. В таблице 6 приведено их сравнение с методом использующим статистические тесты.

Таблица 6 – Сравнение с лучшими результатами для метода на основе адаптивной стратегии

Алгоритм	Статистические тесты		Адаптивная стратегия	
	|B|	Оценка, сек	|B|	Оценка, сек
Bivium	27	$7.40 \cdot 10^{11}$	39	$1.49 \cdot 10^{12}$
Trivium 64	17	$2.03 \cdot 10^{7}$	21	$3.19 \cdot 10^{7}$
Trivium 96	28	$1.46 \cdot 10^{12}$	40	$2.09 \cdot 10^{12}$
ASG 72	10	4794.55	9	5604.8
ASG 96	19	$1.56 \cdot 10^{6}$	16	$3.72 \cdot 10^{6}$
ASG 192	42	$1.74 \cdot 10^{16}$	44	$2.84 \cdot 10^{17}$
Trivium	130	$4.87 \cdot 10^{43}$	143	$1.52 \cdot 10^{43}$
Mickey	153	$2.81 \cdot 10^{50}$	152	$9.73 \cdot 10^{50}$
Grain v0	61	$3.76 \cdot 10^{22}$	73	$4.42 \cdot 10^{23}$
Grain v1	87	$7.37 \cdot 10^{30}$	102	$7.23 \cdot 10^{31}$

Видно, что для большинства криптографических алгоритмов удалось построить декомпозиционные множества с меньшей оценкой, а значит более эффективные guess-and-determine атаки. Этого удалось достичь благодаря увеличению числа просматриваемых точек пространства в процессе оптимизации.

3.1.4.2. SAT и UNSAT иммунность

В работе [9] были описаны такие понятия как SAT и UNSAT иммунность (SAT/UNSAT immunity). И были предложены разные стратегии по исследованию SAT и UNSAT иммунности для криптографических алгоритмов. По сути своей, эта два совершенно разных подхода к построению guess-and-determine атак. То есть, декомпозиционные множества построенные с использованием одного подхода не могут эффективно применяться для проведения guess-and-determine атаки другого типа. Аналогично и для вычисления оценки времени, которое потребуется на проведение атаки.

Таким образом функция 1 используется для построения декомпозиционных множеств нацеленных на быстрое решение невыполнимых (UNSAT) задач, то есть, исследуя UNSAT иммунность. А функции 2 и 3 для быстрого решения выполнимых (SAT) задач, исследуя SAT иммунность.

3.1.4.3. Сравнение с другими автоматизированными методами

Теперь проведем сравнение с результатами построенные другими автоматизированными методами декомпозиции задач обращения криптографических функций. Единственной проблемой является, то что другие результаты было получены в процессе оптимизации целевой функции для формулы 1. И, как было описано в подразделе 3.1.4.2, такие guess-and-determine атаки нацелены на нахождение уязвимости для быстрого решения невыполнимых (UNSAT) задач. В данной работе в качестве функции приспособленности выступает формула 2, в процессе оптимизации которой строятся декомпозиционные множества исследующие SAT иммунность.

Для сравнения были выбраны следующие работы:

— в работе [27] были опубликованы результаты для алгоритмов A5/1, Bivium, Grain v0
— в работе [35] были опубликованы результаты для генераторов ключевого потока ASG 72, ASG 96 и ASG 192
— в работе [36] были опубликованы результаты для алгоритмов Trivium, Mickey и Grain v1

Во всех этих работах guess-and-determine атаки были построены посредством оптимизации целевой функции 1. А в таблице 7 приведено их сравнение с методом использующим статистические тесты.

Таблица 7 – Сравнение с результатами, опубликованными в работах [27, 35, 36]

Алгоритм	Статистические тесты		Другие работы	
	|B|	Оценка, сек	|B|	Оценка, сек
A5/1	26	$3.63 \cdot 10^{11}$	32	$4.64 \cdot 10^{8}$
Bivium	27	$7.40 \cdot 10^{11}$	50	$3.77 \cdot 10^{10}$
Grain v0	61	$3.76 \cdot 10^{22}$	69	$4.37 \cdot 10^{20}$
ASG 72	10	4794.55	21	1116
ASG 96	19	$1.56 \cdot 10^{6}$	30	$1.15 \cdot 10^{6}$
ASG 192	42	$1.74 \cdot 10^{16}$	63	$2.72 \cdot 10^{15}$
Trivium	130	$4.87 \cdot 10^{43}$	145	$1.4 \cdot 10^{41}$
Mickey	153	$2.81 \cdot 10^{50}$	158	$1.56 \cdot 10^{48}$
Grain v1	87	$7.37 \cdot 10^{30}$	109	$4.04 \cdot 10^{30}$

На данный момент не удалось построить более эффективные декомпозиционные множества по сравнению с другими автоматизированными методами. Однако проведенное сравнение нельзя полностью считать корректным, поскольку сравниваются два разных подхода к построению guess-and-determine атак (первый [32] направлен на исследование UNSAT иммунности, а второй [27] – SAT иммунности). Также следует отметить, что получаемые оценки не всегда подтверждаются в процессе проведения самой атаки, или же наоборот, переоценивают время необходимое на её проведение. А для некоторых криптографических алгоритмов невозможно это проверить, ввиду их высокой криптографической стойкости. Также нельзя не учитывать фактор случайности, поскольку неизвестно сколько UNSAT задач необходимо будет решить, пока не будет найдена и решена единственная SAT задача и восстановлен секретный ключ. Поэтому недостаточно провести эксперимент с одной или несколькими криптографическими атаками для подтверждения полученной оценки для построенного декомпозиционного множества. Необходимо провести множество случайных атак с использованием одного декомпозиционного множества и усреднять время, затрачиваемое на их проведение. Такой эксперимент описан в разделе 3.3 для генератора ключевого потока ASG 72.

3.2. Исследование применимости инкрементальных SAT-решателей

Для экспериментов из этого раздела используется следующая конфигурация экспериментальной среды. Эволюционная стратегия (1+1). Для мутации использовался стандартный битовый оператор мутации. Размер выборки Монте-Карло $N_{max} = 500$. В качестве SAT-решателя также использовался инкрементальный Glucose 3 [4] с ограничением на время работы $tl = 10$ секунд. Каждый запуск алгоритма производился на одном узле вычислительного кластера с выделенным бюджетом времени в 24 часа. И для каждого исследуемого криптографического алгоритма с использованием этой конфигурации было произведено 3 независимых запуска.

Результаты проведенных экспериментов для криптографических алгоритмов A5/1, Bivium, Trivium 64 и Trivum 96 приведены в таблице 8.

Таблица 8 – Сравнение результатов для инкрементальной функции приспособленности

Алгоритм	Инкрементальная ФП		Обычная ФП	
	|B|	Оценка, сек	|B|	Оценка, сек
A5/1	24	$4.07 \cdot 10^{12}$	27	$1.44 \cdot 10^{12}$
Bivium	24	∞	38	$1.03 \cdot 10^{13}$
Trivium 64	15	$2.58 \cdot 10^{8}$	15	$2.45 \cdot 10^{8}$
Trivium 96	25	$8.13 \cdot 10^{12}$	30	$3.92 \cdot 10^{13}$

Из результатов в таблице 8 видно, что используя обычную функцию приспособленности, в большинстве случаев (для A5/1, Trivium 64, Bivium), получается строить декомпозиционные множества с лучшими оценками. В процессе оптимизации инкрементальной функции приспособленности из декомпозиционного множества B исключается слишком много переменных, что хорошо заметно для алгоритма Bivium. А поскольку для вычисления финальной оценки для построенного декомпозиционного множества B не предполагается использование инкрементального решения задач, то в процессе декомпозиции получаются слишком сложные задач, которые уже не могут быть решены без использования инкрементальных SAT-решателей.

3.3. Атака на Alternating Step Generator

В рамках данной работы была также проведена guess-and-determine атака на генератор ключевого потока ASG 72 с использованием построенного декомпозиционного множества, которое приведено в таблице B.2 приложения B.

Процесс проведения криптографической атаки соответствовал используемой функции приспособленности (формула 2) и выглядит следующим образом:

— генерируется фрагмент ключевого потока γ и запоминается секретный ключ α, который использовался для его генерации
— перебирается все множество значений β_i для декомпозиционного guessed bits множества B и для каждого такого β_i решается задача $C(\gamma, \beta_i)$ с ограничением времени T
— если было найдено такое β_i, для которого решение задачи $C(\gamma, \beta_i)$ являлось истинным, то это означает, что секретный ключ был найден, и перебор останавливался

Важно уточнить, что от выбранного ограничения времени T зависит не только время на проведения guess-and-determine атаки, но и вероятность её успешного проведения. В таблице 9 представлены результаты экспериментов для разных значений ограничения времени T. Было решено 1000 случайных задач обращения криптографической функции для каждого из значений T. Для самого удачного значения $T = 2.0$ секунды было увеличено число решенных задач обращения до 10000.

Таблица 9 – Экспериментальные результаты проведения атак на алгоритм ASG 72

Число атак	Ограничение времени, сек.	Успешные атаки, %	Усредненное время, сек	Отклонение времени, сек
1000	1.0	64.8	1039.24	17.43
1000	2.0	100	1081.32	492.72
1000	3.0	100	1592.62	281.56
10000	2.0	100	1084.26	486.00

Сравнивая усредненное время на совершение атаки с оценкой, приведенной в таблице 5 для алгоритма ASG 72, можно сделать вывод о том, что используемая формула 2 имеет большой запас прочности. На проведение атаки требуется в среднем в 4.5 раза меньше времени, а в худшем случае – в 3 раза меньше.

Также в результате эксперимента были обнаружены коллизии генератора ключевого потока ASG 72. В 3% случаях результатом решения задачи обращения криптографической функции i являлся секретный ключ α'_i отличный

от исходного α_i. Однако оба этих секретных ключа (α_i и α_i') генерируют один и тот же фрагмент ключевого потока γ.

Выводы по главе 3

а) Был проведено сравнение статистических тестов U-критерия Манна-Уитни и теста Барнарда.

б) Было продемонстрировано увеличение числа просматриваемых точек в пространстве поиска при использовании статистических тестов.

в) Были построены guess-and-determine атаки на ряд криптографических алгоритмов и приведено их сравнение с другими известными результатами.

г) Было проведено экспериментальное исследование предложенной инкрементальной функции приспособленности.

д) Была совершена guess-and-determine атака на генератор ключевого потока Alternating Step Generator, используя построенное декомпозиционное множество.

ЗАКЛЮЧЕНИЕ

В данной работе был сделан обзор существующих автоматизированных методов декомпозиции задач обращения криптографических функций. Было выявлено, что эволюционные алгоритмы ранее не применялись для построения декомпозиционных множеств в алгебраическом криптоанализе. В своей бакалаврской работе автор предложил применять эволюционные алгоритмы для декомпозиции задач обращения криптографических функций. В данной работе была разработана новая адаптивная стратегия подсчета функции приспособленности, основанная на статистических тестах. Главная её идея заключается в том, чтобы сравнивать особи не напрямую через значения функции приспособленности, а, вместо этого, статистически сравнивать измерения из выборок Монте-Карло, на которых вычисляются соответствующие значения приспособленности. Это также позволило использовать статистические тесты для управления размером этой самой выборки, что позволило быстро отсеивать заведомо плохие особи в процессе оптимизации. Нельзя не сказать о том, что разработанная адаптивная стратегия может использоваться с любым другим алгоритмом оптимизации. Более того, может применяться не только для задач криптоанализа, но и для другой любой задачи предполагающей вычисления значения функции приспособленности, используя метод Монте-Карло.

Экспериментальным путем было проведено сравнение различных статистических тестов. А в результате экспериментального исследования было показано, что эволюционный алгоритм с предложенной адаптивной стратегии просматривает в несколько раз больше потенциальных точек для ряда криптографических алгоритмов, чем без неё. Также были построены guess-and-determine атаки на следующие криптографические алгоритмы: A5/1, Bivium, Trivium, Trivium 64, Trivium 96, ASG 72, ASG 96, ASG 192, Mickey, Grain v0, Grain v1. Сравнение полученных результатов показало, что предложенный метод эффективней для всех исследуемых криптографических алгоритмов по сравнению с эволюционным методом, который был предложен автором ранее.

Однако по сравнению с другими автоматизированными методами декомпозиции не удалось построить более эффективные guess-and-determine атаки, но для оценки времени её проведения использовался подход отличный от используемого в данной работе. Нельзя исключать и возможность того, что полученные конкурентами оценки слишком оптимистичны и могут не подтвер-

ждаться в реалиях настоящей криптографической атаки. Также для некоторых алгоритмов подобная оценка криптографической стойкости была впервые проведена с использованием автоматических методов декомпозиции.

По соответствующей теме были сделаны доклады на следующих научных конференциях: VIII Всероссийский конгресс молодых ученых (2019 год, НИУ ИТМО), EvoStar (2019 год, Leipzig, Germany), 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO, 2019 год, Opatija, Croatia), IX Всероссийский конгресс молодых ученых (2020 год, НИУ ИТМО). А также были опубликованы 3 научные работы в трудах крупных зарубежных конференций.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms / J. Derrac [et al.] // Swarm and Evolutionary Computation. — 2011. — Vol. 1, no. 1. — P. 3–18.

2 A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization / S. García [et al.] // Journal of Heuristics. — 2009. — Vol. 15, no. 6. — P. 617.

3 *Anderson R.* A5 (was: Hacking digital phones) // Newsgroup Communication. — 1994.

4 *Audemard G.*, *Lagniez J.-M.*, *Simon L.* Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction // International conference on theory and applications of satisfiability testing. — Springer. 2013. — P. 309–317.

5 *Bard G.* Algebraic cryptanalysis. — Springer Science & Business Media, 2009.

6 *Barnard G. A.* A new test for 2× 2 tables // Nature. — 1945. — Vol. 156. — P. 177.

7 *Bayardo Jr R. J.*, *Schrag R.* Using CSP look-back techniques to solve real-world SAT instances // Aaai/iaai. — Providence, RI. 1997. — P. 203–208.

8 *Beyer H.-G.* The theory of evolution strategies. — Springer Science & Business Media, 2001.

9 *Courtois N. T.*, *Gawinecki J. A.*, *Song G.* Contradiction immunity and guess-then-determine attacks on GOST // Tatra Mountains Mathematical Publications. — 2012. — Vol. 53, no. 1. — P. 65–79.

10 *Courtois N. T.*, *Pieprzyk J.* Cryptanalysis of block ciphers with overdefined systems of equations // International Conference on the Theory and Application of Cryptology and Information Security. — Springer. 2002. — P. 267–287.

11 Cryptanalysis of four-round DES based on genetic algorithm / J. Song [et al.] // 2007 International Conference on Wireless Communications, Networking and Mobile Computing. — IEEE. 2007. — P. 2326–2329.

12 *De Canniere C.*, *Preneel B.* Trivium specifications // eSTREAM, ECRYPT Stream Cipher Project. — Citeseer. 2005.

13 *De Canniere C.*, *Preneel B.* Trivium // New Stream Cipher Designs. — Springer, 2008. — P. 244–266.

14 Encoding cryptographic functions to SAT using Transalg system / I. Otpuschennikov [et al.] // Proceedings of the Twenty-second European Conference on Artificial Intelligence. — IOS Press. 2016. — P. 1594–1595.

15 Evolutionary computation based cryptanalysis: A first study / E. Laskari [et al.] // Nonlinear Analysis: Theory, Methods & Applications. — 2005. — Vol. 63, no. 5–7. — e823–e830.

16 *Fisher R. A.* On the interpretation of χ 2 from contingency tables, and the calculation of P // Journal of the Royal Statistical Society. — 1922. — Vol. 85, no. 1. — P. 87–94.

17 *François O.*, *Lavergne C.* Design of evolutionary algorithms-A statistical perspective // IEEE Transactions on Evolutionary Computation. — 2001. — Vol. 5, no. 2. — P. 129–148.

18 *Garg P.*, *Varshney S.*, *Bhardwaj M.* Cryptanalysis of simplified data encryption standard using genetic algorithm // American Journal of Networks and Communications. — 2015. — Vol. 4, no. 3. — P. 32–36.

19 *Günther C. G.* Alternating step generators controlled by de Bruijn sequences // Workshop on the Theory and Application of of Cryptographic Techniques. — Springer. 1987. — P. 5–14.

20 *Kruskal W. H.*, *Wallis W. A.* Use of ranks in one-criterion variance analysis // Journal of the American statistical Association. — 1952. — Vol. 47, no. 260. — P. 583–621.

21 *Mann H. B.*, *Whitney D. R.* On a test of whether one of two random variables is stochastically larger than the other // The annals of mathematical statistics. — 1947. — P. 50–60.

22 *Marques-Silva J. P.*, *Sakallah K. A.* GRASP: A search algorithm for propositional satisfiability // IEEE Transactions on Computers. — 1999. — Vol. 48, no. 5. — P. 506–521.

23 *Matthews R. A.* The use of genetic algorithms in cryptanalysis // Cryptologia. — 1993. — Vol. 17, no. 2. — P. 187–201.

24 *Metropolis N.*, *Ulam S.* The Monte Carlo Method // J. Amer. Statistical Assoc. — 1949. — Vol. 44, no. 247. — P. 335–341.

25 *Mitchell M.* An introduction to genetic algorithms. — MIT press, 1998.

26 Model design for a reduced variant of a Trivium Type Stream Cipher / A. C. Lechtaler [et al.] // Journal of Computer Science and Technology. — 2014. — Vol. 14, no. 01. — P. 55–58.

27 On cryptographic attacks using backdoors for SAT / A. Semenov [et al.] // Thirty-Second AAAI Conference on Artificial Intelligence. — 2018.

28 Parallel Framework for Evolutionary Black-box optimization with Application to Algebraic Cryptanalysis / A. Pavlenko [et al.] // 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). — IEEE. 2019. — P. 1144–1149.

29 Parallel logical cryptanalysis of the generator A5/1 in BNB-Grid system / A. Semenov [et al.] // International Conference on Parallel Computing Technologies. — Springer. 2011. — P. 473–483.

30 *Pavlenko A.*, *Semenov A.*, *Ulyantsev V.* Evolutionary computation techniques for constructing SAT-based attacks in algebraic cryptanalysis // International Conference on the Applications of Evolutionary Computation (Part of EvoStar). — Springer. 2019. — P. 237–253.

31 *Schaefer E. F.* A simplified data encryption standard algorithm // Cryptologia. — 1996. — Vol. 20, no. 1. — P. 77–84.

32 *Semenov A.*, *Zaikin O.* Algorithm for finding partitionings of hard variants of boolean satisfiability problem with application to inversion of some cryptographic functions // SpringerPlus. — 2016. — Vol. 5, no. 1. — P. 554.

33 *Wilcoxon F.* Individual comparisons by ranking methods // Breakthroughs in statistics. — Springer, 1992. — P. 196–202.

34 *Yasumoto T.*, *Okuwaga T.* Rokk 1.0.1 // SAT Competition. — 2014. — Vol. 2014. — P. 70.

35 *Zaikin O.*, *Kochemazov S.* An improved SAT-based guess-and-determine attack on the alternating step generator // International Conference on Information Security. — Springer. 2017. — P. 21–38.

36 *Zaikin O.*, *Kochemazov S.* Pseudo-boolean black-box optimization methods in the context of divide-and-conquer approach to solving hard SAT instances // DEStech Transactions on Computer Science and Engineering. — 2018. — Optim.

ПРИЛОЖЕНИЕ А. СТРУКТУРА КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ

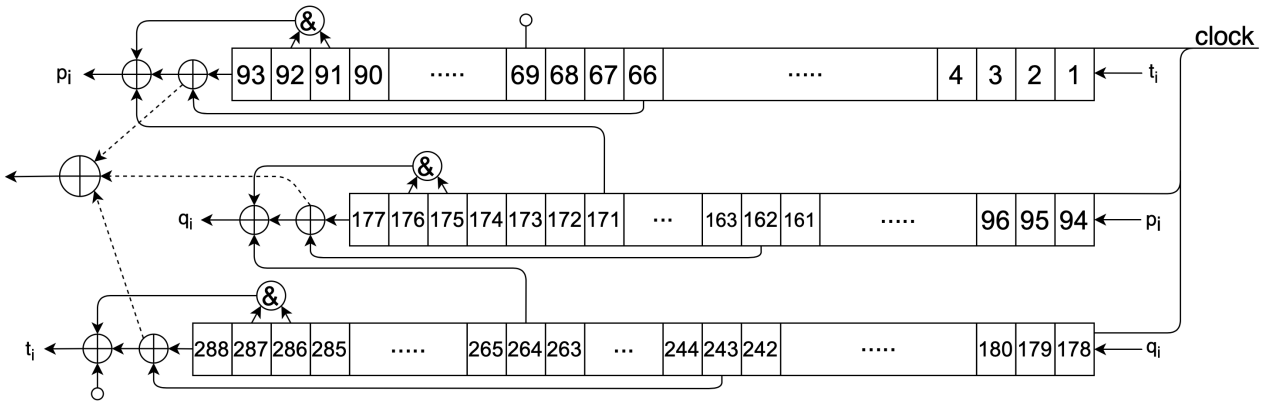

Рисунок А.1 – Структура алгоритма Trivium

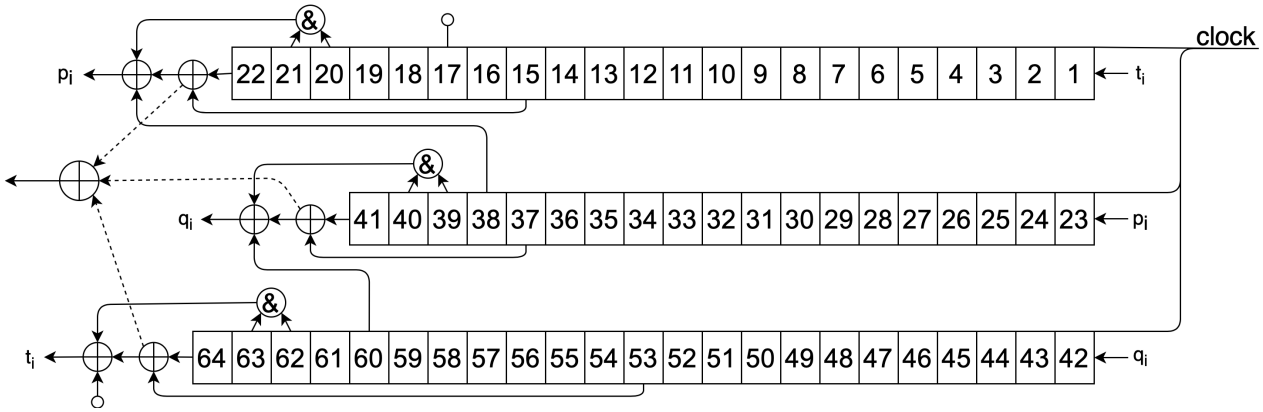

Рисунок А.2 – Структура алгоритма Trivium 64

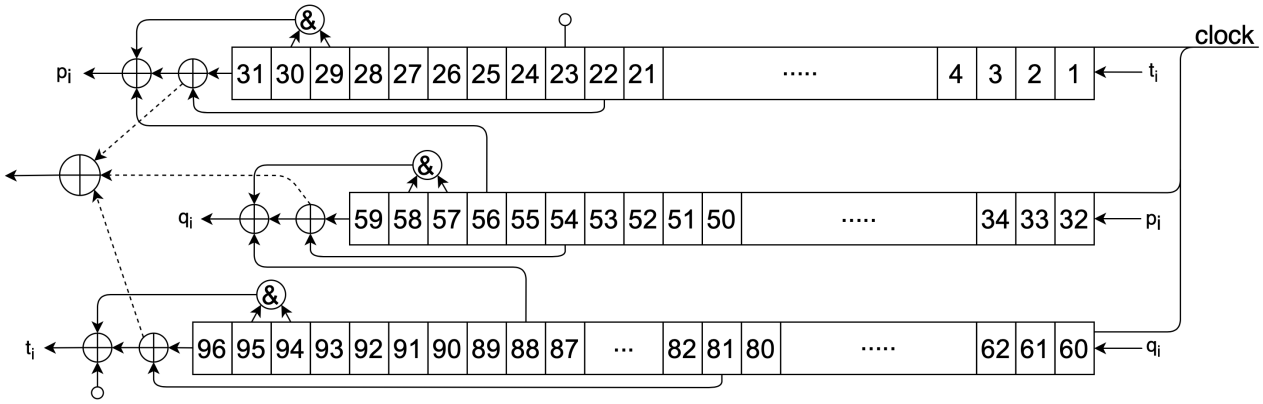

Рисунок А.3 – Структура алгоритма Trivium 96 (Trivium-toy)

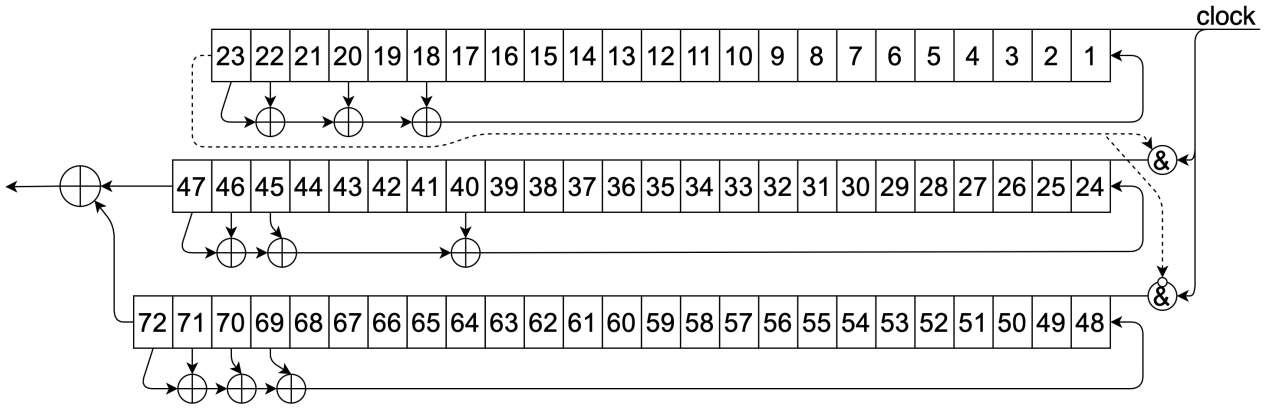

Рисунок А.4 – Структура алгоритма ASG 72

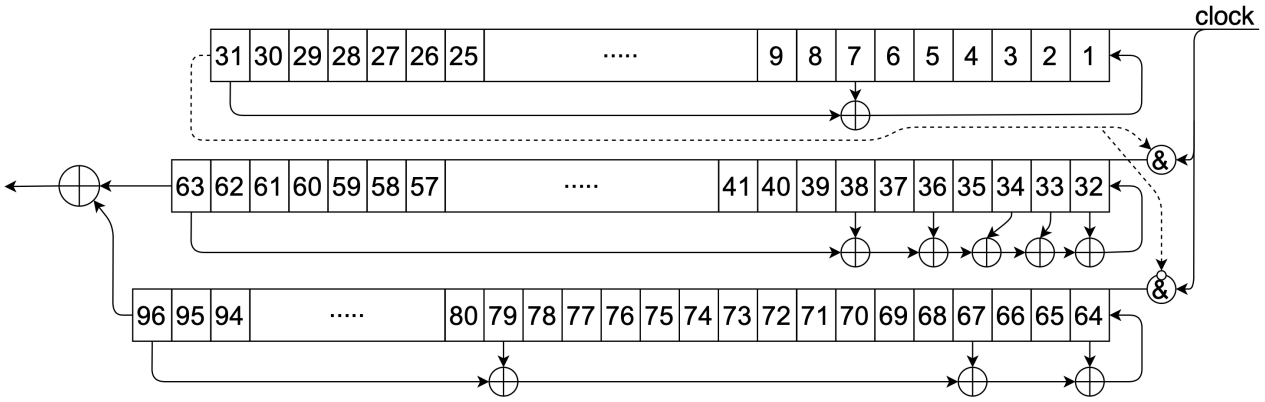

Рисунок А.5 – Структура алгоритма ASG 96

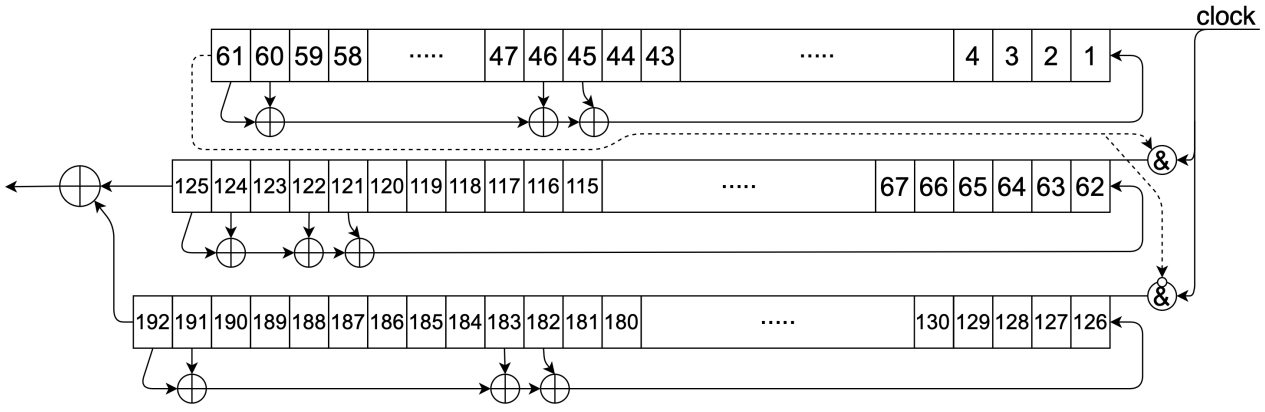

Рисунок А.6 – Структура алгоритма ASG 192

ПРИЛОЖЕНИЕ Б. ПРИМЕРЫ ПРОЦЕССА ОПТИМИЗАЦИИ

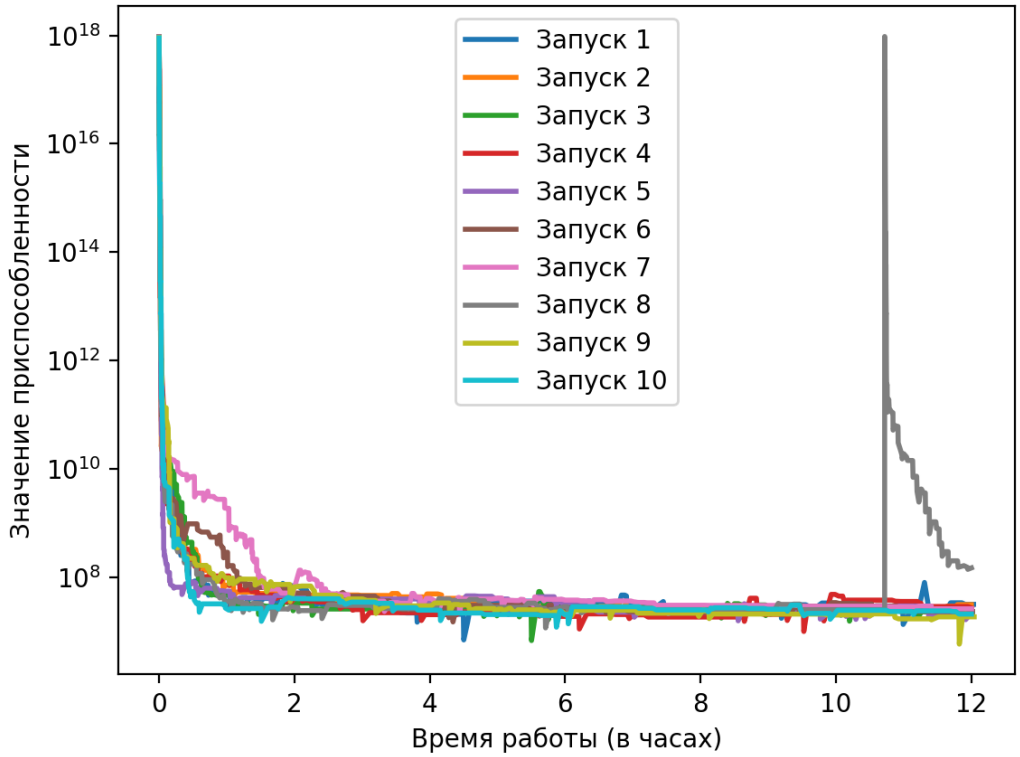

Рисунок Б.1 – Пример процесса оптимизации для алгоритма Trivium 64

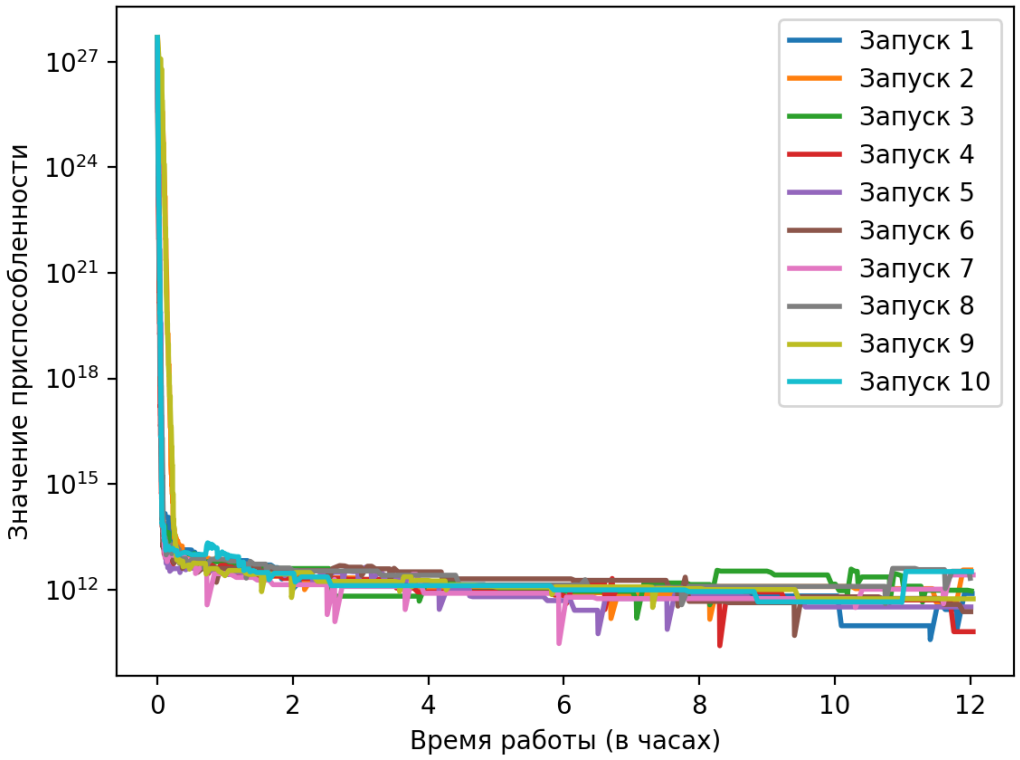

Рисунок Б.2 – Пример процесса оптимизации для алгоритма Trivium 96

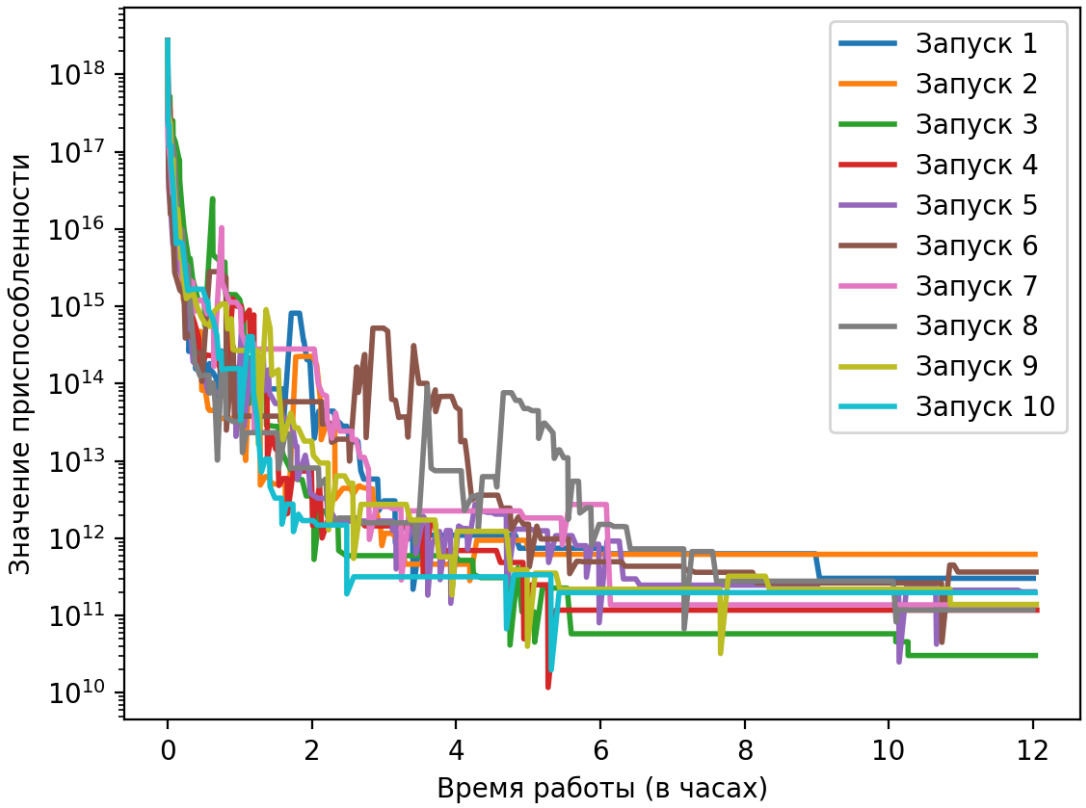

Рисунок Б.3 – Пример процесса оптимизации для алгоритма A5/1

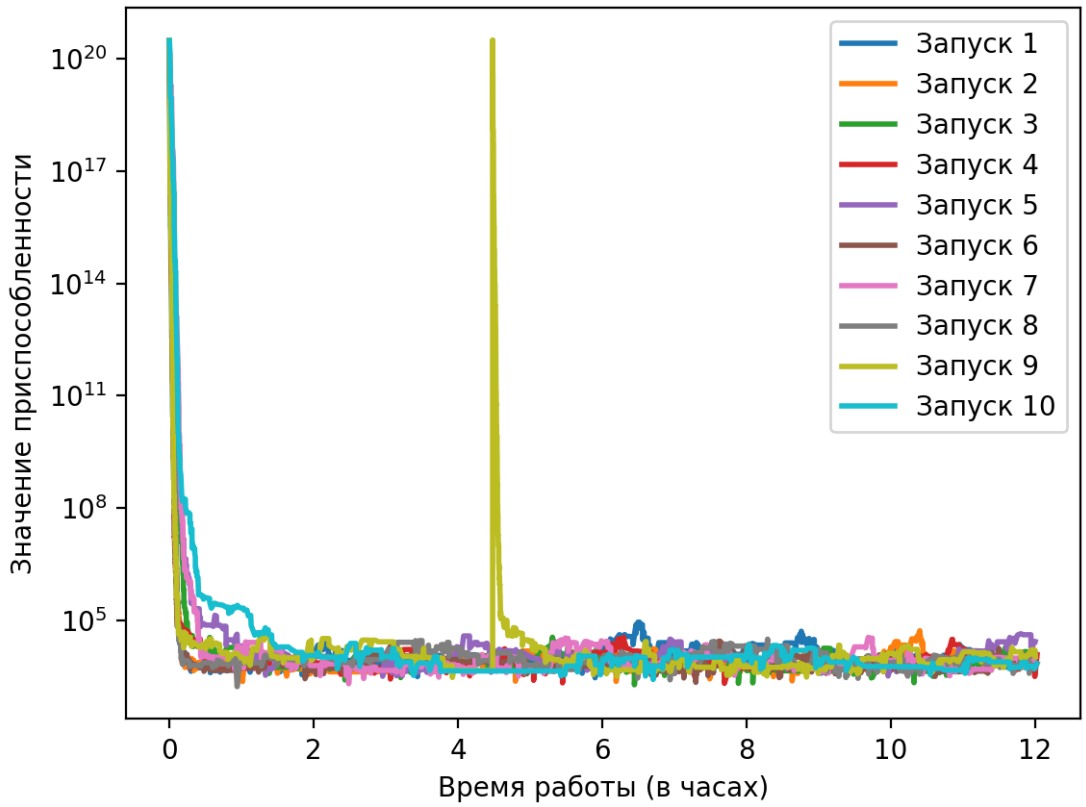

Рисунок Б.4 – Пример процесса оптимизации для алгоритма ASG 72

ПРИЛОЖЕНИЕ В. СПИСОК ПОСТРОЕННЫХ ДЕКОМПОЗИЦИОННЫХ МНОЖЕСТВ

Таблица В.1 – Лучшие декомпозиционные множества построенные генетическим алгоритмом

Алгоритм	Декомпозиционное множество (B)
A5/1	1 3..5 7 9 10 13 16 18..27 30 31 34 37 41 43 44 47 48 50..52 56 60 61 64
Bivium	15 16 18 33 42 43 56 61 70 72 74 85 96 97 100 108 109 115 124 126 127 130 135 138 141 144 145 154
Trivium 64	2 4 6 9 11 16 20 23..28 30 32 36 38 46..48 51
Trivium 96	6 9 11 15 18..21 27 29 32..34 38 39 41 42 45 47 49 53 54 61 63 69..71 75 77 78 84 90..93

Таблица В.2 – Лучшие декомпозиционные множества построенные эволюционной стратегией (1+1)

Алгоритм	Декомпозиционное множество (B)
A5/1	3 4 5 8 9 12 16..19 21..30 34 37 38 43 51 52
Bivium	30 31 33 34 48 57 58 59 61 62 63 71 73 101 104 122 124 125 126 150 152 154 158 161 166 167 171
Trivium	4 5 6 10 11 12 17 20 25 27 29 30 31 34..37 39 41..44 47 49 50 52 55 56 59 62 65 70 72 73 74 77..83 85 86 90 101 102 108 113 116 117 118 121 122 128 130 134 136 137 139 142 146 150 151 156 159..162 165..169 178 179 181 183 184 188 189 193 196 199 204 206..214 217 218 220 221 222 225 226 228 230 233 234 236 238 239 242 245 247 248 251 255 256 264 266 268 269 272..278 280 281 285 286
Trivium 64	1 2 4 9 23..29 33 34 44 47 48 50
Trivium 96	1 4 11 13 15 16 18 22 23 26 42..47 49 51 63 64 66 70 72 73 79 80 87 95
ASG 72	1 3 6 8 11 13 15 16 17 46
ASG 96	2 3 5 7 10 13 15 17 19 20 22 24..27 29 30 31 95
ASG 192	1 8..14 19 20 27 28 32 33 35 39 41 42 45 48 54 55 58 94 99 103 104 107 108 115 117 121 157 159 161 167 176 178 179 184 187 188
Grain v0	5 7 8 10 17 21 27 28 39 47 53 60 63 67 71 75 76 80..85 87 89 90 94..99 102 103 105 106 107 113..116 120 121 122 124 125 126 128 129 131 132 134 135 138 140 141 143 148 155 156 157
Grain v1	4..7 9..13 15..19 21..25 27 28 29 32 34 36 41 43..46 49 51 54 55 58 59 62 63 64 66 67 69 70 72 73 76 78 79 83 85 87 88 90 91 93 94 97 98 99 101 102 108 110 112 115 117 118 119 121..124 126 128 129 131 133 137 139 145 146 148 150 151 152 158 159
Mickey	1..5 8..12 14..31 35 36 39..45 47..61 63 65..70 72..78 81 83..87 91 94 97 98 103 106..111 114..117 119 121 123 124 126..131 133..142 144 145 148..159 161..169 171..179 182 184 186..192 195 197..200