Министерство образования и науки Украины
Харьковский национальный университет радиоэлектроники

На правах рукописи

КАЗИМИРОВ АЛЕКСАНДР ВЛАДИМИРОВИЧ

УДК 681.3.06:519.248.681

МЕТОДЫ И СРЕДСТВА ГЕНЕРАЦИИ НЕЛИНЕЙНЫХ УЗЛОВ ЗАМЕНЫ ДЛЯ СИММЕТРИЧНЫХ КРИПТОАЛГОРИТМОВ

05.13.21 – системы защиты информации

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель
Олейников Роман Васильевич
к.т.н., доцент

Цей примірник дисертації є ідентичним за змістом з усіма іншими, що надішли до спеціалізованої вченої ради

Вчений секретар спеціалізованої вченої ради К 64.052.05 **І.В. Лисицька**

Харьков 2013

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ, ЕДИНИЦ, СОКРАЩЕНИЙ И ТЕРМИНОВ

AES – advanced encryption standard.

AI – алгебраический иммунитет.

CI – корреляционный иммунитет.

CRYPTREC – cryptography research and evaluation committees.

DES – data encryption standard.

FIPS – federal information processing standards.

GF – поле Галуа.

IBM – international business machines.

MAC – код аутентификации сообщения.

MPI – интерфейс передачи сообщений.

NESSIE – new European schemes for signatures, integrity, and encryption.

NIST – the national institute of standards and technology.

NL – нелинейность.

PC – критерий распространения.

SAC – строгий лавинный критерий.

Sage – system for algebra and geometry experimentation.

SAT – задача выполнимости булевых формул.

SHA – secure hash algorithm.

SPN – подстановочно-перестановочная сеть.

SSI – sum-of-squares indicator.

XOR – побитовая операция сложения по модулю 2 (exclusive OR).

БСШ – блочный симметричный шифр.

ГОСТ – государственный стандарт.

ДКЭ – долговременный ключевой элемент.

ДСТУ – державний стандатр України.

КШЗ – Карле-Шарпен-Зиновьев.

ЛЛР – линейный рекуррентный регистр.

НУЗ – нелинейный узел замены.

ОС – операционная система.

СН – совершенно нелинейный.

СРК – схема разворачивания ключа.

ПК – персональный компьютер.

ПБ – почти бент.

ПСН – почти совершенно нелинейный.

ПШ – поточный шифр.

ПО – программное обеспечение.

РА – расширенно аффинная.

ЧРА – частично расширенно аффинная.

ЭВМ – электронно-вычислительная машина.

$\oplus$ – побитовая операция сложения по модулю 2 (XOR).

$\mathcal{F}_2^n$ – векторное пространство размерности n, состоящее из элементов $\{0,1\}$.

$\mathcal{F}_q$ – конечное поле, состоящее из q элементов.

(n,m)-функция – векторная булева функция из $\mathcal{F}_2^n$ в $\mathcal{F}_2^m$.

ВВЕДЕНИЕ

Актуальность темы. Одним из главных стратегических приоритетов государственной политики Украины в информационной сфере является принятие комплексных мер по защите национального информационного пространства [1]. Главной особенностью этого направления является увеличение производительности и повышение уровня безопасности передачи сообщений в информационно-телекоммуникационных системах и сетях. Быстрый и безопасный доступ к информационным и вычислительным ресурсам, большая часть которых входит в состав глобальной сети Интернет, на сегодняшний день может рассматриваться как один из важнейших показателей культурного и экономического развития государства.

Примером заинтересованности государства в развитии национального информационного пространства и защиты его информационного суверенитета является указ Президента Украины № 514 от 2009 г. «Про Доктрину інформаційної безпеки України» [2]. Этим Указом определяются:

– жизненно важные интересы в информационной сфере;

– реальные и потенциальные угрозы информационной безопасности;

– направления развития государственной политики в сфере информационной безопасности Украины.

Основными направлениями, на которые сфокусирована доктрина, являются:

– информационно-психологическое, в частности по обеспечению конституционных прав и свобод человека и гражданина;

– технологическое развитие, в частности инновационное обновление национальных информационных ресурсов, внедрение новейших технологий создания, обработки и распространения информации;

– защита информации, в частности для обеспечения конфиденциальности, целостности и доступности информации, в том числе технической защиты информации в национальных информационных ресурсах от кибернетических атак [2].

Очевидно, что информационные технологии являются неотъемлемой частью нашей повседневной жизни. Эффективность функционирования и применения информационных систем напрямую зависит от их безопасности и во многом определяется совершенством используемых методов защиты информации. На сегодняшний день существует множество областей, где непредсказуемая или нештатная работа информационно-телекоммуникационной системы может привести к серьёзным последствиям. К ним относятся системы управления энергообеспечением, особенно в нефтяной и газовой промышленности, движением всех видов транспорта, телекоммуникациями и др. За последние несколько десятков лет заметно увеличилось количество публикаций и работ, связанных с различными аспектами информационной безопасности. Это свидетельствует о значительном и непрерывно растущем интересе к данной проблеме, который приобретает глобальный характер.

Одновременно с появлением новых проблем совершенствуются и методы их решения [3-5]. Так криптографические методы защиты информации, использование и разработка которых до недавнего времени была доступна лишь специальным государственным службам, сегодня применяются в повседневной жизни на этапах создания, передачи, приёма, обработки, хранения и уничтожения информации [6-11].

Одну из важных ролей в комплексе средств защиты информации, особенно при необходимости обеспечения высокой скорости обработки информации, в Украине продолжают играть блочные симметричные шифры (БСШ) [3, 7, 12-14]. Они чаще других используются для защиты передаваемых данных и получили широкое распространение благодаря

высокой эффективности и низкой сложности реализации [15-17]. Кроме обеспечения конфиденциальности БСШ используются как компоненты в кодах аутентификации сообщений (MAC), хэш-функциях и электронных цифровых подписях для проверки целостности данных, а также при генерации псевдослучайных последовательностей и в составе протоколов подтверждения подлинности [7, 18, 19].

В 2010 году в Украине завершился открытый конкурс алгоритма-прототипа блочного симметричного шифрования [12]. В качестве «эталона» для многих разработчиков был шифр Rijndael [15], который своей победой на конкурсе Advanced Encryption Standard (AES) [20] во многом обязан убедительному обоснованию авторами производительности и высоких показателей стойкости шифра к известным на то время атакам. Многие претенденты на украинском конкурсе были представлены в виде усовершенствованных версий Rijndael. Основной целью было повышение их устойчивости к потенциальным атакам, которые, как предполагается, могут использовать простоту его алгебраического описания и слабости относительно простой схемы разворачивания ключа (СРК) [20-25]. После длительного и детального процесса изучения и исследования представленных решений был отмечен шифр «Калина» [27, 29].

Однако, до адаптации в качестве нового стандарта потребуется ещё некоторый период времени, а пока остаётся ориентироваться на использование действующего стандарта ДСТУ ГОСТ 28147:2009 [7]. В связи с чем, остаются актуальными вопросы по повышению надёжности и защищённости телекоммуникационных систем и сетей передачи данных, при использовании данного стандарта.

Анализ решений, использованных при построении шифра «Калина», позволяет сделать вывод, что он во многом унаследовал идеи победителя конкурса AES [28]. В соответствии с известными принципами Шеннона [38], алгоритмы используют нелинейные операции для перемешивания и

линейные преобразования для рассеивания. Последовательное многократное применение перемешивания и рассеивания позволяет добиться высокого уровня криптографической стойкости. Как и в Rijndael, линейное преобразование основано на матричном преобразовании в поле $GF(2^8)$. Матрица выбиралась таким образом, чтобы код, сгенерированный при помощи этой матрицы, обладал максимально достижимым кодовым расстоянием [29-31].

Одной из отличительных особенностей отмеченного алгоритма шифрования в украинском конкурсе является нелинейный слой замены [13, 26, 29]. В отличие от БСШ Rijndael, где подстановка (S-блок) генерировалась на основе конструкции, приведённой ещё в работах К. Ниберг и Т. Динга [30, 32, 33], т.е. преобразования, использующего вычисление обратного элемента в поле $GF(2^8)$ с последующим аффинным усложнением, «Калина» имеет несколько случайно сгенерированных подстановок. Их основным преимуществом является описание при помощи системы алгебраических уравнений степени 3, в отличие от 2-й – для AES [34-37].

Подстановки для современных симметричных примитивов, в том числе и функций разворачивания ключа, как правило, реализуются в виде таблиц замены. Учитывая, что в большинстве современных БСШ (Rijndael [39], Camellia [40], ARIA [41] и др.) для введения цикловых ключей в алгоритм шифрования используется линейная операция (побитовое сложение по модулю 2), S-блоки оказываются единственными элементами, определяющими нелинейность шифрующего преобразования и уровень его стойкости к криптоаналитическим атакам [43-46]. Необходимое число циклов блочных симметричных шифров вычисляется на основе обеспечения стойкости к известным видам криптографического анализа при условии заданных свойств S-блоков.

Как правило, подстановки отображают n-битый входной блок в выходной длиной m бит. Представление подстановок варьируется в зависимости от алгоритма шифрования. В поточных шифрах (ПШ) узлы нелинейной замены представлены обычно в виде векторных булевых функций [47-50]. Перестановки являются подклассом S-блоков и широко используются в блочных симметричных шифрах в виде нелинейных узлов замены (НУЗ). Подстановка может быть достаточно просто преобразована из одной формы в другую [48, 49].

Для защиты криптографического примитива от различных типов атак S-блоки должны удовлетворять ряду критериев [15, 29, 51-57]. Из-за их большого количества, противоречивости или частичной взаимозависимости, проблематично сформировать подстановку, соответствующую всем известным требованиям. В связи с чем на практике используются НУЗ, удовлетворяющие основным критериям, существенным для конкретного симметричного алгоритма [54]. Такие S-блоки принято называть оптимальными [58]. Критерии оптимальности могут быть как заданы для класса криптопримитивов, так и для каждого по отдельности.

Большинство теоретических методов генерации векторных булевых функций обладают предельными показателями (например, δ-равномерности или нелинейности), при этом не обладают другими необходимыми свойствами (например, высоким показателем алгебраического иммунитета), необходимыми для симметричных криптопримитивов [57]. Поэтому задача генерации оптимальных подстановок с оптимальными показателями является довольно трудоёмкой, особенно для больших значений n и m. В тоже время, она может быть частично решена при помощи классов эквивалентностей векторных булевых функций: расширенно аффинных (РА) и Карле-Шарпен-Зиновьев (КШЗ) [48, 50].

Таким образом, актуальность темы диссертационной работы определяется необходимостью:

– анализа существующих критериев и обоснованному выбору лишь тех из них, которые действительно необходимы для конкретных криптопримитивов;

– разработки теоретически обоснованных рекомендаций для практической генерации S-блоков;

– поиска эффективных методов нахождения нелинейных узлов замены, обеспечивающих высокие показатели стойкости в симметричных криптопримитивах.

Связь работы с научными программами, планами. Диссертационные исследования проводились в рамках научно-исследовательских работ «Розробка перспективних методів та засобів криптографічного захисту інформації в державних відомствах України» (№ДР0102U003739), «Дослідження та розробка перспективних криптографічних систем та протоколів захисту інформації у телекомунікаційних системах та мережах України» (№ ДР 0103U001981).

Цель и задачи исследования. В работе представлены методы формирования (генерации) узлов нелинейной замены для применения в симметричных криптопримитивах, таких как ГОСТ 28147-89 [6], СТБ 34.101.31-2011 [59], ГОСТ Р 34.11-2012 [60], «Калина» [13] и ряде других алгоритмов шифрования, строящихся на основе многоцикловых слоёв подстановок и перестановок [61].

В недавно опубликованных работах [55, 62] особое внимание уделяется исследованию возможности применения случайных таблиц подстановок в симметричных криптоалгоритмах. Такие S-блоки основаны на случайных методах генерации подстановок и последующей проверке их криптографических свойств вместо использования векторных булевых функций с заданной структурой.

В виду того, что блочные симметричные шифры, в частности AES и ДСТУ ГОСТ 28147:2009, сегодня широко используются в области защиты

информации, данная работа направлена на развитие и совершенствование методов генерации подстановок для повышения их криптографических свойств, а также для использования в других средствах симметричной криптографии.

Целью настоящей работы является повышение уровня стойкости современных итеративных криптографических примитивов к дифференциальному, линейному и алгебраическому криптоанализам за счёт разработки методов генерации нелинейных узлов замены.

Для достижения поставленной цели необходимо решить следующие задачи.

1. Провести анализ методов формирования нелинейных отображений в симметричной криптографии.

2. Разработать метод представления линейных отображений, заданных над полем $GF(2^n)$, в матричный вид с целью уменьшения сложности проверки на эквивалентность нелинейных отображений.

3. Усовершенствовать метод оценки стойкости блочных симметричных шифров относительно алгебраической атаки на основе решения системы нелинейных уравнений над полем $\mathcal{F}_2$.

4. Разработать метод формирования долговременных ключевых элементов (ДКЭ) для шифра ДСТУ ГОСТ 28147:2009, подстановки которых принадлежат различным классам расширенно аффинной (РА) эквивалентности и обладают максимальными показателями стойкости к дифференциальному и линейному криптоанализам.

5. Разработать эффективный метод генерации нелинейных узлов замены для перспективных блочных симметричных шифров с учётом алгебраической атаки.

Итогом работы должны стать комплексы программного обеспечения (программные модели), позволяющее осуществлять генерацию таблиц подстановок основываясь на методах построения узлов нелинейной замены с

целью использования в перспективных методах криптографической защиты информации в Украине и за рубежом.

Научная новизна полученных результатов. В результате выполненных в работе научных исследований предложены теоретико-практические методы генерации нелинейных узлов замены, используемые в качестве основных элементов при построении многих современных криптографических алгоритмов. Данные методы направлены на практическое решение задач в области информационной безопасности.

Используя теорию векторных булевых функций для описания свойств S-блоков были получены следующие научные результаты.

1. Впервые предложен метод генерации узлов нелинейной замены для перспективных блочных симметричных шифров с одновременным учётом δ-равномерности, нелинейности и алгебраических показателей на основе векторных булевых функций, что позволяет находить подстановки с улучшенными показателями алгебраического иммунитета и нелинейности при малых затратах ресурсов.

2. Впервые предложен метод формирования долговременных ключевых элементов на основе классов эквивалентностей векторных булевых функций, что позволяет генерировать узлы нелинейной замены, которые принадлежат различным классам РА-эквивалентности и имеют максимальные показатели защиты от дифференциального и линейного криптоанализов.

3. Усовершенствован метод нахождения матрицы линейного отображения, заданного в виде полинома над полем $GF(2^n)$, который, в отличие от известных, для решения системы матричных уравнений использует набор входных векторов бинарного вида с единичным весом Хемминга, использование которого позволяет уменьшить сложность нахождения алгебраической формы высокоуровневых конструкций криптографических алгоритмов и проверки векторных булевых функций на

частично расширенную аффинную (ЧРА) эквивалентность.

4. Получил дальнейшее развитие метод оценки стойкости блочных симметричных шифров к алгебраической атаке, который отличается от известных учётом показателей количества уравнений в системе и её разреженностью, что позволяет уточнить значение верхней границы скложности атаки.

5. Получил дальнейшее развитие метод отбора подстановок для блочных симметричных шифров, который основан на критериальном подходе, в частности с учётом пердложеного алгебраического критерия, и отличается от известных комплексной оценкой стойкости, что позволяет генерировать S-блоки, использование которых в симметричных алгоритмах шифрования увеличивает сложность криптоаналитических атак.

Практическая значимость полученных результатов. Практическая значимость полученных результатов заключается в следующем.

1. Разработан алгоритм нахождения 8-битовых подстановок и его программная реализация на основе предложенного метода генерации нелинейных узлов замены для перспективных блочных симметричных шифров, который позволяет находить подстановки с отсутствием фиксированных точек, нелинейностью 104, минимальной степенью 7, алгебраическим иммунитетом 3 и δ-равномерностью 8 на однопроцессорном компьютере со средним временем работы 3,5 часа.

2. Разработан алгоритм генерации ДКЭ для шифра ДСТУ ГОСТ 29147:2009 и его програмная реализация на основе предложенного метода формирования долговременных ключевых элементов, который позволяет определять подстановки с различных классов РА-эквивалентности с показателями δ-равномерность 4 и нелинейность 4.

3. Разработанные программные средства вычисления показателей минимальной степени, нелинейности, корреляционного иммунитета, δ-равномерности, циклической структуры, алгебраического иммунитета,

абсолютного индикатора, критерия распространения, глобальной характеристики «сумма квадратов», а также программные средства проверки на сбалансированность и отсутствие фиксированных точек произвольных векторных булевых функций, позволили сформировать требования к узлам нелинейной замены.

4. Разработаны комплексы программного обеспечения оценки верхней границы вероятности нахождения подстановок с заданными показателями стойкости к дифференциальному, линейному и алгебраическому криптоанализам на основе моделирования метода случайной генерации подстановок в распределённых кластерных системах.

5. Разработаны практические рекомендации относительно генерации узлов нелинейной замены, которые позволяют сократить время проектирования перспективных симметричных криптопримитивов.

Основные результаты данной работы внедрены в процесс научных и экспериментальных исследований «Института информационных технологий» («ИИТ»), в части создания программного обеспечения и моделирования методов генерации нелинейных узлов замены для существующих (ДСТУ ГОСТ 28147:2009) и перспективных блочных симметричных шифров. Имеются акты об использовании результатов работы в научных разработках ЗАО «ИИТ» и в учебном процессе Харьковского национального университета радиоэлектроники (ХНУРЭ) (см. приложение А).

Апробация результатов диссертации. Результаты исследований, проведённых в работе, докладывались на: 4 международных форумах, включая 3-й Международный радиоэлектронный форум «Прикладная радиоэлектроника. состояние и перспективы развития», 14-й Международный молодёжный форум «Радиоэлектроника и молодёжь в XXI веке» и др. [63-66]; 9 научно-практических конференциях, среди которых XII Международная научно-практическая конференция «Безопасность

информации в информационно-телекоммуникационных системах», Научно-техническая конференция с международным участием «Компьютерное моделирование в наукоёмких технологиях», Международная научно-практическая конференция «Перспективы развития информационных и транспортных технологий в налоговой сфере, внешнеэкономической деятельности и управлении организациями» и др. [67-76]; 3 зарубежных международных конференциях, к которым относятся WAIFI'12, RusCrypto13 и CTCrypt 2013 [77-81]; 3 исследовательских школах – Winter School in Information Security 2012, ECRYPT II Summer School on Tools и IceBreak 2013 [82-84].

Личный вклад соискателя. В работе [34] автором изложены основные идеи алгебраической атаки на блочные симметричные шифры и проведён анализ алгоритма шифрования «Лабиринт». Алгебраический криптоанализ схемы разворачивания ключа шифра «Калина» выполнен в [37]. При подготовке совместной публикации [55] автор участвовал в разработке метода нахождения дифференциальных характеристик для различных мини-версий БСШ, основанных на подстановочно-перестановочной сети (SPN). В работе [51] автором проведён анализ критериев узлов нелинейной замены с точки зрения блочных симметричных шифров. Основные критерии для S-блоков с использованием математического аппарата булевых функций приведены в [57]. В статье [54] рассмотрены криптографические критерии векторных булевых функций, а в [85] – предложен критерий для множества подстановок. Разработанная методика практической оценки мощности множества состояний поточного шифра «Mickey-80» и «Mickey-128», которую можно использовать при анализе циклических свойств S-блоков, изложена в [86] и [87] соответственно. Теоретическое обоснование работ [86] и [87] опубликовано в [88]. В работе [50] предложен метод проверки векторных булевых функций на эквивалентность, а в работе [22] проведена сравнительная характеристика

производительности схем разворачивания ключа современных блочных симметричных шифров. Анализ известных методов построения узлов нелинейной замены изложен в [54]. В работе [58] предложен метод генерации оптимальных подстановок.

Публикация результатов работы. По результат выполненных исследований опубликовано 31 печатное издание. К ним относятся 10 статей, напечатанных в научных специализированных изданиях Украины, и 3 статьи в зарубежных изданиях, которые входят в научно-метрические базы, а также 18 материалов научных конференций.

Структура и объем диссертации. Диссертация состоит из 5 разделов, 2 приложений и изложена на 190 страницах. Список использованных источников содержит 222 наименования.

1 СОВРЕМЕННЫЕ ТЕНДЕНЦИИ В РАЗВИТИИ ТЕОРИИ СИММЕТРИЧНЫХ КРИПТОПРИМИТИВОВ

1.1 Обзор конкурсов по созданию алгоритмов симметричного шифрования

В конце двадцатого века Эли Бихаму и Ади Шамиру удалось теоретически взломать шифр Data Encryption Standard (DES) [44, 89]. Немного позже появились и практические реализации, позволяющие находить ключ шифрования за приемлемое время [90]. В следствии чего в Соединённых Штатах Америки (США) в 1997 был организован конкурс Advanced Encryption Standard (AES), направленный на выбор стандарта блочного симметричного шифрования нового поколения [28]. После нескольких лет исследований в США был введён в действие новый стандарт шифрования – FIPS-197 [20], основанный на алгоритме Rijndael. Данный алгоритм шифрования занял первое место благодаря высокому уровню стойкости, простому описанию и высокой производительности на большинстве платформ того времени.

В ноябре 2000 года начался европейский открытый конкурс New European Schemes for Signatures, Integrity and Encryption (NESSIE) [91]. Основной задачей проекта NESSIE являлся отбор лучших криптографических примитивов среди поданных претендентов со всего мира. В качестве основных критериев отбора претендентов были выбраны безопасность, производительность и гибкость. После окончания конкурса в список рекомендованных для промышленного использования вошли алгоритмы блочных симметричных шифров, хэш-функции, коды аутентификации сообщения (MAC) и алгоритмы цифровой подписи.

Параллельно с NESSIE аналогичный конкурс (CRYPTREC) провело и

японское правительство [92]. В результате нескольких лет исследований криптопримитивов были отобраны лучшие алгоритмы и рекомендованы к использованию в государственном и промышленном секторах.

В Украине рекомендован к использованию блочный симметричный шифр ДСТУ ГОСТ 28147:2009 [6, 7], который был принят в 1989 году и уже уступает по производительности многим современным шифрам, в том числе и AES [20]. В последние несколько лет были успешно проведены теоретические атаки на криптоалгоритм, что позволило уменьшить сложность нахождения ключа с 2^{256} до 2^{192} [94, 95]. Однако, сложность в 2^{192} недостижима для современных компьютеров, что говорит о практической защищённости шифра [95].

С целью поиска альтернативны ГОСТ 28147-89 в Украине был объявлен открытый конкурс на разработку алгоритма-прототипа блочного симметричного шифрования [12]. Высокий уровень стойкости относительно известных видов криптоаналитических атак являлся одним из основных требований к перспективному шифру. При этом, нужно было достичь уровня производительности не меньше, чем у предыдущего стандарта. В 2009 году по результатам конкурса был отмечен алгоритм шифрования «Калина».

На ряду с другими криптографическими алгоритмами на конкурс NESSIE были поданы шесть поточных шифров [91]. Все шесть претендентов были теоретически взломаны. Это привело к тому, что в ноябре 2004 года был объявлен отдельный проект eSTREAM, основная задача которого заключалась в выборе одного или нескольких ПШ для использования в корпоративном секторе [96]. Необходимо отдельно отметить, что поточные шифры были разделены на две категории ориентированных на программное (категория 1) и аппаратное (категория 2) применение [97]. После четырёх лет исследований были отобраны 4 шифра в каждой из категорий. Однако, в 2008 году шифр F-FCSR-H v2 был исключён из списка из-за найденных в нём уязвимостей [98]. В таблице 1.1 приведены поточные шифры относящиеся к

различным категориям и рекомендованные к использованию на сегодняшний день [98].

Таблица 1.1 – Поточные шифры, рекомендованные к использованию по результатам eSTREAM

Категория 1	**Категория 2**
HC-128	Grain v1
Rabbit	MICKEY 2.0
Salsa20/12	Trivium
SOSEMANUK	

В ноябре 2007 года Национальным институтом стандартов и технологий (NIST) был открыт конкурс на разработку хэш-функции SHA-3 [99], которая бы дополнила существующие две версии [100]. По аналогии с AES, NIST объединил усилия разработчиков и криптоаналитиков со всего мира с целью выбора одного или нескольких дополнительных алгоритмов хэширования. В октябре 2012 года было объявлено, что, по результатам конкурса, алгоритм Keccak будет лежать в основе новой хэш-функции SHA-3 [101].

В отличии от США, Россия не объявляла открытого конкурса, а использовала в качестве прототипа хэш-функцию «Стрибог» [80, 102, 103]. Данный алгоритм является единственным известным вариантом проекта государственного стандарта криптографической хэш-функции. С 1 января 2013 стандарт ГОСТ Р 34.11-2012 [60, 104], описывающий алгоритм хэширования, вступил в силу, заменив более раннюю версию ГОСТ Р 34.11-94 [105].

1.2 Обобщённые модели криптоалгоритмов

В данном подразделе рассматриваются обобщённые модели симметричных криптоалгоритмов. В частности, описываются структуры блочного симметричного шифра, поточного шифра и хэш-функции.

1.2.1 Алгебраическая модель симметричного шифра

Пусть X, K, Y – некоторые конечные множества, которые названы множеством открытых текстов, множеством ключей и множеством зашифрованных сообщений соответственно. На прямом произведении $X \times K$ множеств X и K задана функция

$$f : X \times K \mapsto Y \Leftrightarrow f(x,k) = y, \Leftrightarrow f_k(x) = y \quad x \in X, k \in K, y \in Y .$$

Тройка множеств X, K, Y с функцией отображения $f\left(A = \left(X, K, Y, f\right)\right)$ называется алгебраической структурой шифра, если выполнены два условия [31]:

а) функция f сюръективна (каждый элемент множества X является образом хотя бы одного элемента множества Y);

б) для любого $k \in K$ функция f_k инъективна (образы двух различных элементов различны) [31].

Запись $f(x,k) = y$ называется уравнением шифрования. Подразумевается, что открытое сообщение x зашифровывается на ключе k, в результате чего получается зашифрованный текст y. Уравнением расшифрования называют запись вида:

$$f^{-1}(y,k) = x \quad \Leftrightarrow \quad f_k^{-1}\left(y\right) = x,$$

подразумевая, что зашифрованный текст $y = f(x,k)$ расшифровывается на ключе k и получается исходное открытое сообщение x [31].

Произведением шифров $A_1 = (X_1, K_1, Y_1, f_1)$ и $A_2 = (X_2, K_2, Y_2, f_2)$, где $Y_1 \subseteq X_2$ – называют шифр $A = (X_1, K_1 \times K_2, Y_2, f)$, для которого

$$f(x,(k_1,k_2)) = f(f(x,k_1),k_2), \quad (k_1,k_2) \in K_1 \times K_2 .$$

Данная модель шифра отражает лишь некоторые функциональные свойства шифрования и расшифрования симметричных шифров, у которых ключ расшифрования равен или легко вычисляется из ключа зашифрования [3].

Важным параметром любого шифра является ключ, обеспечивающий выбор одного алгоритма криптографического преобразования из множества возможных. В современной криптографии предполагается, что вся секретность криптографического алгоритма должна базироваться на ключе, а не на алгоритме шифрования (принцип Кирхгофа) [3, 31, 106, 107].

1.2.2 Блочный симметричный шифр

Пусть функция $E: \{0,1\}^l \times \{0,1\}^k \mapsto \{0,1\}^l$ принимает ключ K длиной k бит и входное сообщение (открытый текст) M длиной l бит и возвращает сообщение (зашифрованный текст) $E(M,K)$ [85]. Для каждого ключа K определим функцию $E_K: \{0,1\}^l \times \{0,1\}^l$ как $E_K(M) = E(M,K)$. Тогда E – блочный симметричный шифр, при условии что E_K является перестановкой для любого значения ключа K и функции E_K, E_K^{-1} вычисляются эффективно [85].

В основе большинства современных БСШ лежит итерационная процедура [108]. На рисунке 1 она изображена в виде цикловой (раундовой) функции.

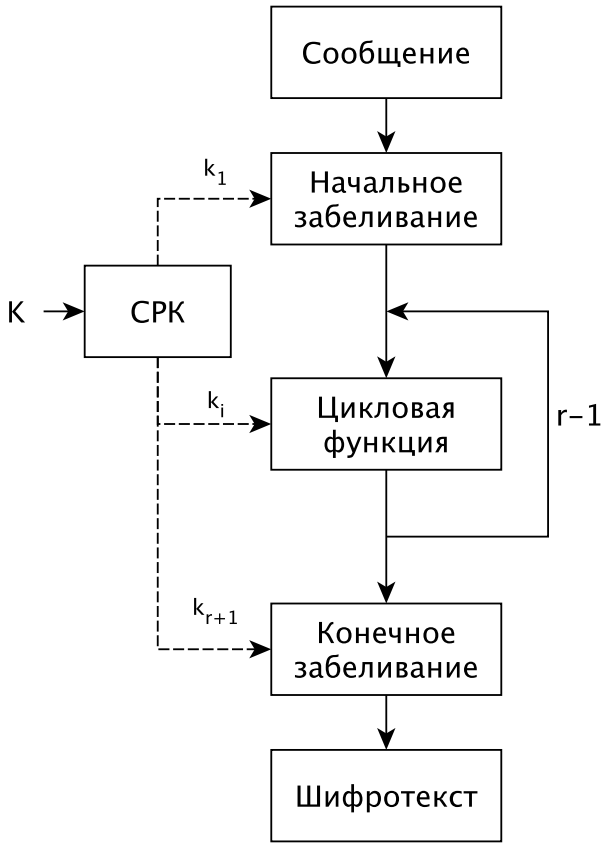

Рисунок 1.1 – Обобщённая структура итеративного БСШ

В общем виде итерационный блочный симметричный шифр математически представляется следующим образом:

$$E_K(M) = PW_{k_{r+1}} \circ \prod_{i=2}^{r} R_{k_i} \circ IW_{k_1}(M),$$

где R – цикловая функция, IW и PW – функции начального и конечного забеливания сообщения соответственно [85]. На рисунке 1.1 схема разворачивания ключа (СРК) представляет собой алгоритм, который генерирует подключи ($k_1, k_2, \ldots, k_{r+1}$) для всех этапов алгоритма шифрования на основе входного мастер-ключа (K) [22, 109].

Процедурой смешивания ключа блочного симметричного шифра называется алгоритм, который описывает шаги введения циклового ключа в алгоритм шифрования [85]. В большинстве современных блочных симметричных шифров в качестве функции смешивания ключа используется

операция исключающее ИЛИ (XOR), что обусловлено простотой её реализации.

Современные БСШ должны удовлетворять следующим критериям [3, 4, 12, 28]:

а) сложность выполнения операции шифрования и расшифрования должна быть соизмерима с действующими стандартами;

б) отсутствие статистических зависимостей между шифротектстом и открытым сообщением;

в) быть защищённым от всех известных на сегодняшний день атак.

На сегодняшний день существует два базовых преобразования в БСШ: блок подстановок (S-блок) и блок перестановок (P-блок) [3, 4, 107, 108]. Можно показать, что любое двоичное преобразование над двоичным блоком фиксированной длины, сводится к S-блоку, но на практике, в силу сложности строения n-разрядного S-блока при больших n, применяют более простые конструкции [28, 110, 111].

В общем виде S-блок может быть представлен как дешифратор (комбинационная схема), преобразующий n-разрядное двоичное сообщение в одноразрядное сообщение по основанию 2^n, систему коммутаторов внутренних соединений (всего соединений $2^n!$) и шифратор (комбинационная схема), переводящий сообщение из одноразрядного 2^n-ричного в n-разрядное двоичное [48, 111]. Анализ n-разрядного S-блока, при большом n крайне сложен, так как число возможных значений слишком большое ($2^n!$). В общем случае S-блок может быть и линейным преобразованием, однако на практике используют нелинейные подстановки, причём меньшей разрядности (2^4, 2^8), но как части более сложных систем. При этом в алгоритме шифрования основной задачей нелинейных преобразований является перемешивание бит [3, 4, 107, 108].

P-блок изменяет положение символов и является линейным устройством [107]. Этот блок может иметь очень большое количество

входов-выходов, однако в силу линейности систему нельзя считать криптоустойчивой. Криптоанализ ключа для *n*-разрядного P-блока проводится путём подачи на вход $n-1$ различных сообщений, каждое из которых состоит из $n-1$ нуля («0») и 1 единицы («1») [111]. Главной задачей линейных преобразований является рассеивание битов данных при шифровании [108, 111].

1.2.3 Поточный шифр

Основным отличием поточного шифра от блочного является генерация случайных бит (гаммы), на основе ключа, с последующим их побитовым сложением по модулю два (XOR) с открытым текстом [110, 112, 113]. На выходе данной операции формируется шифротекст.

На рисунке 1.2 изображена общая модель симметричного поточного шифра [112].

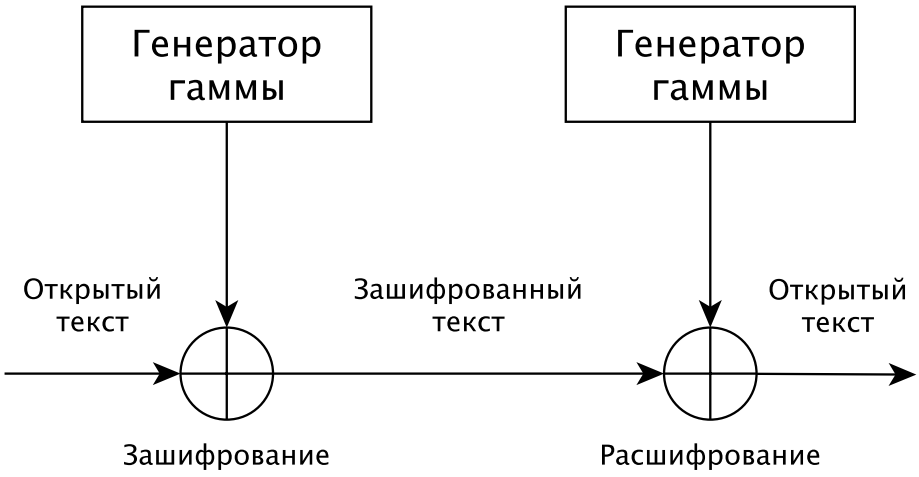

Рисунок 1.2 – Общая модель поточного шифра

Пусть генератор гаммы создаёт последовательность (гамму шифрования) $k_1, k_2, \ldots, k_n$, а $m_1, m_2, \ldots, m_n$ – открытый текст, тогда шифротекст будет иметь вид:

$$c_i = m_i \oplus k_i.$$

Расшифрование происходит аналогичным образом. На приёмной стороне генерируется аналогичная последовательность гаммы $k_1, k_2, \ldots, k_n$, и в результате операции XOR принятого потока данных от другой стороны с ключевым потоком, получается исходный открытый текст:

$$m_i = c_i \oplus k_i.$$

Если последовательность гаммы шифрования не имеет периода и выбирается случайно, то взломать шифр невозможно [3, 114]. Однако на практике такого достичь практически невозможно. Поэтому обычно применяют ключ меньшей длины, с помощью которого генерируется псевдослучайная последовательность удовлетворяющая ряду критериев, включая постулаты Голомба [110, 112, 114].

1.2.4 Криптографическая хэш-функция

Криптографической функцией хэширования называется хэш-функция, если она является стойкой к [3, 115]:

а) восстановлению прообраза;

б) восстановлению второго прообраза;

в) коллизиям.

Вышеперечисленные атаки являются классическими и наиболее общими, т.е. применимы для всех криптографических функций хэшифрования. Однако, практическое применение вносит свои критерии для такого рода криптографических примитивов. Например, производительность и защита от всех известных на сегодняшний день атак стали основными критериями выбора функций конкурса SHA-3 [101].

Следует отметить, что на сегодняшний день теоретически не доказано существование необратимых хэш-функций. Предполагается, что нахождение входного сообщения является вычислительно сложной задачей [116]. Так, например, атака, основанная на «парадоксе дней рождения», позволяет

находить коллизии с длиной хэш-кода, равного *n* битов, приблизительно за $2^{\frac{n}{2}}$ вычислений хэш-функции. Поэтому, хэш-функция является криптостойкой тогда и только тогда, когда не существует алгоритма нахождения коллизии со сложностью меньшей чем $2^{\frac{n}{2}}$. По умолчанию (иногда вводится в качестве критерия) считается, что при малейшем изменении значения входного сообщения (например, одного бита), криптографическая хэш-функция должна сильно изменять значение хэш-кода. Данный критерий известен как лавинный эффект [3, 57].

Современные криптографические хэш-функции имеют три основных этапа вычисления хэш-кода [60, 100]:

а) инициализации;

б) разбиения входного сообщения на блоки и последовательное применение функции сжатия к каждому из них;

в) конечного преобразование и формирования хэш-кода сообщения.

На сегодняшний день большинство хэш-функций построены с использованием схемы Меркле-Дамгорда (Merkle–Damgård) [117, 118]. За последние 10 лет было найдено множество нежелательных свойств данной конструкции, включая удлинение сообщения (length extension) [119, 120, 121], мультиколлизии [122, 123] и другие [119, 124, 125].

Во время проведения конкурса SHA-3 хорошо себя зарекомендовала функция «губка» (sponge function) [120]. С её помощью можно создавать такие криптопримитивы, как блочные симметричные шифры, коды аутентификации сообщения, поточные шифры и хэш-функции. Более того, на основе данной конструкции был спроектирован алгоритм Keccak, ставший победителем конкурса SHA-3 [101].

1.3 Методы криптоанализа блочных симметричных шифров

1.3.1 Дифференциальный криптоанализ

Дифференциальный криптоанализ [44, 57, 108, 126] предполагает наличие упорядоченных пар (α,β), таких, что случайно выбранный открытый текст M и соответствующее ему значение $M-\alpha$ после зашифрования формируют шифртексты C и C', причём наиболее вероятной разностью шифртекстов является $C-C'=\beta$. Под «$-$» понимается некоторая операция, обратная к процедуре смешивания ключа. Упорядоченная пара (α,β) называется дифференциалом, а совокупность дифференциалов на различных раундах – дифференциальной характеристикой [3, 126]. Чем выше вероятность перехода дифференциала (в то же время не равная 1), тем более эффективной является атака. Другими словами, если δ – максимальное значение в таблице распределения дифференциалов (не принимая во внимание значение первой строки, первого столбца) [30, 55, 126], тогда

$$\delta=\max_{\alpha\in\mathcal{F}_2^n,\alpha\neq 0,\beta\in\mathcal{F}_2^m}\#\left\{x\middle|S(x)\oplus S(x\oplus\alpha)=\beta\right\}, \qquad (1.1)$$

где S – подстановка применяемая в алгоритме шифрования.

При дифференциальной атаке злоумышленник изучает, как разность (дифференциал) входных данных (открытого текста) влияет на результирующую разницу (зашифрованных текстов) [126]. Дифференциал, проходящий с наивысшей вероятностью, используется для взлома всего алгоритма шифрования.

1.3.2 Линейный криптоанализ

Линейный криптоанализ основан на Piling-up лемме и был впервые описан в 1994 Нибер, а позже в этом же году применён Мацуи на практике по отношению к блочному симметричному шифру DES [45, 126, 127]. Основная идея заключается в следующем: для случайно выбранных битов ключа, открытого и зашифрованного текстов выражение $\alpha \cdot m + \beta \cdot c + \gamma \cdot k$, где "$\cdot$" обозначает скалярное произведение, есть вероятность отличная от $\frac{1}{2}$. Пусть S – подстановка, применяемая в алгоритме шифрования, λ – максимальное значение аппроксимационной таблицы (без учёта значения ячейки (0,0)), тогда

$$\lambda = \max_{\alpha \in \mathcal{F}_2^n, \alpha \neq 0, \beta \in \mathcal{F}_2^m} \left| \#\left\{ x \left| \bigoplus_{s=0}^{N} \left(x[s] \cdot \alpha[s] \right) = \bigoplus_{t=0}^{N} \left(S(x)[t] \cdot \beta[t] \right) \right. \right\} - 2^{n-1} \right|, \qquad (1.2)$$

где $\zeta[s]$ – s-й бит значения ζ.

Чем больше отклонение от $\frac{1}{2}$, тем более эффективной является атака [126].

1.3.3 Алгебраический криптоанализ

Алгебраическая атака – это метод криптографического анализа, основанный на алгебраических свойствах шифра. Впервые этот метод был применён к блочным шифрам Н. Куртуа (N. Courtois) в 2002 г. [23, 35].

Алгебраические атаки используют внутреннюю структуру шифра, то есть для получения ключа необходимо представить алгоритм шифрования в виде системы уравнений с минимальной степенью многочленов и впоследствии решить данную систему [34, 36, 126, 128-130].

Для реализации атаки необходимо выполнить следующие шаги:

а) декомпозировать алгоритм шифрования на составляющие операции;

б) алгебраически описать каждый из элементов;

в) связать входные значения каждого из элементов с выходными остальных.

Под элементом понимаются линейные и нелинейные операции, применяемые в современных шифрах [3, 39, 126]. Декомпозиция представляет собой разбиение алгоритма шифрования на более мелкие части. Отдельно группируются линейные и нелинейные операции [56, 72].

Под алгебраическим описанием понимается представление преобразования в виде системы, которая связывает входные и выходные значения преобразования. Выходом алгебраического описания блочного симметричного шифра является система уравнений, описывающая все шифрующее преобразование, включая схему разворачивания ключей.

На сегодняшний день существует достаточно большое количество методов решения уравнений над полем GF(2), например, метод Гаусса, XL, T', ElimLim, F4, F5, преобразование в SAT [131, 132]. Однако, сложность большинства из этих методов зависит от разреженности системы. Это позволяет сделать вывод о том, что разреженность уравнений, описывающих S-блок симметричного примитива, напрямую влияет на сложность решения конечной системы уравнений.

N. Courtois в работах [23, 36] предложил вариант, который позволял атаковать шифр AES, зная лишь несколько пар открытого и зашифрованного текстов. Предложенный метод основывается на описании S-блока системой уравнений. Постепенное расширение системы дало возможность описать весь алгоритм шифрования системой уравнений второй степени.

Позднее, больше теоретические чем практические, методы N. Courtois в жизнь воплотил R.-P. Weinmann [133]. В своей работе он атаковал мини-версию шифра AES и тем самым продемонстрировал состоятельность

метода. Похожие результаты получила и E. Kleiman [128, 134], однако, в отличии от Weinmann, в её работе представлен более общий алгоритм по нахождению системы уравнений второй степени на основе построения матрицы, описывающей S-блок.

В 2006 N. Courtois продемонстрировал атаку на полный 6-ти цикловой вариант алгоритма DES. При этом для нахождения ключа понадобилось лишь одна пара открытого текста – шифротекста и персональный компьютер [135].

Применение алгебраического анализа было продемонстрировано и для шифров, поданных на национальный конкурс в Украине [34, 37]. Как и при разработке AES, в шифре Лабиринт для построения S-блока использовалась конструкция K. Nyberg [32, 136], которая позволила добиться предельных показателей для защиты от дифференциального и линейного криптоанализов, при этом привела к возможности применения алгебраической атаки [34, 126].

1.3.4 Атаки на связанных ключах

Атаки на связанных ключах – вид криптоанализа, в котором злоумышленник может наблюдать за работой шифра под действием различных ключей, значения которых первоначально неизвестны, но математические отношения позволяют злоумышленнику установить связь между ними [137-140].

Данная атака предполагает, что криптоаналитик не имеет прямого доступа к искомому ключу (например, ключ прошит в каком-либо аппаратном шифрующем устройстве), но может изменять определенным образом различные фрагменты ключа.

В последнее время атака на связанных ключах используется совместно с атакой типа бумеранг [126, 141], которая основана на дифференциальном криптоанализе.

Основное отличие атаки типа бумеранг от дифференциальной заключается в том, что злоумышленник разбивает шифр на два и ищет дифференциал с наибольшей вероятностью только для одного из них [126, 142, 143].

Стоит отметить, что одним из основных компонентов атаки на основе полного биграфа (biclique) является взаимосвязь ключей [144-146]. Атака получила широкое распространение после успешной реализации на полную версию шифра AES. В работе [145] было теоретически доказано, что ключ шифрования может быть найден со сложностью меньшей, чем полный перебор.

1.3.5 Скользящая (слайд) атака

Предполагается, что даже слабые шифры могут быть криптографически стойкими, если увеличить число циклов и таким образом предотвратить ряд атак. При скользящей атаке, вместо того, чтобы искать уязвимости в шифре, анализируется функция разворачивания ключа, на основе которой, в последствии базируется атака [75, 130, 147-150].

Впервые такой тип атаки предложили Д. Вагнер и А. Бирюков [147]. Она применима преимущественно к многоцикловым шифрам, каждый цикл которых представляет собой преобразование исходного блока с использованием лишь одного ключа. Ключ может как повторятся, так и быть разным для каждого цикла. Важным является то, что циклы должны быть идентичны и обратимы.

Предположим, что есть некий многоцикловый шифр, в каждом цикле которого используется функция $F(x, k)$, где x – выход предыдущего цикла (или открытый текст для первого цикла алгоритма), а k – цикловой ключ. В этом случае для атаки используется следующая пара открытых текстов:

а) случайно выбранный текст P;

б) второй текст – *P’*, представляющий собой результат одноциклового преобразования текста *P*, т.е.:

$$P' = F(P, k).$$

Соответствующие таким открытым текстам шифртексты *C и C’* (для *P* и *P’* соответственно) связаны между собой аналогичным соотношением *C’ = F(C, k)* (рис. 1.3).

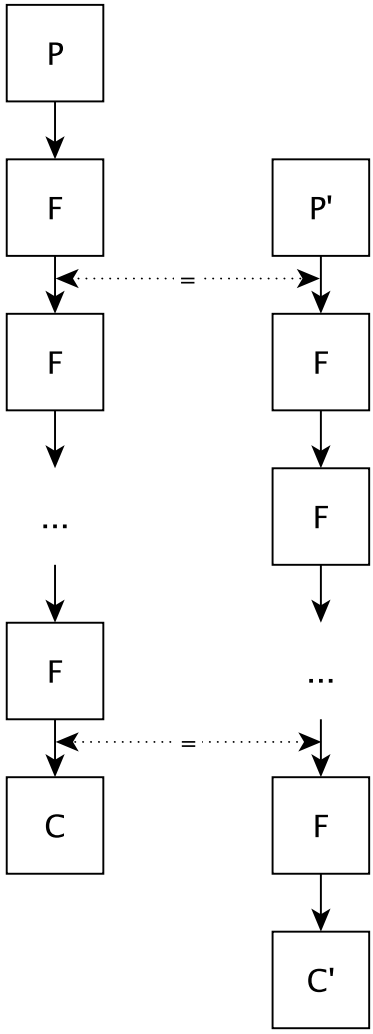

Рисунок 1.3 – Общая структура скользящей атаки

Таким образом, имея две пары текстов (*P*, *P’* и *C*, *C’*), связанные между собой лишь одним циклом шифрования, криптоаналитик способен получить значение ключа цикла *k* [75].

Согласно парадоксу дней рождения, требуемый текст *P’* будет найден

после перебора порядка $2^{\frac{n}{2}}$ текстов, где n – размер шифруемого блока в битах. Стоит отметить, что количество циклов алгоритма в данном случае никак не влияет на успешность его взлома.

На основе данной атаки в работах [75, 151] было показано, что при условии доступа к управлению сеансовым ключом, возможно практическое восстановление долговременного ключевого элемента шифра ГОСТ 28147 [6, 7].

1.3.6 Другие методы криптоанализа БСШ

На сегодняшний день стало практически невозможно применять независимые атаки (т.е. с применением лишь одного из методов) относительно современных шифров. Основной причиной является проектирование шифров с учётов всех существующих методов анализа шифров. Дифференциальный и линейный криптоанализы в той форме, которой были применены к DES [44, 45], уже неэффективны против современных шифров, в следствии чего получили развитие модифицированные или комбинированные варианты атак.

Так, например, из-за простоты описания и понимания, за последние 20 лет появилось множество атак, основанных на дифференциальных свойствах основных составляющих шифра. К ним относятся атаки усечённых дифференциалов, невозможных дифференциалов, бумеранг, дифференциалов высших порядков и другие [3, 39, 126].

В 2011 году впервые была представлена атака на полную версию шифра ГОСТ 28147, основанная на объединении атак «встреча посередине», фиксированных точек и грубая сила [93]. В том же году была представлена первая атака на полную версию шифра AES [145]. Как было указано в 1.3.4,

данная атака состоит из комбинации атаки на связанных ключах и грубая сила с применением полного двудольного графа (biclique) [145, 152].

Таким образом, исследование и применение комбинированных методов криптоанализа является приоритетной областью исследования БСШ.

1.4 Теория булевых функции и подстановок

В разделе приведены теоретические аспекты представления и построения булевых функций, а также связанные с ними актуальные криптографические свойства для симметричных криптоалгоритмов.

1.4.1 Определения и обозначения

Пусть $\mathcal{F}_2^n$ – векторное пространство всех бинарных векторов длины n, где $\mathcal{F}_2$ поле Галуа с двумя элементами {0,1} [4, 48, 153]. Пусть n и m – два натуральных числа, тогда под (n,m)-функцией понимается векторная булева функция $F:\mathcal{F}_2^n \mapsto \mathcal{F}_2^m$ [48]. Такие функции используются в криптографии как нелинейные отображения в псевдослучайных генераторах (потоковых шифрах) или как подстановки (S-блоки) в блочных симметричных шифрах [4, 48]. Булевы функции $f_1,\ldots,f_m$ такие, что $F(x)=(f_1,\ldots,f_m)$, называются координатными функциями F. Линейные комбинации координатных функций с ненулевыми коэффициентами называются компонентными функциями F.

Очевидно, что при $m=1$ векторная булева функция имеет один бит на выходе и эквивалентна обычной булевой функции. Для нахождения алгебраической структуры векторное пространство часто имеет структуру конечного поля $\mathcal{F}_{2^n}$ с некоторым неприводимым полиномом.

1.4.2 Криптографические свойства булевых функции

Пусть $f(x):\mathcal{F}_2^n \mapsto \mathcal{F}_2$, где $x=(x_0,x_1,...,x_{n-1})$ – некоторая булева функция с n переменными. Тогда функция нахождения веса Хемминга задаётся как [4, 57, 153]

$$hw(f)=\sum_{x=0}^{2^n-1} f(x). \tag{1.3}$$

Пусть $f(x)$ и $g(x)$ – булевы функции с n переменными. Тогда расстояние Хемминга между двумя функциями вычисляется по формуле:

$$hd(f,g)=\sum_{x=0}^{2^n-1} f(x)\oplus g(x). \tag{1.4}$$

Алгебраическая нормальная форма (АНФ) некоторой булевой функции имеет вид:

$$\begin{aligned} f(x_0,x_1,\ldots,x_{n-1}) = a \oplus a_0x_0 \oplus a_1x_1 \oplus \ldots \oplus a_{n-1}x_{n-1} \oplus a_{01}x_0x_1 \oplus a_{02}x_0x_2 \oplus \ldots \oplus \\ \oplus a_{(n-2)(n-1)}x_{n-2}x_{n-1} \oplus \ldots \oplus a_{012\ldots n-1}x_0x_1x_2\ldots x_{n-1}. \end{aligned} \tag{1.5}$$

Под алгебраической степенью булевой функции понимается максимальная степень монома с коэффициентом отличным от 0; она обозначается как $\deg(f)$ [4, 57, 153].

Значения корреляции между произвольной булевой функцией $f(x)$ и множеством всех линейных функций определяется как преобразование Уолша в виде:

$$W(\omega)=\sum_{x=0}^{2^n-1}(-1)^{f(x)\oplus l_\omega(x)}, \tag{1.6}$$

где $l_\omega(x)=\omega\cdot x=\omega_0x_0\oplus\omega_1x_1\oplus\ldots\oplus\omega_{n-1}x_{n-1}$.

Функция $f(x)$ от n переменных называется сбалансированной если

$$hw(f)=2^{n-1}. \tag{1.7}$$

Под нелинейностью булевой функции ($NL(f)$) понимается минимальное расстояние Хемминга ко всем аффинным функциям,

состоящими из n переменных. Ниже приведена взаимосвязь между нелинейностью булевой функции и преобразованием Уолша [153]:

$$NL(f) = \frac{1}{2} \cdot \left(2^n - \max\left(\left|W(\omega)\right|\right)\right). \tag{1.8}$$

Автокорреляционной функцией $r_f(\alpha)$ таблицы истинности булевой функции $f(x)$ является производная функции для всех переменных по направлению $\alpha \in \mathcal{F}_2^n$, которая задаётся в виде [4, 57, 153]:

$$r_f(\alpha) = \sum_{x=0}^{2^n-1} (-1)^{f(x) \oplus f(x \oplus \alpha)}. \tag{1.9}$$

Другими словами, $r_f(\alpha)$ показывает на сколько функция $f(x)$ отличается от себя, сдвинутой на α позиций. Значения $r_f(\alpha)$ иногда называют индикаторами [48, 153, 154].

Пусть $\left|AC\right|_{\max}$ - максимальное абсолютное значение функции автокорреляции (абсолютный индикатор [154]), тогда:

$$\left|AC\right|_{\max} = \max_{\alpha} \left|r_f(\alpha)\right|. \tag{1.10}$$

Обозначим через σ глобальную лавинную характеристику (ГЛХ) «сумма квадратов» (sum-of-square indicators) [153, 155], тогда:

$$\sigma = \sum_{\alpha=0}^{2^n-1} r_f^2(\alpha). \tag{1.11}$$

Говорят, что некоторая булева функция $f(x)$ удовлетворяет строгому лавинному критерию (SAC), если для всех s справедлива система уравнений [153, 155]:

$$\begin{cases} hw(s) = 1; \\ \sum_{x=0}^{2^n-1} f(x) \oplus f(x \oplus s) = 2^{n-1}. \end{cases} \tag{1.12}$$

Булева функция $f(x)$ удовлетворяет критерию распространения порядка k ($PC(k)$) тогда и только тогда, когда для ненулевого вектора $\alpha \in \mathcal{F}_2^n$

$$\begin{cases} 1 \le hw(\alpha) \le k; \\ \sum_{x=0}^{2^n-1} f(x) \oplus f(x \oplus \alpha) = 2^{n-1}. \end{cases} \quad (1.13)$$

Булева функция $f(x)$ обладает корреляционным иммунитетом порядка m ($CI(m)$), если система уравнений [153]

$$\begin{cases} 1 \le hw(\omega) \le m; \\ W(\omega) = 0 \end{cases} \quad (1.14)$$

справедлива для всех ω.

Если булева функция $f(x)$ одновременно является сбалансированной и при этом обладает корреляционным иммунитетом *t*-го порядка, то такая функция называется *t*-устойчивой [57, 153].

Пусть функция $g(x)$ – аннигилятор функции $f(x)$, т.е. $f(x) \cdot g(x) = 0$. Тогда, минимальная алгебраическая степень функции $g(x) \neq 0$ называется алгебраическим иммунитетом функции $f(x)$ и обозначается $AI(f)$, если

$$\begin{cases} f(x) \cdot g(x) = 0; \\ (f(x) \oplus 1) \cdot g(x) = 0. \end{cases} \quad (1.15)$$

1.4.3 Криптографические свойства векторных булевых функций

Преобразование Уолша некоторой (n,m) -функции отображает упорядоченную пару $(u,v) \in \mathcal{F}_2^n \times \mathcal{F}_2^m$ в сумму (вычисляемую в кольце целых чисел) [48, 54]:

$$\lambda(u,v) = \sum_{x \in F_2^n} (-1)^{v \cdot F(x) + u \cdot x}, \quad (1.16)$$

где символ “ $\cdot$ ” обозначает скалярное произведение в векторных пространствах $\mathcal{F}_2^n$ и $\mathcal{F}_2^m$. Стоит отметить, что функция $v \cdot F(x)$ является

компонентной функцией F, при условии $v \neq 0$. Преобразование Уолша удовлетворяет соотношению Парсеваля (Parseval) [4, 48, 57]

$$\sum_{u \in F_2^n} \left(\sum_{x \in F_2^n} (-1)^{v \cdot F(x) + u \cdot x} \right)^2 = 2^{2n} \tag{1.17}$$

для произвольного $v \in \mathcal{F}_2^m$. Множество, состоящее из всех значений $\lambda(u,v)$, где $u \in \mathcal{F}_2^n$ и $v \in \mathcal{F}_2^{m*} = \mathcal{F}_2^m \setminus \{0\}$, называется спектром Уолша функции F. Расширенный спектр Уолша состоит из спектра Уолша, каждый элемент которого представлен в виде абсолютного значения.

Произвольная (n,m)-функция F имеет единственное представление в алгебраической нормальной форме [48]:

$$F(x) = \sum_{I \subseteq \{1,\ldots,n\}} a_I \left(\prod_{i \in I} x_i \right), \tag{1.18}$$

где $a_I \in \mathcal{F}_2^m$, а сумма вычисляется в $\mathcal{F}_2^m$.

Алгебраическая степень (n,m) -функции F равна максимальной алгебраической степени её координатных функций. При этом, минимальной степенью векторной булевой функции F называется минимальную алгебраическую степень среди всех компонентных функций, т.е. $\min\{\deg(v \cdot F(x)) \mid v \in \mathcal{F}_2^{m*}, x \in \mathcal{F}_2^n\}$. Векторные функции, используемые в криптографии, должны иметь высокую минимальную степень, чтобы обеспечивать защиту от различных типов атак (например от дифференциальной атаки высоких порядков) [4, 48]. Функция называется аффинной, если её алгебраическая степень равна 1 и квадратичной – в случае второй [48, 77].

Любая (n,n) -функция F имеет единственное одномерное представление над полем $\mathcal{F}_{2^n}$ со степенью, не превышающей $2^n - 1$ [48]:

$$F(x) = \sum_{j=0}^{2^n-1} \delta_j x^j, \quad \delta_j \in \mathcal{F}_{2^n}. \tag{1.19}$$

Пусть $w_2(j)$ – количество ненулевых бит в двоичном представлении числа j. Тогда функция F имеет алгебраическую степень равную $\max\{w_2(j) | \delta_j \neq 0\}$.

Ниже приведена лемма, также известная как критерий Эрмита (Hermite) [156], которая часто используется для определения перестановочного полинома над конечным полем.

Лемма 1.1 Пусть p – характеристика поля $\mathcal{F}_q$. Полиномом $F \in \mathcal{F}_q[x]$ является перестановочным тогда и только тогда, когда выполняются следующие два условия:

– F имеет ровно один корень в поле $\mathcal{F}_q$;

– для всех $t = \{1,\ldots,q-2\}$ и $t \neq 0 (\mathrm{mod}\, p)$ полином $F(x)^t (\mathrm{mod}\, x^q - x)$ имеет степень, не превышающую $q-2$.

Другими словами, функция называется перестановочной тогда и только тогда, когда генерирует перестановку элементов поля. Функция $F : \mathcal{F}_{2^n} \mapsto \mathcal{F}_{2^n}$ называется линейной, если F является линейным полиномом над полем $\mathcal{F}_{2^n}$ [48-50]:

$$F(x) = \sum_{i=0}^{n-1} c_i x^{2^i}, \quad c_i \in \mathcal{F}_{2^n}. \tag{1.20}$$

Аффинная функция состоит из суммы константы и линейной функции.

Если m делит n, тогда любая (n,m)-функция может быть представлена как отображение поля $\mathcal{F}_{2^n}$ в себя, так как $\mathcal{F}_{2^m}$ является подполем поля $\mathcal{F}_{2^n}$ [48, 156]. Поэтому функция имеет уникальное представление вида:

$$F(x) = tr_{n/m}\left(\sum_{j=0}^{2^n-1} \delta_j x^j \right), \quad \delta_j \in \mathcal{F}_{2^n}, \tag{1.21}$$

где $tr_{n/m}(x) = x + x^{2^m} + x^{2^{2m}} + x^{2^{3m}} + \cdots + x^{2^{n-m}}$ – след из поля $\mathcal{F}_{2^n}$ в $\mathcal{F}_{2^m}$.

(n,m)-функция F является сбалансированной тогда и только тогда, когда её компонентные функции являются сбалансированными (т.е. вес Хемминга равен 2^{n-1}) [48].

Нелинейность $NL(F)$ (n,m)-функции F есть минимальное расстояние Хемминга между всеми компонентными функция F и всеми аффинными функциями с n переменными [48]:

$$NL(F) = 2^{n-1} - \frac{1}{2} \max_{v \in \mathcal{F}_2^m \setminus \{0\}; u \in \mathcal{F}_2^n} \left\{ \left| \sum_{x \in F_2^n} (-1)^{v \cdot F(x) + u \cdot x} \right| \right\}. \quad (1.22)$$

Как было указано в 1.3.2, критерий нелинейности определяет уровень стойкости S-блока к линейному криптоанализу [45, 48, 126]. Принимая во внимание уравнение (1.2), получаем:

$$NL(F) = 2^{n-1} - \lambda \quad (1.23)$$

В следующей теореме приводится граница значения нелинейности, также известная как граница Сидельникова-Чабауда-Вауденая (Sidelnikov-Chabaud-Vaudenay) [4, 48].

Теорема 1.1 Пусть n и m – натуральные числа, такие, что $m \geq n-1$. *Обозначим через F произвольную* (n,m)*-функцию. Тогда:*

$$NL(F) \leq 2^{n-1} - \frac{1}{2}\sqrt{3 \times 2^n - 2 - 2\frac{(2^n - 1)(2^{n-1} - 1)}{2^m - 1}}. \quad (1.24)$$

При $m = n-1$ неравенство сводится к $NL(F) \leq 2^{n-1} - 2^{\frac{n}{2}-1}$, которое известно как граница радиуса покрытия (covering radius bound) [48].

Векторная булева функция является бент-функцией, если для всех значений $v \neq 0$ и u преобразование Уолша принимает значения $\pm 2^{\frac{n}{2}}$. Если

значения принадлежат множеству $\left\{0,\pm 2^{\frac{n+1}{2}}\right\}$, то такая функция называется почти бент (ПБ) (almost bent) [4, 48, 50, 57].

С другой стороны, любая (n,m)-функция F является бент-функцией тогда и только тогда, когда её производная функция ($D_a(x)=F(x)+F(x+a)$, $x,a\in\mathcal{F}_2^n$) сбалансирована [48]. По этой причине бент-функции также называются совершенно нелинейными (СН) (perfect nonlinear) [32, 48].

Любая (n,m)-функция F является δ-равномерной, если для любого $a\in\mathcal{F}_2^n\setminus\{0\}$ и $b\in\mathcal{F}_2^m$ уравнение $D_a(x)=b$ имеет не более δ решений [32, 44, 48, 126]. Функция F называется почти совершенно нелинейной (ПСН) (almost perfect nonlinear (APN)) если $\delta=2$. Для произвольной F значение $\delta\geq 2^{n-m}$ и равно 2^{n-m} тогда и только тогда, когда F является совершенно нелинейной [5, 48-50].

1.4.4 Эквивалентность векторных булевых функций

Две функции $F,G:\mathcal{F}_2^n\mapsto\mathcal{F}_2^m$ называются расширенно аффинно (РА) эквивалентными (extended affine (EA) equivalent), если существуют такие аффинно-перестановочные функции $A_1(x)=L_1(x)+c_1$, $A_2=L_2(x)+c_2$ и произвольная линейная функция $L_3(x)$, что [48, 77]:

$$F=A_1(G(A_2(x)))+L_3(x)=A_1\circ G\circ A_2(x)+L_3(x). \qquad (1.25)$$

При выполнении условия $L_3(x)=const$ функции F и G называются аффинно-эквивалентными, а при $L_3(x)=0,c_1=0,c_2=0$ – линейно-эквивалентными. Более того, если хотя бы один из параметров $L_1(x),L_2(x),L_3(x),c_1$ или c_2 отсутствует, то такие функции называются частично расширенно аффинно (ЧРА) эквивалентными (restricted EA-equivalent).

В работе [157] функции F и G рассматриваются в более общем представлении – в виде $G_F(x,y)=\left(\{x,y\}:y=F(x)\right)$. Пусть $\mathcal{L}(x,y)$ – некоторая аффинная перестановочная функция, отображающая векторное пространство $\mathcal{F}_2^{2n}$ в себя, тогда

$$\mathcal{L}(x,y)=(L_1(x)+L_2(y),L_3(x)+L_4(y)) \qquad (1.26)$$

для некоторых аффинных функций L_1, L_2, L_3, L_4 из $\mathcal{F}_2^n$ в себя. Более того, $\mathcal{L}(x,y)$ – аффинная перестановочная функция если система

$$\begin{cases} L_1(x)+L_2(y)=0 \\ L_3(x)+L_4(y)=0 \end{cases}$$

имеет лишь одно решение [48, 157].

Две функции F и G называются Карле-Шарпен-Зиновьев (КШЗ) (Carlet-Charpin-Zinoviev (CCZ))-эквивалентными, если существует аффинная перестановка $\mathcal{L}:\mathcal{F}_2^{2n}\mapsto\mathcal{F}_2^{2n}$, такая что $G_F=\mathcal{L}\left(G_G\right)$. Две функции F и G КШЗ-эквиваленты тогда и только тогда, когда для некоторой аффинной перестановки $\mathcal{L}(x,y)$ выполняются условия:

$$F(x)=F_2\circ F_1^{-1}(x),$$

где $F_1(x)=L_1(x)+L_2\circ G(x)$ – перестановочная функция, а $F_2(x)=L_3(x)+L_4\circ G(x)$ – произвольная.

Для КШЗ- и РА-эквивалентных функций инвариантными остаются следующие криптографические свойства: расширенный спектр Уолша (ПБ, СН, нелинейность) и δ-равномерность (ПСН). Алгебраическая степень сохраняется только для РА-эквивалентных функции. Таким образом, КШЗ-эквивалентность является более общим случаем РА-эквивалентности и обратных функций [48, 158].

1.5 Выводы

К настоящему моменту завершились несколько открытых конкурсов по выбору новых криптоалгоритмов. К кандидатам предъявлялись высокие требования по обеспечению защиты от существующих атак наравне с высокой производительностью. Аналогичные требования предъявлялись и к СРК, на слабостях которых базируется ряд атак [24, 137, 139, 140, 141, 159]. Одной из самых эффективных на сегодняшний день является biclique-атака на действующий стандарт США, сложность которой равна $2^{126.1}$, $2^{189.7}$ и $2^{254.4}$ для AES-128, AES-192 и AES-256 соответственно [25]. Однако, атака до сих пор остаётся теоретической и нереализуемой на практике.

Отличительной особенностью проходившего в Украине конкурса по выбору симметричного алгоритма шифрования являлось применение сложных схем разворачивания ключа. Это позволило защититься от ряда атак, в том числе и от алгебраической [37]. Большинство БСШ, поданных на конкурс, включали в себя применение разномодульного сложения, которое значительно затрудняет применение различных методов криптоанализа [29, 81], включая дифференциальный и алгебраический.

Несмотря на разномодульное сложение, основными нелинейными элементами современных шифров являются подстановки. Существует множество различных критериев S-блоков, определяющих впоследствии стойкость алгоритма шифрования. Однако, на сегодняшний день нет однозначного мнения в необходимости большинства из них. Например, исследования, представленные в работах [53, 55, 62], показывают, что значение частного дифференциала не играет существенной роли при большом количестве циклов шифрования. Поэтому использование в шифрах Калина и Мухомор подстановок с непредельными показателями не оказывает особого влияния на стойкость алгоритма, при этом лишь немного уменьшается динамика достижения средних значений показателя полного

дифференциала [55].

Применение математического аппарата векторных булевых функций позволяет упростить описание основных элементов симметричных алгоритмов. Такое представление даёт возможность обобщения множества критериев, в том числе и применяемых к подстановкам, одновременно позволяя оценить корреляционные, алгебраические и другие свойства S-блоков. Как показывают исследования, применение теоретического подхода не всегда может удовлетворить практические потребности, в частности по генерации подстановок с заданными непредельными показателями.

Таким образом, несмотря на множество существующих решений в области симметричной криптографии, по прежнему остаётся актуальным вопрос по нахождению подстановок, применение которых в алгоритмах шифрования обеспечивает защиту от существующих и перспективных видов атак. Для развития и усовершенствования методов генерации нелинейных узлов замены необходимо решить задачу обоснования критериев, и продолжить исследования в области методов криптоанализа и теории векторных булевых функций.

2 КРИТЕРИАЛЬНЫЙ ПОДХОД К ОЦЕНКЕ СТОЙКОСТИ НЕЛИНЕЙНЫХ УЗЛОВ ЗАМЕНЫ

2.1 Анализ известных характеристик, критериев и показателей

2.1.1 Общие требования к нелинейным блокам поточных шифров

Многие современные потоковые шифры строятся с использованием функции фильтрации [96, 98, 113]. Один из способов построения генераторов гаммы (потоковых шифров) представлен на рисунке 2.1.

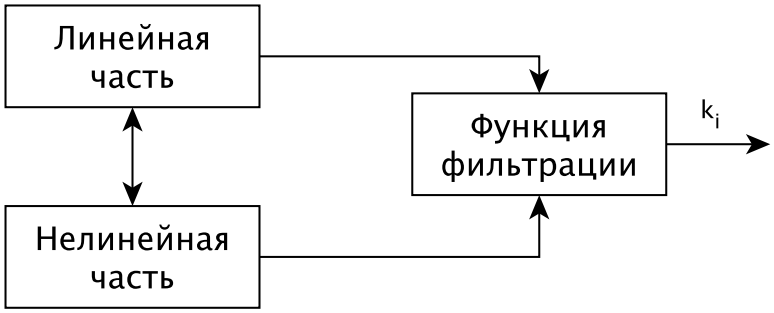

Рисунок 2.1 – Генератор гаммы с фильтрующей функцией

Как видно из рисунка, алгоритм выработки гаммы состоит из трёх частей: функции фильтрации (F), линейной и нелинейной частей. В качестве линейной части используются линейные рекуррентные регистры (ЛРР) с обратной связью. В общем случае функция фильтрации является (n,m) функцией. Однако, из-за того, что в большинстве случаев гамма-последовательность состоит из битов, то чаще всего используются векторные булевы функции с $m=1$, т.е. булевы функции (см. 1.4.1) [86-88, 96].

На сегодняшний день не существует общей теории построения нелинейной части для поточных шифров. Однако, разработчики применяют следующие два способа [96, 98]:

– усложнение ЛРР с использованием нелинейных преобразований (например, подстановки или сложение по модулю 2^n);

– использование нелинейных рекуррентных регистров с обратной связью.

Нелинейные регистры с обратной связью являются весьма перспективным элементами для построения генераторов гаммы [160-163]. Такие криптопримитивы являются эффективными при аппаратной реализации и обеспечивают высокий уровень нелинейности, что имеет решающее значение для обеспечения криптографической стойкости. Тем не менее, работы, посвящённые анализу этих конструкций, главным образом основаны на ряде допущений и не имеют формальных оценок и доказательств. Последние исследования показывают, что неудачный выбор параметров, которые, на первый взгляд, не влияют на стойкость, может значительно повлиять на сложность криптоаналитических атак [86-88, 164].

Основными требованиями к гамме шифрующей являются большая длина периода, оптимальные корреляционные свойства и высокая нелинейность [96]. Чтобы добиться этого, разработчики поточных шифров часто объединяют линейные и нелинейные регистры. При этом предполагается, что линейная часть гарантирует период гаммы шифрующей, а нелинейная обеспечивает стойкость к статистическим и алгебраическим атакам. Кроме того, функция обновления для обоих регистров может включать в себя состояние соседнего регистра и осуществлять так называемое «взаимное управление» [86, 160, 165].

В независимости от того, какой из методов применяется, векторные булевы функции должны обеспечивать защиту от корреляционных, алгебраических, статистических и других видов атак. Поэтому они должны удовлетворять максимальному количеству критериев, указанных в 1.4.2 и 1.4.3. Так, например, лавинный эффект показывает, какое количество выходных бит меняется при изменении некоторого количества входных. Это свойство необходимо для предотвращения статистических атак [4, 96]. Если

векторная булева функция удовлетворяет строгому лавинному критерию, тогда изменение одного входного бита меняет ровно половину выходных [4].

2.1.2 БСШ DES

На конференции Crypto'2000 Дон Копперсмит (Don Coppersmith), будучи одним из разработчиков DES, привёл следующий перечень критериев проектирования подстановок, использованных в данном шифре [126, 166]:

а) 6 входных битов и 4 выходных;

б) отсутствие функций, описывающих выходные биты, близких к линейным;

в) 4 внутренних бита подстановки (крайние биты фиксированные) организовывают перестановку, т.е. $\Delta_{in} = 0wxyz0 \Rightarrow \ \Rightarrow \Delta_{out} \neq 0$;

г) $\Delta_{in} = 001100 \Rightarrow \left|\Delta_{out}\right| \geq 2$;

д) вероятность $P(\Delta_{out} = 0 \mid \Delta_{in}) \leq \frac{8}{32}$;

е) вероятность $P(\Delta_{out} = 0 \mid \Delta_{in})$ может быть более строгая для конкретного случая;

ж) $\Delta_{in} = 11xy00 \Rightarrow \Delta_{out} \neq 0$;

з) реализация должна использовать не более 47 вентилей.

Очевидно, что критерии а), в) и з) являются специфическими для подстановок шифра DES [89]. Известно, что криптографически стойкая подстановка должна обладать максимальной алгебраической степенью (см. 1.4.2-1.4.3) [36, 48, 131]. Эквивалентным критерием для узлов замены шифра DES является пункт б). Остальные критерии в той или иной степени покрываются характеристиками максимума дифференциальной и аппроксимационной таблиц (см. формулы 1.1 и 1.2).

Стоит отметить, что ещё в 1974 году, команда компании International Business Machines (IBM) оценила сложность атаки на основе подобранного шифротекста [126].

2.1.3 БСШ Rijndael

Пусть $S = (f_0, f_1, \ldots, f_{m-1})$ – некоторая $n \times m$ подстановка, где f_i – булева функция от n переменных. Обозначим через g_j множество всех линейных комбинаций f_i. Тогда [48, 51, 54]:

– нелинейность S равна

$$NL(S) = \min_{0<j<2^m} \left(NL(g_j) \right); \tag{2.1}$$

– минимальная степень S определяется как

$$\deg(S) = \min_{0<j<2^m} \left(\deg(g_j) \right); \tag{2.2}$$

– максимум автокорреляции S задаётся в виде

$$\left| AC(S) \right|_{\max} = \max_{0<j<2^m,\, \alpha \in \{1,2,\ldots,2^{n-1}\}} \left(r_{g_j}(\alpha) \right); \tag{2.3}$$

– S удовлетворяет строгому лавинному эффекту, если каждая функция g_j $\left(j = 1,2,...,2^m - 1\right)$ удовлетворяет SAC (см. формулу 1.12);

– S удовлетворяет критерию распространения порядка k, если каждая функция g_j $\left(j = 1,2,...,2^m - 1\right)$ удовлетворяет $PC(k)$ (см. формулу 1.13);

– S обладает корреляционным иммунитетом порядка t, если каждая функция g_j $\left(j = 1,2,...,2^m - 1\right)$ обладает корреляционным иммунитетом порядка t (см. формулу 1.14);

– S сбалансирована (является перестановкой), если каждая функция g_j $\left(j = 1,2,...,2^m - 1\right)$ сбалансированна (см. формулу 1.7);

– S является t-устойчивой подстановкой, если каждая функция g_j $\left(j = 1,2,...,2^m - 1\right)$ t-устойчивая [4].

Аналогичные критерии в терминах теории векторных булевых функций приведены в 1.4.3.

После публикации статей по дифференциальному и линейному криптоанализам [44, 45] были представлены критерии к подстановкам для защиты от этих типов атак [32, 126]. В [32] был предложен метод построения идеального нелинейного узла замены с минимально достижимой (на то время) вероятностью нетривиального преобразования разницы и линейного приближения. Данный метод (см. 1.4.4) был использован разработчиками в шифре Rijndael для генерации S-блока со следующими критериями [15, 39]:

а) обратимость, т.е. подстановка должна быть сбалансированной;

б) минимизация наибольшего нетривиального значения между комбинациями входных и выходных битов;

в) минимизация наибольшего нетривиального значения в дифференциальной таблице;

г) сложность алгебраического представления над F_{2^n};

д) простота описания.

Дополнительно, *S*-блок не должен иметь фиксированных точек ($S(a) = a$) и обратных фиксированных точек ($S(a) = \overline{a}$). Критерии проектирования подстановки выбирались с учётом дифференциального и линейного криптоанализа с одной стороны, и атак с использованием алгебраических преобразований (например, интерполяционная атака [167]) – с другой [39].

Характеристика (см. выше, подразделы 1.4.2 и 1.4.3) узла замены шифра Rijndael представлена в таблице 2.1, где:

а) МТД – максимальное значение таблицы дифференциалов;

б) МТЛА – максимальное значение таблицы линейных аппроксимаций;

в) запись X:Y означает: X – первый элемент в цикле, Y – длина цикла [87];

г) AI/КУ/SP – алгебраический иммунитет, количество уравнений и разреженность соответственно (см. пункт 2.2.3).

Как видно из таблицы, подстановка не удовлетворяет многим критериям, включая критерий распространения, корреляционный иммунитет, строгий лавинный эффект.

Таблица 2.1 – Характеристика *S*-блока шифра Rijndael

Свойства	Значение
Компонентные булевы функций	
Сбалансированность	Да
Нелинейность	112
$\lvert AC\rvert_{\max}$	32
Критерий распространения	0
Корреляционный иммунитет	0
SSI	133120
Минимальная степень	7
t-устойчивость	0
Строгий лавинный критерий	Нет
Подстановка	
Биективность	Да
МТД	4
МТЛА	16
Циклическая структура	115:2, 11:27, 0:59, 1:81, 4:87
AI/КУ/SP	2/39/0,689

Как было показано в [51, 56], максимальное значение алгебраического иммунитета для 8-битных подстановок равно 3, при этом система состоит из

441 уравнений. Наличие у S-блока шифра Rijndael лишь 2-й степени позволило применить алгебраическую атаку [34-36, 128], которая основывается на возможности описания всего шифрующего преобразования при помощи системы уравнений. В отличие от дифференциального и линейного криптоанализов, злоумышленнику необходимо лишь несколько пар открытого текста и соответствующего ему зашифрованного текста. Однако, сложность атаки на сегодняшний день всё ровно больше чем полный перебор [131].

Благодаря применению преобразования Ниберг-Динга [30, 34], остальные свойства являются лучшими из известных для 8-ми битных подстановок. Цикловые свойства можно изменять применяя PA-эквивалентность (см. формулу 1.25).

2.1.4 БСШ ГОСТ 28147-89

Стандарт ГОСТ 28147-89 предусматривает использование долговременных ключевых элементов, которые являются частью секретного ключа [6]. В алгоритме шифрования ДКЭ представлены в виде подстановок и обеспечивают дополнительный уровень защиты, что делает вопрос генерации криптографически стойких *S*-блоков ещё более важным.

2.1.4.1 Критерии случайности подстановок

В 1990-х годах, после обретения Украиной независимости, среди криптографов стал вопрос по отбору подстановок для блочного симметричного шифра ГОСТ 28147-89 [6]. Одной из идей, предложенных в работах [3, 168, 169], является генерация случайных *S*-блоков с последующей проверкой на критерии случайности.

Перестановка S с входным значением равным n бит, считалась случайной, если:

а) количество инверсий (η) в S-блоке удовлетворяет неравенству:

$$\left|\eta - \frac{N(N-1)}{4}\right| \le a\frac{\sqrt{N^3}}{6}, \tag{2.4}$$

где обычно $a = 1$;

б) количество циклов (ς) S лежит в пределах

$$\left|\varsigma - \ln(N)\right| \le a\sqrt{\ln(N)}, \tag{2.5}$$

где обычно $a = 1$;

в) количество возрастаний (θ) перестановки находится в пределах

$$\left|\theta - \frac{N}{2}\right| \le a\sqrt{\frac{n}{12}}, \tag{2.6}$$

где обычно $a = 1$.

Очевидно, что данные критерии защищают лишь от статистических атак и никак не учитывают специализированные атаки на БСШ (например, дифференциальную).

2.1.4.2 Современные критерии

После опубликования и изучения работ по линейному и дифференциальному криптоанализу критерии случайности были дополнены новыми. Множество работ посвящены исследованиям, связанными с доказательством и формированием набора критериев, а также ускорением методов генерации подстановок [11, 33, 51, 54, 57, 58, 62, 77].

В [51] показано, что все подстановки 4 в 4 бита могут быть описаны системой уравнений второй степени. Следовательно, критерий алгебраического иммунитета не играет большой роли для таких подстановок. Но в тоже время он не может равняться 1.

В дополнение к критериям, описанным в 2.1.1, были добавлены следующие [3, 4]:

а) максимизация нелинейности (2.1);

б) минимальная степень должна равняться 3 (2.2);

в) минимизация максимального значения таблицы дифференциалов (1.1).

2.1.5 БСШ Калина

Блочный симметричный шифр Калина был представлен на украинский конкурс по отбору перспективного алгоритма шифрования [13, 29].

Восьмибитовые подстановки для шифра Калина генерировались случайным образом. Далее выбирались те *S*-блоки, которые удовлетворяли следующим условиям [29]:

а) максимальное значение вероятности нетривиальных XOR разниц равно 2^{-5};

б) максимальное абсолютное значение вероятности линейного отклонения равно 2^{-3};

в) минимальная степень равна 7.

В процессе конкурса было показано, что подстановки, используемые в шифре, обладают алгебраическим иммунитетом равным 3. И к существующим трём критериям был добавлен ещё один [51]: количество уравнений 3-й степени, описывающих подстановку, должно быть 441 (минимальное значение для $n = 8$). Критерии, описанные в 2.1.1, в явной степени не использовались, однако проверка на циклические свойства все же проводилась.

2.1.6 Другие симметричные криптоалгоритмы

На сегодняшний день существует множество различных криптоалгоритмов, применяющих векторные булевы функции. Наибольшее распространение в БСШ получили перестановки. После конкурса AES [28] многие разработчики приняли на вооружение критерии, описанные в спецификации к шифру Rijndael [15, 39]. Поэтому на сегодняшний день существует два подхода к генерации узлов нелинейной замены:

а) ориентация на предельные показатели для защиты от определённых атак;

б) защита от всех существующих на сегодняшний день атак.

Из представленных алгоритмов на национальный конкурс первый метод был использован в шифрах Лабиринт и ADE [26, 30]. Процедура генерации нелинейных узлов замены идентична Rijndael, за исключением использования других аффинных преобразований.

В последнее время второй метод применяется всё чаще. Многие криптопримитивы (например, Мухомор [14], Калина [29], Стрибог [60, 102, 104]) в большей мере заботятся о защите от всех существующих атак, нежели о достижении предельных характеристик. Например, достичь необходимый уровень защиты от дифференциального и линейного криптоанализа можно за счёт увеличения количества раундов или применения разномодульного сложения. Поэтому разработчики все больше уделяют внимание другим потенциальным атакам (например, алгебраической).

2.2 Обоснование критериев отбора таблиц подстановок БСШ

На сегодняшний день наибольшее развитие и распространение получили атаки, основанные на дифференциальном и линейном криптоанализе. Другие виды анализа, например «встреча по середине» [3],

являются специфическими для каждого шифра и, в большинстве случаев, используют общую структуру криптографического преобразования, а не отдельных её компонентов.

2.2.1 Обязательные критерии

Из подразделов 1.3 и 2.1 следует, что подстановки, применяемые в криптоалгоритмах, должны удовлетворять критериям б) и в) из 2.1.3. Необходимость максимального значения минимальной степени подстановки и обратной ей обусловлена защитой от интерполяционной и статистических атак [4]. Проверка обратной подстановки необходима из-за того, что КШЗ-эквивалентность не сохраняет минимальную степень векторной булевой функции. В качестве примера можно привести функцию x^{-1}, которая обладает наибольшим значением минимальной степени [32, 153, 158], при этом, обратная функция будет линейной. Дополнительно, в большинстве случаев, подстановки используемые в БСШ должны быть биективными [170, 171], т.е. перестановками.

Последние работы [53, 55] показывают, что значение максимума дифференциала всего шифра не зависит от нелинейных свойств *S*-блоков после определённого количества циклов. Однако, необходимо отметить, что данный показатель влияет на динамические свойства, т.е. как быстро шифр достигает теоретического среднего максимума дифференциальной таблицы. Это означает, что подстановки с более низким значением δ-равномерности более эффективные, чем случайные *S*-блоки.

К подстановкам, применяемым в блочных шифрах, дополнительно ко всем вышеперечисленным критериям, необходимо добавить следующие: расширенный критерий алгебраического иммунитета (см. пункт 2.2.3) и отсутствие фиксированных точек (см. пункт 2.2.4) [108]. Последний необходимо учитывать лишь при определённых условиях [85].

2.2.2 Повторяющиеся и несущественные критерии

Критерий корреляционного иммунитета в основном рассматривается в потоковых шифрах для защиты от корреляционных атак [112, 154]. В [172, 173] описана χ^2 атака на шифры RC-5 и RC-6. До сегодняшнего дня не была показана ни одна атака, применимая к БСШ, основанная на подстановках, не удовлетворяющим данному критерию.

Нелинейность векторных булевых функций непосредственно связана с максимальным значением аппроксимационной таблицы как $NL(F) = 2^{n-1} - \lambda$. Очевидно, что эти два критерия являются взаимозаменяемыми. Таким образом, зная значение нелинейности подстановки, можно легко вычислить значение максимума аппроксимационной таблицы и наоборот. Это свойство позволяет ускорить подсчет критериев, используя самый быстрый метод подсчета для одного из них.

Критерий распространения включает в себя строгий критерий распространения. Из уравнения 1.13 можно легко увидеть, что при $k = 1$ PC становится SAC. Таким образом, при генерации подстановки необходимо проверить её лишь на удовлетворение критерию распространения.

2.2.3 Расширенный критерий алгебраического иммунитета

Впервые упоминание об алгебраической атаке в том виде, который известен на сегодняшний день, было сделано Клодом Шеноном [38]. Алгебраическая атака активно стала развиваться в начале 2000-х годов. Было показано, что в некоторых случаях, алгебраический криптоанализ применительно как к поточные шифрам [174, 175], так и к блочным [129, 130, 133], является более эффективным, чем существующие методы. Самой последней и эффективной является атака Рёнйом-Хеллесета [176]. На

рисунке 2.2 приведено развитие алгебраического криптоанализа, где “???” означает потенциальные атаки на блочные симметричные шифры.

Отдельно стоит отметить, что решение задачи выполнимости булевых формул (SAT) [131, 132], точнее нахождение полиномиального алгоритма, приведёт к уменьшению сложности большинства алгоритмов, представленных на рисунке 2.2.

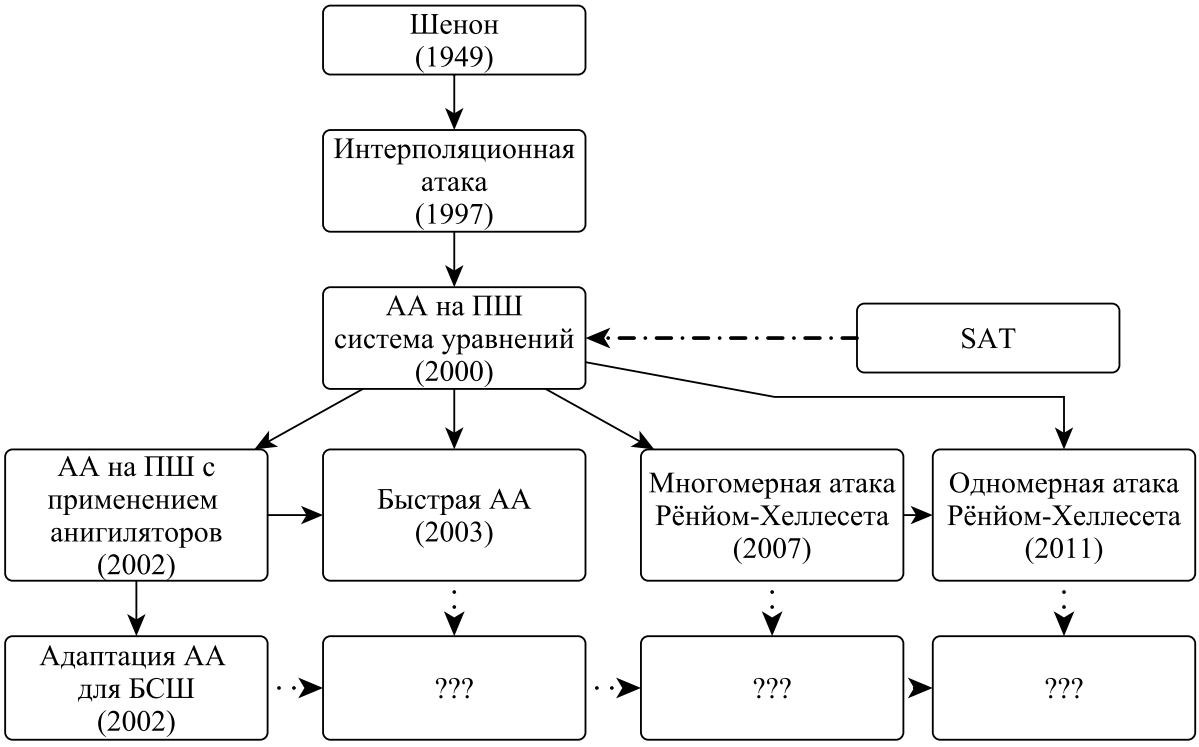

Рисунок 2.2 – Развитие алгебраического криптоанализа

Как отмечалось в 1.4.1, *S*-блок может быть представлен как набор булевых функций вида $F(x)=\left(f_1,\ldots,f_m\right)$. Однако, существует и альтернативное описание – системой уравнений, где используются все возможные произведения комбинаций входных и выходных переменных над полем *GF(2)*. При таком представлении можно получить более низкую степень термов, чем при функциональной зависимости выхода от входа. В работе [37] показаны принципы построения такой системы для случая, когда

степень каждого из термов не превышает вторую. В общем случае для описания *S*-блока можно использовать произвольную степень [72].

Для поиска системы уравнений можно воспользоваться алгоритмом, основанным на построении матрицы, описывающей все возможные значения термов для всех вариантов комбинаций входных переменных *S*-блока [72, 134].

Пусть X – значение, которое подаётся на вход *S*-блока, а Y – значение на выходе *S*-блока. Тогда строка матрицы A, необходимой для формирования системы уравнений, будет включать в себя все возможные сочетания битов X и Y до максимальной степени термов системы и константу 1. Так, для нахождения матрицы, описывающей *S*-блок, с термами не выше третей степени, строка будет содержать все возможные произведения комбинаций входных и выходных переменных третей степени, второй степени, биты входа, выхода *S*-блока и константу 1.

Размерность матрицы, описывающей *S*-блок с n битами на входе и m битами на выходе, равна:

$$|A| = \left(2^n\right) \times \left(N_C\right); \tag{2.7}$$

$$N_C = \sum_{i=0}^{d} C_{n+m}^{i}, \tag{2.8}$$

где C_g^h – число сочетаний из g элементов по h;

d – максимальная степень искомой системы.

Системой уравнений (если такая существует), описывающей *S*-блок, является ядро матрицы A [67, 68].

В работе [35] показано, что необходимое количество уравнений r, при помощи которых можно описать произвольную подстановку при $N_C > 2^n$, находится по формуле:

$$r \geq N_C - 2^n. \tag{2.9}$$

Однако формула (2.9) описывает лишь частный случай. Из теории матриц известно, что:

$$N_C = \#\left|NullSpace(A)\right| + Rank(A), \tag{2.10}$$

где N_C – количество столбцов матрицы A;

$\#\left|NullSpace(A)\right|$ – количество столбцов ядра матрицы A;

$Rank(A)$ – ранг матрицы A.

Таким образом, в общем виде r находится по следующей формуле:

$$r = N_C - Rank(A) = \sum_{i=0}^{d} C_{n+m}^{i} - Rank(A). \tag{2.11}$$

Для того, чтобы найти минимальную степень матрицы (системы) d_{min}, при которой описываются все подстановки из n в m, необходимо, чтобы выполнялось условие:

$$N_C > Rank(A). \tag{2.12}$$

Так как максимальный ранг матрицы $A = 2^n$, то d_{min} находится из соотношения:

$$\left(\sum_{i=0}^{d_{\min}} C_{2k}^{i}\right) > 2^n. \tag{2.13}$$

Говорят, что S-блок криптографически стойкий, если $Rank(A) = 2^n$. Из (2.11) получаем, что если $Rank(A) = 2^n$, тогда d имеет максимальное значение. Стоит отметить, что данные результаты были получены независимо от [36].

Увеличение степени d до максимального значения приводит к увеличению размерности системы, и следовательно увеличению необходимой памяти и времени на обработку системы уравнений. Исходя из этого, предлагается расширить критерий выбора подстановок для случая, когда S-блоки обладают одинаковым алгебраическим иммунитетом [51].

Определим разрежённость системы как отношение количества ненулевых элементов в системе к их максимальному количеству. Тогда, лучшую защиту от алгебраической атаки обеспечивает подстановка, система которой:

а) имеет более высокую степень;

б) является менее разреженной (количество ненулевых термов в системе больше) в каноническом представлении;

в) обладает меньшим количеством уравнений.

Стоит отметить, что каноническое представление системы уравнений эквивалентно матрице приведенного ступенчатого вида по строкам [132].

Эти три показателя не исключают полностью алгебраическую атаку, однако позволяют значительно усложнить её. Увеличение степени (d) уравнений на единицу влечёт за собой и увеличение размерности системы на C_{n+m}^{d+1}, а следовательно необходимость в большей производительности и требуемого объёма памяти для обработки системы уравнений. Сложность решения конечной системы зависит от её разрежённости (см. пункт 1.3.3), соответственно, уменьшение разрежённости ведёт к усложнению криптоанализа [132].

Рассмотрим *S*-блоки распространённых симметричных блочных алгоритмов, в том числе представленных на открытый национальный конкурс в Украине [13, 20, 30, 40]:

- «Калина» (*S*-блоки S0-S7);
- «Лабиринт» (Л);
- Camellia (*S*-блоки S1-S4, обозначенные C1-C4);
- AES/Rijndael.

Для каждого *S*-блока были построены переопределенные системы уравнений 2 и 3 степеней. Результаты расчёта показателей алгебраических свойств приведены в таблице 2.2.

Таблица 2.2 – Результаты расчёта показателей алгебраических свойств различных подстановок

S-блок	Всего возможно термов	Число ненул. термов в системе	Разреженность, %	Кол-во свобод. членов	Число термов 1-й степени	Число термов 2-й степени	Число термов 3-й степени
S0	307377	56939	0.815	219	3552	26537	26631
S1	307377	57066	0.814	214	3529	26681	26642
S2	307377	56657	0.816	230	3491	26284	26652
S3	307377	56978	0.815	230	3508	26432	26808
S4	307377	56992	0.815	217	3531	26402	26842
S5	307377	56996	0.815	227	3551	26444	26774
S6	307377	56736	0.815	221	3544	26438	26533
S7	307377	56868	0.815	225	3536	26451	26656
Л	5343	1625	0.696	16	316	1293	-
	328287	42262	0.871	202	3244	16363	22453
C1	5343	1615	0.698	18	317	1280	-
	328287	43495	0.868	239	3548	17152	22556
C2	5343	1647	0.692	16	333	1298	-
	328287	43837	0.866	242	3622	17388	22585
C3	5343	1656	0.690	22	313	1321	-
	328287	43400	0.868	203	3535	17023	22639
C4	5343	1676	0.686	15	326	1335	-
	328287	44325	0.865	220	3684	17642	22779
AES	5343	1661	0.689	16	303	1342	-
	328287	42219	0.871	194	3181	16342	22502

Для *S*-блоков, применяемых в шифре «Калина», минимальная степень системы, которая описывает подстановку, равна 3 (теоретический

максимум). Для алгоритмов «Лабиринт», Camellia и AES/Rijndael минимальная степень системы равна 2, но также возможно построение системы 3-й степени.

Как следует из таблицы 2.2 наибольший уровень защищённости симметричного криптографического алгоритма будет обеспечивается *S*-блоками алгоритма «Калина» (по показателями степени системы и наименьшей части нулевых термов в системе). Среди *S*-блоков шифра «Калина» лучшим с точки зрения алгебраического критерия является S1 (с минимальным отрывом от других подстановок этого алгоритма).

2.2.4 Критерий отсутствия фиксированных точек

Одним из критериев при построении подстановок шифра AES было отсутствие фиксированных и обратных фиксированных точек [15]. Разработчики современных шифров пытаются учесть данный критерий при построении шифров. Далее будет показано, что данный критерий не может рассматриваться независимо от других преобразований блочных симметричных шифров. Дополнительно, приводится пример достижения фиксированных точек на примере алгоритма шифрования AES, который основан на подстановочно-перестановочной сети.

В работах [177, 178] был приведён общий метод построения и описания изоморфных шифров. Очевидно, что алгоритмы шифрования и основных преобразований могут иметь множество представлений (изоморфизмов). Шифр BES является хорошо известным примером изоморфного AES [131].

Как было отмечено в пункте 1.2.2, большинство современных блочных симметричных шифров основаны на итеративной процедуре. AES является одним из таких шифров, с длиной обрабатываемого блока равного 128 бит и с 128, 192 или 256-битными длинами ключа [20]. Количество циклов зависит

от размера ключа и равно 10, 12 или 14 соответственно. Итеративная процедура состоит из 4-х основных функций: AddroundKey (σ_k), SubBytes (γ), ShiftRows (π) и MixColumns (θ).

Весь алгоритм шифрования имеет следующий вид (рис. 2.3):

$$E_K(M) = \sigma_{k_{r+1}} \circ \pi \circ \gamma \circ \prod_{i=2}^{r} \left(\sigma_{k_i} \circ \theta \circ \pi \circ \gamma \right) \circ \sigma_{k_1}(M). \qquad (2.14)$$

В SubBytes преобразовании состояние шифра обрабатывается побайтно и независимо друг от друга. Подстановка, используемая в преобразовании, была сгенерирована на основе нахождения обратного элемента в поле $\mathcal{F}_{2^8}$ с последующим применением аффинного преобразования [39]. В терминах уравнения (1.25) преобразование имеет следующий вид:

$$F = A_1\left(x^{-1}\right) = L_1(x^{-1}) + c_1, \qquad (2.15)$$

где все операции выполняются в поле $\mathcal{F}_{2^8}$ с неприводимым полиномом $f(x) = x^8 + x^4 + x^3 + x + 1$.

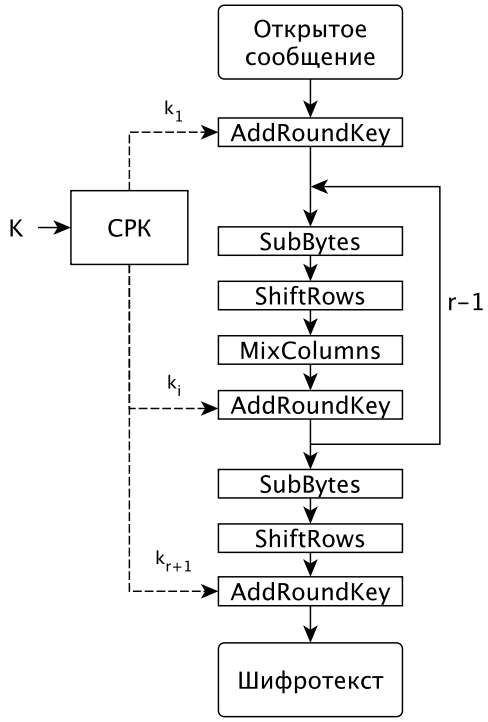

Рисунок 2.3 – Общее представление алгоритма шифрования AES

Подстановка, генерируемая векторной булевой функцией $F: \mathcal{F}_{2^8} \to \mathcal{F}_{2^8}$, обладает следующим свойствами:

а) максимальная вероятность нетривиальной XOR разницы равна 2^{-6};

б) максимальная вероятность абсолютного значения линейной аппроксимации равна 2^{-4};

в) минимальная степень равна 7 [32, 39].

Стоит отметить, что выбранный полином x^{-1} позволяет описать S-блок и весь алгоритм шифрования при помощи переопределённой системы уравнений 2-й степени [35-37, 131]. В тоже время БСШ устойчив к дифференциальному, линейному и другим статистическим методам криптоанализа. В дополнение к общим свойствам, подстановка шифра AES была выбрана таким образом, чтобы в ней отсутствовали фиксированные точки.

Преобразование MixColumns представляет собой умножение столбца состояния на фиксированную матрицу M. Каждый из 4-х столбцов матрицы обрабатывается независимо друг от друга [15]:

$$y = M \cdot x, \tag{2.16}$$

где y – выходное состояния после преобразования;

x – входное состояние (столбец матрицы).

В AES матрица M обладает МДР свойством [15], которое связано с количеством ветвлений (β):

$$\beta = \min_{x \neq 0} \left(W(x) + W(y) \right), \tag{2.17}$$

где $W(z)$ – вес вектора z в байтовом представлении.

Известно, что $m \times m$ МДР матрица обладает максимальным количеством ветвлений ($m+1$) [126, 179]. Поэтому они обладают совершенными рассеивающим свойством и часто используются в линейном слое.

Умножение в поле $\mathcal{F}_{2^n}$ является линейным преобразованием относительно операции XOR, поэтому оно сохраняет свойство линейности [180]:

$$\theta(x+y)=\theta(y)+\theta(y) \tag{2.18}$$

ShiftRows преобразование обрабатывает текущее состояние, используя сдвиг последних трёх строк на различное количество позиций [15]. Более детально, *i*-я строка сдвигается на i байт для $0 \le i \le 3$. Как и MixColums, преобразование ShiftRows является линейной функцией, которая сохраняет $\pi(x+y)=\pi(y)+\pi(y)$ свойство.

Оба линейных преобразования позволяют добиться максимального количества активных подстановок после нескольких раундов шифрования [39]. Эти функции лежат в основе защиты шифра AES от дифференциального и линейного криптоанализов.

В AES функция смешивания (AddRoudKey) задаётся операцией XOR. Длина циклового ключа равна размеру текущего состояния. Операция XOR может быть выполнена над каждым битом независимо друг от друга. Поэтому AddRoudKey может быть применена к каждому из байтов текущего состояния независимо.

Выше было упомянуто, что алгоритм шифрования имеет различные представления. Один из подходов основан на линейных свойствах основных функций (MixColumns и ShiftRows).

При построении шифра AES авторы использовали принцип простоты, который даёт возможность увеличить производительность и компактность кода на большинстве современных платформ. Для увеличения производительности при программной реализации использовались таблицы предвычислений и линейные свойства базовых функций.

Алгоритм расшифрования для произвольно шифротекста C имеет следующий вид (рис. 2.4) [15]:

$$D_K(C) = \sigma_{k_1} \circ \gamma^{-1} \circ \pi^{-1} \circ \prod_{i=2}^{r} \left(\theta^{-1} \circ \sigma_{k_{r-i+2}}^{-1} \circ \gamma^{-1} \circ \pi^{-1} \right) \circ \sigma_{k_{r+1}}(C) \qquad (2.19)$$

Для того, чтобы использовать таблицы предвычислений необходимо сделать процедуру расшифрования похожей на алгоритм шифрования. Из-за того, что функции γ^{-1} и π^{-1} вычисляются независимо, они обладают коммутативным свойством $\gamma^{-1} \circ \pi^{-1} = \pi^{-1} \circ \gamma^{-1}$ [180]. Аналогичным свойством обладают функции θ^{-1} и σ:

$$\theta^{-1} \circ \sigma_{k_{r-i+2}} = \sigma_{\theta^{-1}(k_{r-i+2})} \circ \theta^{-1}, \qquad (2.20)$$

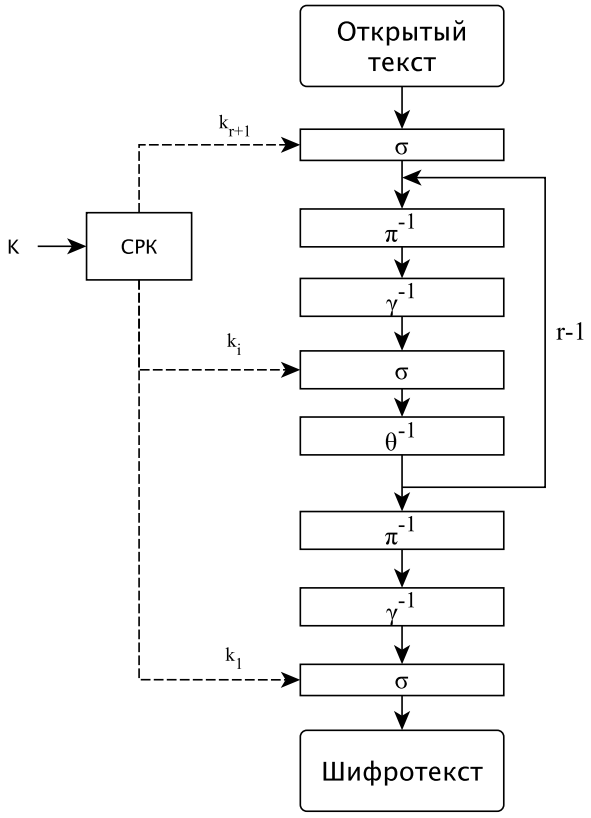

Рисунок 2.4 – Алгоритм расшифрования AES

Таким образом, весь алгоритм расшифрования можно представить в следующем виде (рис. 2.5):

$$D_K(C) = \sigma_{k_1} \circ \pi^{-1} \circ \gamma^{-1} \circ \prod_{i=2}^{r} \left(\sigma_{\theta^{-1}(k_{r-i+2})}^{-1} \circ \theta^{-1} \circ \pi^{-1} \circ \gamma^{-1} \right) \circ \sigma_{k_{r+1}}(C) \qquad (2.21)$$

Элементарные модификации криптоалгоритма позволяют значительно увеличить производительность процедуры расшифрования благодаря изоморфным свойствам базовых функций [15].

Очевидно, что такая же техника может быть применена и к алгоритму шифрования. Однако, основной задачей становится нахождение представления шифра, в котором подстановка будет иметь фиксированную точку. Для упрощения представления предположим, что все подключи (цикловые ключи) являются независимыми.

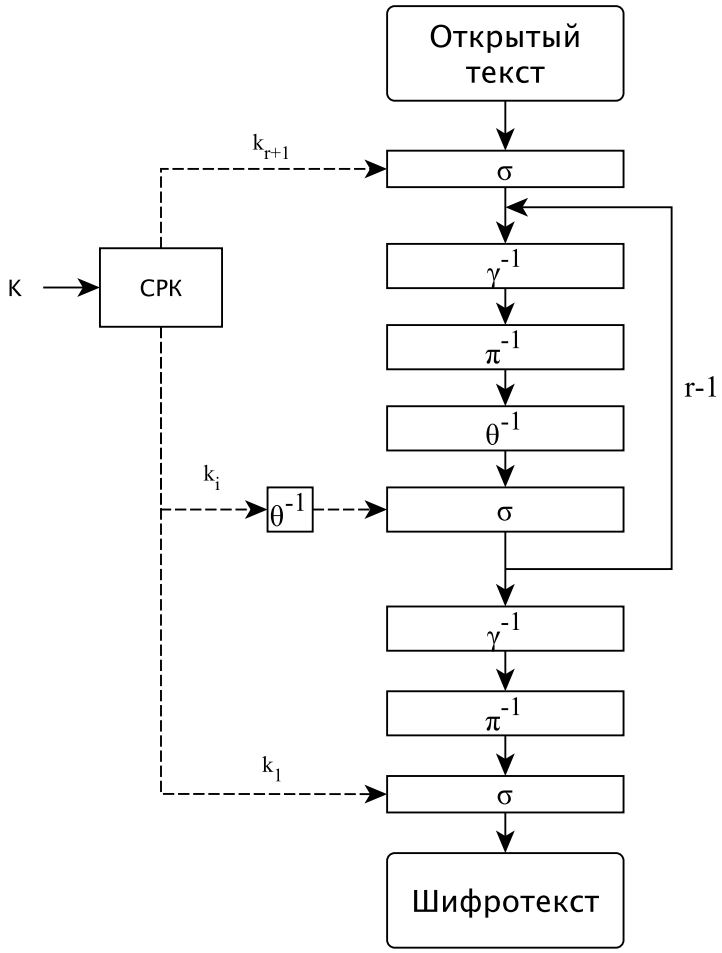

Рисунок 2.5 – Модифицированная процедура расшифрования AES

Для начала перепишем уравнение (2.14) в форме (рис. 2.6):

$$E_K(M) = \pi \circ \sigma_{\pi^{-1}(k_{r+1})} \circ \gamma \circ \prod_{i=2}^{r} \left(\theta \circ \pi \circ \sigma_{\pi^{-1} \circ \theta^{-1}(k_i)} \circ \gamma \right) \circ \sigma_{k_1}(M). \qquad (2.22)$$

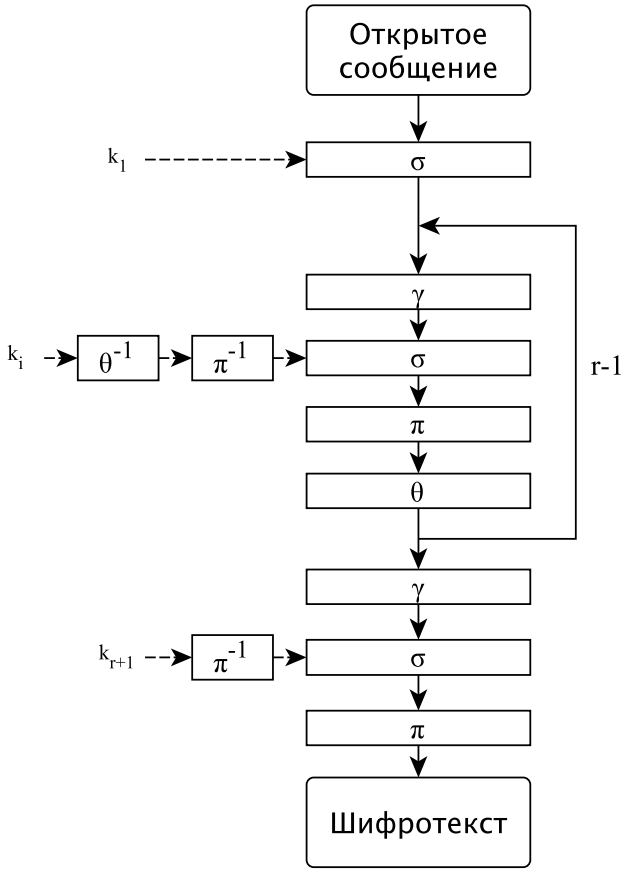

Рисунок 2.6 – Модифицированный алгоритм шифрования AES

Из уравнения видно, что последнее преобразование ShiftRows является избыточным. Как было показано выше, наличие данной функции необходимо для увеличения производительности алгоритма расшифрования.

Из [50] известно, что произвольная подстановка S может быть представлена в виде векторной булевой функции $F : \mathcal{F}_{2^n} \to \mathcal{F}_{2^m}$

$$F(x) = F'(x) + F(0) \tag{2.23}$$

Вследствие того, что характеристика поля равна 2, константа может быть перенесена в цикловой ключ. Пусть ξ - функция, в которой константа из подстановки AES [15] побитово складывается с каждым байтом циклового ключа. Обозначим через $k_i^{'}$ преобразование $\pi^{-1} \circ \theta^{-1} \circ \xi(k_i)$. Тогда процедура шифрования примет вид (рис. 2.7):

$$E_K(M) = \pi \circ \sigma_{\pi^{-1} \circ \xi(k_{r+1})} \circ \gamma' \circ \prod_{i=2}^{r} \left(\theta \circ \pi \circ \sigma_{k_i^{'}} \circ \gamma' \right) \circ \sigma_{k_1}(M), \tag{2.24}$$

где γ' – SubBytes функция, использующая функцию $F(x) = L(x^{-1})$ (см. формулу 2.15) в качестве подстановки.

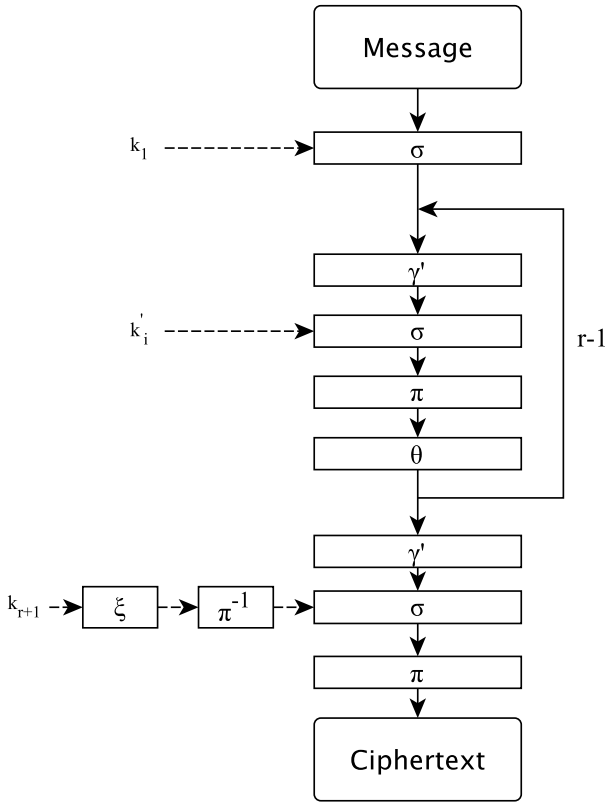

Рисунок 2.7 – Модифицированный алгоритм шифрования AES с подстановкой обладающей фиксированной точкой

Из рисунка 2.7 видно, что структура шифра практически не изменилась. Очевидно, что если злоумышленник сможет найти ключи в модифицированном шифре, то он автоматически получит соответствующие ключи в оригинальном шифре. Это свойство сохраняется из-за линейной зависимости ключей k_i' и k. Стоит отметить, что подстановка $F(x) = L(x^{-1})$ обладает фиксированной точкой при $x = 0$. Соответственно, подстановка не удовлетворяет требованиям, предъявляемым к подстановке из [15].

Описанные особенности шифра возникают из-за линейных свойств операций XOR, MixColumns и ShiftRows. Если заменить функцию смешивания на некоторую нелинейную (например, сложение по модулю 2^n), тогда будет невозможно найти изоморфный шифр такого вида. Таким

образом, функция смешивания, основанная на сложении по модулю 2^n, более криптографически стойкая, чем операция XOR.

Однако, критерий отсутствия фиксированных точек используется в большинстве шифров для защиты от статистических атак [131]. Данного критерия можно легко достичь при помощи аффинной эквивалентности [49].

2.2.5 Предложенные свойства оптимальной подстановки

На сегодняшний день нет однозначного набора критериев для идеального S-блока. Поэтому, на основании проведённого исследования было предложено ориентироваться на достижение предельных показателей δ -равномерности, нелинейности, алгебраического иммунитета и минимальной степени в процессе формирования подстановок, используемых в блочных симметричных шифрах.

Определение 2.1. Подстановка является оптимальной, если достигнута совокупность предельных значений показателей, известных на текущий момент, которая определяет стойкость симметричного преобразования к методам дифференциального, линейного и алгебраического криптоанализа.

В данной работе под оптимальной подстановкой понимается перестановка с максимальными показателями минимальной степени и алгебраического иммунитета; с предельными показателями (при фиксированных предыдущих) δ -равномерности и нелинейности; отсутствием фиксированных точек (циклов длиной 1).

Например, для $n = 8$ оптимальная подстановка будет иметь минимальную степень 7, алгебраический иммунитет 3 (441 уравнение), δ-равномерность равную 8 или меньше, нелинейность не меньше 104 и не иметь фиксированных точек.

2.2.6 Предложенный критерий для нескольких *S*-блоков

Очевидно, что техника, применяемая в 2.2.4, может быть расширена с использованием обратимых линейных функций. Пусть $\boldsymbol{\tau}$ – произвольная обратимая линейная (аффинная) функция. Применение данной функции, как показано на рисунке 2.8, приводит к новому изоморфному шифру, при этом значения шифротекста не меняются.

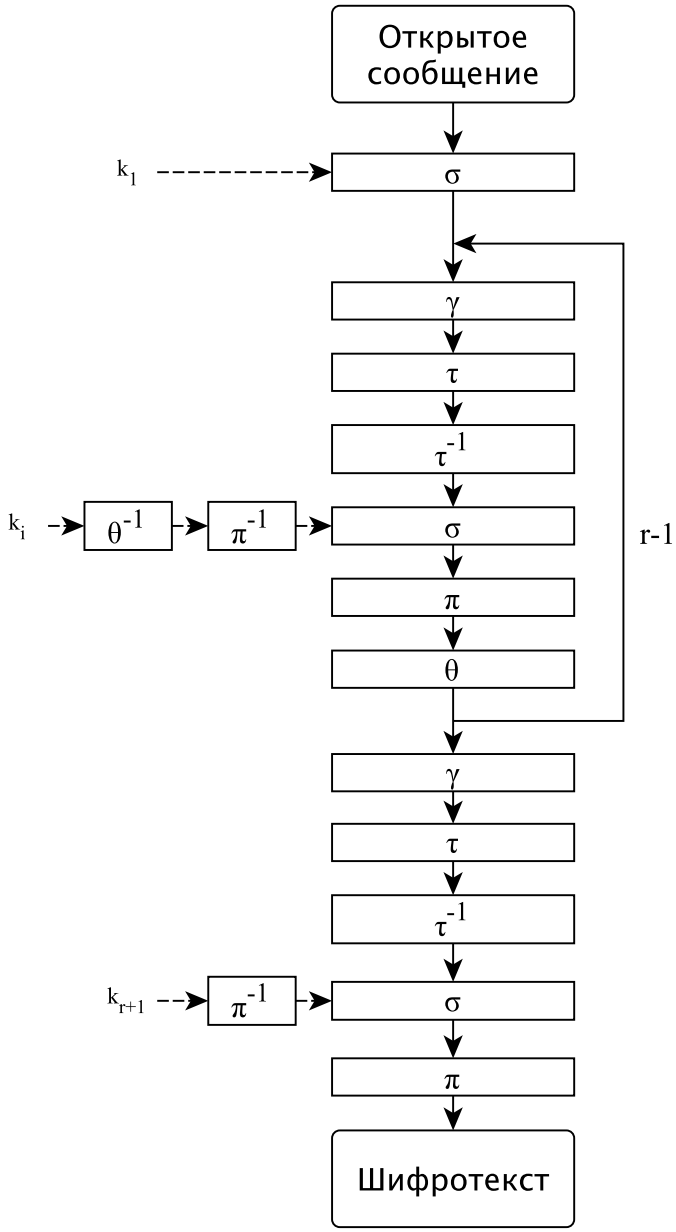

Рисунок 2.8 – Изоморфный алгоритм шифрования AES с дополнительным линейным слоем

Более того, τ может быть объединена с нелинейным слоем, а τ^{-1} перенесена в цикловой ключ и объединена с π функцией (рис. 2.9). При таком представлении, цикловые свойства подстановки будут зависеть от выбранной функции τ.

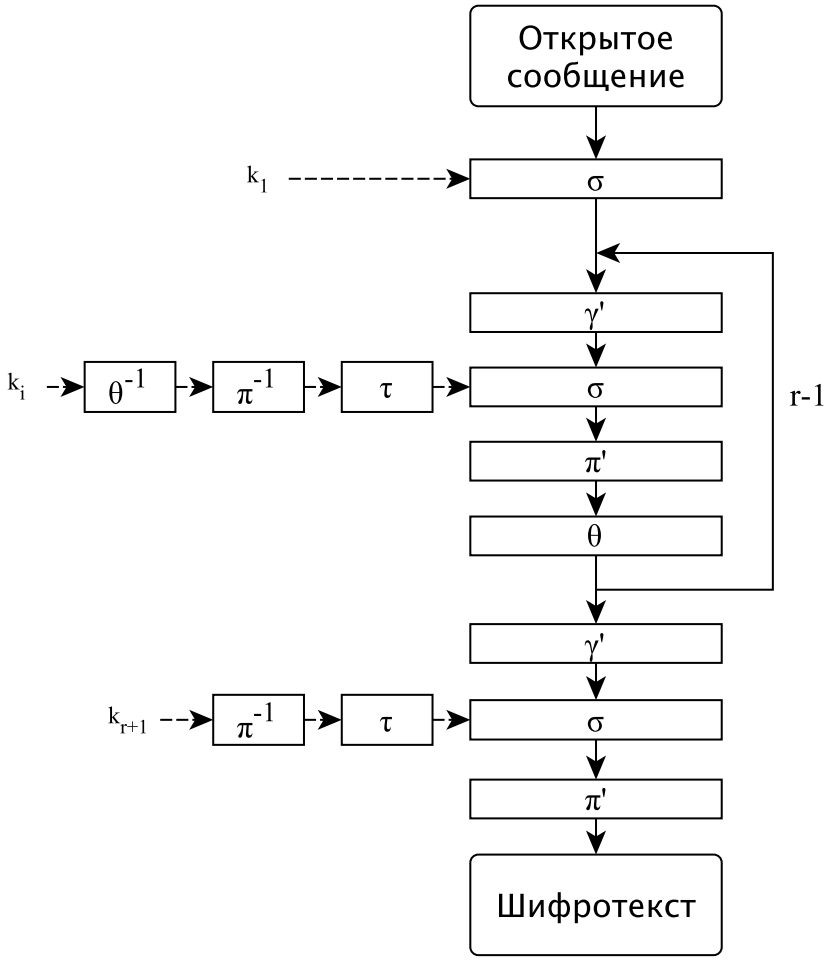

Рисунок 2.9 – Изоморфный алгоритм шифрования AES с модифицированной подстановкой

Таким образом, злоумышленник, в случае линейной функции смешивания, может контролировать отсутствие фиксированных точек и циклические свойства подстановки, что приводит к следующему критерию.

Утверждение 2.1. Используемые в нелинейном слое подстановки $S_1, S_2, \ldots, S_k$ ($k \geq 2$) *должны принадлежать различным классам эквивалентности.*

Очевидно, что если подстановки принадлежат к одному классу (например, РА-эквивалентные), то злоумышленник может найти изоморфный шифр, который состоит из одной подстановки и модифицированного линейного слоя. Следовательно, не будет никаких преимуществ использования множества подстановок. Предложенный критерий может применяться как для новых шифров, так и для анализа уже существующих [29, 41]. Так как КШЗ-эквивалентность является наиболее общей на сегодняшний день, то есть смысл проверять подстановки на принадлежность к различным КШЗ-эквивалентным классам.

Некоторые полиномиальные алгоритмы для проверки эквивалентностей представлены в следующем подразделе [50].

2.3 Метод проверки векторных булевых функций на эквивалентность

После нахождении одной или нескольких новых векторных булевых функций часто возникает вопрос о их принадлежности к уже существующим классам эквивалентности. Эта же задача может быть адаптирована для симметричных криптоалгоритмов: при генерации нескольких подстановок для алгоритма шифрования необходимо проверить их на эквивалентность [48].

Сложность полного перебора проверки двух функций из $\mathcal{F}_2^n$ в $\mathcal{F}_2^n$ на принадлежность одному классу РА-эквивалентности равна $O(2^{3n^2+2n})$. Даже при $n=6$ эта сложность равна 2^{120}, что приводит к невозможности полного перебора на существующих электронных вычислительных машинах (ЭВМ).

Авторы статьи [181] показали, что в случае, когда векторные булевы функции являются перестановками в $\mathcal{F}_2^n[x]$, сложность проверки на частично расширенную аффинную эквивалентность (ЧРА) равна $O(n^2 \cdot 2^n)$ для случая линейной эквивалентности и $O(n \cdot 2^{2n})$ при аффинной эквивалентности.

Ниже приведены более общие случаи ЧРА-эквивалентности для функций из $\mathcal{F}_2^n$ в $\mathcal{F}_2^m$ и рассмотрены условия, при которых сложность проверки может быть уменьшена до полиноминальной.

Произвольная аффинная функция $A:\mathcal{F}_2^n \mapsto \mathcal{F}_2^m$ может быть представлена в матричном виде:

$$A(x) = M \cdot x \oplus C, \tag{2.25}$$

где M – некоторая $m \times n$ матрица;

$C \in \mathcal{F}_2^m$.

Все операции проводятся в $\mathcal{F}_2$, поэтому (2.25) можно переписать в виде:

$$\begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{m-1} \end{pmatrix}_x = \begin{pmatrix} k_{0,0} & \cdots & k_{0,n-1} \\ k_{1,0} & \cdots & k_{1,n-1} \\ \vdots & \ddots & \vdots \\ k_{m-1,0} & \cdots & k_{m-1,n-1} \end{pmatrix} . \begin{pmatrix} x_0 \\ x_1 \\ \cdots \\ x_{n-1} \end{pmatrix} \oplus \begin{pmatrix} c_0 \\ c_1 \\ \cdots \\ c_{m-1} \end{pmatrix}, \tag{2.26}$$

где $a_i, x_i, c_i, k_{j,s} \in \mathcal{F}_2$.

Таким образом, РА-эквивалентность (см. уравнение (1.25)) в матричном представлении записывается как:

$$F(x) = M_1 \cdot G\left(M_2 \cdot x \oplus V_2\right) \oplus M_3 \cdot x \oplus V_1, \tag{2.27}$$

где элементы $\{M_1, M_2, M_3, V_1, V_2\}$ имеют размерности $\{m \times m, n \times n, m \times n, m, n\}$.

Различные типы ЧРА эквивалентностей, рассматриваемые в этом пункте, представлены в таблице 2.3.

Таблица 2.3 – Рассматриваемые типы ЧРА-эквивалентности

ЧРА-эквивалентность	Тип
$F(x) = M_1 \cdot G(x) \oplus V_1$	I
$F(x) = G(M_2 \cdot x \oplus V_2)$	II
$F(x) = G(x) \oplus M_3 \cdot x \oplus V_1$	III
$F(x) = M_1 \cdot G(x) \oplus M_3 \cdot x \oplus V_1$	IV

Отсюда и далее предполагается, что сложность вычисления значения $F(x)$ равна $O(1)$. Функций $F(x), F^{-1}(x)$ и соответствующие им подстановки являются входными значениями алгоритма. Дополнительно не учитывается сложность представления функции в необходимой форме и место, необходимое для хранения всех значений. Эти допущения вводятся для того, чтобы иметь возможность сравнивать сложности алгоритмов с результатами работы [181], где были сделаны те же допущения.

Существует $2^{n \cdot m}$ различных вариантов линейного отображения. Сложность нахождения $m \times n$ матрицы M удовлетворяющей уравнению

$$F(x) = M \cdot G(x), \tag{2.28}$$

используя метод полного перебора равна $O(2^n \cdot 2^{m \cdot n})$, где $O(2^{m \cdot n})$ и $O(2^n)$ сложности проверки всех матриц для всех значений $x \in \mathcal{F}_2^n$ соответственно.

Очевидно, что для нахождения хотя бы одного решения (2.28) можно воспользоваться методом Гаусса или Уильямса [131, 182]. Сложность последнего равна $O(s^{2.3727})$, где $s = \max\{n, m\}$. Более того, для $s \le 64$ можно использовать 64 битовые инструкции процессора, что приводит к сложности $O(s^2)$ (две строки матрицы складываются за один шаг) [183, 184]. Так как любая система с *m* уравнениями и *n* переменными может быть рассмотрена как система с *s* уравнениями и *s* переменными, то сложность решения такой системы равна

$$\mu = O(s^2). \tag{2.29}$$

Утверждение 2.2. Любая линейная функция $L : \mathcal{F}_2^n \mapsto \mathcal{F}_2^m$ может быть преобразована в матрицу со сложностью $O(n)$.

Доказательство. Необходимо найти $m \times n$ матрицу *M,* удовлетворяющую уравнению $L(x) = M \cdot x$ для всех $x \in \mathcal{F}_2^n$.

Пусть $\text{rows}_M(i) = (k_{ij}), \forall j \in \{0,1,\ldots,n-1\}$ и $cols_M(j) = (k_{ij}), \forall i \in \{0,1,\ldots,m-1\}$ – значения i-й строки и j-го столбца матрицы M соответственно.

Каждое значение $x \in \{2^i \mid 0 \le i \le n-1\}$ эквивалентно вектору со значением 1 в i-й строке

$$2^0 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad 2^1 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad \cdots \quad 2^{n-1} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}. \tag{2.30}$$

Очевидно, что каждый столбец, за исключением i-го, матрицы M обнуляется при умножении на x. Поэтому матрица M может быть найдена при помощи уравнения:

$$\text{cols}_M(i) = L(2^i), \quad i \in \{0,1,\ldots,n-1\}. \tag{2.31}$$

Сложность такого преобразования равна $O(n)$, так как для нахождения всей матрицы необходимо вычислить n значений функции $L(2^i)$, $0 \le i \le n-1$. □

Утверждение 2.3. Любая $n \times n$ матрица M может быть преобразована в линейную функцию $L : \mathcal{F}_2^n \mapsto \mathcal{F}_2^n$ со сложностью $O(n^3)$ операций в поле.

Доказательство. Любая линейная функция имеет следующий вид:

$$M \cdot x = L(x) = \sum_{i=0}^{n-1} \delta_i x^{2^i}, \tag{2.32}$$

где $\delta_i \in \mathcal{F}_{2^n}$. Тогда, подставляя $x = 2^i, 0 \le i \le n-1$, в (2.32), уравнение может быть переписано в виде:

$$\begin{pmatrix} \text{cols}_M(0) \\ \text{cols}_M(1) \\ \vdots \\ \text{cols}_M(n-1) \end{pmatrix} = \begin{pmatrix} (2^0)^{2^0} & (2^0)^{2^1} & \cdots & (2^0)^{2^{n-1}} \\ (2^1)^{2^0} & (2^1)^{2^1} & \cdots & (2^1)^{2^{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ (2^{n-1})^{2^0} & (2^{n-1})^{2^1} & \cdots & (2^{n-1})^{2^{n-1}} \end{pmatrix} \times \begin{pmatrix} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_{n-1} \end{pmatrix}. \tag{2.33}$$

Сложность нахождения вектора $\{\delta_0,\delta_1,\ldots,\delta_{n-1}\}$ равна сложности нахождения обратной матрицы, т.е. равна $O(n^3)$ операций в поле. □

На практике интерполяционная формула Лагранжа [156] работает быстрее для $n \le 6$ и гораздо медленнее для остальных n.

Утверждение 2.4. Пусть $F,G:\mathcal{F}_2^n \mapsto \mathcal{F}_2^m$ *и* $G'(x)=G(x)\oplus G(0)$. *Тогда сложность проверки* F *и* G *на ЧРА–эквивалентность типа I равна:*

а) $O(2^{n+1})$ *для случая, когда для всех* $i\in\{0,\ldots,m-1\}$ *существует* $x\in\mathcal{F}_2^n$ *такой, что* $G'(x)=2^i$;

б) $O(m\cdot 2^{2n})$ *для произвольной функции* G.

Доказательство. Пусть $F'(x)=F(x)\oplus F(0)$. Тогда ЧРА–эквивалентность I типа

$$F'(x)\oplus F(0)=M_1\cdot G'(x)\oplus M_1\cdot G(0)\oplus V_1 \qquad (2.34)$$

можно переписать в следующей форме:

$$\begin{cases} F(0)=M_1\cdot G(0)\oplus V_1; \\ F'(x)=M_1\cdot G'(x). \end{cases} \qquad (2.35)$$

Для случая $G(0)=0$, V_1 равно $F(0)$. Однако, в общем виде сначала необходимо найти M_1 из уравнения $F'(x)=M_1\cdot G'(x)$. Если множество $\{2^i \mid 0\le i\le m-1\}$ является подмножеством множества значений функции G', тогда проблема нахождения $m\times m$ матрицы M_1 эквивалентна проблеме преобразования линейной функции в матричный вид с дополнительной проверкой всех значений $x\in\mathcal{F}_2^n$.

Из утверждения 2.2 следует, что сложность нахождения M_1 равна $O(m)$. Сложности нахождения прообразов G' элементов 2^i, $\forall i\in\{0,\ldots,m-1\}$ и проверки $F'(x)=M_1\cdot G'(x)$, $\forall x\in\mathcal{F}_2^n$ равны $O(2^n)$.

В криптографии, в большинстве случаев, $2^n >> m$, поэтому сложностью $O(m)$ можно пренебречь. Следовательно, общая сложность проверки F и G на ЧРА–эквивалентность равна $O(2^n + 2^n + m) \approx O(2^{n+1})$.

Рассмотрим случай, когда G – произвольная функция. Пусть $F'(x)_i$ – i-й бит векторной булевой функции $F'(x)$. Обозначим через $\mathrm{img}(G')$ множество значений функции G' , тогда $u_{G'} = \left|\mathrm{img}(G')\right|$ – количество элементов $\mathrm{img}(G')$. Пусть $N_{G'}$ – произвольное подмножество $\mathcal{F}_2^n$, такое что $\left|N_{G'}\right| = u_{G'}$ и $\left|\left\{G'(a) \,\middle|\, a \in N_{G'}\right\}\right| = u_{G'}$.

Тогда для нахождения M_1 необходимо решить систему уравнений для всех значений $i \in \{0,\ldots,m-1\}$

$$F'(x_j)_i = \mathrm{rows}_{M_1}(i) \cdot G'(x_j), \quad \forall x_j \in N_{G'}, 0 \le j \le u_{G'} - 1 \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} F'(x_0)_i = \mathrm{rows}_{M_1}(i) \cdot G'(x_0); \\ F'(x_1)_i = \mathrm{rows}_{M_1}(i) \cdot G'(x_1); \\ \ldots \\ F'(x_{u_{G'}-1})_i = \mathrm{rows}_{M_1}(i) \cdot G'(x_{u_{G'}-1}). \end{cases} \tag{2.36}$$

Сложность решения данной системы очень сильно зависит от значений $u_{G'}$ и m и равна $O(\max\{u_{G'}, m\}^2)$ в соответствии с (2.29). Следовательно, сложность нахождения M_1 для всех m бит равна $O(m \cdot \max\{u_{G'}, m\}^2)$. Если значение $u_{G'} \approx 2^n$, тогда $O(\mathrm{m} \cdot 2^{2n})$. □

Замечание 2.1. Если заранее известно, что функции F и G из Утверждения 2.4 являются ЧРА–эквивалентными, то сложность проверки $F'(x) = M_1 \cdot G'(x)$ может быть упущена, и общая сложность нахождения параметров ЧРА–эквивалентности, для случая $\left\{2^i \,\middle|\, 0 \le i \le m-1\right\} \subset img(G')$, равна $O(2^n)$.

Утверждение 2.5. Пусть G является перестановочной функцией и $F,G:\mathcal{F}_2^n \mapsto \mathcal{F}_2^n$. *Тогда сложность проверки F и G на ЧРА–эквивалентность II типа равна* $O(n^2)$.

Доказательство. Пусть $H(x) = G^{-1}(F(x))$, тогда уравнение $F(x) = G(M_2 \cdot x \oplus V_2)$ будет иметь вид:

$$H(x) = M_2 \cdot x \oplus V_2 . \qquad (2.37)$$

Очевидно, что для удовлетворения ЧРА–эквивалентности функция H должна быть линейной и обратимой.

Произвольная линейная функция из $\mathcal{F}_2^n$ в $\mathcal{F}_2^n$ имеет форму (2.32) и состоит из не более чем n мономов. Предположим, что $\psi(n)$ – сложность нахождения двоичного веса Хемминга степени монома векторной булевой функции H. Для большинства современных процессоров, включая Intel, значение $\psi(n)$ равно $O(1)$ [185]. Следовательно, общая сложность проверки функции H на линейность равна $O(n \cdot \psi(n)) = O(n)$.

При $x = 0$ значение $V_2 = H(0)$ и эквивалентность приобретает вид $H'(x) = M_2 \cdot x$, где $H'(x) = H(x) \oplus H(0)$. Чтобы удовлетворять условиям ЧРА–эквивалентности, матрица M_2 должна быть обратимой. Поэтому общая сложность проверки F и G на ЧРА-эквивалентность равна сумме проверки функции H на линейность ($O(n)$), нахождения матрицы M_2 ($O(n)$) и проверки её на обратимость ($O(n^2)$), и приблизительно равна $O(n^2)$. □

Утверждение 2.6. Пусть $F,G:\mathcal{F}_2^n \mapsto \mathcal{F}_2^m$, *тогда сложность проверки F и G на ЧРА-эквивалентность III типа равна* $O(n)$.

Доказательство. Обозначим через $H(x) = F(x) \oplus G(x)$, тогда ЧРА-эквивалентность

$$F(x) = G(x) \oplus M_3 \cdot x \oplus V_1 \qquad (2.38)$$

сводится к виду:

$$H(x) = M_3 \cdot x \oplus V_1 . \qquad (2.39)$$

Данное уравнение идентично (2.37), однако для произвольной $m \times n$ матрицы. Поэтому сложность нахождения M_3 и V_1 равна $O(n)$. □

Любая векторная булева функция H может быть представлена в форме:

$$H(x) = H'(x) \oplus L_H(x) \oplus H(0), \qquad (2.40)$$

где H' – функция, состоящая из всех алгебраических степеней больше 2;

L_H – линейная функция.

Утверждение 2.7. Пусть G' задаётся (2.40) и $F, G : \mathcal{F}_2^n \mapsto \mathcal{F}_2^m$. Тогда сложность проверки F и G на ЧРА-эквивалентность IV типа равна

- $O(2^{n+1})$ *в случае* $\left\{2^i \mid 0 \le i \le m-1\right\} \subset img(G')$,

- $O(m \cdot 2^{2n})$ *в случае произвольной* G.

Доказательство. Используя (2.40), ЧРА-эквивалентность IV типа можно представить в виде:

$$F'(x) \oplus L_F(x) \oplus F(0) = M_1 \cdot G'(x) \oplus M_1 \cdot L_G(x) \oplus M_3 \cdot x \oplus M_1 \cdot G(0) \oplus V_1,$$

что приводит к системе уравнений:

$$\begin{cases} F'(x) = M_1 \cdot G'(x) \\ L_F(x) = M_1 \cdot L_G(x) \oplus M_3 \cdot x \\ F(0) = M_1 \cdot G(0) \oplus V_1 \end{cases} \qquad (2.41)$$

Можно заметить, что при известной матрице M_1 легко находится M_3 и V_1 из уравнения 2 и 3 соответственно. Первое уравнение приводит к двум различным случаям рассмотренными в утверждении 2.4. Поэтому общая сложность нахождения M_1 равна $O(2^{n+1})$, если $\left\{2^i \mid 0 \le i \le m-1\right\} \subset img(G')$, и $O(m \cdot 2^{2n})$ в другом случае. Стоит заметить, что сложность нахождения матрицы M_3 не учитывается из-за того, что $2^{n+1} >> n$. □

Очевидно, что если добавить одно из значений V_1 или V_2 к ЧРА-эквивалентности, то сложность подсчёта увеличится в 2^m или 2^n соответственно. Сравнение сложностей с V_1, V_2 представлено в таблице 2.4, где СПП – сложность полного перебора, а СИМ – сложность известных методов.

Стоит отметить, что I и II типы ЧРА эквивалентности являются частными случаями IV типа. Однако, необходимо проверять все случаи вследствие различных условий накладываемых на функцию G.

Таблица 2.4 – Сравнение сложностей решения проблемы ЧРА эквивалентности

№	ЧРА эквивалентность	СПП	СИМ	$G(x)$
1	$F(x) = M_1 \cdot G(M_2 \cdot x)$	$O\left(\left(\prod_{i=1}^{n}(2^n - 2^{i-1})\right)^2\right)$	$O(n^2 \cdot 2^n)$	Перестановка*
2	$F(x) = M_1 \cdot G(M_2 \cdot x \oplus V_2) \oplus V_1$	$O\left(\left(\prod_{i=1}^{n}(2^n - 2^{i-1})\right)^2 \cdot 2^{2n}\right)$	$O(n \cdot 2^{2n})$	Перестановка*
3	$F(x) = M_1 \cdot G(x \oplus V_2) \oplus V_1$	$O\left(\prod_{i=1}^{m}(2^m - 2^{i-1}) \cdot 2^{n+m}\right)$	$O(2^{2n+1})$	†
4	$F(x) = M_1 \cdot G(x \oplus V_2) \oplus V_1$	$O\left(\prod_{i=1}^{m}(2^m - 2^{i-1}) \cdot 2^{n+m}\right)$	$O(m \cdot 2^{3n})$	Произвольная
5	$F(x) = G(M_2 \cdot x \oplus V_2) \oplus V_1$	$O\left(\prod_{i=1}^{n}(2^n - 2^{i-1}) \cdot 2^{n+m}\right)$	$O(n^2 \cdot 2^n)$	Перестановка
6	$F(x) = G(x \oplus V_2) \oplus M_3 \cdot x \oplus V_1$	$O\left(2^{n \cdot m+n+m}\right)$	$O(n \cdot 2^n)$	Произвольная
7	$F(x) = M_1 \cdot G(x \oplus V_2) \oplus M_3 \cdot x \oplus V_1$	$O\left(\prod_{i=1}^{m}(2^m - 2^{i-1}) \cdot 2^{n \cdot m+n+m}\right)$	$O(2^{2n+1})$	‡
8	$F(x) = M_1 \cdot G(x \oplus V_2) \oplus M_3 \cdot x \oplus V_1$	$O\left(\prod_{i=1}^{m}(2^m - 2^{i-1}) \cdot 2^{n \cdot m+n+m}\right)$	$O(m \cdot 2^{3n})$	Произвольная

* – сложности представленные в работе [181].

† – при выполнении условия $\left\{2^i \middle| 0 \le i \le m-1\right\} \subset img(G')$, где $G'(x) = G(x) + G(0)$.

‡ – при выполнении условия $\left\{2^i \middle| 0 \le i \le m-1\right\} \subset img(G')$, где $G'(x)$ определена как (2.40).

Как видно из таблицы 2.4, сложности предложенных методов нельзя напрямую сравнивать с уже известными. Поэтому, в таблице 2.5 представлен расчёт сложностей (в логарифмическом виде с основанием 2) для часто используемых значений n.

Таблица 2.5 – Сравнение сложностей решения проблемы ЧРА эквивалентности для часто используемых значений n

№ ЧРА эквивалентности	$n=6$		$n=8$		$n=10$		$n=12$		$n=14$	
	СПП	СИМ	СПП	СИМ	СПП	СИМ	СПП	СИМ	СПП	СИМ
1	69	12	125	14	197	17	285	20	389	22
2	81	15	141	19	217	24	309	28	417	32
3	47	13	79	17	119	21	167	25	223	29
4		21		27		34		40		46
5		12		14		17		20		22
6	48	9	80	11	120	14	168	16	224	18
7	83	13	143	17	219	21	311	25	419	29
8		21		27		34		40		46

2.4 Выводы

В ходе исследований было проанализировано множество критериев для подстановок, применяемых в симметричных криптоалгоритмах, и показано, что большинство из них не применимы к блочным симметричным шифрам, а играют роль лишь в поточных шифрах при использовании векторных булевых функций. Свойства подстановок, которые разрабатывались для

БСШ DES и ГОСТ 28147 более 20 лет назад, на сегодняшний день не являются актуальными. Современные критерии ориентированы на защиту от существующих видов криптоанализа: линейного, алгебраического и различных вариаций дифференциального.

Многие исследования, включая проведённые в рамках данной работы, показывают, что идеальных подстановок, вероятнее всего, не существует. Поэтому, вводится термин «оптимальная подстановка», критерии которой определяются для конкретного алгоритма шифрования (или группы алгоритмов) и являются оптимальными с точки зрения защиты от существующих видов атак.

Эквивалентность векторных булевых функций позволяет находить различные варианты представления отображений. В связи с чем, к множеству уже существующих критериев был добавлен новый, связанный с принадлежностью подстановок к различным классам эквивалентности. Данный критерий применим лишь в том случае, когда в шифре используется более одного узла нелинейной замены. В следствии чего возникает необходимость нахождения методов проверки нескольких векторных булевых функций на различные виды эквивалентности.

Основываясь на разработанном полиномиальном методе преобразования линейных функций, заданных над полем, в матричное представление, были предложены несколько методов проверки функций на частичную РА эквивалентность, сложность которых, при определённых условиях, равна полиномиальной.

Кроме того, модификация метода преобразования линейных функций позволяет восстанавливать высокоуровневые конструкции симметричных криптопреобразований. В частности, в [80] было показано, что базовое преобразование хэш-функции ГОСТ Р 34.11-2012, заданное в алгоритмическом виде, спроектировано на основе шифра Rijndael, с применением теории полей и кодирования.

Таким образом, приведённые методы не только позволяют решать практические задачи, но и являются дополнительными инструментами для развития теории векторных булевых функций.

3 АНАЛИЗ ПУТЕЙ СОВЕРШЕНСТВОВАНИЯ ИЗВЕСТНЫХ МЕТОДОВ ГЕНЕРАЦИИ ПОДСТАНОВОК

3.1 Генерация случайных подстановок с заданными характеристиками

Генерация случайных подстановок и проверка их криптографических свойств является самым очевидным методом построения узлов нелинейной замены. По-видимому данный метод использовался при генерации подстановок блочных симметричных шифров DES, ГОСТ 28147-89 и Калина [6, 29, 89].

Пусть $\mathrm{X}_k = \{\chi_1, \chi_2, \ldots, \chi_k\}$, где $k \geq 1$, – множество критериев, которым должен удовлетворять нелинейных узел замены S. Обозначим через $\mathrm{X}_s (s \leq k)$ множество критериев оптимальной подстановки (см. пункт 2.2.5). Тогда метод генерации S-блока задаётся в следующем виде:

а) генерация случайной подстановки S;

б) повторять пункт а) до тех пор, пока S не будет удовлетворять всем критериям.

Очевидно, что в данном методе самым сложным является шаг б). Поэтому, если этап проверки свойств упростить до минимума, то производительность увеличится колоссально по сравнению с неоптимизированной версией.

Существует два способа решение данной задачи. Первый основан на уменьшении сложности подсчёта показателей. При максимальной эффективности данного метода сложность нахождения каждого из них будет равна $O(n)$. Однако, это идеализированный случай и на практике, с использованием современных персональных компьютеров, не достижим.

Более того, данный способ содержит ряд ограничений. Во-первых, при увеличении размерности обрабатываемого блока (n) становится сложнее вычислять показатели подстановки. Во-вторых, увеличивается пространство возможных вариантов, так при $n = m = 8$ количество возможных S-блоков приблизительно ровно 2^{1684}. Ещё одним недостатком является необходимость в наличии генератора псевдослучайных чисел, что само по себе является трудной криптографической задачей [3, 5].

Второй способ представляет собой генерацию подстановок с заданными параметрами. Обычно подстановки, сформированные при помощи этого метода, обладают предельными (максимальными или минимальными зависит от свойства) показателями. Более подробно данный способ описан в подразделе 3.2.

3.1.1 Обоснование временных ограничений при реализации метода случайной генерации подстановок с заданными параметрами

На начальном этапе разработки алгоритмов генерации нелинейных узлов замены (см. раздел 4) была исследована практическая возможность нахождения подстановок с оптимальными показателями. Как будет показано в подразделе 3.2, существуют эффективные способы нахождение S-блоков для $n = m = 6$, поэтому следующим значением и наиболее частым, с точки зрения применения в современных блочных симметричных шифрах [13, 59], является значение $n = 8$.

Основная задача заключалась в нахождении как минимум 4-х КШЗ-неэквивалентных подстановок с нелинейностью не меньше 100, δ – равномерностью не превышающей 8, с алгебраическим иммунитетом 3 и минимальной степенью 7.

Для решения данной практической задачи было написано программное обеспечение (ПО), описанное в подразделе 5.3. При реализации был

использован алгоритм из подраздела 3.1, т.е. программа генерирует случайную перестановку и проверяет её на оптимальность. После 12 часов работы кластера (см. пункт 5.3.1) было найдено 27 оптимальных подстановок, 4 из которых оказались КШЗ-неэквивалентными. Пример одной из подстановок в шестнадцатеричном представлении приведён в таблице 3.1, а её криптографические свойства – в таблице 3.2.

Таблица 3.1 – Случайно сгенерированная перестановка

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	14	9d	b9	e7	67	4c	50	82	ca	e5	1d	31	0a	c6	b2	51
1	a2	d8	54	90	d0	ce	2d	7d	c7	7e	d7	94	df	83	8e	6c
2	66	d2	6f	16	1e	76	fe	cc	aa	5a	8f	17	bd	2c	ac	ea
3	7b	65	a9	10	c0	92	ee	be	6a	6e	48	96	95	e9	32	bc
4	a1	42	d5	a7	81	b4	5f	e6	c2	5d	ad	3a	b7	0c	8d	01
5	98	fd	12	02	75	13	0f	6b	22	e2	ab	f7	7f	ba	97	d1
6	64	d9	c4	59	af	23	33	37	de	ae	60	05	63	a8	52	a5
7	4e	e0	dd	71	f2	24	34	57	47	a4	b3	9e	2f	c1	b8	cb
8	2b	d4	0d	36	91	8b	9c	26	25	61	a3	d6	eb	35	53	f4
9	2e	88	80	e4	30	db	fc	0e	77	8c	93	a6	78	06	e1	ec
A	f9	03	a0	27	da	ef	5c	00	7a	45	e8	40	1a	4b	5e	73
B	c3	ff	f5	f3	b0	c5	49	21	fa	11	39	84	43	38	85	07
C	f0	79	46	f8	e3	1f	09	b6	cd	55	1c	1b	fb	7c	ed	6d
D	15	56	86	20	68	4a	41	4f	d3	99	08	f6	3f	89	62	04
E	cf	c8	69	9f	19	5b	44	9b	87	b1	3d	bb	dc	2a	bf	58
F	3c	8a	18	3e	72	0b	28	4d	b5	9a	c9	74	29	f1	3b	70

Таблица 3.2 – Криптографические свойства подстановки из таблицы 3.1

Свойства	Значение
Компонентные булевы функций	
Сбалансированность	Да
Нелинейность	100
$\lvert AC\rvert_{\max}$	96
Критерий распространения	0
Корреляционный иммунитет	0

Продолжение таблицы 3.2

SSI	267520
Минимальная степень	7
t-устойчивость	0
Строгий лавинный критерий	Нет
Подстановка	
Биективность	Да
МТД	8
МТЛА	28
Циклическая структура	34:5, 7:16, 11:44, 0:191
AI/КУ/SP	3/441/0,816

Дополнительно был проведён поиск подстановок с нелинейностью равной 102 и более используя метод случайной генерации. Однако, после 48 часов работы кластера, что эквивалентно приблизительно 22 годам работы одного однопроцессорного (с одним ядром) компьютера, ни одной подстановки не нашлось. Таким образом, можно прийти к выводу, что с практической точки зрения генерация таких подстановок является вычислительно сложной задачей.

3.2 Аналитические методы генерации векторных булевых функций с предельными показателями

Известно, что любая подстановка может быть представлена в виде векторной булевой функций (см. формулу (1.19)). Следовательно, применяя интерполяционную формулу Лагранжа [156], при $x, y \in \mathcal{F}_{2^{\max(n,m)}}$, можно всегда найти функциональное представление произвольного S-блока. Из чего

следует, что всегда существует взаимосвязь между подстановкой и векторной булевой функцией.

Известно несколько классов степенных ПСН (см. пункт 1.4.2) функций. В таблице 3.3 приводятся все известные на сегодняшний день экспоненты d вплоть до эквивалентности, такие, что функция x^d является ПСН над полем $\mathcal{F}_{2^n}$ [48]. Для чётного n Голд (Gold), Касами (Kasami), Вэлш (Welch) и Нихо (Niho) функции также являются ПБ [48, 158]. Когда n чётное, обратные функции ($d = -1$) являются дифференциально 4-распределёнными перестановками [54]. Данный вид функций, вместе с аффинными преобразованиями, использовался при выборе подстановок в алгоритмах шифрования AES [20, 39] и Лабиринт [30].

Таблица 3.3 – Известные степенные ПСН функции x^d над полем $\mathcal{F}_{2^n}$

Название	Экспонента d	Условие	Источник
Голд	$2^i + 1$	$НОД(i,n) = 1$	[186]
Касами	$2^{2i} - 2^i + 1$	$НОД(i,n) = 1$	[187]
Вэлш	$2^t + 3$	$n = 2t + 1$	[188]
Нихо	$2^t + 2^{\frac{t}{2}} - 1$	$n = 2t + 1$, t – чётное	[189]
	$2^t + 2^{\frac{3t+1}{2}} - 1$	$n = 2t + 1$, t – нечётное	
Обратные	$2^{2t} - 1$	$n = 2t + 1$	[32]
Доббертин	$2^{4t} + 2^{3t} + 2^{2t} + 2^t - 1$	$n = 5t$	[190]

Существует множество других нестепенных классов функций, которые ранее были рассмотрены в [48, 158]. До недавнего времени считалось, что ПСН перестановочных функций для чётного значения n не существует [158]. Однако, в 2010 году была представлена работа [191], где был приведен пример такой функции для $n = 6$. Авторы при ПСН такой функции

объединили результаты теории кодирования, её связь с КШЗ-эквивалентностью и вычислительные мощности того времени.

Теорема 3.1. Пусть α *– мультипликативный генератор поля* $\mathcal{F}_{2^6}$ *с неприводимым полиномом* $f(x) = x^6 + x^4 + x^3 + x + 1$. *Тогда ПСН функция*

$$F(x) = \alpha x^3 + \alpha^5 x^{10} + \alpha^4 x^{24}$$

является КШЗ-эквивалентной ПСН перестановочной функцией над полем $\mathcal{F}_{2^6}$ [191].

Например, для функции

$$\mathcal{L}(x, y) = (tr_{6/3}(\alpha^4 x) + \alpha\, tr_{6/3}(y), tr_{6/3}(\alpha x) + \alpha\, tr_{6/3}(\alpha^4 y))$$

где $tr_{6/3}(x) = x + x^{2^3}$, $y = F(x)$, функция $G_F = \mathcal{L}(G_H)$ является ПСН перестановочной. Однако, на сегодняшний день остаётся открытым вопрос о существовании ПСН перестановок для $n \geq 8$.

В последние несколько лет получило развитие исследование квадратичных функций. Во-первых это связано с тем, что многие учёные (например, [192]) предполагают, что не существует больше классов степенных ПСН функции, чем приведённые в таблице 3.3. Во-вторых, квадратичные функции являются следующими по сложности в анализе, после степенных.

Другим направлением в данной области является нахождение КШЗ-эквивалентных функций или доказательство принадлежности функций к различным КШЗ-эквивалентным классам [157]. Адаптированные результаты данных исследований так же могут быть использованы при генерации подстановок [54].

3.3 Генерация булевых функций методом градиентного спуска

Метод построения булевых функций на основе градиентного спуска был представлен в [193, 194]. Напомним, что булевы функции являются подклассом векторных булевых функций с $m = 1$.

Основная идея метода заключается в понижении нелинейности заданных бент-последовательностей. Другими словами, в заданной бент-последовательности (таблице истинности) изменяются некоторые биты таким образом, чтобы новая последовательность была сбалансированной, а нелинейность была близкой к нелинейности бент-функции.

Очевидно, что произвольное изменение битов может негативно сказаться на нелинейности получившейся последовательности. Поэтому для инвертирования выбираются позиции, изменение которых влечёт за собой увеличение или уменьшение максимального значения преобразования Уолша на равное количество, так чтобы нелинейность получившейся функции была близка к предыдущей.

Данный метод, например, может быть использован для генерации криптографических функций, применяемых в поточных шифрах. Однако, для современных ПШ фильтрующая функция должна быть как минимум 20 бит или более [96]. В связи с чем эффективность данного метода снижается, так как помимо нахождения самой последовательности ещё необходимо восстанавливать алгебраическую нормальною форму [48], которая должна содержать минимальное количество мономов для эффективной реализации алгоритма шифрования.

Другая область применения заключается в наборе булевых функций в нелинейный узел замены, который заранее обладает высокими криптографическими свойствами.

3.4 Метод генерации подстановок на основе набора булевых функций

Концепция построения данного метода базируется на использовании в качестве входных значений булевых функций, обладающих заведомо хорошими криптографическими свойствами. Основная цель заключается в наборе булевых функций в векторную булеву функцию с сохранением сбалансированности (см. пункт 1.4.3) [48]. Выходом данного метода, при $n = m$, является перестановка с высокими показателями криптографической стойкости.

Метод генерации подстановок описывается следующим образом:

а) создаётся множество случайных булевых функций $J = \{f_1, f_2, \ldots, f_k\}$, где $f_i : \mathcal{F}_2^n \to \mathcal{F}_2$ и $k \geq m$, с заданными параметрами (например, с высокой нелинейностью и сбалансированные);

б) к множеству S (начальное значение которого ровно $\varnothing$) добавляется случайная функция $g \in J$;

в) если векторная булева функция $F = S$ не сбалансирована, то g удаляется из множества S;

г) шаги б) и в) повторяются до тех пор, пока размер множества S не равен m.

Сложность данного метода зависит от двух показателей: начальных криптографических свойств булевых функций, а следовательно и выходной функции, и начальном выборе функций f. Стоит отметить, при маленьких значениях k и больших n (8 и более) вероятность генерации подстановки стремится к нулю.

В качестве примера в статье [194] приводится подстановка ($n = m = 8$), сгенерированная при помощи модифицированного алгоритма, описанного выше. Первые четыре булевых функций имеют нелинейность 112, а остальные были выбраны случайным образом так, чтобы выходная векторная

функция была перестановочной. В результате, выходная подстановка (таб. 3.4) имеет показатели, приведённые в таблице 3.5.

Все значения из таблицы определены в подразделах 1.4 и 2.1. Напомним, что циклы задаются в виде:

$$sp : l,$$

где sp – начальная точка цикла;

l – длина цикла.

Таблица 3.4 – Перестановка сгенерированная из отдельных булевых функций

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	f0	4a	ed	f4	d1	db	bc	67	7d	f7	a0	fb	6c	d6	51	ba
1	eb	7f	46	c3	aa	ef	e7	02	f6	72	ab	3e	37	52	3b	4e
2	3f	15	ec	85	5a	a4	4d	66	03	39	30	b9	ca	78	91	0a
3	54	11	e3	82	fd	f1	b6	13	48	2c	2b	6e	f9	ac	7b	2e
4	5f	b5	f2	69	71	cb	4c	97	92	e8	ff	25	dc	96	79	ea
5	14	60	99	0c	6a	be	d7	c2	89	6d	40	c1	bf	12	7a	ae
6	70	c8	93	d8	ee	94	2d	a6	cc	76	6f	b4	e2	a8	81	3a
7	6b	9f	58	3d	65	01	16	63	a7	53	64	41	09	1c	8a	ce
8	a1	bb	cd	b3	08	da	ad	36	d2	88	9d	24	33	e9	1f	f5
9	4b	af	e6	d3	2a	0e	b7	32	28	8c	95	c0	c9	5c	84	b0
A	7e	d4	8d	26	2f	05	7c	07	3c	56	8f	d0	1d	df	8e	c5
B	04	90	55	b2	45	31	86	5b	06	62	d5	10	27	a2	c4	50
C	9e	74	43	68	b1	1a	dd	fe	bd	87	00	9b	f3	29	1e	75
D	34	20	a9	18	9a	de	77	22	c7	73	38	cf	d9	fc	44	e0
E	e1	1b	42	49	4f	a5	9c	47	23	19	80	0b	0d	57	5e	e5
F	35	21	c6	a3	8b	0f	98	5d	17	83	e4	61	59	b8	fa	f8

Таблица 3.5 – Криптографические свойства подстановки из таблицы 3.4

Свойства	Значение
Компонентные булевы функций	
Сбалансированность	Да
Нелинейность	14
$\lvert AC\rvert_{\max}$	128
Критерий распространения	0
Корреляционный иммунитет	0
SSI	999424
Минимальная степень	6
t-устойчивость	0
Строгий лавинный критерий	Нет
Подстановка	
Биективность	Да
МТД	16
МТЛА	114
Циклическая структура	94:4, 0:252
AI/КУ/SP	2/8/0,730

Авторы статьи утверждают, что данный нелинейный узел замены может быть использован в криптографических средствах защиты информации [194]. Однако, из таблицы 3.5 видно, что выходная подстановка обладает гораздо более низкими криптографическим показателями, чем отдельные булевы функции.

В более поздней работе [195], авторы отмечают, что нахождение перестановок с $m \geq 6$, которые удовлетворяют критериям высокой нелинейности, низкой автокорреляции и высокой алгебраической степени,

используя данный алгоритм, не представляется возможным с точки зрения вычислительных ресурсов.

3.5 Другие методы генерация подстановок

Существует множество других методов генерации подстановок. К ним можно отнести:

– оптимизированные методы генерации случайных подстановок [196];

– эффективные методы нахождения ПСН векторный булевых функций с определёнными ограничениями на их представление [158];

– различные вариации метода градиентного спуска, включая градиентный подъём, метод «имитации отжига» и др. [197, 198];

– новые направления использующие матричное представление квадратичных функций [199, 200].

Стоит отметить, что новые направления являются перспективными и, скорей всего, эффективными. В тоже время прошло не так много времени с момента их представления, поэтому они требуют дополнительных всесторонних независимых экспертиз и проверок.

Метод, представленный в работе [58], был найден автором настоящей работы лишь после разработки метода представленного в подразделе 4.2. В статье не представлена информация о производительности предложенного метода. Из личной переписки, известно, что время генерации одной оптимальной подстановки (с нелинейностью 104, алгебраическим иммунитетом 3 и 8-равномерной и вероятностью 90%) на персональном компьютере занимает до 44 часов. К сожалению, более детальной информации, такой как количество использованных ядер, т.е. использовалась ли принципы параллелизма, количество затраченной памяти и т.д., не удалось получить. Поэтому предполагается, что приведённые данные соответствуют генерации одного S-блока на одном процессоре (ядре) за 44

часа. Очевидно, что данное предположение является заниженными, если принять во внимание факт использования нескольких ядер.

3.6 Выводы

Проведённый анализ показывает, что большинство современных методов генерации подстановок основаны на теоретическом подходе, где за основу берутся векторные булевы функции с теоретически доказанными свойствами. Однако, в большинстве случаев рассматривается лишь одно или два из них, в то время как на практике необходимо учитывать 4 и более.

Альтернативным направлением является эвристический подход, методы которого ориентированы на генерацию псевдослучайных подстановок с дальнейшей проверкой их криптографических свойств.

В последнее время увеличилась сложность аналитического подхода нахождения новых функции, которые были бы неэквивалентны уже существующим. С целью развития данного направления предпринимаются попытки анализировать функции, не принадлежащие к степенным классам и с алгебраической степенью большей или равной второй. Однако, при увеличении пространства рассматриваемых полиномов и размерности поля значительно сложнее становится доказывать и проверять их свойства.

Ограничения эвристических методов обусловлены вычислительными возможностями современных компьютеров и сложностью методов генерации. К сожалению, в большинстве случаев алгоритмы генерации подстановок основаны на некотором случайном или псевдослучайном процессе, что приводит к невозможности уменьшения сложности генерации S-блоков до полиномиальной.

Такие ограничения приводят к поиску других путей нахождения нелинейных узлов замены. Один из них основан на дальнейшем развитии математического аппарата [198, 199]. Другой путь включает в себя

модификацию эвристических методов с использованием алгебраических структур. Применение такого подхода даёт возможность значительно уменьшить сложность практического поиска подстановок с высокими показателями криптографической стойкости.

4 РАЗРАБОТКА НОВЫХ МЕТОДОВ ГЕНЕРАЦИИ *S*-БЛОКОВ

4.1 Предложенный метод генерации ДКЭ для шифра ДСТУ ГОСТ 28147:2009

Исследование подстановок в виде векторных булевых функций предоставляет возможность значительно оптимизировать формирование узлов замены симметричных криптографических примитивов. Например, позволяет находить целые классы функций со схожими свойствами, использование которых сокращает время поиска оптимальной подстановки. Так, в таблице 3.3 представлен ряд функций с минимальными показателями δ-равномерности. Как отмечалось в разделе 3.2, функции, представленные в таблице, не являются полным списком. Существуют и нестепенные функции, классификация которых продолжается и до настоящего времени.

Тем не менее, для маленьких значений n выполнить полную классификацию возможно. Так в работах [201, 202] были классифицированы все 4-х битные перестановки с оптимальными показателями. Оптимальной считалась биективная подстановка с нелинейностью равной 4 и 4-равномерная [201]. Всего было получено 16 различных перестановочных векторных булевых функций, не являющихся аффинно-эквивалентными. Все 16 перестановок и соответствующие им векторные булевы функции представлены в таблицах 4.1 и 4.2.

Таблица 4.1 – Все перестановки для $n = 4$ вплоть до аффинной эквивалентности

F_i	Перестановка															
F_1	0	1	2	d	4	7	f	6	8	b	c	9	3	e	a	5
F_2	0	1	2	d	4	7	f	6	8	b	e	3	5	9	a	c

Продолжение таблицы 4.1

F_3	0	1	2	d	4	7	f	6	8	b	e	3	a	c	5	9
F_4	0	1	2	d	4	7	f	6	8	c	5	3	a	e	b	9
F_5	0	1	2	d	4	7	f	6	8	c	9	b	a	e	5	3
F_6	0	1	2	d	4	7	f	6	8	c	b	9	a	e	3	5
F_7	0	1	2	d	4	7	f	6	8	c	b	9	a	e	5	3
F_8	0	1	2	d	4	7	f	6	8	c	e	b	a	9	3	5
F_9	0	1	2	d	4	7	f	6	8	e	9	5	a	b	3	c
F_{10}	0	1	2	d	4	7	f	6	8	e	b	3	5	9	a	c
F_{11}	0	1	2	d	4	7	f	6	8	e	b	5	a	9	3	c
F_{12}	0	1	2	d	4	7	f	6	8	e	b	a	5	9	c	3
F_{13}	0	1	2	d	4	7	f	6	8	e	b	a	9	3	c	5
F_{14}	0	1	2	d	4	7	f	6	8	e	c	9	5	b	a	3
F_{15}	0	1	2	d	4	7	f	6	8	e	c	b	3	9	5	a
F_{16}	0	1	2	d	4	7	f	6	8	e	c	b	9	3	a	5

Таблица 4.2 – Полиномиальное представление перестановок из таблицы 4.1

F_i	Значение векторной булевой функции
F_1	$g^{12}x^{14} + g^{12}x^{13} + g^4x^{12} + g^2x^{11} + g^2x^{10} + g^2x^9 + g^{12}x^8 + g^3x^6 + g^{10}x^5 + g^5x^4 + g^{10}x^3 + g^{10}x^2$
F_2	$g^{12}x^{14} + g^2x^{13} + g^{12}x^{12} + g^{13}x^{11} + x^{10} + gx^9 + g^{11}x^8 + gx^7 + g^9x^6 + g^{12}x^5 + g^{10}x^4 + g^{12}x^3 + g^{10}x^2 + g^{13}x$
F_3	$g^3x^{14} + g^9x^{13} + gx^{12} + g^{13}x^{10} + g^9x^9 + g^4x^8 + g^5x^7 + gx^6 + g^{14}x^5 + g^5x^4 + gx^3 + g^5x^2 + g^9x$
F_4	$gx^{14} + g^{12}x^{13} + g^2x^{12} + g^4x^{11} + g^6x^9 + g^8x^8 + g^9x^7 + g^5x^6 + g^{12}x^5 + g^3x^4 + g^5x^3 + g^{12}x^2$
F_5	$g^4x^{14} + x^{13} + g^{11}x^{12} + g^5x^{11} + g^{12}x^{10} + g^2x^9 + x^8 + g^6x^6 + g^5x^5 + g^6x^4 + g^{11}x^3 + g^5x^2 + g^{12}x$

Продолжение таблицы 4.2

F_6	$g^{10}x^{14} + g^8x^{13} + g^4x^{12} + gx^{11} + g^5x^{10} + g^2x^9 + g^5x^8 + g^2x^7 + g^3x^6 + g^7x^5 + g^7x^4 + g^{10}x^3 + g^6x^2 + x$
F_7	$x^{14} + g^4x^{13} + g^3x^{12} + g^2x^{11} + x^{10} + g^{11}x^9 + g^2x^8 + gx^7 + g^2x^6 + g^9x^5 + g^4x^4 + g^9x^3 + g^{12}x^2 + g^{11}x$
F_8	$g^4x^{14} + g^5x^{13} + g^{13}x^{12} + g^5x^{11} + g^{11}x^{10} + g^{10}x^7 + g^{14}x^6 + g^5x^5 + g^{12}x^4 + g^6x^3 + g^{10}x^2 + g^{13}x$
F_9	$g^{11}x^{14} + g^{12}x^{13} + g^{11}x^{12} + g^6x^9 + g^{10}x^8 + g^8x^7 + g^5x^5 + g^4x^4 + g^9x^3 + g^7x^2 + g^2x$
F_{10}	$g^5x^{13} + g^{12}x^{12} + g^{10}x^{11} + g^6x^{10} + gx^9 + gx^8 + g^8x^7 + g^9x^6 + g^{10}x^5 + g^6x^4 + g^{12}x^3 + g^5x^2 + g^{11}x$
F_{11}	$g^4x^{13} + g^2x^{12} + g^{11}x^{11} + g^{11}x^{10} + g^3x^9 + g^{14}x^7 + g^5x^6 + g^9x^5 + g^{11}x^4 + gx^3 + g^{10}x$
F_{12}	$g^2x^{14} + g^3x^{13} + g^2x^{12} + g^9x^{11} + x^{10} + g^3x^9 + g^{13}x^8 + gx^7 + g^5x^6 + g^8x^5 + g^3x^4 + gx^3 + gx^2 + g^{11}x$
F_{13}	$g^2x^{14} + g^{12}x^{13} + g^3x^{12} + g^6x^{11} + gx^{10} + g^6x^9 + g^7x^8 + g^4x^7 + g^{11}x^6 + g^6x^5 + g^{10}x^4 + gx^3 + g^7x^2 + g^2x$
F_{14}	$gx^{14} + g^{13}x^{13} + g^9x^{12} + g^9x^{11} + x^{10} + g^2x^9 + g^9x^8 + x^7 + g^7x^6 + gx^5 + x^4 + g^{13}x^3 + g^2x^2 + x$
F_{15}	$g^7x^{14} + x^{13} + g^2x^{12} + g^8x^{11} + g^8x^{10} + gx^9 + g^7x^7 + g^2x^6 + g^{14}x^5 + g^{11}x^4 + g^9x^3 + g^4x^2 + g^6x$
F_{16}	$g^{11}x^{14} + g^2x^{13} + g^9x^{12} + gx^{10} + g^{11}x^9 + g^{11}x^8 + g^{11}x^7 + g^{13}x^6 + g^7x^5 + g^6x^4 + x^3 + g^{12}x^2 + g^{14}x$

Переменная g является мультипликативным генератором поля $\mathcal{F}_{2^4}$ с примитивным полиномом $x^4 + x + 1$. Стоит заметить, что хотя все функции и являются аффинно-неэквивалентными, некоторые из них – КШЗ-эквивалентные [48]:

$$F_1 \sim F_2 \sim F_3 \sim F_9; \quad F_8 \sim F_{12} \sim F_{13};$$

$$F_4 \sim F_6; \quad F_5 \sim F_7; \quad F_{10} \sim F_{11}; \quad F_{15} \sim F_{16}.$$

Как видно из соотношений выше функция F_{14} не является КШЗ-эквивалентной ни одной другой. С практической точки зрения, приведённая взаимосвязь между функциями означает, что любая 4-равномерная перестановка с нелинейностью 4 будет эквивалентна одной из семи функций, представленных выше. Следовательно, не существует долговременных

ключевых элементов (ДКЭ) для шифра ДСТУ ГОСТ 28147:2009, удовлетворяющих утверждению 2.1, в случае КШЗ-эквивалентности.

В криптографии, помимо линейных и дифференциальных показателей, важным является критерий минимальной степени компонентных функций (см. подраздел 1.4) [48]. Для 4-битной перестановки значение данного показателя должно быть равно 3. В таблице 4.3 приведены 8 из 16 векторных булевых функций в каноническом виде (коэффициент при старшей степени равен 1), удовлетворяющих данному критерию.

Таблица 4.3 – Полиномиальное представление оптимальных подстановок для $n = 4$, принадлежащих различным аффинным классам

	Полином
F_4	$x^{14}+g^{11}x^{13}+gx^{12}+g^3x^{11}+g^5x^9+g^7x^8+g^8x^7+g^4x^6+g^{11}x^5+g^2x^4+g^4x^3+g^{11}x^2$
F_5	$x^{14}+g^{11}x^{13}+g^7x^{12}+gx^{11}+g^8x^{10}+g^{13}x^9+g^{11}x^8+g^2x^6+gx^5+g^2x^4+g^7x^3+gx^2+g^8x$
F_6	$x^{14}+g^{13}x^{13}+g^9x^{12}+g^6x^{11}+g^{10}x^{10}+g^7x^9+g^{10}x^8+g^7x^7+g^8x^6+g^{12}x^5+g^{12}x^4+x^3+g^{11}x^2+g^5$
F_7	$x^{14}+g^4x^{13}+g^3x^{12}+g^2x^{11}+x^{10}+g^{11}x^9+g^2x^8+gx^7+g^2x^6+g^9x^5+g^4x^4+g+x^3+g^{12}x^2+g^{11}x$
F_8	$x^{14}+gx^{13}+g^9x^{12}+gx^{11}+g^7x^{10}+g^6x^7+g^{10}x^6+gx^5+g^8x^4+g^2x^3+g^6x^2+g^9x$
F_{12}	$x^{14}+gx^{13}+x^{12}+g^7x^{11}+g^{13}x^{10}+gx^9+g^{11}x^8+g^{14}x^7+g^3x^6+g^6x^5+gx^4+g^{14}x^3+g^{14}x^2+g^9x$
F_{13}	$x^{14}+g^{10}x^{13}+gx^{12}+g^4x^{11}+g^{14}x^{10}+g^4x^9+g^5x^8+g^2x^7+g^9x^6+g^4x^5+g^8x^4+g^{14}x^3+g^5x^2+x$
F_{14}	$x^{14}+g^{12}x^{13}+g^8x^{12}+g^8x^{11}+g^{14}x^{10}+gx^9+g^8x^8+g^{14}x^7+g^6x^6+x^5+g^{14}x^4+g^{12}x^3+gx^2+g^{14}x$

Стоит отметить, что из приведённых в таблице 4.3 КШЗ-эквивалентными функциями являются:

$$F_4 \sim F_6; \quad F_5 \sim F_7;$$

$$F_8 \sim F_{12} \sim F_{13}.$$

Отсюда и далее для $n = 4$ под оптимальной понимается подстановка:

а) биективная;

б) 4-равномерная;

в) с нелинейностью 4;

г) с минимальной степенью 3;

д) отсутствием фиксированных точек (нет циклов длиной 1).

На основе функций, описанных в таблице 4.3, был предложен метод генерации долговременных ключевых элементов для шифра ДСТУ ГОСТ 28147:2009 [7].

На первом шаге каждой из 8 подстановок ДКЭ (K_1, …, K_8) ставится в соответствие уникальная векторная булева функция F_i ($i = \{4,5,6,7,8,12,13,14\}$). Количество различных вариантов равно $8! = 4320$. Очевидно, что каждая из подстановок, образованных при помощи векторных булевых функции, будет удовлетворять пунктам а)-г). Далее к каждому полиному последовательно применяются различные аффинно-эквивалентные преобразования (см. формулу (1.25)) до тех пор, пока функция (подстановка) не будет удовлетворять пункту д). Количество возможных аффинно-эквивалентных преобразований равно $\left(\prod_{i=1}^{n}(2^n - 2^{i-1})\right)^2 \cdot 2^{2n}$. Для случая n=4 общее количество ДКЭ, которые можно получить при помощи данного метода, не превышает $\left(\prod_{i=1}^{4}(2^4 - 2^{i-1})\right)^2 \cdot 2^8 \cdot 8! \approx 2^{51}$ и равно ему в случае отсутствия пункта д). Как было описано в пункте 1.4.4, аффинное преобразование сохраняет свойства а)-г).

Таким образом, предлагаемый метод позволяет сократить время, необходимое на генерацию и проверку криптографических параметров подстановок для шифра ДСТУ ГОСТ 28147:2009, за счёт заранее выбранных векторных булевых функций с параметрами а)-г). Пример ДКЭ с оптимальными показателями, сгенерированного по предложенному методу, представлен в таблице 4.4.

Таблица 4.4 – Пример долговременного ключевого элемента для шифра ДСТУ ГОСТ 28147:2009 с оптимальными показателями

	Ключ
K_1	[5, 11, 13, 10, 8, 4, 1, 0, 6, 12, 3, 15, 2, 9, 7, 14]
K_2	[7, 8, 12, 10, 2, 1, 15, 14, 11, 13, 5, 9, 0, 3, 6, 4]
K_3	[15, 14, 7, 5, 3, 13, 9, 2, 10, 6, 11, 1, 8, 0, 12, 4]
K_4	[15, 8, 9, 14, 1, 4, 13, 11, 3, 5, 6, 12, 0, 2, 7, 10]
K_5	[5, 10, 6, 15, 8, 4, 2, 3, 9, 7, 13, 0, 14, 1, 12, 11]
K_6	[7, 9, 12, 8, 10, 2, 13, 14, 0, 5, 4, 6, 3, 15, 1, 11]
K_7	[8, 14, 11, 5, 1, 4, 7, 6, 13, 2, 9, 15, 3, 10, 12, 0]
K_8	[13, 14, 6, 10, 2, 15, 0, 5, 12, 1, 11, 4, 9, 8, 3, 7]

4.2 Предложенный метод генерации нелинейных узлов замены для перспективных симметричных криптопримитивов

Основные этапы построения функций на основе градиентного спуска были описаны в подразделе 3.3. В предлагаемом методе предлагается использовать тот же подход, однако с двумя существенными отличиями [54]:

- вместо булевых функций использовать векторные булевы функции;
- вместо бент-функций (последовательностей) использовать векторные булевы функции (подстановки) с максимальными показателями δ-равномерности или с максимальным значением нелинейности.

Напомним, в [193] было показано, что само по себе изменение необходимого количества бит в бент-последовательности не гарантирует достижение нелинейности, близкой к максимальной. Для векторного случая в [203] было доказано следующее утверждение.

Утверждение 4.1. Пусть $F : \mathcal{F}_{2^n} \mapsto \mathcal{F}_{2^n}$. *Определим функцию* G *следующим образом:*

$$\begin{cases} G(p_1) = F(p_2), \quad p_1 \neq p_2; \\ G(p_2) = F(p_1); \\ G(x) = F(x), \quad x \neq p_1, p_2. \end{cases}$$

Тогда

$$\delta(F) - 4 \leq \delta(G) \leq \delta(F) + 4,$$
$$NL(F) - 2 \leq NL(G) \leq NL(F) + 2.$$

Из утверждения 4.1 видно, что при обмене местами двух значений в некоторой перестановке значения, нелинейности и δ-равномерности будут отличаться на небольшое значение. Основываясь на описанном выше, предлагается новый алгоритм генерации подстановок на основе векторных булевых функций.

Метод принимает на вход перестановочную векторную функцию F с минимальным показателем δ-равномерности и количество значений (NP), которые необходимо поменять местами для достижения оптимальных криптографических показателей.

Основные шаги метода представлены ниже.

а) Генерация подстановки S на основе выбранной перестановочной (биективной) векторной булевой функции F.

б) Случайный обмен местами NP значений подстановки S и формирование подстановки S_t.

в) Последовательная вычисление показателей в зависимости от их вычислительной сложности. Если подстановка S_t удовлетворяет всем критериям, кроме цикловых, тогда применяется PA-эквивалентность для достижения критерия отсутствия фиксированных точек. При несоответствии хотя бы одному из критериев переход в п. б).

В результате оптимальная подстановка будет храниться в S_t. В виду того, что значение NP является входным параметром, то возможна ситуация (при маленьком значении NP), когда метод не найдёт ни одного S-блока. В тоже время, при $NP = 2^n$, вышеописанный метод идентичен методу случайной генерации подстановок (см. подраздел 3.1) с соответствующей сложностью генерации.

4.3 Формирование оптимальных подстановок

Как уже отмечалось в разделе 3, генерация подстановок с оптимальными криптографическими показателями является одновременно практически и теоретически сложной задачей. Данные утверждения подтверждаются многими исследованиями, включая результаты из пункта 3.1.1.

Отсюда и далее для $n = 8$ оптимальной является подстановка, если она является перестановкой без фиксированных точек, с показателями:

– минимальная степень 7;

– алгебраический иммунитет 3;

– 8-равномерная;

– нелинейность 104.

Предложенный в подразделе 4.2 метод позволяет найти оптимальные подстановки. В качестве примера была взята функция $F(x) = x^{-1}$ [32]. Увеличивая значения NP (начиная с 1), экспериментальным путём найдено значение $NP = 22$, при котором достигаются все необходимые свойства подстановки. Пример такой перестановки представлен в таблице 4.5, а её свойства приведены в таблице 4.6.

Таблица 4.5 – Пример оптимальной подстановки, сгенерированной по алгоритму из подраздела 4.2

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
0	c4	ca	ff	b1	be	2c	6f	c2	aa	62	3f	84	2b	f0	5c	30
1	86	a5	6b	da	bf	31	4b	40	52	3b	02	79	27	ea	ba	61
2	bd	69	44	63	0c	72	b0	1a	3c	70	76	e7	cb	19	14	c8
3	7b	22	11	8b	99	9b	b9	20	92	fc	7a	6a	dd	d0	4c	eb
4	74	c1	53	d5	ae	ab	09	34	c0	f1	59	b8	57	f5	d4	db
5	95	1d	15	a3	e8	a1	d9	c5	88	67	39	a2	e1	96	f2	37
6	a0	41	fb	47	cc	46	4d	56	8d	3a	a6	fe	4a	bb	04	b4
7	d8	94	ad	87	75	33	83	de	68	06	51	18	0e	bc	a4	e4
8	f9	64	e3	85	8e	66	f7	d3	b5	cf	32	f8	60	ce	17	ed
9	7f	49	8f	4e	5f	e5	e9	1e	b7	0a	7c	4f	a9	0d	c7	0f
a	b6	77	01	5e	13	d1	af	91	9d	36	2a	48	58	a7	5b	fd
b	d7	d6	16	5d	93	1b	98	80	dc	c3	7e	cd	2f	3e	03	f3
c	54	6c	0b	b3	35	e0	38	e6	c9	ec	5a	7d	73	21	9a	25
d	f6	c6	42	90	6e	12	07	8a	8c	df	9f	82	29	81	89	ee
e	1c	00	28	05	2e	10	26	43	08	65	9c	9e	78	fa	3d	45
f	ef	ac	a8	71	50	1f	97	2d	24	6d	b2	55	e2	23	d2	f4

Таблица 4.6 – Криптографические свойства подстановки из таблицы 4.5

Свойства	Значение
Компонентные булевы функций	
Сбалансированность	Да
Нелинейность	102
$\lvert AC\rvert_{\max}$	88
SSI	217600
Минимальная степень	7
Подстановка	
Биективность	Да
МТД	8
МТЛА	26
Циклическая структура	0:26, 1:230
AI/КУ/SP	3/441/0,825

Изначально для поиска представленной подстановки было написано программное обеспечение на языке Sage (см. подраздел 5.2). За 12 часов на обычном компьютере сгенерировалось достаточно подстановок для того, чтобы найти 4 КШЗ-неэквивалентных перестановок, что говорит об эффективности предложенного метода. Дополнительные тесты при использовании данного ПО показали, что для нелинейности больше 102, подстановки не являются оптимальными по показателю алгебраического иммунитета. Однако, существуют перестановки с нелинейностью 104 и алгебраическим иммунитетом 2, в которых количество уравнений системы незначительное. Пример такой подстановки и соответствующие ей характеристики представлены в таблицах 4.7 и 4.8 соответственно.

Таблица 4.7 – Пример подстановки с маленьким количеством уравнений и алгебраическим иммунитетом 2

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
0	49	95	69	a4	2c	e7	ef	d8	b8	8b	6c	1e	be	30	5f	04
1	42	cf	e1	1f	c2	a5	cc	ff	84	af	bd	2b	b0	3f	dd	78
2	b9	39	37	c7	77	61	f2	72	c5	8d	b1	47	4f	52	7b	bc
3	89	ac	79	7e	ba	44	12	e8	74	93	34	9b	97	f9	29	10
4	eb	5d	96	d1	85	bb	b6	cd	6f	75	ad	8a	8e	ee	ed	f5
5	c3	9c	19	3b	c0	2a	5a	62	b7	07	87	06	b3	60	54	c1
6	31	86	d2	1d	76	28	43	e4	27	66	b5	6d	a1	f1	e9	0f
7	4e	e0	08	2f	cb	55	45	fd	81	f0	d7	68	ec	0d	6a	82
8	26	00	0b	05	7a	3c	09	e2	a9	3e	fc	21	33	b4	71	ca
9	9e	1a	90	1c	a8	8f	83	91	46	b2	fb	94	c4	64	6e	13
a	f6	51	a0	bf	6b	a7	0a	c6	db	20	59	9d	d3	58	a3	a6
b	92	a2	d6	5e	aa	56	da	38	03	7c	63	ab	18	14	25	70
c	23	ae	32	57	80	e5	53	fe	f8	50	3a	35	e6	d4	4d	d5
d	c8	02	41	4b	df	ce	40	01	65	7f	de	7d	fa	d9	16	e3
e	9a	2e	4a	98	15	24	88	ea	2d	f4	99	d0	36	67	4c	0c
f	22	17	f7	11	c9	f3	5c	8c	48	73	3d	1b	9f	5b	dc	0e

Таблица 4.8 – Криптографические свойства подстановки из таблицы 4.7

Свойства	Значение
Компонентные булевы функций	
Сбалансированность	Да
Нелинейность	104
$\|AC\|_{\max}$	80
Критерий распространения	0
Корреляционный иммунитет	0
SSI	186880
Минимальная степень	7
t-устойчивость	0
Строгий лавинный критерий	Нет
Подстановка	
Биективность	Да
МТД	8
МТЛА	24
Циклическая структура	2:5, 11:6, 69:6, 10:47, 0:192
AI/КУ/SP	2/1/0,518

ПО Sage является хорошим средством для так называемого «доказательства концепции». Однако, чтобы утверждать с практической точки зрения, что не существует подстановок с лучшими характеристиками, необходимо провести множество вычислений. Для решения данной задачи было написано отдельное программное обеспечение для использования на кластере (см. подраздел 5.3).

После 1 часа работы кластера было сгенерировано 1152 перестановок с нелинейностью 104 и алгебраическим иммунитетом 3. Пример такой

подстановки и её характеристика приведены в таблицах 4.9 и 4.10 соответственно.

Очевидно, что если значения функции *F* меняются случайным образом, тогда время необходимое на генерацию одного оптимального S-блока на однопроцессорном компьютере (с одним ядром) равно 3,5 часам.

Таблица 4.9 – Пример оптимальной подстановки с нелинейностью 104 и алгебраическим иммунитетом 3

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
0	ab	99	2a	0d	1a	6c	90	c3	65	4a	d1	e5	36	95	b3	ff
1	e1	43	19	53	1c	6d	ec	fe	60	17	fa	5d	05	4d	ea	b1
2	50	d0	92	40	ee	9c	22	4c	b9	88	07	18	68	0f	57	dd
3	ae	a0	aa	7f	5c	02	89	a4	b4	f2	db	f9	64	3f	b7	c1
4	87	3c	0e	a5	bf	1d	06	5f	cf	7a	da	28	d7	d5	bc	93
5	e2	96	9a	ba	dc	7d	ed	9f	d6	32	08	c4	30	c8	61	49
6	91	bd	e6	4e	3d	e9	f7	ac	3e	b2	56	e8	7c	d2	29	2b
7	42	5e	f8	48	ce	26	cd	a8	c6	df	2f	f0	3a	9e	81	4f
8	e3	e7	8d	76	62	34	1e	f3	af	2e	8c	38	51	bb	66	54
9	1f	77	5a	0a	eb	23	fb	3b	70	2c	0c	01	04	4b	ca	13
a	10	41	d8	d4	94	b0	6e	82	39	fc	71	a1	52	b5	58	74
b	31	a9	25	5b	86	97	15	f1	cb	67	f4	27	21	8b	7b	09
c	c7	6b	8a	e0	6f	fd	c2	37	80	a2	ad	be	ef	8f	7e	69
d	a6	c5	16	6a	f6	79	55	98	84	b8	45	35	d3	33	85	78
e	c0	de	9b	59	b6	75	72	9d	f5	03	e4	44	14	cc	1b	a7
f	d9	24	0b	00	11	12	c9	83	73	20	8e	47	46	2d	63	a3

Приведённые результаты показывают, что разработанный метод более чем на порядок эффективней метода основанного на генетическом алгоритме [204].

Таблица 4.10 – Криптографические свойства подстановки из таблицы 4.9

Свойства	Значение
Компонентные булевы функций	
Сбалансированность	Да
Нелинейность	104

Продолжение таблицы 4.10

$\lvert AC\rvert_{\max}$	80
Критерий распространения	0
Корреляционный иммунитет	0
SSI	194944
Минимальная степень	7
t-устойчивость	0
Строгий лавинный критерий	Нет
Подстановка	
Биективность	Да
МТД	8
МТЛА	24
Циклическая структура	123:6, 6:7, 0:243
AI/КУ/SP	3/441/0,825

После проверки первых 6-ти подстановок, 4 из них оказались КШЗ-неэквивалентными. Данное исследование свидетельствует о том, что предложенный метод может быть использован при формировании оптимальных узлов замены для перспективных блочных симметричных шифров [13, 59, 205].

В таблице 4.11 приведена сравнительная характеристика всех подстановок, приведённых в данной работе, с отличающимися свойствами. Данные результаты, а также значения из таблицы 4.12, являются дополнительными аргументами в пользу выбора критериев оптимальности приведённых в пункте 2.2.5.

Таблица 4.11 – Сравнение криптографических характеристик сгенерированных подстановок

Свойства	Значения			
	Таблица 3.1	Таблица 4.5	Таблица 4.7	Таблица 4.9
Нелинейность	100	102	104	104
$\lvert AC\rvert_{\max}$	96	88	80	80
SSI	267520	217600	186880	194944
МТД	8	8	8	8
МТЛА	28	26	24	24
Циклическая структура	34:5, 7:16, 11:44, 0:191	0:26, 1:230	2:5, 11:6, 69:6, 10:47, 0:192	123:6, 6:7, 0:243
AI/КУ/SP	3/441/0,822	3/441/0,830	2/1/0,518	3/441/0,828

Сравнительная характеристика S-блока из таблицы 4.9 («П»), подстановок блочных симметричных шифров AES («А»), СТБ 34.101.31-2011 («С»), ГОСТ Р 34.11-2012 («Г») и нелинейного узла замены S0 алгоритма «Калина», приведена в таблице 4.12.

Таблица 4.12 – Сравнение криптографических характеристик сгенерированной подстановки с известными

Свойства	Значения				
	«А»	«С»	«Г»	«S0»	«П»
Компонентные булевы функций					
Сбалансированность	Да	Да	Да	Да	Да
Нелинейность	112	102	100	96	104
$\lvert AC\rvert_{\max}$	32	88	96	88	80
SSI	133120	232960	258688	244480	194944

Продолжение таблицы 4.12

Минимальная степень	7	6	7	7	7
Подстановка					
МТД	4	8	8	8	8
МТЛА	16	26	28	32	24
Циклическая структура	115:2, 11:27, 0:59, 1:81, 4:87	22:7, 1:35, 3:78, 0:136	21:13, 0:243	85:4, 4:24, 1:41, 2:78, 0:109	123:6, 6:7, 0:243
AI/КУ/SP	2/39/0,687	3/441/0,823	3/441/0,824	3/441/0,826	3/441/0,828

Как видно из таблицы 4.12, подстановка «П» обладает наилучшими свойствами среди всех представленных с алгебраическим иммунитетом равным 3.

Дополнительно была проверена возможность генерации биективных S-блоков с нелинейностью 106 и более, алгебраическим иммунитетом 3, минимальной алгебраической степенью 7 и δ-равномерностью меньше 8. После 107 часов работы кластера, что эквивалентно 50 годам работы однопроцессорного компьютера, не было найдено ни одной такой подстановки. Это свидетельствует о невозможности нахождения таких подстановок практическим путём, используя известные методы. При этом остаётся открытым вопрос о существовании подстановок с показателями, описанными выше.

В таблице 4.13 приведена сравнительная характеристика нелинейных узлов замены по показателям оптимальности (см. подпункт 2.2) из стандартов FIPS 197 [20], ГОСТ Р34.11-2012 [60], СТБ 34.101.31-2011 [59], первым S-блоком (S0) БСШ «Калина» [13] и полученной подстановкой.

Таблица 4.13 – Сравнительная характеристика предложенного с существующими нелинейными узлами замены

Свойства	AES	СТБ 34.101.31 -2011	ГОСТ Р 34.11-2012	Клина S0	Табл. 4.9
δ -равномер-ность	4	8	8	8	8
Нелинейность	112	102	100	96	104
Минимальная степень	7	7	7	6	7
AI/КУ/SP	2/39/0,687	3/441/0,823	3/441/0,824	3/441/0,826	3/441/0,828

4.4 Оценка сложности криптоаналитических атак на примере шифра «Калина 128/128» с применением различных узлов нелинейной замены

Как известно нелинейный узел замен сам по себе не обеспечивает достаточный уровень стойкости криптоалгоритма к распространённым видам атак [15]. Расчёт сложности нахождения ключа шифрования необходимо проводить для конкретного алгоритма с фиксированными (определёнными) параметрами, такими как нелинейный слой, линейный слой, функция смешивания ключа и схема разворачивания ключа.

В качестве примера для расчёта сложностей был выбран алгоритм «Калина 128/128» (длина ключа и обрабатываемого блока равны 128 битам) [13], который был отмечен на открытом национальном конкурсе блочного симметричного шифра.

Для анализа были взяты подстановки из спецификации и сгенерировано несколько новых S-блоков с различными параметрами,

значения которых приведены в таблице 4.14. В колонке «Набор подстановок» введены следующие обозначения:

а) O – подстановка S0 шифра «Калина»;

б) C – случайный S-блок;

в) C_{max} – подстановка, полученная при помощи метода случайной генерации (см. подраздел 3.1);

г) D_{min} – нелинейный узел замены, который описывается системой уравнений 2-й степени, состоящей из одного уравнения;

д) A – подстановка, с алгебраическим иммунитетом 2 и небольшим количеством уравнений;

е) AES – S-блок сгенерирован на основе обратной функции (см. подраздел 3.2);

ж) П – нелинейный узел замены полученный при помощи предложенного метода.

Таблица 4.14 – Характеристики подстановок используемых при анализе шифра «Калина 128/128»

Набор подстановок	Характеристика				
	Нели-нейность	δ-равно-мерность	AI	SP	КУ
O	96	8	3	0,8115	441
C	90	10	3	0,8260	441
C_{max}	100	8	3	0,8281	441
D_{min}	100	8	2	0,5547	1
A	102	8	2	0,5300	13
AES	112	4	2	0,6873	39
П	104	8	3	0,8281	441

Напомним, что разреженность определяется отношением ненулевых мономов в системе уравнений, уписывающей S-блок, к общему (возможному) количеству мономов.

Для расчёта сложности дифференциальной и линейной атаки на шифр «Калина 128/128» воспользуемся методом приведённой в [126]. Напомним, что S-блок называется активным, если вероятность прохождения через него дифференциала отлична от 1 и в тоже время не равна 0. В [206, 207] БСШ рассматривается с заменой функции смешивания ключа со сложения по модулю 2^{32} на операцию XOR. В следствии чего, можно применить метод «широкого следа» для расчёта сложности дифференциальной атаки [39, 44, 126]. Десяти-цикловая версия алгоритма «Калина», с длиной обрабатываемого блока равным 128 бит, имеет $a^{(r)} = 63$ активных S-блока.

Традиционный подход к оценке стойкости БСШ к дифференциальному криптоанализу основан на подсчёте минимального числа активных подстановок в дифференциальной характеристике [39, 126]. Для того, чтобы вычислить верхнюю границу вероятности – необходимо максимальное значение вероятности ($p_{D_{\max}}$) прохождения ненулевой разности через нелинейный слой замены возвести в степень числа активных подстановок, то есть [206]:

$$P_{DC}^{(r)} = (p_{D_{\max}})^{a^{(r)}}, \tag{4.1}$$

где $a^{(r)}$ – минимальное количество активных S-блоков после *r* циклов шифра.

По аналогии вычисляется верхняя граница вероятности линейной характеристики [206]:

$$P_{LC}^{(r)} = (p_{L_{\max}})^{a^{(r)}}, \tag{4.2}$$

где $p_{L_{\max}}$ – максимальная вероятность линейной аппроксимации S-блока.

Для подсчёта сложности алгебраической атаки воспользуемся методом представленным в [36, 56, 72, 128, 134, 135]. Основная идея состоит в описании всего шифрующего преобразования в виде системы состоящей из m уравнений степени d с n неизвестными. Пусть $Nr = 10$ – количество циклов БСШ «Калина» [13]. Обозначим через *r* количество уравнений

степени d в системе, которая описывает нелинейный узел замены. Тогда исходя из спецификации алгоритма шифрования [13]:

$$n = 16 \cdot r \cdot Nr;$$
$$m = 128 \cdot 3 + 128 \cdot (Nr - 1),$$

где $\frac{128}{8} = 16$ – количество байт в текущем состоянии;

$128 \cdot 3$ – количество неизвестных битов ключа (см. схему разворачивания ключа [13, 37]);

$128 \cdot (Nr - 1)$ – количество неизвестных битов промежуточных состояний алгоритма шифрования.

Согласно [37, 56, 131, 132], ожидаемое количество уравнений после применения XL метода ровно $R = C_n^{D-d} \cdot m$, при этом количество неизвестных становиться $T = C_n^D$ (число сочетаний из n элементов по D), а система имеет степень D. Для того, чтобы система была разрешимой необходимо, чтобы выполнялось условие:

$$R \geq T \quad \Rightarrow \quad C_n^{D-d} \cdot m \geq C_n^D, \tag{4.3}$$

при этом количество линейно независимых уравнений было как минимум T. Другими словами, параметр D выбирается таким образом, чтобы выполнялось неравенство (4.3). Тогда нижняя граница сложности атаки рассчитывается по формуле:

$$O(T) = T^{\omega} = \left(C_n^D\right)^{\omega}, \tag{4.4}$$

где ω – экспонента решения системы линейных уравнений [131].

Значение ω зависит от вида системы и её разреженности [132]. В общем виде $\omega \leq 3$. Наиболее употребляемым значением в криптологии является $\omega = 2{,}376 \approx 2{,}4$ [131, 132, 181], однако существуют работы показывающие более низкие значения [183].

В таблице 4.15 и на рисунке 4.1 приведены расчёты сложности дифференциальной («Диф.»), линейной («Лин.») и алгебраической («XL»)

атак для различных подстановок (см. таблицу 4.14), используя формулы (4.1), (4.2) и (4.4). На рисунке 4.1 вертикальной линией обозначена сложность атаки полного перебора.

Таблица 4.15 – Значение сложностей атак на шифр «Калина 128/128» с различными нелинейными слоями

Набор подстановок	Сложность криптоанализа		
	Диф.	Лин.	XL
O	2^{315}	2^{252}	$2^{607,3}$
C	2^{294}	$2^{220,8}$	$2^{607,3}$
C_{max}	2^{315}	$2^{276,3}$	$2^{607,3}$
D_{min}	2^{315}	$2^{276,3}$	$2^{607,3}$
A	2^{315}	$2^{289,7}$	$2^{555,8}$
AES	2^{378}	2^{378}	$2^{361,1}$
П	2^{315}	$2^{304,3}$	$2^{607,3}$

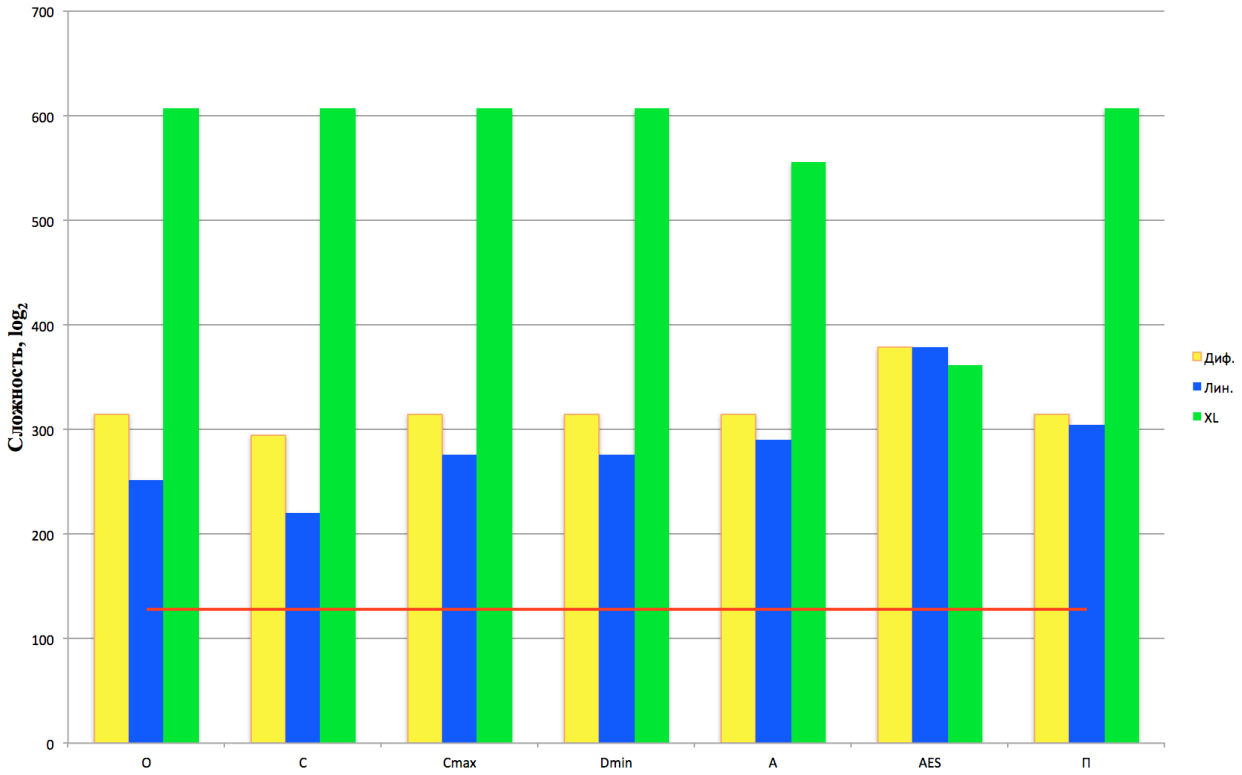

Рисунок 4.1 – Сравнение сложностей атак на шифр «Калина 128/128» с различными нелинейными слоями.

Изображённый выше рисунок показывает преимущества и недостатки различных подстановок. Узел нелинейной замены, полученный при помощи метода предложенного в 4.2, значительно увеличивает сложность линейного криптоанализа по сравнению с S-блоком из шифра «Калина», при этом сохраняет на высоком уровне сложности дифференциальной и алгебраической атак.

4.5 Выводы

Применение математического аппарата векторных булевых функций позволяет находить эффективные методы формирования нелинейных подстановочных конструкций для симметричных криптографических примитивов с оптимальными показателями.

Теорию, разрабатываемую в рамках векторных булевых функций, можно использовать для генерации долговременных ключевых элементов шифра ДСТУ ГОСТ 28147:2009. Предложенный быстрый метод генерации нелинейных узлов замены для данного БСШ обеспечивает формирование перестановок (принадлежащих различным аффинно-эквивалентным классам с нелинейностью 4, минимальной степенью 3, 4-равномерных и с отсутствием фиксированных точек), применение которых позволяет обеспечить значительный запас стойкости к известным видам криптоанализа.

Проведённые вычислительные эксперименты показали ограничения нахождения оптимальных подстановок при использовании метода случайной генерации. В то же время, комбинация алгебраических и эвристических методов позволяет преодолеть эти ограничения.

За основу предложенного в подпункте 4.2 метода был взят известный метод градиентного спуска для генерации булевых функции. В отличие от известного, где находились лишь отдельные булевы функции, предложенный метод позволяет формировать векторные булевы функции с заданными

параметрами. Таким образом, он может быть использован для генерации оптимальных нелинейных узлов замены, применяемых в современных блочных симметричных шифрах с требованиями к обеспечению высокого уровня стойкости.

5 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИССЛЕДОВАНИЯ И ГЕНЕРАЦИИ ПОДСТАНОВОК ДЛЯ СИММЕТРИЧНЫХ КРИПТОГРАФИЧЕСКИХ ПРЕОБРАЗОВАНИЙ

5.1 Анализ существующих программных средств и их недостатки

С целью изучения свойств и методов генерации подстановок, на первом этапе обычно проводится ряд исследований уже существующих методов и алгоритмов. Для решения данной задачи необходимо иметь инструменты для проведения тестирования.

Существует множество готовых решений [181, 196, 208, 209]. Однако у всех них есть свои ограничения. Например, класс SBox [208] из пакета mq в Sage [210] оптимизирован лишь для маленьких значений n. При увеличении размерности обрабатываемого блока (например, при $n = 8$), функции класса не возвращают ожидаемый результат. В качестве примера можно привести отсутствие системы уравнений второй степени для подстановки из блочного симметричного шифра AES (рис. 5.1) [131, 134].

Большинство других программ (библиотек) ориентированы на работу с ограниченным количеством свойств и/или лишь булевыми функциями. В рамках данного исследования рассматривалось наиболее общее представление подстановок с n битами входа и m битами выхода. Поэтому, было разработано новое программное обеспечение (ПО), с использованием уже известных оптимизированных алгоритмов, для анализа (n,m) векторных булевых функций.

Предложенная реализация написана как расширение функционала ПО Sage и представлена в виде библиотеки «Sbox» [211]. На сегодняшний день она позволяет проверять все криптографические свойства, описанные в разделе 1, для произвольной подстановки.

```
    sage: S
    (99, 124, 119, 123, 242, 107, 111, 197, 48,
1, 103, 43, 254, 215, 171, 118, 202, 130, 201,
125, 250, 89, 71, 240, 173, 212, 162, 175, 156,
164, 114, 192, 183, 253, 147, 38, 54, 63, 247,
204, 52, 165, 229, 241, 113, 216, 49, 21, 4, 199,
35, 195, 24, 150, 5, 154, 7, 18, 128, 226, 235,
39, 178, 117, 9, 131, 44, 26, 27, 110, 90, 160,
82, 59, 214, 179, 41, 227, 47, 132, 83, 209, 0,
237, 32, 252, 177, 91, 106, 203, 190, 57, 74, 76,
88, 207, 208, 239, 170, 251, 67, 77, 51, 133, 69,
249, 2, 127, 80, 60, 159, 168, 81, 163, 64, 143,
146, 157, 56, 245, 188, 182, 218, 33, 16, 255,
243, 210, 205, 12, 19, 236, 95, 151, 68, 23, 196,
167, 126, 61, 100, 93, 25, 115, 96, 129, 79, 220,
34, 42, 144, 136, 70, 238, 184, 20, 222, 94, 11,
219, 224, 50, 58, 10, 73, 6, 36, 92, 194, 211,
172, 98, 145, 149, 228, 121, 231, 200, 55, 109,
141, 213, 78, 169, 108, 86, 244, 234, 101, 122,
174, 8, 186, 120, 37, 46, 28, 166, 180, 198, 232,
221, 116, 31, 75, 189, 139, 138, 112, 62, 181,
102, 72, 3, 246, 14, 97, 53, 87, 185, 134, 193,
29, 158, 225, 248, 152, 17, 105, 217, 142, 148,
155, 30, 135, 233, 206, 85, 40, 223, 140, 161,
137, 13, 191, 230, 66, 104, 65, 153, 45, 15, 176,
84, 187, 22)
    sage: S.polynomials(degree=2)
[]
```

Рисунок 5.1 – Пример неправильного результата функции polynomials класса mq.SBox

Другим направлением практических исследований является нахождение оптимальных подстановок для применения в современных симметричных криптоалгоритмах. Очевидно, что для решения задач, связанных с формированием узлов нелинейной замены, необходим большой объём ресурсов. Следовательно, скриптовые языки, к которым относиться Sage [210], не подходят для решения данной задачи из-за низкой производительности. Самым распространённым высокопроизводительным языком программирования для распределённых вычислений на сегодняшний день является C/C++ [212]. Вследствие чего, для поиска оптимальных подстановок был выбран именно этот язык с использованием библиотеки «интерфейса передачи сообщений» (MPI) [213].

5.2 Библиотека проверки криптографических свойств подстановок «Sbox»

Данный подраздел описывает использование и специфику реализации библиотеки «Sbox», которая была написана с использованием системы компьютерной алгебры Sage [210]. Перед началом использования расширения рекомендуется ознакомиться с детальными инструкциями по установке, учебным пособием и справочным руководством Sage [210].

5.2.1 Общие сведения

Библиотека «Sbox» состоит из 20 файлов. Исходные коды основных из них приведены в приложении Б.1.

Для функционирования программы необходимо следующее программное обеспечение:

– операционная система (ОС) Windows, Linux или Mac OS X;

– система компьютерной алгебры Sage не ниже версии 5.8.

5.2.2 Функциональное назначение

Основной функционал библиотеки «Sbox» включает:

– генерацию подстановок, алгоритмы которых представлены в подразделах 3.1, 3.2, 4.2-4.4;

– нахождение эквивалентных функций (см. пункт 1.4.4);

– проверку на криптографические свойства произвольных векторных булевых функций (см. подраздел 1.4).

Дополнительно предусмотрена возможность добавления нового функционала. Это может быть актуально при нахождении более оптимальных алгоритмов генерации подстановок или подсчёта их

показателей. Две версии алгоритмов могут одновременно находится в библиотеке до тех пор, пока идёт разработка и тестирование одного из них.

5.2.3 Описание логической структуры

Общая блок-схема работы библиотеки представлена на рисунке 5.2.

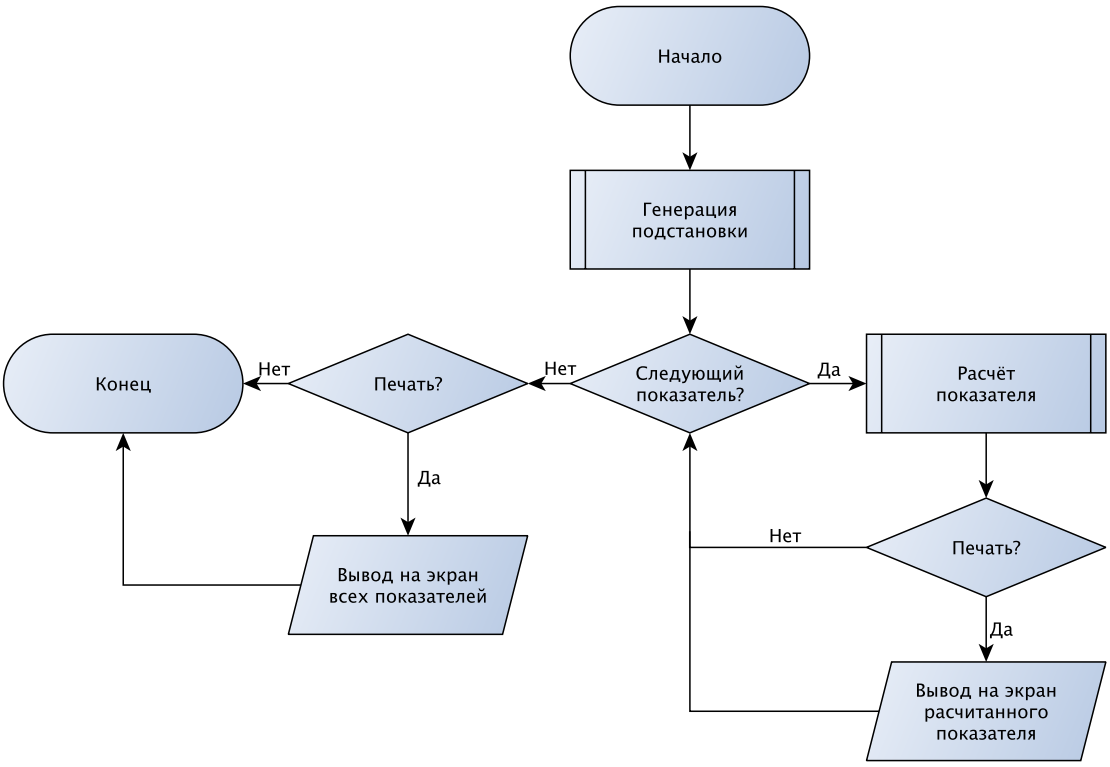

Рисунок 5.2 – Блок-схема библиотеки «Sbox»

Класс «Sbox» является основным объектом библиотеки «Sbox». Через него осуществляется взаимодействие со всеми другими функциями (включая C/C++) и внутренним представлением векторных булевых функций (подстановками).

Все методы класса Sbox можно разделить на 4 большие группы:

а) генерация подстановок (GSbox);

б) проверка криптографических свойств (CSbox);

в) общие методы, предназначенные для связи предыдущих двух;

г) методы расширения функциональности Sage (например, вычисление следа полинома над полем [48]).

Все функции из первого класса начинаются с «gen_». Большинство из них основаны на теоретических методах, приведённых в подразделе 3.2. Соответствие между всеми функциями реализованными в библиотеке и их аналогами представлено в таблице 5.1.

Таблица 5.1 – Соответствие между известными классами векторных булевых функций и их аналогами в библиотеке

Теоретические функции/методы	Аналоги с библиотеке «Sbox»	Источник
Голд	gen_gold	подраздел 3.2
Касами	gen_kasami	
Вэлш	gen_welch	
Нихо	gen_niho	
Обратные	gen_inverse	
Доббертин	gen_dobbertin	
ПСН перестановка для $n = 6$	gen_APN6	
Диксон	gen_dicson	[156]

Стоит отдельно отметить, что среди методов для генерации подстановок присутствует функции для нахождения КШЗ- и РА-эквивалентности. Сами по себе эквивалентности не позволяют находить функции с определёнными характеристиками, тем не менее, они являются одним из важных инструментов для поиска зависимостей и закономерностей уже существующих векторных булевых функций.

Для полноты и дополнительных экспериментов библиотека включает методы генерации случайных подстановок и перестановок, которые задаются

методами «gen_random_substitution», и «gen_random_permutation» соответственно.

Группа б) содержит методы начинающиеся с «cr_». Как и в группе а), большинство методов основаны на теоретических алгоритмах [43, 48, 156, 214, 215]. Вычисление некоторых показателей, например подсчёт алгебраического иммунитета, было оптимизировано в процессе научного исследования [211]. Таблица 5.2 отображает соответствие между свойствами из подраздела 1.4 и их аналогами в библиотеке «Sbox».

Таблица 5.2 – Соответствие между свойствами подстановок и функциями в библиотеке

Свойства подстановок	Аналоги в библиотеке «Sbox»
Сбалансированность	cr_balanced
Нелинейность	cr_nonlinearity
Абсолютный индикатор	cr_autocorrelation
Критерий распространения	cr_PC
Корреляционный иммунитет	cr_CI
SSI	cr_SSI
Минимальная алгебраическая степень	cr_minimum_degree
t-устойчивость	cr_resilient
Строгий лавинный критерий	cr_SAC
Биективность	cr_is_bijection
Максимум дифференциальной таблицы	cr_maximal_diff_table
Максимум таблицы линейных аппроксимаций	cr_maximal_linear_table
Циклы	cr_cycles
Алгебраический иммунитет	cr_algebraic_immunity_sbox

Очевидно, что группа данных методов должна обладать максимальной производительностью. Для преодоления данной проблемы были выбраны

встроенные решения в Sage, а именно использование комбинации Cython и C/C++ [212, 216].

Применяя такой подход можно объединить преимущества интерпретируемых и высокоэффективных компилируемых языков, тем самым добиться максимальной производительности при сохранении удобного интерфейса для конечного пользователя.

Большинство методов, описанных в таблице 5.2, были написаны на языках C/C++ в виде отдельных функций. Преимущества такого подхода заключаются в возможности использования функции не только в библиотеке «Sbox», но и в любых других программных продуктах (пример см. в подразделе 5.3).

Помимо проверки свойств, данная группа содержит ряд вспомогательных функций, таких как создание системы уравнений, описывающей подстановку [34, 57], или нахождение полинома, генерирующего подстановку [48, 156]. Краткое описание данных функций приведено в таблице 5.3.

Таблица 5.3 – Описание вспомогательных функций группы б)

Функция	Описание
cr_interpolation_polynomial	Нахождение полинома генерирующего подстановку.
cr_check_polynomial	Проверка на эквивалентность полинома и подстановки.
cr_create_system	Создание системы уравнений описывающей подстановку.
cr_check_system	Проверка уравнений сгенерированных функцией cr_create_system.
cr_is_APN	Проверка подстановки на ПСН.

Продолжение таблицы 5.3

cr_is_CCZ_equivalent	Проверка двух функций на КШЗ эквивалентность
cr_difference_distribution_matrix_full	Выводит все значения таблицы дифференциалов.

На рисунке 5.3 изображена иерархическая структура библиотеки «Sbox». Классы «Sbox», «GSbox», «CSbox» и «Test» написаны на языке Sage (Python).

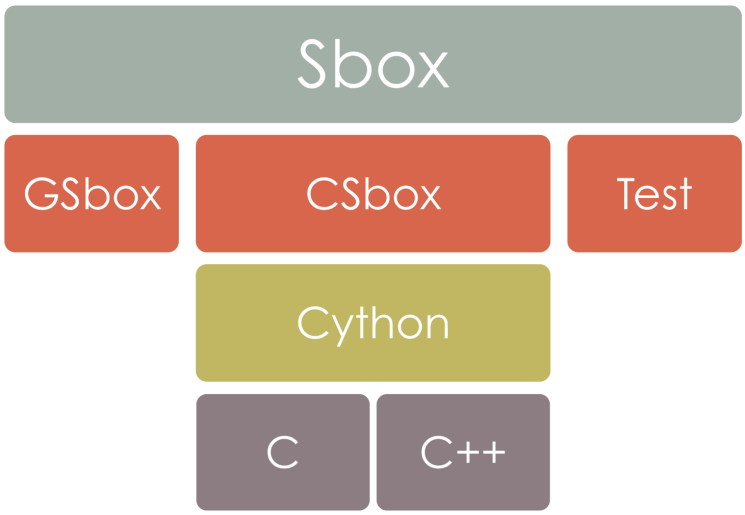

Рисунок 5.3 – Блок-схема библиотеки «Sbox»

Методы, представленные в группах в) и г), являются вспомогательными и, в большинстве случаев, не используются конечным пользователем. Для полноты описания библиотеки «Sbox», в таблице 5.4 приведены лишь самые важные функции из данных групп.

Таблица 5.4 – Описание основных функций групп в) и г)

Функция	Описание
generate_sbox	Функция вызова функций генерации подстановок.
get_field	Возвращает поле $\mathcal{F}_{2^{\max\{n,m\}}}$.
get_ring	Возвращает кольцо $\mathcal{F}_{2^{\max\{n,m\}}}[x]$.

Продолжение таблицы 5.4

get_mg	Возвращает мультипликативный генератор поля $\mathcal{F}_{2^{\max\{n,m\}}}$.
get_sbox	Возвращает значение подстановки.
get_polynomial	Возвращает значение полинома соответствующего подстановке
set_sbox	Задание подстановки для дальнейшего анализа.
Tr_pol	Вычисление следа полинома.
g2p	Преобразование полинома с мультипликативным элементом во внутреннее представление.
p2g	Обратная функция к g2p.

Как видно из таблицы, по названию методов можно определить их функциональное предназначение.

5.2.4 Используемые технические средства

В качестве тестовых стендов использовались компьютеры с 32- и 64-битными процессорами и различными операционными системами, характеристики которых приведены в таблице 5.5.

Таблица 5.5 – Характеристики тестовых компьютеров

Архитектура процессора	Центральный процессор	Операционная система	ОЗУ, ГБ
32 бит	Intel T2050, 1.60 Гц	Ubuntu 12.04 LTS	2
32 бит	Intel T2050, 1.60 Гц	Windows 7 SP1	2
64 бит	Intel i7 870, 2.93 Гц	Ubuntu 12.10	16
64 бит	Intel i7 3615QM 2.30 Гц	Mac OS X 10.8.3	8

Минимальные системные требования, необходимые для запуска библиотеки «Sbox», соответствуют требованиям, приведёнными в описании по компиляции и инсталляции Sage [210]. Работоспособность предложенного решения тестировалась на различных версиях Sage, начиная с версии 4.5. Текущая версия библиотеки была протестирована с последней доступной на сегодняшний день версией Sage 5.12 [210].

5.2.5 Вызов и загрузка

Чтобы использовать библиотеку необходимо:

а) установить систему компьютерной алгебры Sage;

б) скопировать файлы, относящиеся к библиотеке, на жёсткий диск;

в) из командной строки запустить файл «Main.sage».

В зависимости от поставленной задачи, необходимо применять различные комбинации функций, описанные в пункте 5.2.3.

5.2.6 Входные данные

Входными данными являются подстановки или векторные булевы функции, для которых необходимо посчитать характеристики.

5.3 Реализация эффективного алгоритма генерации оптимальных подстановок

В подразделе приводится описание эффективного программного обеспечения генерации оптимальных подстановок и изложены основные понятия параллельного программирования, необходимые для понимания внутренней логики работы программы.

5.3.1 Аппаратные средства распределённой высокопроизводительной системы

Высокопроизводительные вычисления (ВВ) довольно общее определение. Чаще всего данное понятие ассоциируется с суперкомпьютером или с распределёнными вычислениями [213, 217, 218]. ВВ являются эффективным, надёжным и высокоскоростным процессом параллельной обработки данных для решения прикладных задач. В качестве примера можно привести задачу о нахождении всех возможных решений в большой разреженной системе линейных уравнений над полем $\mathcal{F}_2$ [132, 134].

Одним из ключевых элементов в суперкомпьютерах, влияющем на производительность системы в целом, является организация модели памяти, которую можно разделить на два больших класса [219]. Первая из них называется моделью с общей памятью и представляет собой процессоры, соединённые с общедоступной (глобальной) памятью (рис. 5.4).

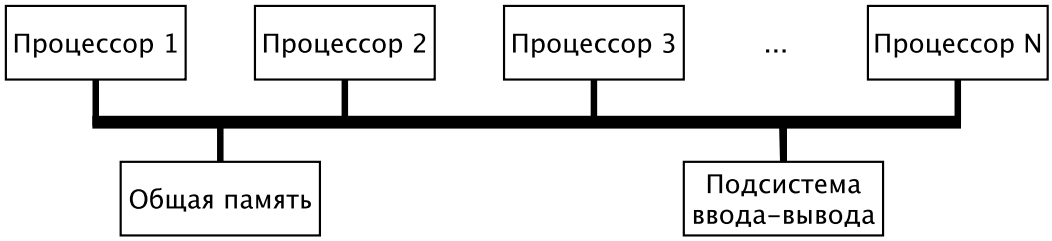

Рисунок 5.4 – Модель с общей памятью

Вторая модель основана на распределённой памяти, т.е. каждый процессор имеет собственную (выделенную) память (рис. 5.5).

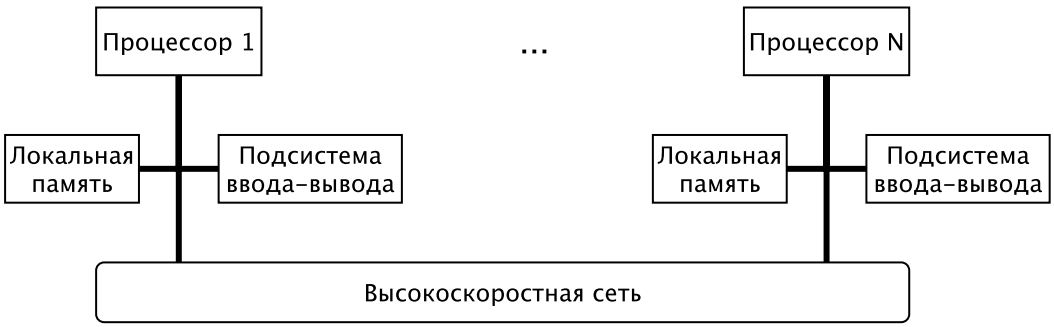

Рисунок 5.5 – Модель с распределённой памятью

Наиболее часто применяемой стала модифицированная модель второго типа, в которой каждый процессор имеет доступ не только к выделенной собственной памяти, но и к общей [219].

Организация памяти в системе Cray XE6m-200 [220, 221], которая была доступна для исследований, является распределённой, поэтому ПО было реализовано именно для данной модели. Для написание кода, в качестве библиотеки параллельного программирования, был взят интерфейс передачи сообщений (MPI).

При классической реализации алгоритма программы, в MPI используются сообщения вида MPI_Send и MPI_Receive, которые применяются для организации коммуникации между двумя процессами [213]. Дополнительно, в библиотеке присутствует функция MPI_Broadcast, которая позволяют передавать однотипную информацию для всех процессов одновременно.

Данная библиотека может быть использована с такими языками программирования как Фортран, C или C++ для обмена сообщениями, в то время как для последовательных операции используются встроенные в язык программирования команды циклов, условных выражений и вычислений. В данном случае предпочтение было отдано MPI C из-за простоты необходимых вычислений и более высокой производительности по сравнению с C++ [213].

Параллельное программирование может быть более сложным, чем последовательное [213, 219]. Оно приводит к необходимости решения новых задач, таких как состояние гонки и взаимной блокировки. Состояние гонки имеет место в том случае, когда разные (параллельные) потоки пытаются обновить одну и ту же переменную или одновременно записать данные в одну и ту же область памяти, включая память находящуюся на жёстком

диске. Взаимная блокировка возникает в тот момент, когда один поток обращается к другому, который не ожидает вызова.

Не все алгоритмы получают преимущества при использовании параллелизма. Некоторые из них строго последовательны, так как содержат блоки зависимых друг от друга вычислений. Повышение производительности от применения параллелизма в оптимальном случае должно быть линейным, т.е. увеличение количества процессорных элементов вдвое должно в два раза сокращать время выполнения [219].

Заметим, что для получения максимальной производительности программы, необходимо иметь эффективный алгоритм и сократить до минимума время на передачу сообщений между процессами.

5.3.2 Общие сведения

Исходные коды программы «maxNL», которая состоит из 11 файлов, приведены в приложении Б.2. Для её функционирования необходимо ОС Linux или Mac OS X с предустановленной библиотекой MPI.

5.3.3 Функциональное назначение

Основной функцией программы «maxNL» является генерация оптимальных подстановок на основе метода, описанного в подразделе 4.3. Помимо непосредственного использования ПО, возможна быстрая модификация функциональности с целью адаптации для других алгоритмов. Например, необходимо несколько минут для изменения на алгоритм генерации случайной подстановки, описанного в подразделе 3.1, с проверкой большинства критериев.

5.3.4 Описание логической структуры

Алгоритм работы программы «maxNL» представлен в виде блок-схемы на рисунке 5.6.

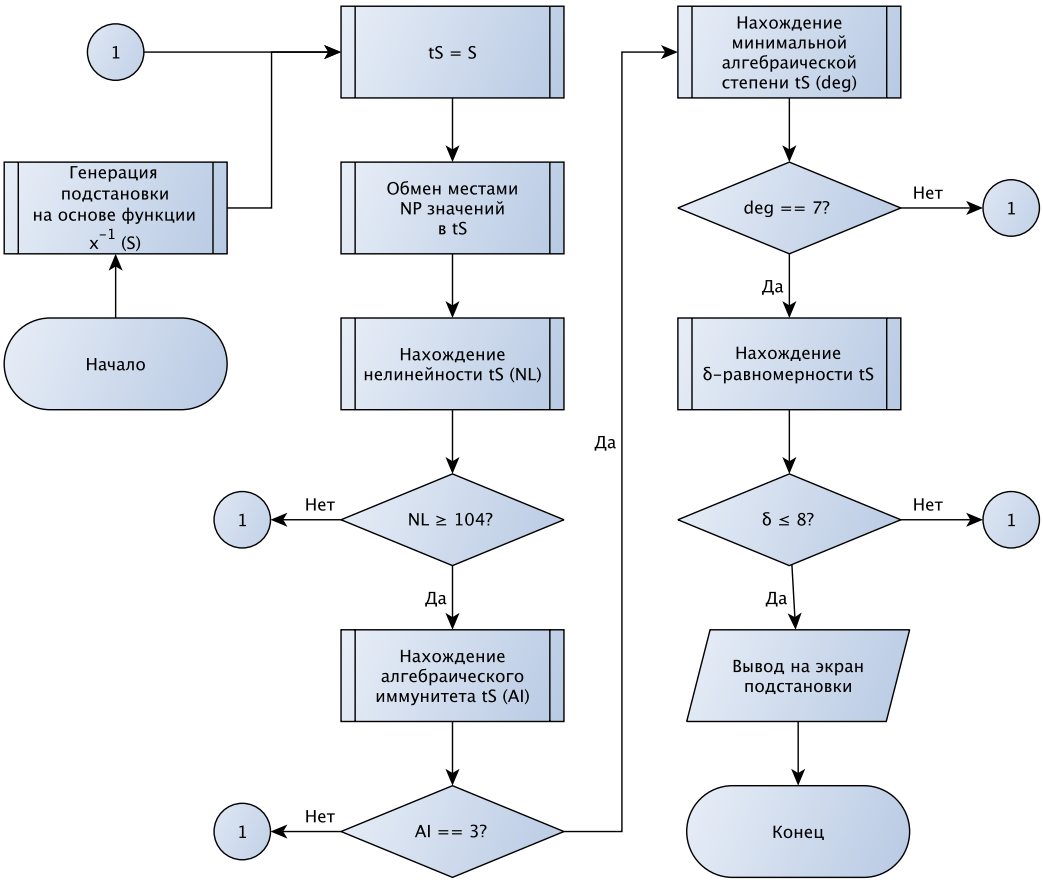

Рисунок 5.6 – Блок-схема программы «maxNL»

Как видно из рисунка, приведённая реализация основывается на нескольких подпрограммах. Во время разработки был выбран функциональный подход к реализации каждой такой части.

Все функций можно разделить на 4 большие группы относящиеся к:

а) нахождению криптографических характеристик отдельных булевых функций;

б) нахождению криптографических характеристик подстановок;

в) работе с матрицами;

г) вспомогательным.

Первые две группы включают в себя функции, непосредственно относящиеся к нахождению характеристик, представленные на рисунке 5.6.

Группа в) состоит из минимального набора функции, необходимых для нахождения алгебраического иммунитета (см. пункт 2.2.3). К ним относятся: приведение к ступенчатому виду по строкам; нахождение ранга и обратной матрицы; сложение и умножение матриц; дополнительные функции, необходимые для работоспособности предыдущих (например, обмен местами двух строк, сложение строки с константой, инициализация матрицы и т.д.).

Последняя группа из реализованных алгоритмов состоит из: преобразования чисел по основанию 10 в числа по основанию 2 и обратно, вычисления количества сочетаний, нахождения компонентных функций, генерации случайных чисел и др. Как можно заметить, перечисленные функции непосредственно не вычисляют показатели, однако необходимы для их нахождения.

Как было указано в пункте 5.3.1, для максимальной производительности программы необходимо до минимума свести взаимодействие между процессами. Очевидно, что подпрограммы алгоритма, представленного на рисунке 5.6, не зависят друг от друга и могут запускаться параллельно.

После проведённых экспериментальных исследований было выяснено, что максимальная производительность достигается в том случае, если всю программу от начала и до конца запускать на отдельном процессоре. Таким образом, в ПО «maxNL» генерация подстановок происходит независимо друг от друга, а MPI используется для запуска необходимого количества процессов на каждом ядре системы.

5.3.5 Используемые технические средства

Тестирование и отладка программы «maxNL» проводились на компьютере Mac Book Pro, который имеет следующие характеристики: процессор Intel i7 3615QM 2.30 Гц, 8 Гб оперативной памяти, твердотельный накопитель ёмкостью 256 Гб и операционную систему Mac OS X 10.8.3.

Запуск программы и получение оптимальных подстановок, описанных в подразделах 3.1, 4.2 и 4.3, проводилось на кластере «Hexagon» [220], который был построен на основе системы Cray XE6 [221]. Этот кластер обладает распределённой системой памяти и состоит из 696 узлов, каждый из которых имеет по два 16-ядерных процессора AMD «Interlagos» (2,3 ГГц) и 32 Гб оперативной памяти. Все узлы между собой соединены широкополосной сетью «Cray Gemini», которая обладает малым временем задержки [220, 221].

В соответствии с общими ограничениями [222], максимальный возможный объем ресурсов кластера «Hexagon» на одного пользователя составляет:

– 4096 процессорных ядер (общее количество для всех задач);

– до 8 одновременно запущенных задач;

– 2 бездействующие задачи (например, приостановленные).

5.3.6 Вызов и загрузка

Для запуска программы необходимо в командной строке:

а) выполнить компиляцию программы при помощи команды «make» или самостоятельно при помощи компилятора с поддержкой MPI;

б) ввести команду запуска MPI программы, например «mpiexec –n 4 maxNL», где параметр 4 означает количество одновременно запущенных процессов.

С целью максимизации производительности параметр «n», в зависимости от используемой системы, может варьироваться от $1 \cdot ncpu$ до $1,2 \cdot ncpu$, где $ncpu$ – количество процессоров в системе [222].

5.3.7 Входные данные

По умолчанию программа «maxNL» не требует никаких входных параметров. Подстановка, соответствующая векторной булевой функции x^{-1} для $n = 8$, была сгенерирована при помощи библиотеки «Sbox», описание которой приведено в подразделе 5.2.

При использовании другой векторной булевой функции необходимо изменить переменную «SboxInv» на значение новой подстановки.

Дополнительно стоит отметить, что программа рассчитана на работу с векторными булевыми функциями при $n = m$, значение которых задаётся через переменную $BITS$.

5.4 Практические выходные данные

В зависимости от способа использования библиотеки «Sbox», выходными параметрами могут быть:

а) сгенерированные векторные булевы функции;

б) рассчитанные показатели подстановок.

Очевидно, что выходные параметры могут быть объединены, т.е. можно сгенерировать подстановку и подсчитать её криптографические характеристики одновременно. Выходные данные, полученные с использованием библиотеки «Sbox», представлены на рисунке 5.7.

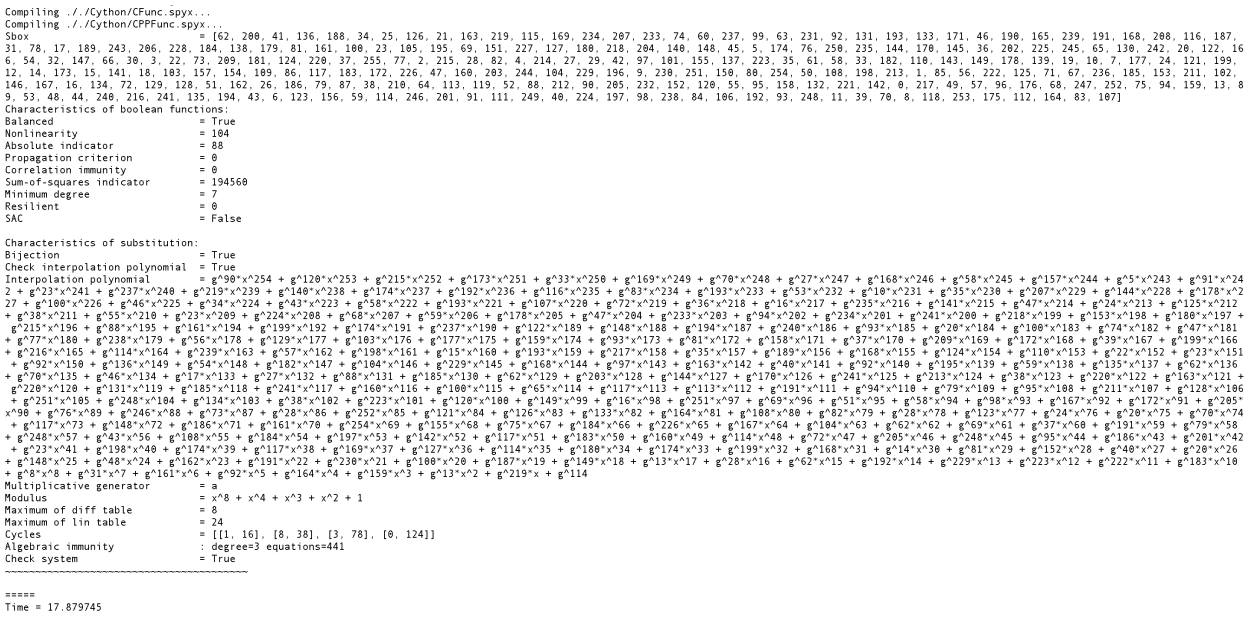

Рисунок 5.7 – Рассчитанные показатели для одной из подстановок, сгенерированных при помощи алгоритма из подраздела 4.3

В качестве входных параметров можно использовать результат программы «maxNL» (рис. 5.8), которая записывает значение оптимальной подстановки в файл «sboxNCPU.txt», где NCPU – номер ядра.

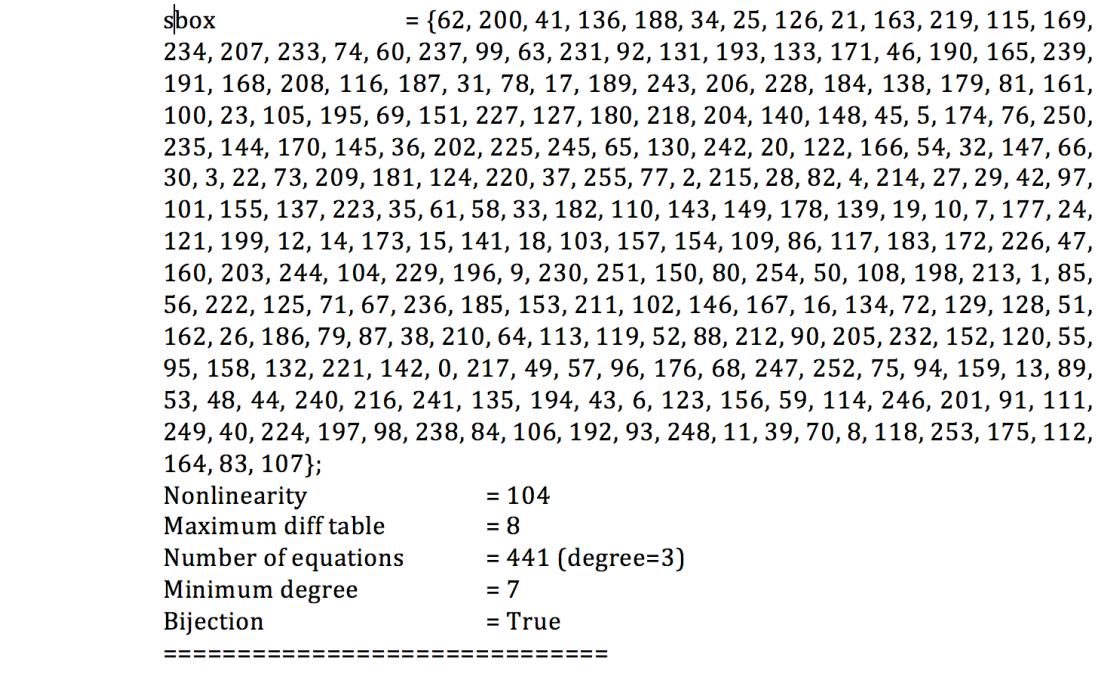

Рисунок 5.8 – Результат выполнения программы «maxNL»

Ещё один пример нелинейного узла замены и посчитанные для него характеристики представлены в таблицах 5.6 и 5.7 соответственно.

Таблица 5.6 – Пример оптимальной перестановки сгенерированной программой «maxNL»

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	fa	15	d7	59	cc	03	4a	46	4f	db	8a	39	6c	a4	d4	48
1	04	51	40	19	1d	2c	f9	bf	92	cd	93	25	0e	47	08	2a
2	2e	bb	76	95	87	b7	4e	75	44	21	af	8b	53	ba	1c	f8
3	f1	74	12	9d	5a	fb	9c	94	fd	7c	28	96	d3	50	71	d8
4	b6	ce	49	c7	e8	3e	f6	27	68	bc	33	69	fe	8e	77	a0
5	dd	56	b5	0a	23	1b	7a	be	d9	61	e3	81	b1	aa	d2	66
6	c0	c2	ec	14	a1	6a	5c	fc	3b	2d	88	cb	b2	98	1a	e4
7	8c	31	c6	37	df	9a	29	7f	ab	85	35	58	83	4b	79	11
8	82	5e	72	0f	07	ac	e7	6d	ad	64	13	c3	cf	ed	0b	b3
9	05	3c	06	9f	7d	a2	43	73	41	f7	e5	f5	89	f3	45	97
A	34	86	01	e9	f4	ae	0d	ee	09	26	6b	00	17	42	1f	6e
B	0c	e2	62	eb	54	dc	e6	91	8f	67	20	9b	52	a7	a6	b9
C	d1	80	65	d0	d6	a9	c4	90	57	8d	5b	c1	38	78	7b	d5
D	e1	f0	c8	60	ea	22	bd	70	2f	c5	b0	de	02	16	55	3f
E	9e	4d	f2	30	32	ef	c9	3d	b8	1e	18	7e	a3	3a	10	5f
F	b4	a5	a8	99	36	84	5d	63	4c	2b	6f	e0	ff	ca	da	24

Таблица 5.7 – Криптографические свойства подстановки из таблицы 5.6

Свойства	Значение
Компонентные булевы функций	
Сбалансированность	Да
Нелинейность	104
$\lvert AC\rvert_{\max}$	88
SSI	194560
Минимальная степень	7
Подстановка	
Биективность	Да
МТД	8
МТЛА	24
Циклическая структура	2:94, 0:162
AI/КУ/SP	3/441/

5.5 Выводы

Практическая реализация теоретически полученных результатов является, в большинстве случаев, трудоёмким процессом. Из подраздела 5.1 следует, что на сегодняшний день не существует достаточно быстрых и эффективных средств нахождения характеристик для произвольных подстановок.

В рамках проведённых исследований были улучшены алгоритмы нахождения некоторых свойств. В частности, был предложен, а позже и реализован, метод нахождения алгебраического иммунитета для произвольных n и m. Данный метод позволяет находить степень системы уравнений, описывающей S-блок, без необходимости её построения.

Реализованная библиотека «Sbox» включает в себя наибольшее количество методов генерации подстановок и функций (известных на сегодняшний день) для подсчёта их свойств. Более того, её архитектура предусматривает расширение функционала.

Доказательство практической сложности генерации оптимальных подстановок возможно лишь с использованием больших вычислительных ресурсов, таких как кластер. Оптимизация и нахождение высокопроизводительных параллельных алгоритмов является одной из сложнейших задач в данной области. В рамках исследования методов быстрой генерации оптимальных подстановок было показано, что задача не является последовательной и, следовательно, может рассматриваться как множество более мелких подзадач, выполняемых параллельно. Данный результат приводит к уменьшению времени генерации S-блоков.

ВЫВОДЫ

Исследования, проведённые в работе, позволили решить одну из актуальных научно-исследовательских задач разработки теоретической и практической базы проверки криптографических свойств и генерации нелинейных узлов замены для перспективных и существующих средств криптографической защиты информации с оптимальными показателями.

Результаты проведённого анализа показывают, что основными критериями для подстановок, применяемых в БСШ, являются биективность, отсутствие фиксированных точек, δ-равномерность, минимальная степень, алгебраический иммунитет и нелинейность. Стоит отметить, что перспективный алгебраический криптоанализ ещё не до конца изучен, и границы его применения не выяснены. Тем не менее, анализ шифров, представленных на украинский конкурс, показал, что стойкость шифра к алгебраической атаке зависит не только от алгебраического иммунитета, но и от количества уравнений и разрежённости системы, описывающей криптоалгоритм. Таким образом, был предложен расширенный критерий алгебраического иммунитета, который позволяет отбирать нелинейные узлы замены, обеспечивающие максимальную защиту от алгебраической атаки.

Более того, перспективные шифры вносят дополнительные требования к S-блокам. Одним из таких требований является принадлежность всех подстановок, используемых в одном алгоритме шифрования, к различным классам эквивалентности. Соответствие данному критерию приводит к уменьшению количества изоморфных представлений раундовой функции и всего шифрующего преобразования. Вследствие чего, возникает необходимость в нахождении эквивалентных преобразований, которые могут быть использованы для построения изоморфных представлений.

В ходе работы были предложены новые методы проверки на

эквивалентность двух нелинейных отображений. Данные методы основаны на методе преобразования линейной функции, заданной над полем $\mathcal{F}_{2^n}$, в матричную форму. Последний отличается от известных полиномиальной сложностью. Вариации данных методов позволяет находить изначальные представления высокоуровневых конструкций таких как ГОСТ Р 34.11-2012.

Предложенный метод генерации нелинейных узлов замены для перспективных блочных симметричных шифров основан на комбинации теории векторных булевых функций и эвристическом методе генерации S-блоков. Он позволяет генерировать подстановки с улучшенными показателями алгебраического иммунитета и нелинейности при малых затратах ресурсов. В частности, применение метода генерации оптимальных нелинейных узлов замены для национальных симметричных алгоритмов шифрования при $n = 8$ позволяет получить перестановки с отсутствием фиксированных точек и показателями:

- δ-равномерность 8;
- нелинейность 104;
- минимальная степень 7;
- алгебраический иммунитет 3.

Из чего следует, что применение таких подстановок в отмеченном алгоритме на национальном конкурсе по выбору перспективного алгоритма шифрования «Калина» позволяет увеличить нелинейность с 96 до 104 при сохранении всех остальных показателей.

Применение теории векторных булевых функций на практике приводит к разработке метода, сокращающего время генерации долговременных ключевых элементов для шифра ДСТУ ГОСТ 28147:2009 до 1 с. Каждый ДКЭ состоит из подстановок, принадлежащих различным РА-эквивалентным классам. При этом каждый S-блок обладает следующими характеристиками:

а) биективностью;

б) 4-равномерностью;

в) нелинейностью 4;

г) минимальной степенью 3;

д) отсутствием фиксированных точек (нет циклов длиной 1).

Такие показатели обеспечивают высокий уровень стойкости к дифференциальному и линейному криптоанализам. Дополнительно была определена верхняя граница таких ДКЭ, которая равна 2^{51}.

Таким образом, главным научным результатом работы следует считать разработку методов генерации оптимальных подстановок для перспективных симметричных криптоалгоритмов и долговременных ключевых элементов для блочного симметричного шифра ДСТУ ГОСТ 28147:2009.

Основным практическим результатом является разработка программного обеспечения для генерации и проверки криптографических свойств нелинейных узлов замены, которое позволяет формировать базовые параметры для схем симметричного криптопреобразования, что приводит к возможности применения разработанных методов для решения задач обеспечения безопасности в информационно-телекоммуникационных системах Украины.

Показатели подстановок, полученные с использованием разработанных методов, значительно превосходят аналоги, применяемые в стандартах СТБ 34.101.31-2011, ГОСТ Р 34.11-2012 и отмеченного на национальном конкурсе по выбору перспективного алгоритма шифрования «Калина». Данные методы могут быть использованы при решении задач генерации параметров средств криптографической защиты информации, находящихся на стадии разработки, и с целью улучшения национальных стандартов шифрования.

ПЕРЕЧЕНЬ ССЫЛОК

1. Про основи національної безпеки України [Текст] : закон України від 19 черв. 2003 р. № 964-IV // Офіційний вісник України. – 2003. – № 29. – С. 38.

2. Про Доктрину інформаційної безпеки України [Текст] : указ Президента України від 8 лип. 2009 р. № 514 // Офіційний вісник України. – 2009. – № 52. – С. 7.

3. Горбенко І. Д. Прикладна криптологія [Текст] / І. Д. Горбенко, Ю. І. Горбенко. – Х. : Форт, 2012. – 870 с.

4. Булевы функции в теории кодирования и криптологии [Текст] / О. Логачев, А. Сальников, С. Смышляев, В. Ященко ; Ин-т проблем информ. безопасности МГУ. – 2-е изд., доп. – М. : МЦНМО, 2012. – 583 с.

5. Menezes A. J. Handbook of applied cryptography [Text] / Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. – Boca Raton : CRC press, 2010. – 816 p.

6. ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования [Текст]. – Введ. 01–07–1990. – К. : Изд-во стандартов, 1989. – 28 с.

7. ДСТУ ГОСТ 28147:2009. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования [Текст]. – Взамен ГОСТ 28147–89 ; введ. 01–02–2009. – К. : Изд-во стандартов, 2009.

8. Горбенко И. Д. Критерии отбора случайных таблиц подстановок для алгоритма шифрования по ГОСТ 28147–89 [Текст] / И. Д. Горбенко, И. В. Лисицкая // Радиотехника : Всеукр. межвед. науч.-техн. сб. – 1997. – Вып. 103.

9. Courtois N. T. Security Evaluation of GOST 28147–89 in view of international standardisation [Text] / N. T. Courtois // Cryptologia. – 2012. – Vol. 36, № 1. – P. 2–13.

10. Moharir M. TrueCrypt: An Innovative Approach for Hard Disk Security [Text] / M. Moharir, A. V. Suresh // Advances in Computational Sciences and Technology. – 2010. – Vol. 3, № 3. – P. 305–312.

11. Олейников Р. В. Исследование свойств подстановок ГОСТ 28147–89, построенных на основе анализа свойств координатных функций. [Текст] / Р. В. Олейников, И. В. Лисицкая // Правове, нормативне та метрологичне забезпечення системи захисту інформації в Україні : наук.-техн. зб. – К, 2002. – № 5.

12. Положення про проведення відкритого конкурсу криптографічних алгоритмів [Електронний ресурс] / ДССЗІУ. – Режим доступу : www. URL: http://www.dstszi.gov.ua/dstszi/control/uk/publish/article?art_ id=48383&cat_id=38710. – 04.07.2013.

13. Перспективний блоковий симетричний шифр «Калина» – основні положення та специфікація [Текст] / І. Д. Горбенко, В. І. Долгов, Р. В. Олійников [та ін.] // Прикладная радиоэлектроника. – Х., 2007. – Т. 6, № 2. –С. 195–208.

14. Обгрунтування вимог та розробка основних рішень з побудування та властивості перспективного БСШ «Мухомор» [Текст] / М. Ф. Бондаренко, І. Д. Горбенко, В. І. Долгов [та ін.] // Прикладная радиоэлектроника. – Х., 2007. – Т. 6, № 2. –С. 174–185.

15. Daemen J. AES Proposal: Rijndael, AES algorithm submission [Electronic resource] / J. Daemen, V. Rijmen. – Mode of access : www. URL: http://csrc.nist.gov/archive/aes/index.html.

16. PRESENT: An ultra-lightweight block cipher [Text] / A. Bogdanov et al. // Cryptographic Hardware and Embedded Systems – CHES 2007 / ed. P. Paillier, I. Verbauwhede. – Berlin ; Heidelberg : Springer, 2007. – P. 450–466. –

(Lecture Notes in Computer Science ; vol. 4727).

17. PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications [Text] / J. Borghoff et al. // Advances in Cryptology – ASIACRYPT 2012. – Berlin ; Heidelberg : Springer, 2012. – P. 208–225. – (Lecture Notes in Computer Science ; vol. 7658).

18. Dworkin M. Recommendation for Block Cipher Modes of Operation: Methods and Techniques [Electronic resource] : NIST Special Publication 800-38A 2001 / M. Dworkin. – Mode of access : www. URL: http://www.researchgate.net/publication/235183820_Recommendation_for_Block_Cipher_Modes_of_Operation._Methods_and_Techniques.

19. Gligor V. D. Fast encryption and authentication: XCBC encryption and XECB authentication modes [Text] / V. D. Gligor, P. Donescu // Fast Software Encryption. – Berlin ; Heidelberg : Springer, 2002. – P. 92–108.

20. Advanced Encryption Standard (AES) (FIPS PUB 197) [Electronic resource]. – 26.11.2001 – Mode of access : www. URL: http://techheap.packetizer.com/cryptography/encryption/fips-197.pdf.

21. Олейников Р. В. Построение переопределенной системы уравнений для описания алгоритма шифрования AES [Текст] / Р. В. Олейников, А. И. Шумов, В. И. Руженцев // Безпека інформації в інформаційно-телекомунікаційних системах : матеріали VII Міжнар. наук.-практ. конф., 12–14 трав. 2004 р. – К., 2004.

22. Казимиров А. В. Сравнение функций разворачивания ключа симметричных блочных шифров [Текст] / А. В. Казимиров, Р. В. Олейников, // Защита информации : сб. науч. тр. / НАУ. – К., 2010. – Вып. 17. – С. 162–165.

23. Courtois N. T. Cryptanalysis of block ciphers with overdefined systems of equations [Text] / N. T. Courtois, J. Pieprzyk // Advances in Cryptology – ASIACRYPT 2002 : proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New

Zealand, December 1–5, 2002. – Berlin ; Heidelberg : Springer, 2002. – P. 267–287. – (Lecture Notes in Computer Science ; vol. 2501).

24. Biryukov A. Distinguisher and related-key attack on the full AES-256 [Text] / A. Biryukov, D. Khovratovich, I. Nikolić // Advances in Cryptology – CRYPTO'99 : proceedings of the 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 1999. – Berlin ; Heidelberg : Springer, 1999. – P. 231–249. – (Lecture Notes in Computer Science ; vol. 1666).

25. Bogdanov A. Biclique cryptanalysis of the full AES [Text] / A. Bogdanov, D. Khovratovich, C. Rechberger // Advances in Cryptology – ASIACRYPT 2011 : proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4–8, 2011. – Berlin ; Heidelberg : Springer, 2011. – P. 344–371. – (Lecture Notes in Computer Science ; vol. 7073).

26. Results of Ukrainian national public cryptographic competition [Text] / R. V. Oliynykov, I. D. Gorbenko, V. I. Dolgov, V. I. Ruzhentsev // Tatra Mt. Math. Publ. – 2010. – Vol. 47. – P. 99–114.

27. Горбенко И. Д. Стандартизация алгоритмов шифрования. Требования к проекту национального стандарта блочного симметричного шифрования на современном этапе развития криптографии о стандартах шифрования [Текст] / И. Д. Горбенко, И. В. Лисицкая // Радиотехника : Всеукр. межвед. науч.-техн. сб. – Х., 2011. – Вып. 66. – С. 5–10.

28. Announcing development of a federal information processing standard for advanced encryption standard [Electronic resource]. – 1997. – Mode of access : www. URL: http://csrc.nist.gov/archive/aes/pre-round1/aes_9701.txt.

29. Принципи побудування та властивості блокового симетричного шифру «Калина» [Текст] / І. Д. Горбенко, В. І. Долгов, Р. В. Олійников [та ін.] // Прикладная радиоэлектроника. – 2007. – Т. 6, № 2. – С. 209–216.

30. Головашич С. А. Алгоритм блочного симметричного шифрования «Лабиринт» [Текст] / С. А. Головашич. – Х., 2007. – 46 с.

31. Бабаш А. В. Криптография [Текст] / А. В. Бабаш, Г. П. Шамкин. – М. : СОЛОН-Р, 2002. – 512 с.

32. Nyberg K. Perfect nonlinear S-boxes [Text] / K. Nyberg // Advances in Cryptology – EUROCRYPT '91 : proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8–11, 1991. – Berlin ; Heidelberg : Springer, 1991. – P. 378–386. – (Lecture Notes in Computer Science ; vol. 547).

33. Carlet C. Highly nonlinear mappings [Text] / C. Carlet, C. Ding // Journal of Complexity. – 2004. – Vol. 20, № 2–3. – P. 205–244.

34. Олейников Р. В. Построение переопределенной системы уравнений для описания алгоритма шифрования «Лабиринт» [Текст] / Р. В. Олейников, А. В. Казимиров // Прикладная радиоэлектроника. – 2009. – Т. 8, № 3. – С. 247–251.

35. Efficient Algorithms for solving Overdefined System of Multivariate Polynomial Equations [Text] / N. Courtis, A. Klimov, J. Patarin, A. Shamir // Advances in Cryptology – EUROCRYPT 2000 : proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14–18, 2000. – Berlin ; Heidelberg : Springer-Verlag, 2000. – P. 392–407. – (Lecture Notes in Computer Science ; vol. 1807).

36. Courtis N. T. Higher Order Correlation Attacks, XL Algorithm and Cryptanalysis of Toyocrypt [Electronic resource] / N. T. Courtois. – Mode of access : www. URL: http://eprint.iacr.org/2002/087.pdf.

37. Казимиров А. В. Алгебраические свойства схемы разворачивания ключей блочного симметричного шифра «Калина» [Текст] / А. В. Казимиров, Р. В. Олейников // Радіоелектронні і комп'ютерні системи. – Х. : ХАІ, 2010. – № 5 (46). – С. 61–66.

38. Shannon C. E. Communication Theory of Secrecy Systems [Text] / C. E. Shannon // Bell System Technical Journal. – 1949. – Vol. 28. – P. 656–715.

39. Daemen J. The design of Rijndael. AES – The Advanced Encryption

Standard [Text] / J. Daemen, V. Rijmen. – Berlin : Springer-Verlag, 2002. – 238 p.

40. Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms [Electronic resource] / K. Aoki, T. Ichikawa, M. Kanda et al. – Mode of access : www. URL: https://www.cosic.esat.kuleuven.be/nessie/updatedPhase2 Specs/camellia/camellia-v2.pdf.

41. New block cipher: ARIA [Text] / D. Kwon et al. // Information Security and Cryptology – ICISC 2003 : proceedings of the 6th International Conference, Seoul, Korea, November 27–28, 2003. – Berlin ; Heidelberg : Springer, 2004. – P. 432–445. – (Lecture Notes in Computer Science ; vol. 2971).

42. Design of block ciphers and coding theory [Text] / D. Kwon et al. // Trends in Mathematics. – 2005. – Vol. 8, № 1. – P. 13–20.

43. Schneier B. Applied Cryptography [Text] / B. Schneier. – New York : Wiley, 1996.

44. Biham E. Differential Cryptanalysis of the Data Encryption Standard [Text] / E. Biham, A. Shamir. – Berlin ; Heidelberg : Springer-Verlag, 1993.

45. Matsui M. Linear Cryptanalysis Method for DES Cipher [Electronic resource] / M. Matsui. – Mode of access : www. URL: http://homes.esat .kuleuven.be/~abiryuko/Cryptan/matsui_des.PDF.

46. Авдошин С. М. Криптоанализ: современное состояние и перспективы развития [Текст] / С. М. Авдошин, А. А. Савельева // Информационные технологии. – 2007. – № 3. – С. 1–32.

47. Rueppel R. A. Stream ciphers [Text] / R. A. Rueppel // Contemporary Cryptology: The Science of Information Integrity / ed. G. Simmons. – New York : IEEE Press, 1991. – P. 65–134.

48. Carlet C. Vectorial Boolean Functions for Cryptography [Text] / C. Carlet // Boolean Models and Methods in Mathematics, Computer Science, and Engineering / ed. Y. Crama, P. Hammer. – Cambridge : Cambridge University Press, 2010. – P. 398–469.

49. Budaghyan L. Verification of restricted EA-equivalence for vectorial

boolean functions [Text] / L. Budaghyan, O. Kazymyrov // Arithmetic of Finite Fields : 4th International Workshop, WAIFI 2012, Bochum, Germany, July 16–19, 2012. – Berlin ; Heidelberg : Springer, 2012. – P. 108–118. – (Lecture Notes in Computer Science ; vol. 7369).

50. Kazymyrov O. Verification of restricted EA-equivalence for vectorial boolean functions [Text] / O. Kazymyrov, L. Budaghyan // Сиситемы обработки информации. – Х., 2013. – Вып. 1 (108). – С. 155–160.

51. Олейников Р. В. Выбор S-блоков для симметричных криптографических алгоритмов на основе анализа алгебраических свойств [Текст] / Р. В. Олейников, А. В. Казимиров // Вісн. Харк. нац. ун-ту. Сер. Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. – Х., 2010. – № 925. – С. 79–86.

52. Казимиров А. В. Сравнение производительности функций разворачивания ключа блочных симметричных алгоритмов шифрования [Текст] / А. В. Казимиров // Радиоэлектроника и молодежь в XXI веке : материалы 14-го междунар. молодежного форума, 18–20 марта 2010 г. – Х. : ХНУРЭ, 2010. – Ч. 2. – С. 82.

53. Исследование циклических и дифференциальных свойств уменьшенной модели шифра «Лабиринт» [Текст] / В. И. Долгов, И. В. Лисицкая, А. В. Григорьев, А. В. Широков // Прикладная радиоэлектроника. – 2009. – Т. 8, № 3. – С. 283–289.

54. Казимиров А. В. Использование векторных функций при генерации подстановок для симметричных криптографических преобразований [Текст] / А. В. Казимиров, Р. В. Олейников // Системи обробки інформації. – 2012. – № 6 (104). – С. 97–102.

55. Долгов В. И. Вариации на тему шифра Rijndael [Текст] / В. И. Долгов, И. В. Лисицкая, А. В. Казимиров // Прикладная радиоэлектроника. – 2010. – Т. 9, № 3. – С. 321–325.

56. Казимиров А. В. Построение переопределённой системы уравне-

ний для описания алгоритма шифрования «Лабиринт» [Текст] / А. В. Казимиров, Р. В. Олейников // Прикладная радиоэлектроника. – 2009. – Т. 8, № 3. – С. 247–251.

57. Oliynykov R. An Impact Of S-Box Boolean Function Properties To Strength Of Modern Symmetric Block Ciphers [Text] / R. Oliynykov, O. Kazymyrov // Радиотехника : Всеукр. межвед. науч.-техн. сб. – 2011. – Вып. 166 : Информационная безопасность. – С. 11–16.

58. Казимиров А. В. Метод построения нелинейных узлов замены на основе градиентного спуска [Текст] / А. В. Казимиров, Р. В. Олейников // Радиотехника : Всеукр. межвед. науч.-техн. сб. – 2013. – Вып. 172 : Информационная безопасность. – С. 104–108.

59. СТБ 34.101.31–2011. Информационные технологии и безопасность. Защита информации. Криптографические алгоритмы шифрования и контроля целостности [Текст]. – Взамен СТБ П 34.101.31–2007 ; введ. 31–01–2011. – Минск, 2011. – 35 с.

60. ГОСТ Р 34.11–2012. Информационная технология. Криптографическая защита информации. Функция хэширования [Текст]. – Взамен ГОСТ Р 34.11–94 ; введ. 01–01–2013. – М. : Стандартинформ, 2012.

61. Kam J. B. Structure and design of substitution-permutation encryption networks [Text] / J. B. Kam, G. I. Davida // IEEE Trans. Comput. – 1979. – Vol. 28, № 10. – P. 747–753.

62. Новая идеология оценки стойкости блочных симметричных шифров к атакам дифференциального и линейного криптоанализа [Текст] / И. Д. Горбенко, В. И. Долгов, И. В. Лисицкая, Р. В. Олейников // Прикладная радиоэлектроника. – 2010. – Т. 9, № 3. – С. 312–320.

63. Казимиров А. В. Подход к криптоанализу блочного симметричного шифра «Лабиринт» [Текст] / А. В. Казимиров, Р. В. Олейников, А. Б. Небывайлов // Прикладная радиоэлектроника. Состояние и перспективы развития : материалы 3-го междунар. радиоэлектрон. форума (МРФ'2008), 22–24

окт. 2008 г. / АНПРЭ, ХНУРЭ. – Х. : ХНУРЭ, 2008. – Т. 5 : Междунар. конф. "Информационные компьютерные технологии и системы" (ИКТС–2008). – С. 262–263.

64. Казимиров А. Сравнение функций разворачивания ключа симметричных блочных шифров [Текст] / А. Казимиров, Олейников Р. // Захист інформації в інформаційно-комунікаційних системах : матеріали наук.-практ. конф., 24–26 трав. 2010 р. – К., 2010. – С. 162–165.

65. Казимиров А. В. Програмний комплекс для аналізу властивостей підставних конструкцій на диференційні показники криптографічних перетворень [Текст] / А. В. Казимиров // Радиоэлектроника и молодежь в XXI веке : материалы 14-го междунар. молодежного форума, 18–20 марта 2010 г. – Х. : ХНУРЭ, 2010.

66. Казимиров А. В. Подходы к формированию подстановок с оптимальными показателями [Текст] / А. В. Казимиров // Радиоэлектроника и молодежь в XXI веке : материалы XV Юбилейного Междунар. молодежного форума, 18–20 апр. 2011 г. – Х. : ХНУРЭ, 2011. – Т. 5. – С. 155.

67. Олейников Р. Алебраическая атака на модифицированный вариант алгоритма "Лабиринт" [Текст] / Р. Олейников, А. Казимиров // Безпека інформації в інформаційно-телекомунікаційних системах : матеріали XII Міжнар. наук.-практ. конф., 19–22 трав. 2009 р. – К., 2009. – С. 29.

68. Олейников Р. Выбор S-блоков для симметричных криптографических алгоритмов на основе анализа алгебраических свойств [Текст] / Р. Олейников, А. Казимиров // Компьютерное моделирование в наукоемких технологиях : материалы науч.-техн. конф. с междунар. участием (КМНТ–2010), 18–21 мая 2010 г. – Х. : Изд-во ХНУ. – Ч. 2. – С. 177–179.

69. Горбенко И. Выбор узлов нелинейного преобразования на основе анализа алгебраических свойств подстановок [Текст] / И. Горбенко, Р. Олейников, А. Казимиров // Безопасность информации в информационно-телекомуникационных системах : тез. докл. XIII Междунар. науч.-практ.

конф., 18–21 мая 2010 г. – К., 2010. – С. 36.

70. Анализ усовершенствований шифра Rijndael [Текст] / И. Лисицкая, А. Казимиров, Е. Мельничук и др. // Безопасность информации в информационно-телекомуникационных системах : тез. докл. XIII Междунар. науч.-практ. конф., 18–21 мая 2010 г. – К., 2010. – С. 42.

71. Олейников Р. Сравнение функций разворачивания ключа симметричных блочных шифров [Текст] / Р. Олейников, А. Казимиров // Захист інформації в інформаційно-комунікаційних системах : матеріали наук.-практ. конф., 24–26 трав. 2010 р. – К., 2010. – С. 138.

72. Олейников Р. В. Алгебраические свойства схемы разворачивания ключей блочного симметричного шифра Калина [Текст] / Р. В. Олейников, А. В. Казимиров // Гарантоздатні системи, сервіси та технології (DESSERT 2010) : матеріали 5-ї Міжнар. наук.-техн. конф. – Харків ; Полтава ; Кіровоград, 2010.

73. Лисицкая И. В. Об участии S-блоков в формировании максимальных значений дифференциальных вероятностей блочных симметричных шифров [Текст] / И. В. Лисицкая, А. В. Казимиров // Proceedings International Conference SAIT 2011, May 23–28, 2011, Ukraine. – Kyiv, 2011 – P. 460.

74. Казимиров А. В. Анализ графа обратных состояний шифра Mickey [Текст] / А. В. Казимиров, Р. В. Олейников // Перспективи розвитку інформаційних та транспортно-митних технологій у митній справі, зовнішньоекономічній діяльності та управлінні організаціями : матеріали Міжнар. наук.-практ. конф., 2 груд. 2011 р. – Дніпропетровськ, 2011. – С. 211–212.

75. Казимиров А. В. Восстановление ключей шифра ГОСТ 28147 на основе слайд атаки [Текст] / А. В. Казимиров // Наука и социальные проблемы общества: информатизация и информационные технологии : тез. докл. VI Междунар. науч.-практ. конф., 24–25 мая 2011 г. – Х., 2011. – С. 272–273.

76. Казимиров А. В. Генерация подстановок на основе векторных

функций [Текст] / А. В. Казимиров // Безопасность информации в информационно-телекомуникационных системах : тез. докл. XV Междунар. науч.-практ. конф., 22–25 мая 2013 г. – К., 2013.

77. Budaghyan L. Verification of restricted EA-equivalence for vectorial boolean functions [Text] / L. Budaghyan, O. Kazymyrov // Arithmetic of Finite Fields : 4th International Workshop, WAIFI 2012, Bochum, Germany, July 16–19, 2012. – Berlin ; Heidelberg : Springer, 2012. – P. 108–118. – (Lecture Notes in Computer Science ; vol. 7369).

78. О создании эффективных программных реализаций отечественных криптографических стандартов [Электронный ресурс] / А. В. Казимиров, С. В. Смышляев, С. Е. Леонтьев, В. О. Попов // РусКрипто'2013 : материалы XV науч.-практ. конф., 27–30 марта 2013 г., Россия, Моск. обл. – Режим доступа : http://www.ruscrypto.ru/accotiation/archive/rc2013/.

79. Kazymyrov V. A Method For Generation Of High-Nonlinear S-Boxes Based On Gradient Descent [Text] / V. Kazymyrov, O. Kazymyrov, R. Oliynykov // Second workshop on Current Trends in Cryptology (CTCrypt 2013), Ekaterenburg, Russian, June 23–24, 2013. – Ekaterenburg, 2013. – P. 107–115.

80. Kazymyrov V. Algebraic Aspects of the Russian Hash Standard GOST R 34.11–2012 [Text] / V. Kazymyrov, O. Kazymyrov // Second workshop on Current Trends in Cryptology (CTCrypt 2013), Ekaterenburg, Russian, June 23–24, 2013. – Ekaterenburg, 2013. – P. 160–176.

81. Kazymyrov V. Extended Criterion for Absence of Fixed Points [Text] / V. Kazymyrov, O. Kazymyrov // Second workshop on Current Trends in Cryptology (CTCrypt 2013), Ekaterenburg, Russian, June 23–24, 2013. – Ekaterenburg, 2013. – P. 177–191.

82. Kazymyrov O. Prototype of Russian Hash Function "Stribog" [Text] / O. Kazymyrov // Winter School in Information Security, Finse, Norway. – 2012.

83. Kazymyrov O. Prototype of Russian Hash Function "Stribog" [Text] / O. Kazymyrov // ECRYPT II Summer School on Tools, May 28 – June 1, 2012,

Mykonos, Greece.

84. Kazymyrov O. Open Problems in the Generation Substitution Field [Text] / O. Kazymyrov // Ice Break 13, June 6–12, 2013, Iceland. – Reykjavik, 2013.

85. Kazymyrov O. Extended Criterion for Absence of Fixed Points [Text] / O. Kazymyrov // Прикладная радиоэлектроника. – 2013. – Т. 12, № 2. – С. 209–214.

86. Казимиров А. В. Оценка количества допустимых внутренних состояний в поточном алгоритме Mickey [Текст] / А. В. Казимиров, Р. В. Олейников // Прикладная радиоэлектроника. – 2011. – Т. 10, № 2. – С. 112–115.

87. Казимиров А. В. Криптоанализ шифра Mickey на основе анализа внутренних состояний [Текст] / А. В. Казимиров, Р. В. Олейников // Радиотехника : Всеукр. межвед. науч.-техн. сб. – Х., 2012. – Вып. 171 : Информационная безопасность. – С. 24–28.

88. State space cryptanalysis of the MICKEY cipher [Text] / T. Helleseth, C. J. A. Jansen, O. Kazymyrov, A. Kholosha // Information Theory and Applications Workshop (ITA), Feb. 10–15, 2013, San Diego, CA. – P. 1–10.

89. FIPS 46–3. Data Encryption Standard (DES) [Electronic resource] / National Bureau of Standards, USA. – 1993. – Mode of access : www. URL: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

90. Kocher P. Differential power analysis [Text] / P. Kocher, J. Jaffe, B. Jun // Advances in Cryptology – CRYPTO 99 : proceedings of the 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 1999. – Berlin ; Heidelberg : Springer, 1999. – P. 388–397. – (Lecture Notes in Computer Science ; vol. 1666).

91. Final report of European project number IST-1999-12324, named New European Shames for Signatures, Integrity, and Encryption [Electronic resource] / B. Preneel, A. Biryuov, C. De Canniere et al. – Mode of access : www. URL: https://www.cosic.esat.kuleuven.be/nessie/Bookv015.pdf.

92. CRYPTREC Report 2002 [Electronic resource]. – Mode of access : http://cryptrec.nict.go.jp/eng_info_page/cryptrec_03_0829_c02_report.htm.

93. Isobe T. A single-key attack on the full GOST block cipher [Text] / T. Isobe // Journal of cryptology. – 2013. – Vol. 26, № 1. – P. 172–189.

94. Dinur I. Improved attacks on full GOST [Text] / I. Dinur, O. Dunkelman, A. Shamir // Fast Software Encryption. – Berlin ; Heidelberg : Springer, 2012. – P. 9–28.

95. Алексеев Е. К. ГОСТ 28147–89: «Не спеши его хоронить». Часть 1. Стойкость алгоритма [Электронный ресурс] / Е. К. Алексеев, С. В. Смышляев. – Режим доступа : http://www.cryptopro.ru/blog/2013/08/27/gost-28147-89-ne-speshi-ego-khoronit-chast-1-stoikost-algoritma. – 02.09.2013.

96. New stream cipher designs: the eSTREAM finalists [Text] / ed. M. Robshaw, O. Billet. – Berlin ; Heidelberg : Springer-Verlag, 2008. – (Lecture Notes in Computer Science ; vol. 4986).

97. ECRYPT: The home page eSTREAM, the ECRYPT Stream Cipher Project [Electronic resource]. – Mode of access : www. URL: http://www.ecrypt.eu.org/stream/. – 02.09.2013.

98. The eSTREAM Portfolio (rev. 1) [Electronic resource] / Steve Babbage, Christophe De Canni`ere, Anne Canteaut et al. – Mode of access : www. URL: http://www.ecrypt.eu.org/stream/portfolio_revision1.pdf. – 02.09.2013.

99. Cryptographic Hash Project [Electronic resource] / National Institute of Standards and Technology. – Mode of access : www. URL: http://csrc.nist.gov/groups/ST/hash/index.html. – 02.09.2013.

100. FIPS PUB 180–4 [Electronic resource] : federal information processing standards publication / National Institute of Standards and Technology. – March 2012. – Mode of access : www. URL: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf.

101. Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition [Electronic resource] / R. Perlner et al. ; US Department of Com-

merce, National Institute of Standards and Technology. – 2012. – Mode of access : www. URL: http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf.

102. «Стрибог». Информационная технология. Криптографическая защита информации. Функция хэширования [Электронный ресурс]. – Режим доступа : www. URL: http://infotecs.ru/laws/gost/proj/gost3411.pdf. – 02.09.2013.

103. Матюхин Д. В. Перспективный алгоритм хэширования [Электронный ресурс] / Д. В. Матюхин, В. Рудской // РусКрипто'2010 : материалы науч.-практ. конф., 1–4 апр. 2010 г., Россия, Моск. обл. – Режим доступа : www. URL: http://www.ruscrypto.ru/accotiation/archive/rc2010/.

104. Лебедев П.А. Сравнение старого и нового стандартов РФ на криптографическую хэш-функцию на ЦП и графических процессорах Nvidia [Электронный ресурс] / П.А. ЛЕБЕДЕВ. – Режим доступа : www. URL: http://paco2012.ipu.ru/procdngs/F108.pdf. – 02.09.2013.

105. ГОСТ Р 34.11–94. Информационная технология. Криптографическая защита информации. Функция хэширования. – Введ. 01–01–1995. – М., 1994. – 20 с.

106. Kerckhoffs A. La Cryptographie Militaire [Electronic resource] / Auguste Kerckhoffs. – Mode of access : www. URL: http://www.petitcolas.net/fabien/kerckhoffs/la_cryptographie_militaire_i.htm. – 02.09.2013.

107. Брюс Ш. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке Си [Текст] / Ш. Брюс. – 2-е изд. – М. : Триумф, 2002.

108. Daemen J. The design of Rijndael: AES-the advanced encryption standard [Text] / J. Daemen, V. Rijmen. – Berlin ; Heidelberg : Springer, 2002.

109. Carter G. Key schedule classification of the AES candidates [Text] / G. Carter, E. Dawsony, L. Nielseny // Proceedings of the end AES Conference, Rome, Italy. – Rome, 1999. – P. 1–14.

110. Golomb S. W. Shift Register Sequences [Text] / S. W. Golomb. – San Francisco : Holden-Day, 1967.

111. Feistel H. Cryptography and Computer Privacy [Text] / H. Feistel // Scientific American. – 1973. – Vol. 228. – P. 15–23.

112. Rueppel R. A. Analysis and design of stream ciphers [Text] / R. A. Rueppel. – New York : Springer-Verlag, Inc., 1986.

113. Robshaw M. J. B. Stream ciphers [Electronic resource] / M. J. B. Robshaw // Technical Notes and Reports / RSA Laboratories. – 1995. – Mode of access : www. URL: http://www.networkdls.com/Articles/tr-701.pdf.

114. Selmer E. S. Linear recurrence relations over finite fields [Text] / Ernst S. Selmer. – Bergen, 1966. – 424 p.

115. Stinson D. R. Cryptography: theory and practice [Text] / D. R. Stinson. – Boca Raton : CRC Press, Inc., 2006. – 434 p.

116. Papadimitriou C. H. Computational complexity [Text] / Christos H. Papadimitriou. – Ventura, CA : Academic Internet Publ., 2007. – 49 p.

117. Damgård I. B. Design Principle for Hash Functions [Text] / I. B. Damgård // Advances in Cryptology – CRYPTO 1989 : proceedings of the 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 1989 / ed. G. Brassard. – Heidelberg : Springer, 1990. – P. 416–427. – (Lecture Notes in Computer Science ; vol. 435.)

118. Merkle R. One way hash functions and DES [Text] / R. Merkle // Advances in Cryptology – CRYPTO 1989 : proceedings of the 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 1989 / ed. G. Brassard. – Heidelberg : Springer, 1990. – P. 428–446. – (Lecture Notes in Computer Science ; vol. 435.)

119. Merkle-Damgård revisited: How to construct a hash function [Text] / J. S. Coron et al. // Advances in Cryptology – CRYPTO 2005 : proceedings of the 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14–18, 2005. – Berlin ; Heidelberg : Springer, 2005. – P. 430–448. –

(Lecture Notes in Computer Science ; vol. 3621).

120. Keccak sponge function family main document [Electronic resource] / G. Bertoni et al. – Mode of access : http://keccak.noekeon.org/Keccak-main-2.1.pdf.

121. Bellare M. Pseudorandom functions revisited: The cascade construction and its concrete security [Text] / M. Bellare, R. Canetti, H. Krawczyk // Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, Oct. 14–16, 1996. – Burlington, 1996. – P. 514–523.

122. Joux A. Multicollisions in iterated hash functions. Application to cascaded constructions [Text] / A. Joux // Advances in Cryptology – CRYPTO 2004 : proceedings of the 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 2004. – Berlin ; Heidelberg : Springer, 2004. – P. 306–316. – (Lecture Notes in Computer Science ; vol. 3152).

123. Nandi M. Multicollision attacks on some generalized sequential hash functions [Text] / Mridul Nandi, Douglas R. Stinson // IEEE Transactions on Information Theory. – 2007. – Vol. 53, № 2. – P. 759–767.

124. Kelsey J. Herding hash functions and the Nostradamus attack [Text] / J. Kelsey, T. Kohno // Advances in Cryptology – CRYPTO 2006 : proceedings of the 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 2006. – Springer Berlin Heidelberg, 2006. – P. 183–200. – (Lecture Notes in Computer Science ; vol. 4117).

125. Dodis Y. Salvaging merkle-damgård for practical applications [Text] / Y. Dodis, T. Ristenpart, T. Shrimpton // Advances in Cryptology – CRYPTO 99 : proceedings of the 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 1999. – Berlin ; Heidelberg : Springer, 1999. – P. 371–388. – (Lecture Notes in Computer Science ; vol. 1666).

126. Knudsen L. R. The block cipher companion [Text] / L. R. Knudsen, M. J. B. Robshaw. – New York : Springer, 2011.

127. Nyberg K. Linear approximation of block ciphers [Text] / K. Nyberg

// Advances in Cryptology – EUROCRYPT '94 : proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9–12, 1994. – Berlin ; Heidelberg : Springer, 1995. – P. 439–444. – (Lecture Notes in Computer Science ; vol. 950).

128. Kleiman E. The XL and XSL attacks on Baby Rijndael [Electronic resource] / E. Kleiman ; Iowa State University. – Iowa, 2005. – Mode of access : http://orion.math.iastate.edu/dept/thesisarchive/MS/EKleimanMSSS05.pdf.

129. Courtois N. T. Algebraic Cryptoanalysis of the Data Encryption Standart [Electronic resource] / N. T. Courtois, G. V. Bard. – Mode of access : www. URL: http://eprint.iacr.org/2006/402.pdf.

130. Courtois N. T. Algebraic and slide attacks on KeeLoq [Text] / N. T. Courtois, G. V. Bard, D. Wagner // Fast Software Encryption. – Berlin ; Heidelberg : Springer, 2008. – P. 97–115.

131. Bard G. V. Algebraic cryptanalysis [Text] / G. V. Bard. – Dordrecht : Springer, 2009.

132. Albrecht M. Alorithmic Algebraic Techniques and Their Application to Block Cipher Cryptanalysis [Electronic resource] / M. Albrecht ; University of London. – 2010. – Mode of access : http://www.sagemath.org/files/thesis/albrecht-thesis-2010.pdf.

133. Weinmann R. Evaluating Algebraic Attacks on the AES [Electronic resource] / R. Weinmann. – Mode of access : www. URL: http://www.coderpunks.org/rpw/diplomarbeit.pdf.

134. Kleiman E. High Performance Computing techniques for attacking reduced version of AES using XL and XSL methods [Text] : doctor thesis / E. Kleiman. – 2010.

135. Courtois N. T. Algebraic Cryptanalysis of the Data Encryption Standard [Electronic resource] : report 2006/402 / Nicolas T. Courtois, Gregory V. Bard // Cryptology ePrint Archive. – 2006. – Mode of access : www. URL: http://eprint. iacr. org/2006/402.

136. Nyberg K. Differentially uniform mappings for cryptography [Text] / K. Nyberg // Advances in Cryptology – Eurocrypt '93 : proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23–27, 1993. – Berlin : Springer-Verlag, 1993. – P. 55–64. – (Lecture Notes in Computer Science ; vol. 765).

137. Biham E. New types of cryptanalytic attacks using related keys [Text] / E. Biham // Journal of Cryptology. – 1994. – Vol. 7, № 4. – P. 229–246.

138. Biryukov A. Distinguisher and related-key attack on the full AES-256 [Text] / A. Biryukov, D. Khovratovich, I. Nikolić // Advances in Cryptology – CRYPTO 2009 : proceedings of the 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16–20, 2009. – Berlin ; Heidelberg : Springer, 1999. – P. 231–249. – (Lecture Notes in Computer Science ; vol. 5677).

139. The related-key rectangle attack–application to SHACAL-1 [Text] / J. Kim et al. // Information Security and Privacy. – Berlin ; Heidelberg : Springer, 2004. – P. 123–136.

140. Biham E. A related-key rectangle attack on the full KASUMI [Text] / E. Biham, O. Dunkelman, N. Keller // Advances in Cryptology – ASIACRYPT 2005 : proceedings of the 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4–8, 2005. – Berlin ; Heidelberg : Springer, 2005. – P. 443–461. – (Lecture Notes in Computer Science ; vol. 3788).

141. Biryukov A., Khovratovich D. Related-key Cryptanalysis of the Full AES-192 and AES-256 [Text] / A. Biryukov, D. Khovratovich // Advances in Cryptology – ASIACRYPT 2009 : proceedings of the 5th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6–10, 2009. – Berlin ; Heidelberg : Springer, 2009. – P. 1–18. – (Lecture Notes in Computer Science ; vol. 5912).

142. Wagner D. The boomerang attack [Text] / D. Wagner // Fast Software Encryption. – Berlin ; Heidelberg : Springer, 1999. – P. 156–170.

143. Biryukov A. The boomerang attack on 5 and 6-round reduced AES [Text] / A. Biryukov // Advanced Encryption Standard – AES : proceedings of the 4th International Conference, AES 2004, Bonn, Germany, May 10–12, 2004. – Berlin ; Heidelberg : Springer, 2005. – P. 11–15. – (Lecture Notes in Computer Science ; vol. 3373).

144. Khovratovich D. Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family [Text] / D. Khovratovich, C. Rechberger, A. Savelieva // Fast Software Encryption. – Berlin ; Heidelberg : Springer, 2012. – P. 244–263.

145. Bogdanov A. Biclique cryptanalysis of the full AES [Text] / A. Bogdanov, D. Khovratovich, C. Rechberger // Advances in Cryptology – ASIACRYPT 2011 : proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4–8, 2011. – Berlin ; Heidelberg : Springer, 2011. – P. 344–371. – (Lecture Notes in Computer Science ; vol. 7073).

146. Hong D. Biclique attack on the full HIGHT [Text] / D. Hong, B. Koo, D. Kwon // Information Security and Cryptology – ICISC 2011 : proceedings of the 14th International Conference, Seoul, Korea, November 30 – December 2, 2011. – Berlin ; Heidelberg : Springer, 2012. – P. 365–374. – (Lecture Notes in Computer Science ; vol. 7259).

147. Biryukov A. Slide Attacks [Text] / A. Biryukov, D. Wagner // Proceedings of the 6th International Workshop on Fast Software Encryption (FSE '99). – Rome : Springer-Verlag, 1999. – P. 245–259.

148. Biryukov A. Advanced slide attacks [Text] / A. Biryukov, D. Wagner // Advances in Cryptology – EUROCRYPT 2000 : proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14–18, 2000. – Berlin ; Heidelberg : Springer, 2000. – P. 589–606. – (Lecture Notes in Computer Science ; vol. 1807).

149. A practical attack on KeeLoq [Text] / S. Indesteege et al. // Advances in Cryptology – EUROCRYPT 2008 : proceedings of the 7th Annual International

Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13–17, 2008. – Berlin ; Heidelberg : Springer, 2008. – P. 1–18. – (Lecture Notes in Computer Science ; vol. 4965).

150. Biham E. Improved slide attacks [Text] / E. Biham, O. Dunkelman, N. Keller // Fast Software Encryption. – Berlin ; Heidelberg : Springer, 2007. – P. 153–166.

151. Saarinen M. J. A chosen key attack against the secret S-boxes of GOST [Electronic resource] / M. J. Saarinen // Unpublished manuscript. – 1998. – Mode of access : www. URL: http://citeseer. ist. psu. edu/saarinen98chosen. html.

152. Свами М. Графы, сети и алгоритмы [Текст] : пер. с англ. / М. Свами, К. Тхуласираман. – М. : Мир, 1984. – 455 с.

153. Carlet C. Boolean functions for cryptography and error correcting codes [Text] / C. Carlet // Boolean Models and Methods in Mathematics, Computer Science, and Engineering. – 2010. – Vol. 2. – P. 257.

154. Maitra S. Autocorrelation properties of correlation immune Boolean functions [Text] / S. Maitra // Progress in Cryptology – INDOCRYPT 2001 : proceedings of the Second International Conference on Cryptology in India, Chennai, India, December 16–20, 2001. – Berlin ; Heidelberg : Springer, 2001. – P. 242–253. – (Lecture Notes in Computer Science ; vol. 2247).

155. Zhang X.-M. GAC – the criterion for global avalanche characteristics of cryptographic functions [Text] / X.-M. Zhang, Y. Zheng // Journal of Universal Computer Science. – 1995. – Vol. 1, № 5. – P. 316–333.

156. Lidl R. Finite Fields [Text] / R. Lidl, H. Niederreiter. – Cambridge : Cambridge University Press, 1997. – 755 p. – (Encyclopedia of Mathematics and its Applications ; vol. 20).

157. Carlet C. Codes, bent functions and permutations suitable for DES-like cryptosystems [Text] / C. Carlet, P. Charpin, V. Zinoviev // Designs, Codes and Cryptography. – 1998. – Vol. 15, № 2. – P. 125–156.

158. Budaghyan L. The equivalence of almost bent and almost perfect non-

linear functions and their generalizations [Text] : PhD Thesis / L. Budaghyan. – Otto-von-Guericke-Universität Magdeburg, 2005.

159. Rudskoy V. On zero practical significance of "Key recovery attack on full GOST block cipher with zero time and memory" [Electronic resource] / V. Rudskoy. – Mode of access : www. URL: http://eprint.iacr.org/2010/111.pdf.

160. Babbage S. The MICKEY stream ciphers [Text] / S. Babbage, M. Dodd // New Stream Cipher Designs. – Berlin ; Heidelberg : Springer, 2008. – P. 191–209. – (Lecture Notes in Computer Science ; vol. 4986).

161. Jansen C. J. A. Cascade jump controlled sequence generator and Pomaranch stream cipher [Text] / C. J. A. Jansen, T. Helleseth, A. Kholosha // New Stream Cipher Designs. – Berlin ; Heidelberg : Springer, 2008. – P. 224–243. – (Lecture Notes in Computer Science ; vol. 4986).

162. De Canniere C. Trivium [Text] / C. De Canniere, B. Preneel // New Stream Cipher Designs. – Berlin ; Heidelberg : Springer, 2008. – P. 244–266. – (Lecture Notes in Computer Science ; vol. 4986).

163. A stream cipher proposal: Grain-128 [Text] / M. Hell, T. Johansson, A. Maximov, W. Meier // 2006 IEEE International Symposium on Information Theory, USA, July 9–14, 2006. – Seattle, WA, 2006. – P. 1614–1618.

164. Jansen Cees J. A. The State Space Structure of the MICKEY Stream Cipher [Electronic resource] / Cees J. A. Jansen // Proceedings of the 32nd Symposium on Information Theory in the Benelux, Brussels, Belgium, May 10–11, 2011. – Mode of access : www. URL: http://www.w-i-c.org/upload/files/wicsp2011_proceedings.pdf.

165. Jansen Cees J. A. Stream Cipher Design Based on Jumping Finite State Machines [Electronic resource] : report 2005/267 / Cees J. A. Jansen // Cryptology ePrint Archive. – 2005. – Mode of access : www. URL: http://eprint.iacr.org/2005/267.

166. Coppersmith D. The Data Encryption Standard and its strength against attacks [Text] / D. Coppersmith // IBM Journal of Research and Develop-

ment. – 1994. – Vol. 38, № 3. – P. 243–250.

167. Jakobsen T. The interpolation attack on block ciphers [Text] / T. Jakobsen, L. R. Knudsen // Fast Software Encryption. – Berlin ; Heidelberg : Springer, 1997. – P. 28–40.

168. Лисицкая И. В. К вопросу построения долговременных ключей для алгоритма ГОСТ-28147-89 [Текст] / И. В. Лисицкая // Информационно-управляющие системы на железнодорожном транспорте. – 1997. – № 3. – С. 54–57.

169. Горбенко И. Д. Анализ стойкости алгоритма ГОСТ 28147-89 при использовании подстановок случайного типа [Текст] / И. Д. Горбенко, И. В. Лисицкая, А. С. Коряк // Радиоэлектроника и информатика. – 1998. – № 1 (02). – С. 39–43

170. Kwangjo K. I. M. A study on the construction and analysis of substitution boxes for symmetric cryptosystems [Electronic resource] : thesis / K. I. M. Kwangjo. – 1990. – Mode of access : www. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.5755&rep=rep1&type=pdf.

171. Adams C. The structured design of cryptographically good S-boxes [Text] / C. Adams, S. Tavares // Journal of Cryptology. – 1990. – Vol. 3, № 1. – P. 27–41.

172. Hinoue T. The security of RC6 against asymmetric chi-square test attack [Text] / T. Hinoue, A. Miyaji, T. Wada // Information and Media Technologies. – 2007. – Vol. 2, № 4. – P. 1052–1061.

173. Knudsen L. R. Correlations in RC6 with a reduced number of rounds [Text] / L. R. Knudsen, W. Meier // Fast Software Encryption : proceedings of the 7th International Workshop, FSE 2000, New York, NY, USA, April 10–12, 2000. – Berlin ; Heidelberg : Springer, 2001. – P. 94–108. – (Lecture Notes in Computer Science ; vol. 1978).

174. Courtois N. T. Algebraic attacks on stream ciphers with linear feed-

back [Text] / N. T. Courtois, W. Meier // Advances in Cryptology – EUROCRYPT 2003 : proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003. – Berlin ; Heidelberg : Springer, 2003. – P. 345–359. – (Lecture Notes in Computer Science ; vol. 2656).

175. Courtois N. T. Fast algebraic attacks on stream ciphers with linear feedback [Text] / N. T. Courtois // Advances in Cryptology – CRYPTO 2003 : proceedings of the 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 2003. – Berlin ; Heidelberg : Springer, 2003. – P. 176–194. – (Lecture Notes in Computer Science ; vol. 2729.)

176. Rønjom S. Attacking the filter generator over GF(2^m) [Text] / S. Rønjom, T. Helleseth // Arithmetic of Finite Fields. – Berlin ; Heidelberg : Springer, 2007. – P. 264–275.

177. Rostovtsev A. Changing probabilities of differentials and linear sums via isomorphisms of ciphers [Electronic resource] : report 2009/117 / A. Rostovtsev // IACR Cryptology ePrint Archive. – 2009. – Mode of access : www. URL: http://eprint.iacr.org/2009/117.

178. Rostovtsev A. AES-like ciphers: are special S-boxes better then random ones? (Virtual isomorphisms again) [Electronic resource] : report 2013/148 / A. Rostovtsev // IACR Cryptology ePrint Archive. – 2013. – Mode of access : www. URL: http://eprint.iacr.org/2013/148.

179. Ailan W. Analysis of corresponding structure of differential branch of MDS matrixes on finite field [Text] / W. Ailan, L. Yunqiang, Z. Xiaoyong // Intelligent Networks and Intelligent Systems (ICINIS'2010) : proceedings of the 3rd International Conference, Washington, DC, USA / IEEE Computer Society. – Washington, 2010. – P. 381–384.

180. Murphy S. Essential algebraic structure within the AES [Text] / S. Murphy, M. J. B. Robshaw // Advances in Cryptology – CRYPTO 2002 : proceedings of the 22nd Annual International Cryptology Conference, Santa Barbara, Cali-

fornia, USA, August 18–22, 2002. – Berlin ; Heidelberg : Springer, 2002. – P. 1–16. – (Lecture Notes in Computer Science ; vol. 2442).

181. A toolbox for cryptanalysis: Linear and affine equivalence algorithms [Text] / Biryukov A. et al. // Advances in Cryptology – Eurocrypt 2003 : proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003. – Berlin ; Heidelberg : Springer, 2003. – P. 33–50. – (Lecture Notes in Computer Science ; vol. 2656).

182. Williams V. V. Multiplying matrices faster than Coppersmith-Winograd [Electronic resource] / Virginia Vassilevska Williams. – November 2011. – Mode of access : http://www.cs.berkeley.edu/~virgi/matrixmult.pdf.

183. Robinson S. Toward an optimal algorithm for matrix multiplication [Text] / S. Robinson // SIAM news. – 2005. – Vol. 38, № 9. – P. 1–5.

184. Group-theoretic algorithms for matrix multiplication [Text] / Cohn H. et al. // Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), October 23–25, 2005, Pittsburgh, PA, USA. – Pittsburgh, 2005. – P. 379–388.

185. Intel® SSE4 Programming Reference [Electronic resource] : D91561-003. – July 2007. – Mode of access : www. URL: http://home.ustc.edu.cn/~shengjie/REFERENCE/sse4_instruction_set.pdf.

186. Gold R. Maximal recursive sequence with 3-valued recursive cross-correlation functions [Text] / R. Gold // IEEE Transactions on Information Theory. – 1968. – Vol. 14, № 1. – P. 154–156.

187. Kasami T. The weight enumerators for several classes of subcodes of the second order binary Reed-Muller codes [Text] / T. Kasami // Information and Control. – 1971. – Vol. 18. – P. 369–394.

188. Dobbertin H. Almost perfect nonlinear power functions over GF(2n): The Welch case [Text] / H. Dobbertin // IEEE Transactions on Information Theory. – 1999. – Vol. 45, № 4. – P. 1271–1275.

189. Dobbertin H. Almost perfect nonlinear power functions over GF(2n):

The Niho case [Text] / H. Dobbertin // International Journal of Computer and Information Sciences. – 1999. – Vol. 151. – P. 57–72.

190. Dobbertin H. Almost perfect nonlinear power functions over GF(2n): A new case for n divisible 5 [Text] / H. Dobbertin // Finite Fields and Applications : proceedings of the Fifth International Conference on Finite Fields and Applications Fq 5, Augsburg, Germany, August 2–6, 1999. – Berlin : Springer, 2001. – P. 113–121.

191. An APN permutation in dimension 6 [Text] / K. Browning, J. F. Dillon, M. McQuistan, A. J. Wolfe // Contemporary Mathematics. – 2010. – Vol. 518 : proceedings of the 9th International Conference on Finite Fields and their Applications, University College Dublin, July 13–17, 2009. – P. 33–42.

192. Edel Y. A new almost perfect nonlinear function which is not quadratic [Electronic resource] : report 2008/313 / Yves Edel, Alexander Pott // IACR Cryptology ePrint Archive. – 2008. – Mode of access : www. URL: http://eprint.iacr.org/2008/313.

193. Кузнецов А. А. Метод построения криптографически стойких булевых функций на основе градиентного спуска [Текст] / А. А. Кузнецов, Ю. А. Избенко, И. Московченко // Зб. наук. пр. Харк. ун-ту Повітр. Сил. – Х. : ХУПС, 2007. – С. 63–66.

194. Вероятностная модель формирования нелинейных узлов замен для симметричных криптографических средств защиты информации [Текст] / Л. С. Сорока и др. // Системи обробки інформації. – Х., 2009. – Вип. 3 (77). – С. 286–294.

195. Кузнецов А. А. Исследование вероятностных методов формирования нелинейных узлов замен [Текст] / А. А. Кузнецов, С. А. Исаев // Системи обробки інформації. – Х., 2011. – Вип. 7 (97) : Проблеми і перспективи розвитку IT-індустрії. – С. 132–133.

196. Burnett L. D. Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography [Text] : PhD thesis / L. D. Burnett. – Queensland

University of Technology, 2005.

197. Millan W. Smart hill climbing finds better boolean functions [Text] / W. Millan, A. Clark, E. Dawson // Proceedings of the Workshop on Selected Areas in Cryptography, SAC 1997, Ottawa, Ontario, Canada. – Berlin : Springer-Verlag, 1997. – P. 50–63.

198. Кузнецов А. А. Вычислительный метод синтеза регулярных нелинейных узлов замен с использованием недвоичных криптографических функций [Текст] / А. А. Кузнецов, С. А. Исаев, В. В. Фролов // Зб. наук. пр. Харк. ун-ту Повітр. Сил. – Х. : ХУПС, 2012. – Вип. 4 (33). – С. 148–153.

199. Yu Y. A Matrix Approach for Constructing Quadratic APN Functions [Electronic resource] : report 2013/007 / Yuyin Yu, Mingsheng Wang, Yongqiang Li // IACR Cryptology ePrint Archive. – 2013. – Mode of access : www. URL: eprint.iacr.org/2013/007.

200. Weng G. On Quadratic Almost Perfect Nonlinear Functions and Their Related Algebraic Object [Electronic resource] / G. Weng, Y. Tan, G. Gong // International Workshop on Coding and Cryptography (WCC 2013), April 15–19, 2013, Bergen, Norway. – Mode of access : www. URL: http://www.selmer.uib.no/WCC2013/pdfs/Weng.pdf.

201. Leander G. On the Classification of 4 Bit S-boxes [Text] / G. Leander, A. Poschmann // Proceedings of the 1st international workshop on Arithmetic of Finite Fields (WAIFI '07), Madrid, Spain. – Berlin ; Heidelberg : Springer-Verlag, 2007. – P. 159–176.

202. Saarinen M. J. O. Cryptographic analysis of all 4×4-bit s-boxes [Text] / M. J. O. Saarinen // Selected Areas in Cryptography : proceedings of the 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15–16, 2012. – Berlin ; Heidelberg : Springer, 2013. – P. 118–133. – (Lecture Notes in Computer Science ; vol. 7707).

203. Yu Y. Constructing Differentially 4 Uniform Permutations from Known Ones [Text] / Yuyin Yu, Mingsheng Wang, Yongqiang Li // Chinese Jour-

nal of Electronics. – 2013. – Vol. 22, № 3. – P. 495–499.

204. TESAŘ P. A New Method for Generating High Non-linearity S-Boxes [Text] / Petr TESAŘ // Radioengineering. – 2010. – Vol. 19, № 1. – P. 23–26.

205. Шишкин В. А. Принципы синтеза перспективного алгоритма блочного шифрования с длиной блока 128 бит [Электронный ресурс] / В. А. Шишкин // РусКрипто'2013 : материалы XV науч.-практ. конф., 27–30 марта 2013 г., Россия, Моск. обл. – Режим доступа : www. URL: http://www.ruscrypto.ru/accotiation/archive/rc2013/.

206. Криптостійкість шифру “Калина” [Текст] / Р. В. Олійников, І. Д. Горбенко, В. І. Долгов та ін. // Прикладная радиоэлектроника. – Х., 2007. – Т. 6, № 2. – С. 217–229.

207. Оценки практической стойкости блочного шифра «Калина» относительно разностного, линейного и билинейного методов криптоанализа [Текст] / А. Н. Алексейчук и др. // Материалы Четвертой международной научной конференции по проблемам безопасности и противодействия терроризму, 30–31 окт. 2008 г. – М. : МЦНМО, 2009. – С. 15.

208. Albrecht M. Tools for Algebraic Cryptanalysis [Text] / M. Albrecht // Proceedings of the ECRYPT Workshop on Tools for Cryptanalysis 2010, June 22–23, 2010, Royal Holloway, Egham, UK. – London, 2010. – P. 13–15.

209. Lafitte F. Cryptographic Boolean Functions with R [Text] / F. Lafitte, D. Van Heule, Julien Van hamme // The R Journal. – 2011. – Vol. 3, № 1. – P. 44–47.

210. Sage Mathematics Software (Version 5.11) [Electronic resource] / William A. Stein et al. – Mode of access : www. URL: http://www.sagemath.org.

211. Kazymyrov O. Source code of class Sbox [Electronic resource] / Oleksandr Kazymyrov. – Mode of access : www. URL: https://github.com/okazymyrov/sbox.

212. Stroustrup B. Programming: principles and practice using C++ [Text]

/ B. Stroustrup. – Boston : Addison-Wesley Professional, 2008. – 1264 p.

213. MPI – The Complete Reference [Text] / M. Snir et al. – Cambridge : MIT Press, 1998. – Vol. 1 : The MPI Core.

214. Donald E. K. The art of computer programming [Text] / E. Knuth Donald. – Reading, Massachusetts : Addison-Wesley, 1998. – Vol. 3 : Sorting and Searching. – 780 p.

215. Donald E. K. The art of computer programming [Text] / E. Knuth Donald. – Reading, Massachusetts : Addison-Wesley, 1997. – Vol. 1 : Fundamental Algorithms. – 650 p.

216. Cython: The best of both worlds [Text] / S. Behnel et al. // Computing in Science & Engineering. – 2011. – Vol. 13, № 2. – P. 31–39.

217. Ekanayake J. High performance parallel computing with clouds and cloud technologies [Text] / J. Ekanayake, G. Fox // Cloud Computing. – Berlin ; Heidelberg : Springer, 2010. – P. 20–38.

218. The Grid 2: Blueprint for a new computing infrastructure [Electronic resource] / ed. I. Foster, C. Kesselman. – Mode of access : www. URL: http://www.elsevierdirect.com/v2/companion.jsp?ISBN=9781558609334.

219. Introduction to parallel computing [Text] / B. Barney et al. // Lawrence Livermore National Laboratory. – 2010. – Vol. 6, № 13. – P. 10.

220. Hardware of «Hexagon» [Electronic resource]. – Mode of access : www. URL: https://www.notur.no/hardware/hexagon/.

221. A preliminary evaluation of the hardware acceleration of the Cray Gemini interconnect for PGAS languages and comparison with MPI [Text] / H. Shan et al. // ACM SIGMETRICS Performance Evaluation Review. – 2012. – Vol. 40, № 2. – P. 92–98.

222. Job execution (Hexagon). General job limitation [Electronic resource]. – Mode of access : www. URL: http://www.bccs.uni.no/hpcdoc/Job_execution_(Hexagon)#General_job_limitations. – 17.09.13.

ПРИЛОЖЕНИЕ А АКТЫ ВНЕДРЕНИЯ РЕЗУЛЬТАТОВ НАУЧНЫХ ИССЛЕДОВАНИЙ

"ЗАТВЕРДЖУЮ"
Генеральний директор АТ "ІІТ"

_____ С.Ю. Сінаюк
«5» 07 _____ 2013 р.

АКТ

впровадження результатів наукових досліджень
в діяльність Приватного акціонерного товариства "Інститут інформаційних технологій"
Казимирова Олександра Володимировича

Комісія у складі голови комісії, заступника Головного конструктора з ЗБС, кандидата технічних наук, професора Качко О. Г., та членів комісії, начальника відділу КЗІ, кандидата технічних наук Погребняка К. А., і аналітика систем захисту інформації Бойка А. О. встановила наступне.

Протягом 2011-2013 рр. у Приватному акціонерному товаристві "Інститут інформаційних технологій" впроваджені наступні результати, що одержані аспірантом кафедри безпеки інформаційних технологій Харківського національного університету радіоелектроніки Казимировим Олександром Володимировичем у процесі виконання його наукової діяльності:

– алгоритм формування довгострокових ключових елементів для блокового симетричного алгоритму ДСТУ ГОСТ 28147:2009, що дало змогу зменшити час генерування нелінійних вузлів заміни, які належать різним класам еквівалентності, до 1 с;

– алгоритм генерації підстановок для перспективних симметричних криптоалгоритмів на основі запропоновоного ним методу, який дозволив зменшити час генерації однієї оптимальної 8-бітної підстановки з 44 годин до 3,5;

– комплекси програмного забезпечення (програмні моделі), що реалізують розроблені методи генерації нелінійних відображень, та проведене програмне моделювання, на основі яких отримані властивості швидкодії алгоритмів.

Голова комісії, к.т.н., проф. О.Г. Качко

Члени комісії:
к.т.н. К. А. Погребняк

А. О. Бойко

"ЗАТВЕРДЖУЮ"
Проректор з науково-педагогічної роботи
Харківського національного університету
радіоелектроніки

_____ проф. Н. С. Лєсна

«17» 07 2013 р.

АКТ

впровадження результатів наукових досліджень
в навчальний процес Харківського національного університету радіоелектроніки
Казимирова Олександра Володимировича

Комісія у складі голови комісії, завідувача кафедри БІТ, доктора технічних наук, професора Горбенка І.Д. і членів комісії, кандидата технічних наук, Балагури Д.С. та кандидата технічних наук Лєншиної Ю.М. встановила, що у Харківському національнму університеті радіоелектроніки впроваджені наступні результати, що одержані аспірантом кафедри безпеки інформаційних технологій Харківського національного університету радіоелектроніки Казимировим Олександром Володимировичем у процесі виконання дисертаційних досліджень

1. По дисципліні "Криптографічні системи та протоколи" для напряму ІБ при підготовці та читанні лекцій за темами:

№6 "Вимоги та принципи побудування перспективних симметричних блокових шифрів";

№7 "Принципи побудови перспективного БСШ «Калина»";

2. Для студентів 5-го курсу та магістрів спеціальності БІКС в лекціях за темами:

№3 "Застосування таблиць предобчислень для оцінки стійкості симметричних криптографічних перетворень";

3. Розробки по дослідженню та аналізу показників нелінійних узлів заміни використані при виконанні магістерських та бакалаврських атестаційних робіт на четвертому на п'ятому курсах.

Голова комісії, д.т.н., проф. _____ І.Д. Горбенко

Члени комісії:

к.т.н. _____ Д.С. Балагура

к.т.н. _____ Ю.М. Лєншина

ПРИЛОЖЕНИЕ Б ИСХОДНЫЕ КОДЫ

Б.1 Исходные коды библиотеки «Sbox»

Ниже представлены исходные коды основной части программы «Main.sage», базового класса «Sbox», определённого в файле «Sbox.sage», и некоторые функции для расчёта показателей подстановок.

```
# Main.sage
#!/usr/bin/env sage

load ./Sage/TestFunctions.sage

def main(argv=None):
      bits=8

      t1=cputime()

      #test_all_functions(bits=bits)
      #test_APN(bits=bits)
      #test_APN_12(bits=bits)
      #test_DKE(bits=bits)
      #test_claude(bits=bits)
      #test_cr_EA(bits=bits)
      #test_cr_EA_pEA4(bits=bits)
      #test_crypto_functions()
      #test_gen_AI(bits=bits)
      #test_gen_CCZ(bits=bits)
      #test_gen_cycles(bits=bits)
      #test_gen_NL_102(bits=bits)
      #test_general_polynomial(bits=bits)
      #test_is_CCZ_equivalent(bits=bits)
      #test_system_of_equations()
      test_temp(bits=bits)
      #test_IsEquivalentToPermutation(bits=bits)

      t2=cputime()

      print "====="
      print "Time = {0}".format(t2-t1)


if __name__ == "__main__":
      sys.exit(main())
```

```
# Sbox.sage
r'''
    Cryptanalysis and generating substitutions
'''
```

```
load ./Cython/CFunc.spyx
load ./Cython/CPPFunc.spyx
load ./Sage/GSbox.sage
load ./Sage/CSbox.sage
load ./Sage/Tools.sage

from random import shuffle
from sage.crypto.boolean_function import random_boolean_function
from inspect import currentframe # for debug
from time import time # for debug

class Sbox(SageObject):
      def __init__(self, n=None, m=None, **kwargs):
            r"""
                  n     - number of input bits of Sbox
                  m     - number of output bits of Sbox

                  kwargs:
                        modulus     - irreducible polynomial
                        bijection   - if True, then all function will
generate bijective substitution
                        sbox        - predefined sbox
            """

            if n is None:
                  raise TypeError("n is not defined")

            if m is None:
                  raise TypeError("m is not defined")

            self._modulus = kwargs.get('modulus',None)
            self._bijection = kwargs.get('bijection',None)
            self._n=n
            self._m=m
            self._length=1<<self._n
            self._LFoEA = [None,None,None,None,None]

            if self._modulus is None:
                  self._K                                                =
GF(1<<max(self._m,self._n),'a',modulus='conway')
            else:
                  self._K                                                =
GF(1<<max(self._m,self._n),'a',modulus=ZZ['x'](self._modulus))

            self._P = PolynomialRing(self._K,'x')

            self._S = kwargs.get('sbox',None)
            self._polynomial = self.g2p(kwargs.get('polynomial',None))
            self._alpha = self._K.multiplicative_generator()

            if self._polynomial is not None:
                  self.EA(G=self._polynomial)

      def g2p(self,pol=None,**kwargs):
            r"""
                  Convert polynomial with primitive element to internal
representation.

                  Parameter
                        pol         - polynomial
```

```
                """
                if pol is not None:
                        return
self._P(pol.replace("g","({0})".format(self._K.multiplicative_generator())))

        def generate_sbox(self, method=None, T=None, **kwargs):
                r'''
                        'method' can be:
                                random_permutation
                                random_substitution
                                gold
                                kasami
                                welch
                                niho
                                inverse
                                dobbertin
                                APN6
                                dicson
                                polynomial
                                two_xor

                        'T' can be:
                                CCZ
                                EA
                                A
                                L

                        'G' is some function for 'method=polynomial'
                '''
                self._polynomial = None
                self._S = None

                methods=dict(AI=self.AI,
                                APN6=self.APN6,
                                claude_matrix=self.claude_matrix,
                                dobbertin=self.dobbertin,
                                dicson=self.dicson,
                                gold=self.gold,
                                inverse=self.inverse,
                                kasami=self.kasami,
                                lilia=self.lilia,
                                lilia_base=self.lilia_base,
                                niho=self.niho,
                                optimal_permutation=self.optimal_permutation,
                                polynomial=self.EA,
                                random_permutation=self.random_permutation,
                                random_substitution=self.random_substitution,
                                welch=self.welch
                                )

                if not method in methods:
                        raise           TypeError("Unsupported           method
'{0}'".format(method))

                if method != 'polynomial':
                        methods[method](**kwargs)

                if    method    ==    'random_permutation'    or    method    ==
'random_substitution':
                        return
```

```
                if T == 'CCZ':
                      self.CCZ(**kwargs)
                      return
                elif T == 'EA':
                      if kwargs.has_key('M1') is not True:
                            if kwargs.has_key('M2') is not True:
                                  if kwargs.has_key('M3') is not True:
                                        if   kwargs.has_key('V1')   is   not
True:
                                              if   kwargs.has_key('V2')   is
not True:
                                                    kwargs['M1'] = 'random'
                                                    kwargs['M2'] = 'random'
                                                    kwargs['M3'] = 'random'
                                                    kwargs['V1'] = 'random'
                                                    kwargs['V2'] = 'random'
                elif T == 'A':
                      if kwargs.has_key('M1') is not True:
                            if kwargs.has_key('M2') is not True:
                                  if kwargs.has_key('V1') is not True:
                                        if   kwargs.has_key('V2')   is   not
True:
                                              kwargs['M1'] = 'random'
                                              kwargs['M2'] = 'random'
                                              kwargs['V1'] = 'random'
                                              kwargs['V2'] = 'random'
                elif T == 'L':
                      if kwargs.has_key('M1') is not True:
                            if kwargs.has_key('M2') is not True:
                                  kwargs['M1'] = 'random'
                                  kwargs['M2'] = 'random'

                self.EA(**kwargs)

          def get_field(self):
                return self._K

          def get_linear_functions(self):
                r"""
                      Returns  [M1,M2,M3,V1,V2]  from  EA-equivalence  F(x)  =
M1 * G( M2 * x + V2 ) + M3 * x + V1
                """
                return self._LFoEA

          def get_mg(self):
                return self._K.multiplicative_generator()

          def get_modulus(self):
                return self._K.modulus()

          def get_ring(self):
                return self._P

          def get_sbox(self):
                return self._S

          def get_polynomial(self):
                return self._polynomial

          def from_field(self,x):
                return x.integer_representation()
```

```
            def set_sbox(self,sbox=None):
                  if sbox is None:
                        raise TypeError("sbox is not defined")
                  self._polynomial = None
                  self._S = sbox[:]

            def to_field(self,x):
                  return self._K(ZZ(x).digits(base=2))

            def l2m(self,L=None):
                  r'''
                        Convert a linear function L to a matrix M
                  '''
                  if L is None:
                        raise TypeError("L is not defined")

                  if isinstance(L,basestring):
                        L = self.g2p(L)

                  M=matrix(GF(2),self._n,self._m)

                  for i in xrange(self._n):

      M.set_column(i,L.subs(self._K(ZZ(2^i).digits(2)))._vector_())

                  return M

            def m2l(self,M=None,representation="generator"):
                  r'''
                        Convert a matrix M to a linear function L
                  '''
                  if M is None:
                        return None

                  T = matrix(self._K,self._n)
                  C = vector(self._K,self._n,[self._K(M.column(g)) for g in
xrange(self._n)])

                  for i in xrange(self._n):
                        T.set_row(i,[self._K.fetch_int(2^i)^(2^g) for g in
xrange(self._n)])

                  T = (T.inverse()*C).list()

                  L = sum([self._P("({0})*x^(2^{1})".format(T[g],g)) for g in
xrange(len(T))])

                  if representation == "internal":
                        return L
                  else:
                        return self.p2g(pol=L)

            def p2g(self,pol=None,**kwargs):
                  r"""
                        Convert internal representation to polynomial with
primitive element.

                        Parameters
                              pol         - polynomial
```

```
                              test - if "True" then function computes
selftesting
                              form - represent polynomial int the form
'integer' or 'logarithmic' (default)
                  """
                  pol                                                 =
self._P(pol).mod(self._P("x^{0}+x".format(self._length)))
                  test = kwargs.get('test',True)
                  form = kwargs.get('form','log')

                  if pol is None:
                        return "None"

                  g_pol = ""
                  if form == 'integer':
                        raise TypeError("'integer' option isn't implemented")
                  elif form == 'log':
                        for i in xrange(self._length-1,1,-1):
                              t                                       =
pol[i].log(self._K.multiplicative_generator())
                              if pol[i] != 0:
                                    if t == 1:
                                          g_pol += "g*x^{0} + ".format(i)
                                    elif t == 0:
                                          g_pol += "x^{0} + ".format(i)
                                    else:
                                          g_pol     +=     "g^{0}*x^{1}     +
".format(t,i)

                        t = pol[1].log(self._K.multiplicative_generator())
                        if pol[1] != 0:
                              if t == 1:
                                    g_pol += "g*x + "
                              elif t == 0:
                                    g_pol += "x + "
                              else:
                                    g_pol += "g^{0}*x + ".format(t)

                        t = pol[0].log(self._K.multiplicative_generator())
                        if pol[0] != 0:
                              if t == 1:
                                    g_pol += "g"
                              elif t == 0:
                                    g_pol += "1"
                              else:
                                    g_pol += "g^{0}".format(t)
                        else:
                              if g_pol[-3:] == " + ":
                                    g_pol = g_pol[:-3]

                        if g_pol == "":
                              g_pol = "0"

                        if test == True:
                              if
self._P(g_pol.replace("g","({0})".format(self._K.multiplicative_generator()))
) == pol:
                                    return g_pol
                              else:
                                    raise TypeError("p2g: selftesting fail")
                        else:
```

```
                        return g_pol

            algebraic_immunity_sbox=cr_algebraic_immunity_sbox
            autocorrelation=cr_autocorrelation
            APN6=gen_APN6
            balanced=cr_balanced
            CCZ=gen_CCZ
            check_polynomial=cr_check_polynomial
            check_system=cr_check_system
            CI=cr_CI
            claude_matrix=gen_e_claude_matrix # experemental
            create_system=cr_create_system
            cycles=cr_cycles
            dicson=gen_dicson
            difference_distribution_matrix_full=cr_difference_distribution_ma
trix_full
            dobbertin=gen_dobbertin
            EA=gen_EA
            AI=gen_e_AI # experemental
            gold=gen_gold
            interpolation_polynomial=cr_interpolation_polynomial
            inverse=gen_inverse
            is_APN=cr_is_APN
            is_bijection=cr_is_bijection
            is_CCZ_equivalent=cr_is_CCZ_equivalent
            IsEquivalentToPermutation=cr_IsEquivalentToPermutation          #
experemental
            is_EA_equivalent=cr_is_EA_equivalent # experemental
            kasami=gen_kasami
            lilia=gen_e_lilia # experemental
            lilia_base=gen_e_lilia_base # experemental
            maximal_diff_table=cr_maximal_diff_table
            maximal_difference_probability=cr_maximal_difference_probability
            maximal_linear_bias=cr_maximal_linear_bias
            maximal_linear_table=cr_maximal_linear_table
            minimum_degree=cr_minimum_degree
            niho=gen_niho
            nonlinearity=cr_nonlinearity
            optimal_permutation=gen_optimal_permutation
            PC=cr_PC
            random_permutation=gen_random_permutation
            random_substitution=gen_random_substitution
            resilient=cr_resilient
            SAC=cr_SAC
            SSI=cr_SSI
            welch=gen_welch


      # CSbox.sage
      def cr_create_system(self, degree=2, groebner=False):
            #t1=time()

            P  =  BooleanPolynomialRing(self._n+self._m,  ["x%d"%i  for  i  in
range(self._n)] + ["y%d"%i for i in range(self._m)])
            X = [P("x%d"%(self._n-i-1)) for i in range(self._n)]
            Y = [P("y%d"%(self._m-i-1)) for i in range(self._m)]

            gens = X+Y

            bits = []
            for i in xrange(self._length):
```

```
                  bits.append(
list(reversed(ZZ(i).digits(base=2,padto=self._n)))                              +
list(reversed(ZZ(self._S[i]).digits(base=2,padto=self._m))) )

            nrows = self._length
            ncols     =     sum(binomial(self._m+self._n,i)     for     i     in
range(0,degree+1))

            A = Matrix(GF(2), nrows , ncols)

            exponents = []
            for d in xrange(degree+1):
                  exponents  +=  IntegerVectors(d,  max_length=self._n+self._m,
min_length=self._n+self._m, min_part=0, max_part=1).list()

            col = 0
            variables = []
            for exponent in exponents:
                  variables.append(   mul([gens[i]**exponent[i]   for   i   in
range(len(exponent))]))
                  for row in xrange(self._length):
                        A[row,col]    =    mul([bits[row][i]    for    i    in
range(len(exponent)) if exponent[i]])
                  col +=1

            system=A.right_kernel()
            system=system.matrix()

            gens=[]
            length=len(variables)
            for j in xrange(len(system.rows())):
                  gens.append(sum(variables[i]*system[j][i]    for    i    in
xrange(length)) )

            return gens

      def cr_interpolation_polynomial(self, representation="generator"):
            if self._polynomial is not None:
                  if representation == "internal":
                        return self._polynomial
                  else:
                        return self.p2g(pol=self._polynomial)

            l = []
            for i in xrange(self._length):
                  l.append(                          (self._K(ZZ(i).digits(2)),
self._K(ZZ(self._S[i]).digits(2))) )

            self._polynomial = self._P.lagrange_polynomial(l)
            if representation == "internal":
                  return self._polynomial
            else:
                  return self.p2g(pol=self._polynomial)
```

Б.2 Исходные коды программного обеспечения «maxNL»

Главный файл «Main.c» программы «maxNL» имеет следующий вид:

```
      #include <stdio.h>
      #include <mpi.h>
      #include "main.h"

      #define BITS 8

      const unsigned long long SboxInv[1<<BITS] = {0, 1, 204, 124, 142, 244,
176, 228, 102, 76, 68, 79, 62, 224, 24, 223, 71, 167, 84, 27, 122, 186, 29,
161, 88, 192, 15, 92, 114, 216, 155, 188, 51, 110, 60, 13, 38, 139, 59, 206,
34, 80, 149, 154, 169, 207, 174, 182, 31, 74, 183, 190, 112, 208, 8, 156, 12,
171, 197, 100, 225, 21, 7, 123, 173, 157, 105, 83, 221, 152, 245, 2, 42, 75,
189, 148, 131, 160, 11, 220, 61, 170, 49, 73, 93, 150, 45, 39, 128, 85, 199,
162, 222, 144, 172, 9, 44, 138, 198, 159, 96, 86, 130, 28, 137, 72, 235, 242,
46, 111, 191, 175, 57, 81, 5, 70, 108, 237, 194, 52, 195, 164, 135, 229, 94,
64, 238, 107, 151, 14, 115, 16, 55, 65, 120, 118, 30, 43, 168, 58, 136, 48,
133, 125, 19, 193, 10, 153, 203, 99, 145, 25, 147, 69, 98, 90, 103, 22, 210,
247, 17, 217, 201, 211, 40, 209, 143, 3, 196, 185, 20, 63, 77, 23, 230, 240,
218, 121, 234, 181, 233, 251, 231, 227, 87, 36, 134, 177, 91, 202, 226, 241,
129, 26, 113, 41, 37, 97, 246, 200, 213, 248, 163, 158, 95, 54, 101, 184, 56,
35, 67, 249, 104, 140, 214, 215, 4, 166, 165, 53, 78, 146, 18, 89, 6, 187,
253, 126, 219, 119, 255, 127, 236, 32, 250, 116, 50, 47, 243, 180, 254, 252,
117, 232, 132, 205, 212, 66, 141, 82, 109, 33, 179, 239, 178, 106};

      int search(int ccpu)
      {
          int i=0, d = 0;
          unsigned long long *Sbox = NULL, SboxLength = 0, t = 0, r1 = 0, r2
= 0;
          char file_name[FILENAME_MAX] = {}, **fs = NULL;
          FILE *output = NULL;

          sprintf(file_name,"./sbox%d.txt",ccpu);

          SboxLength = 1 << BITS;

          Sbox = (unsigned long long*)calloc(SboxLength,sizeof(unsigned long
long));

          fs=(char**)calloc(FunctionsCount,sizeof(char*));

          for(i=0;i<FunctionsCount;i++)
              fs[i]=(char*)calloc(SboxLength,sizeof(char));

          while(1)
          {
              memcpy(Sbox,SboxInv,(1<<BITS)*sizeof(unsigned long long));

                for(i=0;i<26;i++)
                {
                        r1 = get_random_number(BITS);
                        r2 = get_random_number(BITS);

                  r1 = (r1+r2) & 0xFF;
                  r2 = ((r1+1)*r2) & 0xFF;

                        t = Sbox[r1];
                        Sbox[r1] = Sbox[r2];
                        Sbox[r2] = t;
                }

              splitSboxTLF(Sbox,fs);
```

```
                d = diff_Sbox(Sbox,0);

                if (NL_Sbox(fs,0) >= 102)
                {
                    if(alg_eq_Sbox(fs,2,0) == 0)
                    {
                          if(deg_Sbox(fs,0) == 7)
                          {
                                      if(d <= 8)
                                      {
                                            init_output(file_name,&output);

                                            fprintf(output,"sbox\t\t\t= {");
                                            for(i=0;i<SboxLength-1;i++)
                                            {

      fprintf(output,"0x%.2llX,",Sbox[i]);
                                            }

      fprintf(output,"0x%.2llX};\n",Sbox[SboxLength-1]);

                                            fprintf(output,"Nonlinearity\t\t\t=
%d\n",NL_Sbox(fs,0));
                                            fprintf(output,"Maximum diff
table\t\t= %d\n",d);
                                            for (i=1;i<=BITS;i++)
                                {
                                    if ( alg_eq_Sbox(fs,i,0) != 0)
                                    {
                                        fprintf(output,"Number of
equations\t\t= %d (degree=%d)\n",alg_eq_Sbox(fs,i,0),i);
                                        break;
                                    }
                                }
                                            fprintf(output,"Minimum
degree\t\t\t= %d\n",deg_Sbox(fs,0));
                                            fprintf(output,"Bijection\t\t\t\t=
%s\n",bijection_Sbox(Sbox,0) ? "True" : "False");

      fprintf(output,"==============================\n");
                                            close_output(output);
                                      }
                                }
                          }
                    }

                if(d == 2)
                {
                    init_output(file_name,&output);

                    fprintf(output,"sbox\t\t\t= {");
                    for(i=0;i<SboxLength-1;i++)
                    {
                        fprintf(output,"0x%.2llX,",Sbox[i]);
                    }
                    fprintf(output,"0x%.2llX};\n",Sbox[SboxLength-1]);

                    fprintf(output,"Nonlinearity\t\t\t= %d\n",NL_Sbox(fs,0));
                    fprintf(output,"Maximum diff table\t\t= %d\n",d);
                    for (i=1;i<=BITS;i++)
```

```
                {
                    if ( alg_eq_Sbox(fs,i,0) != 0)
                    {
                        fprintf(output,"Number of equations\t\t= %d
(degree=%d)\n",alg_eq_Sbox(fs,i,0),i);
                        break;
                    }
                }
                fprintf(output,"Minimum degree\t\t\t=
%d\n",deg_Sbox(fs,0));
                fprintf(output,"Bijection\t\t\t\t=
%s\n",bijection_Sbox(Sbox,0) ? "True" : "False");
                fprintf(output,"============================\n");
                close_output(output);
            }
        }

        if(Sbox)
            free(Sbox);

        if(fs)
        {
            for(i = 0; i < FunctionsCount; i++)
            {
                free(fs[i]);
            }

            free(fs);
        }

        return 0;
    }

    int main(int argc, char** argv)
    {
        int ccpu, nprocs;

        SboxLength = (1<<BITS);
        SboxBitIn=BITS;
        SboxBitOut=BITS;
        FunctionsCount=(1<<SboxBitOut)-1;
        FunctionsLength=SboxLength;

        MPI_Init(&argc, &argv);
        MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
        MPI_Comm_rank(MPI_COMM_WORLD, &ccpu);

        if ( search(ccpu) != 0)
        {
          MPI_Finalize();
          return -1;
        }

        MPI_Finalize();
        return 0;
    }
```

Ниже представлены несколько функций для расчёта показателей нелинейного узла замены.

```
	int diff_Sbox (unsigned char *Sbox, int debug)
	{
		int max=0,j=0,i=0;
		unsigned char *DifTable=NULL;

		DifTable=(unsigned char*)calloc(SboxLength,sizeof(unsigned
char));

		for(max=0,j=0;j<SboxLength;j++)
		{
			for(i=0;i<SboxLength;i++)
			{
				DifTable[Sbox[i]^Sbox[j^i]]++;
			}

			if( j == 0 )
			{
				DifTable[0]=0;
				continue;
			}

			for(i=0;i<SboxLength;i++)
			{
				if(max<DifTable[i])
					max=DifTable[i];

				DifTable[i]=0;
			}
		}

	int NL_Sbox(char **fs, int debug)
	{
		int i=0, min = 1<<SboxBitOut, retf = 1;

		if(!fs)
		{
			return -1;
		}
		else
		{
			for(i=0;i<FunctionsCount;i++)
					if(!fs[i])
						return -1;
		}

		if(debug)
			fprintf(output,"Debug from NL_Sbox:\n");

		for(i=0;i<FunctionsCount;i++)
		{
			retf=NL_fast(fs[i],0);
			//retf=NL(fs[i],0);

			if(debug)
				fprintf(output,"f[%d] = %d\n",i,retf);

			if( min > retf )
				min=retf;
		}

		return min;
```

```
	}
	    if(DifTable)
	    {
	      free(DifTable);
	    }

	      return max;
	}

	int deg_Sbox(char **fs, int debug)
	{
		int i=0, s_deg=0, t_deg=0;

		for(s_deg=SboxBitOut,i=0;i<FunctionsCount;i++)
		{
			t_deg=deg(fs[i],0);
			if ( s_deg>t_deg )
				s_deg=t_deg;
			if(debug)
				fprintf(output,"s_deg=%d t_deg=%d\n",s_deg,t_deg);
		}

		return s_deg;
	}

	int alg_eq_Sbox(char **Functions, int deg,int debug)
	{
		int *tBuf=NULL, *cEq=NULL, x=0, y=0, z=0;
		int columns=1+(SboxBitOut<<1), c=0, r=0, d=0, ret=0, counter=0,
column=0;
		char **Matrix=NULL, **fs=NULL;
	// columns in matrix
		for(c=2;c<=deg;c++)
		{
			columns+=(int)binom(SboxBitOut<<1,c);
		}
	// init functions
		tBuf=(int*)calloc((SboxBitOut<<1)+2,sizeof(int)); // buffer for
generating all combinations
		cEq=(int*)calloc((SboxBitOut<<1),sizeof(int)); // current vector
for combination
		fs=(char**)calloc(FunctionsLength,sizeof(char*)); // columns

		for(r=0;r<FunctionsLength;r++) // rows
		{
			fs[r]=(char*)calloc((SboxBitOut<<1),sizeof(char));
		}
	// filling first SboxBitOut functions
		for(c=0;c<SboxBitOut;c++)
		{
			for(r=0;r<FunctionsLength;r++)
			{
				fs[r][c]=GET_BIT(r,SboxBitOut-1-c);
			}
		}
	// filling second SboxBitOut functions
		for(c=SboxBitOut;c<(SboxBitOut<<1);c++)
		{
			for(r=0;r<FunctionsLength;r++)
			{
				fs[r][c]=Functions[(1<<(c-SboxBitOut))-1][r];
```

```
                }
        }

        init_matrix((char***)&Matrix,FunctionsLength,columns);
    // filling matrix
        for(r=0;r<FunctionsLength;r++)
        {
                for(c=0;c<columns;c++)
                {
                        Matrix[r][c]=1;
                }
        }

        for(d=1,column=1;d<=deg;d++)
        {
                inittwiddle(d, (SboxBitOut<<1), tBuf);

                for( counter = 0; counter != (SboxBitOut<<1)-d; counter++)
                {
                        cEq[counter] = 0;
                }

                while( counter != (SboxBitOut<<1) )
                {
                        for(r=0;r<FunctionsLength;r++)
                        {
                                Matrix[r][column]&=fs[r][counter];
                        }
                        cEq[counter++] = 1;
                }

                column++;

                while(!twiddle(&x, &y, &z, tBuf))
                {
                        cEq[x] = 1;
                        cEq[y] = 0;

                        for(counter = 0; counter != (SboxBitOut<<1);
counter++)
                        {
                                if(cEq[counter])
                                {
                                        for(r=0;r<FunctionsLength;r++)
                                        {
                                                Matrix[r][column]&=fs[r][counter];
                                        }
                                }
                        } // counter
                        column++;
                } // while
        } // d
    // get rank matrix
        ret=rank_matrix(Matrix,FunctionsLength,columns);
    // debug
    /*
        printf("{\n");

        for(r=0;r<FunctionsLength;r++)
        {
                printf("{");
```

```
            for(c=0;c<columns-1;c++)
            {
                  printf("%d,",Matrix[r][c]);
            }
            printf("%d},\n",Matrix[r][columns-1]);
      }
      printf("};\n");
*/
// end debug
      free_matrix(Matrix,FunctionsLength);
// delete temporary functions
      for(c=0;c<(SboxBitOut<<1);c++)
      {
            free(fs[c]);
      }

      free(fs);

      if(tBuf)
            free(tBuf);

      if(cEq)
            free(cEq);

      //printf("ret=%d columns=%d\n",ret,columns);

      return (columns-ret) < 0 ? 0 : (columns-ret);
}
```