На правах рукописи

Родионов Евгений Юрьевич

ЗАЩИТА ПРОГРАММНЫХ РЕАЛИЗАЦИЙ АЛГОРИТМОВ, ОСНОВАННЫХ НА ПРЕОБРАЗОВАНИЯХ РЕГИСТРОВОГО ТИПА, ОТ АНАЛИЗА В НЕДОВЕРЕННЫХ СРЕДАХ

Специальность: 05.13.19 – методы и системы защиты информации, информационная безопасность

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Автор: 

Москва – 2012

Работа выполнена в Национальном исследовательском ядерном университете «МИФИ» (НИЯУ МИФИ)

Научный руководитель: Кандидат физико-математических наук, доцент кафедры «Криптология и дискретная математика» НИЯУ МИФИ Варфоломеев Александр Алексеевич

Официальные оппоненты: Доктор технических наук, доцент, профессор кафедры «Информационная безопасность банковских систем» НИЯУ МИФИ Запечников Сергей Владимирович

Кандидат физико-математических наук, доцент кафедры «Информационная безопасность» МГТУ им. Баумана Жуков Алексей Евгеньевич

Ведущая организация: Институт проблем информатики Российской академии наук

Защита состоится «28» мая 2012 г. в 14 часов 30 минут на заседании диссертационного совета ДМ 212.130.08 при Национальном исследовательском ядерном университете «МИФИ»: 115409, г. Москва, Каширское ш., д.31. Тел. для справок: +7 (495) 323-95-26, 324-73-34.

С диссертацией можно ознакомиться в библиотеке Национального исследовательского ядерного университета «МИФИ».

Отзывы в двух экземплярах, заверенные печатью, просьба направлять по адресу: 115409, г. Москва, Каширское ш., д.31, диссертационные советы НИЯУ МИФИ, тел.: +7 (495) 323-95-26.

Автореферат разослан «25» апреля 2012 г.

Ученый секретарь диссертационного совета Горбатов В.С.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. В настоящее время тенденции в области развития средств вычислительной техники, автоматизированной обработки и передачи информации способствуют децентрализации обрабатываемой информации и позволяют обрабатывать бóльшие объемы информации, используя мощности входящих в сеть вычислительных устройств. Увеличение интенсивности информационных потоков между узлами вычислительной сети, развитие гетерогенных распределенных вычислительных сетей, а так же широкое распространение мобильных устройств изменили парадигму обработки информации и создали основу для таких технологий, как облачные вычисления, мобильные агенты и технические средства защиты авторских прав.

Облачные вычисления это – модель обеспечения повсеместного и удобного удаленного доступа по требованию к общему пулу конфигурируемых вычислительных ресурсов, которые могут быть оперативно предоставлены или освобождены с минимальными эксплуатационными затратами и/или обращениями к поставщику услуг. Облачные вычисления позволяют их потребителям значительно снизить издержки на инфраструктуру информационных технологий и гибко реагировать на изменения в потребности вычислительных ресурсов.

Технология мобильных агентов – специального программного обеспечения, предназначенного для выполнения определенных заданий и способного самостоятельно перемещаться между узлами вычислительной сети, – позволяет получить такие преимущества как, возможность распараллеливания вычислительных задач, динамическая адаптация, уменьшение сетевого трафика. Мобильные агенты имеют довольно широкую область применения, в которую входят, например, автоматизированные системы заказа авиабилетов и управления онлайн аукционами.

Технологии технических средств защиты авторских прав позволяют предотвратить неправомерный доступ к информации, защищаемой авторским правом. Как правило, защищаемая с помощью таких средств, информация передается пользователям в зашифрованном виде и расшифровывается для дальнейшего воспроизведения на рабочих местах авторизованных пользователей.

С развитием выше приведенных технологий возрастает актуальность защиты программного обеспечения, выполняющегося в недоверенных средах. Программное обеспечение, выполняющееся в недоверенных средах, может быть подвержено декомпиляции и внесению изменений в код программы с целью отклонить ее от заданного функционирования. Программные реализации алгоритмов защиты информации в большей степени подвержены таким атакам. Ключевая информация, используемая алгоритмом защиты информации и встроенная в его программную реализацию, может быть извлечена с помощью методов статического и/или динамического анализа программного обеспечения, и в дальнейшем использована нарушителем. В результате, функционируя в недоверенных средах, существующие средства защиты информации оказываются не эффективными, так как не способны противостоять нарушителю, имеющему полный контроль над вычислительным устройством, на котором обрабатывается и/или хранится защищаемая информация.

Алгоритмы защиты информации, основанные на преобразования регистрового типа, являются важным примитивом, лежащим в основе систем защиты информации, и служат строительным блоком для хэш-функций, генераторов псевдослучайных последовательностей, поточных и блочных шифров. Надежность и

производительность систем защиты информации напрямую зависят от аналогичных характеристик алгоритмов защиты информации, лежащих в их основе.

Приведенные выше технологии в той или иной мере используют программные реализации алгоритмов защиты информации для обеспечения конфиденциальности, целостности и аутентичности обрабатываемой и/или хранимой информации. Таким образом, обеспечение безопасности программных реализаций алгоритмов защиты информации от анализа в недоверенных средах, является необходимым условием для комплексного обеспечения безопасности информации обрабатываемой и/или хранимой с помощью выше описанных технологий, и, следовательно, является актуальной задачей.

Задачей обеспечения безопасности алгоритмов защиты информации от анализа в недоверенных средах за последнее десятилетие занимался ряд ученых и организаций, среди которых следует выделить:

– Кузюрин Н. Н., проводивший исследования по защите программного обеспечения от анализа в недоверенных средах с помощью запутывающих преобразований;

– Варновский Н. П., исследовавший возможность существования доказуемо стойких запутывающих преобразований;

– B. Barak, получивший результаты опровергающие возможность существования универсальных запутывающих преобразований, применимых к произвольному алгоритму;

– S. Chow, предложивший защищенные программные реализации алгоритмов DES и AES-128;

– B. Wyseur, защитивший в 2009 году диссертацию по теме “White-box Cryptography”, автор некоторых атак на программные реализации алгоритмов DES и AES-128;

– O. Billet, автор работы, посвященной анализу запутанной программной реализации алгоритма AES-128;

– W. Michiels, предложивший обобщенный метод анализа запутанных программных реализаций алгоритмов защиты информации, основанных на преобразованиях замены и перестановки.

Автором данной работы предложена методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа, устойчивая к методу разностного анализа по ошибкам вычислений.

Объектом исследования являются алгоритмы защиты информации, основанные на преобразованиях регистрового типа.

Предметом исследования являются способы программной реализации алгоритмов защиты информации, основанных на преобразованиях регистрового типа.

Целью диссертационной работы является повышение стойкости алгоритмов защиты информации, основанных на преобразованиях регистрового типа, от анализа в недоверенных средах.

Методы исследования: теория алгоритмов, теория графов, теория вероятностей и математическая статистика.

Научная задача заключается в синтезе запутывающих преобразований для алгоритмов защиты информации, основанных на преобразованиях регистрового типа, в недоверенных средах.

В рамках решения научной задачи необходимо:

− провести анализ существующих методов обеспечения безопасности программных реализаций алгоритмов защиты информации, от анализа в недоверенных средах;

− провести исследование существующих методов анализа программных реализаций алгоритмов защиты информации и определить условия их применимости;

− создать методику запутывания программных реализаций алгоритмов защиты информации, основанных на преобразования регистрового типа, от анализа в недоверенных средах;

− провести оценку стойкости запутанных программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа, к известным методам анализа в недоверенных средах.

Научная новизна работы состоит в следующем:

– определены условия применимости рассмотренных методов анализа программных реализаций алгоритмов защиты информации в недоверенных средах;

– предложена модель табличной реализации отображений, изложенная в терминах теории графов, позволяющая описать отображения с помощью совокупности таблиц замены;

– предложена методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа, в недоверенных средах.

– получена оценка сложности анализа запутанных программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа, к существующим методам анализа в недоверенных средах.

Практическая значимость результатов заключается в следующем:

– получена запутанная программная реализация алгоритма ГОСТ 28147-89 в режиме простой замены, стойкая к методу разностного анализа в недоверенных средах.

– приведены рекомендации выбора параметров запутанной программной реализации алгоритма ГОСТ 28147-89 в режиме простой замены.

Результаты диссертационной работы представляют практическую ценность для разработки и реализации систем защиты информации в следующих областях информационных технологий: «облачные технологии», мобильные агенты, технические средства защиты авторских прав.

Внедрение результатов исследования. Полученная методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа, была использована при разработке системы защиты приложений от несанкционированного копирования в компании «Актив».

Практические результаты диссертационной работы использовались при разработке системы предотвращения утечки конфиденциальной информации из корпоративной информационной системы в программном комплексе «Модуль Контроля Действий Пользователей» в ОАО Банк «Возрождение».

Теоретические результаты исследования, полученные в процессе выполнения диссертационной работы, использованы в курсе «Защита программного обеспечения» кафедры «Криптология и Дискретная Математика» НИЯУ МИФИ для создания лабораторных работ и лекционного материала.

Публикации и апробация работы: Результаты диссертации были изложены в 12 публикациях, 4 из которых были опубликованы в журналах рецензируемых ВАК

РФ. Результаты работы докладывались на конференциях и семинарах различного уровня:

– X Международная конференция «РусКрипто». 2008 г.;
– XIV Международная телекоммуникационная конференция студентов и молодых ученых «Молодежь и наука», г. Москва, 2010 г.;
– Инфофорум, г. Москва, 2010 г.;
– Научная сессия НИЯУ МИФИ, г. Москва, 2011, 2012 гг.;
– Летняя школа Microsoft Research Summer School, Кембридж, Великобритания, 2011 г.;
– XIX Всероссийская научно-практическая конференция «Проблемы информационной безопасности в системе высшей школы», г. Москва, 2012 г;
– научный семинар кафедры «Информационная безопасность» Московского Государственного Технического Университета им. Н.Э. Баумана, 2012 г. (25 апреля 2012года).

Основные положения, выносимые на защиту:

− результаты анализа атак на программные реализации алгоритмов защиты информации, условия для успешной реализации рассмотренных атак;
− модель табличной реализации отображений, входящих в состав алгоритмов защиты информации;
− методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа;
− выбор параметров запутанной программной реализации алгоритма ГОСТ 28147-89 в режиме простой замены.

Структура и объем работы. Работа состоит из введения, четырех глав, заключения, списка литературы, включающего 107 наименований, и одного приложения. Текст диссертации изложен на 125 страницах, включая 20 рисунков и 3 таблицы.

СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обосновывается актуальность темы диссертационной работы, выделяются и формулируются цель и задачи исследования, описывается структурно-логическая схема диссертационной работы.

В **первой главе** осуществляется исследование способов защиты и анализа программных реализаций алгоритмов защиты информации. Существующие подходы к обеспечению безопасности программных реализаций алгоритмов защиты информации основаны на использовании запутывающих преобразований. В настоящее время существует несколько методов запутывания алгоритмов защиты информации, которые можно разделить на три группы в зависимости от используемых преобразований:

– табличные;
– алгебраические;
– автоматные.

В основе табличных методов запутывания алгоритмов защиты информации лежит реализация отображений с помощью таблиц замены. Такие таблицы содержат ключевую информацию и защищены от анализа с помощью маскирующих преобразований. Некоторые отображения сами по себе допускают эффективную реализацию в виде таблиц замены, например, отображение S-бокс, входящее в состав цикловой функции большинства современных алгоритмов защиты информации, тривиально реализуется с помощью таблицы замены. В общем случае, для произвольного отображения эффективное табличное представление отсутствует. В

диссертационной работе автором предложены табличные реализации некоторых классов отображений, используемых в современных алгоритмах защиты информации:

- аффинные отображения;
- отображение сложения с константной в кольце вычетов целых чисел;
- отображение управляемого циклического сдвига координат двоичного вектора.

Алгебраические методы запутывания программных реализаций алгоритмов защиты информации основаны на представлении алгоритма в виде системы уравнений над конечным полем: в результате, алгоритм, заданный отображением $E_k: V_n \to V_n, k \in V_m$ реализуется в виде (2):

$$\Psi_i^k = \begin{cases} \Psi_{i,1}^k(x_1, x_2, \dots, x_t), \\ \Psi_{i,2}^k(x_1, x_2, \dots, x_t), \\ \dots \\ \Psi_{i,s}^k(x_1, x_2, \dots, x_t). \end{cases} \quad (1)$$

$$E_k(x) = \Psi_R^k \circ \dots \circ \Psi_1^k(x) \quad (2)$$

где $i \in \{1, \dots, R\}$; $x_j \in GF(2^p)$; $\Psi_{i,j}^k: V_n \to V_q$; $s \cdot q = t \cdot p = n$; R – число циклов алгоритма. Для того чтобы скрыть ключевую информацию, содержащуюся в координатных функциях системы (2) к ним применяются маскирующие преобразования.

Автоматные методы запутывания используют теорию конечных автоматов для запутывания программной реализации алгоритма защиты информации. Алгоритм подлежащий защите реализуется в виде конечного автомата к которому применяются запутывающие преобразования.

Основным недостатком автоматных и алгебраических методов запутывания программных реализаций алгоритмов защиты информации является большая емкостная сложность запутанных программных реализаций.

В настоящей работе был проведен обзор существующих решений по обеспечению безопасности программных реализаций алгоритмов защиты информации от анализа в недоверенных средах. Результаты обзора приведены в таблице 1. В правой колонке таблицы приведена сложность наилучшей атаки на программную реализацию алгоритма защиты информации, выраженная в количестве выполнений анализируемого алгоритма.

Таблица 1 – Трудоемкость анализа существующих защищенных реализаций

Запутанная программная реализация алгоритма защиты информации	**Сложность атаки, кол-во выполнений алгоритма**
Защищенный DES, Chow S., 2002	2^{14}
Защищенный AES-128, Chow S., 2002	2^{30}
Защищенный AES-128, Bringer J., 2006	2^{17}
Защищенный AES-128, Щелкунов Д., 2010	2^{24}

По результатам проведенного анализа было выявлено, что все рассмотренные способы запутывания программных реализаций алгоритмов защиты информации от анализа в недоверенных средах обладают существенными недостатками. Предлагаемая в работе методика запутывания лишена выявленных недостатков.

Во **второй главе** предлагается модель табличной реализации отображений в терминах теории графов. Производится построение модели нарушителя. Приводятся результаты исследования атак на программные реализации алгоритмов защиты информации, основанные на табличном представлении отображений.

В диссертационной работе рассматривается угроза извлечения нарушителем ключевой информации из программной реализации алгоритмов защиты информации. Нарушитель обладает учетной записью привилегированного пользователя, имеет физический доступ к СВТ и способен:

- считывать/записывать значение произвольного регистра/ячейки памяти СВТ в любой момент времени;
- использовать средства статического и динамического анализа ПО;
- вносить ошибки в работу алгоритма защиты информации.

При этом нарушитель может:

- точно выбрать время внесения ошибки;
- точно выбрать значение вносимой ошибки;
- вносить ошибку, которая влияет на значение произвольного числа заданных промежуточных переменных;
- вносить ошибку, которая приводит к изменению произвольного числа бит целевых переменных.

Для обеспечения безопасности программных реализаций алгоритмов защиты информации в соответствии с приведенной моделью нарушителя в диссертационной работе решается задача синтеза запутывающих преобразований. Пусть $\mathcal{F} = \{E_k: V_n \to V_n | \ k \in V_m\}$ – семейство отображений, где V_n – множество двоичных векторов длины n, $E_k \in \mathcal{F}$ – алгоритм защиты информации, подлежащий запутыванию, $\tilde{\mathcal{F}}$ – множество запутанных программных реализаций алгоритмов из $\mathcal{F}$. Преобразование $\mathcal{O}: \mathcal{F} \to \tilde{\mathcal{F}}$, называется запутывающим преобразованием, если:

- $\mathcal{O}(E_k)$ реализует отображение эквивалентное отображению $E_k \in \mathcal{F}$;
- временная и емкостная сложность $\mathcal{O}(E_k)$ полиномиально ограничена временной и емкостной сложностью E_k;
- выполняется свойство (3):

$$\left|Pr\left[\mathcal{A}\left(\mathcal{O}(E_k)\right) = 1\right] - Pr\left[\mathcal{S}^{E_k}\left(1^{|E_k|}\right) = 1\right]\right| = \alpha(|E_k|), \qquad (3)$$

где $|E_k|$ – емкостная сложность E_k, $\mathcal{O}(E_k)$ – запутанная программная реализация алгоритма защиты информации E_k, $\mathcal{A}$ – вероятностная полиномиальная машина Тьюринга (ПМТ), моделирующая нарушителя, $\mathcal{S}^{E_k}$ – вероятностная ПМТ, моделирующая нарушителя имеющего доступ к реализации алгоритма защиты информации E_k как к «черному ящику», $\alpha = o(1)$.

Постановка задачи. Для алгоритма защиты информации, заданного отображением (4):

$$E_k: V_n \to V_n, k \in V_m \quad (4)$$

синтезировать запутывающее преобразование $\mathcal{O}$.

Для исследования методов анализа программных реализаций алгоритмов защиты информации от анализа в недоверенных средах в диссертационной работе предлагается модель табличной реализации отображений. Отображению $F: V_n \to V_m$ ставится в соответствие кортеж, состоящий из четырех элементов:

$$(\mathbb{T}, \mathbb{G}, IN, OUT), \quad (5)$$

где:

- $\mathbb{T} = \{\mathbb{T}_\alpha \mid \alpha \in \mathrm{A}\}$ – совокупность таблиц замены;
- $\mathbb{G} = (\mathrm{A}, U)$ – ориентированный граф со множеством вершин $\mathrm{A} = \{\alpha_1, \dots, \alpha_z\}$ и множеством дуг U;
- $IN: V_n \to V_{w_1} \times \dots \times V_{w_s}$ – входное отображение;
- $OUT: V_{v_1} \times \dots \times V_{v_t} \to V_m$ – выходное отображение.

Каждая таблица $\mathbb{T}_\alpha \in \mathbb{T}$ реализует отображение $F_\alpha: V_{n_{1,\alpha}} \times \dots \times V_{n_{i,\alpha}} \to V_{m_\alpha}$. Множество A соответствует множеству индексов таблиц замены. Дуга $(\alpha_i, \alpha_j) \in U$, тогда и только тогда, когда в процессе вычисления значения отображения $y = F(x)$ выходное значение таблицы T_{α_i} является входным значением таблицы T_{α_j}.

Входное и выходное значения $x \in V_n, y \in V_m$; $y = F(x)$ отображения F определяется состояниями $(x_1, \dots, x_s) \in V_{w_1} \times \dots \times V_{w_s}$ и $(y_1, \dots, y_t) \in V_{v_1} \times \dots \times V_{v_t}$ соответсвенно. Входное отображение IN ставит в соответствие входному значению $x \in V_n$ начальное состояние $(x_1^0, \dots, x_s^0)$, выходное отображение OUT по завершающему состоянию $(y_1, \dots, y_t)$ возвращает выходное значение $y \in V_m$. В процессе вычисления значения $y = F(x)$ происходит последовательное вычисление значений координат состояния $(y_1, \dots, y_v)$, таким образом, граф $\mathbb{G} = (A, U)$ состоит из v компонент связанности каждая из которых является деревом. Ниже приведен алгоритм, позволяющий по заданной табличной реализации $(\mathbb{T}, \mathbb{G}, IN, OUT)$ отображения F для входного значения $x \in V_n$ вычислить выходное значение $y = F(x) \in V_m$.

Пусть $\mathbb{G}_\mathrm{i}$ – дерево входящее в состав графа $\mathbb{G}$ с корнем, соответствующим координате y_i выходного состояния $(y_1, \dots, y_t)$; $L_i(\mathbb{G}_\mathrm{i})$ – множество вершин дерева $\mathbb{G}_\mathrm{i}$ находящихся на i-ом ярусе. При вычислении значения отображения получается следующая последовательность состояний $x^0 = (x_1^0, \dots, x_{s_0}^0)$, $x^1 = (x_1^1, \dots, x_{s_1}^1)$, ..., $x^{D-1} = (x_1^{D-1}, \dots, x_{s_{D-1}}^{D-1})$, где $D = diam(\mathbb{G}_\mathrm{F})$. Перенумеруем таблицы замены $\mathbb{T}_\alpha$ следующим образом: $\alpha = (i, j) \Leftrightarrow \alpha \in L_i(\mathbb{G}_\mathrm{F}), x_j^{i+1} = \mathbb{T}_\alpha(x^i)$.

Входные данные алгоритма:

- $(\mathbb{T}, \mathbb{G}, IN, OUT)$ – табличная реализация отображения $F: V_n \to V_m$;
- $x \in V_n$ – входное значение.

Выходные данные алгоритма:

- $y = F(x) \in V_m$ – выходное значение.

Описание алгоритма:

1. Вычислить начальное состояние $x^0 = (x_1^0, \dots, x_{s_0}^0) = IN(x)$;
2. Для i от 0 до $D + 1$ выполнить:

2.1. Вычислить состояние $x^{i+1} = \left(x_1^{i+1}, \dots, x_{s_{i+1}}^{i+1}\right)$:

$$\begin{cases} \mathrm{x}_1^{\mathrm{i+1}} = \mathbb{T}_{\mathrm{i},1}\left(\mathrm{x}_{\mathrm{j}_{1,1}}, \dots, \mathrm{x}_{\mathrm{j}_{1,\mathrm{l}}}\right), \\ \mathrm{x}_2^{\mathrm{i+1}} = \mathbb{T}_{\mathrm{i},2}\left(\mathrm{x}_{\mathrm{j}_{2,1}}, \dots, \mathrm{x}_{\mathrm{j}_{2,\mathrm{l}}}\right), \\ \mathrm{x}_3^{\mathrm{i+1}} = \mathbb{T}_{\mathrm{i},3}\left(\mathrm{x}_{\mathrm{j}_{3,1}}, \dots, \mathrm{x}_{\mathrm{j}_{3,\mathrm{l}}}\right), \\ \dots \\ \mathrm{x}_{\mathrm{k_{i+1}}}^{\mathrm{i+1}} = \mathbb{T}_{\mathrm{i},\mathrm{k_{i+1}}}\left(\mathrm{x}_{\mathrm{j}_{\mathrm{k_{i+1}},1}}, \dots, \mathrm{x}_{\mathrm{j}_{\mathrm{k_{i+1}},\mathrm{l}}}\right), \end{cases} \quad (6)$$

3. Вычислить выходное значение $y = OUT(x^{D-1})$.

Пусть $|\mathbb{T}_\alpha|$ – количество элементов содержащихся в таблице $\mathbb{T}_\alpha \in \mathbb{T}$. Определим емкостную сложность табличной реализации отображения (5) как $\eta_F = \sum_{\alpha \in A} |\mathbb{T}_\alpha|$ равную суммарному размеру таблиц замены, входящих в состав табличной реализации. Временная сложность табличной реализации отображения (5) определяется как $\tau_F = |U|$ количество дуг в графе $\mathbb{G}$.

Для начальных состояний $x = (x_1, \dots, x_s), x' = (x'_1, \dots, x'_s) \in V_{w_1} \times \dots \times V_{w_s}$ и соответствующих завершающих состояний $y = (y_1, \dots, y_t), y' = (y'_1, \dots, y'_t) \in V_{v_1} \times \dots \times V_{v_t}$ определим разностную характеристику табличной реализации отображения F $\delta_{a,b} = P\{(\gamma_1, \dots, \gamma_t) = b\}$ при случайном и равновероятном выборе $x, x' \in V_{w_1} \times \dots \times V_{w_s}$, где $a = x \oplus x',\ a \in V_{w_1} \times \dots \times V_{w_s},\ b \in V_t$,

$$\gamma_i = \begin{cases} 0, \text{если } y_i = y'_i, \\ 1, \text{если } y_i \neq y'_i. \end{cases} \quad (7)$$

для $i = 1, \dots, t$. Разностная характеристика табличной реализации отображений позволяет оценить сложность атаки на запутанную программную реализацию алгоритма защиты информации. Количество материала необходимое для анализа запутанной программной реализации алгоритмов защиты информации обратно пропорционально преобладанию разностной характеристики табличной реализации отображений.

Ниже приведены исследованные в диссертационной работе атаки на программные реализации алгоритмов защиты информации:

- Chow S., 2002;
- Jacob M., 2002;
- Link H.E., 2005;
- Goubin L., 2007;
- Wyseur B., 2007.

Рассмотренные в работе атаки на программные реализации алгоритмов защиты информации, основанные на табличном представлении отображений, являются разновидностями разностного анализа по ошибкам вычислений и осуществляются посредством выполнения следующей последовательности шагов:

- выбирается отображение алгоритма защиты информации, на который будет производиться атака, с целью определения неизвестного параметра π;
- выдвигается гипотеза о неизвестном параметре π;
- генерируется набор входных данных, с последующим применением к ним выбранного отображения;

– производится анализ получившихся выходных значений, в результате чего выдвинутая гипотеза о неизвестном параметре π либо подтверждается, либо опровергается.

В результате, происходит последовательное определение неизвестных параметров отображений в составе алгоритма защиты информации, на основании которых нарушитель вычисляет ключевую информацию.

Методы анализа программных реализаций алгоритмов защиты информации, рассмотренные в работе, используют разностные характеристики табличной реализации отображений, существенно отличающиеся от значения 2^{-t}, при генерации входных данных. Это позволяет значительно сократить объем необходимого материала для проведения атаки. Таким образом, для того чтобы противостоять методу разностного анализа по ошибкам вычислений необходимо, чтобы для различных разностей $a \in V_{w_1} \times \ldots \times V_{w_s},\ b \in V_t$ выполнялось условие $\delta_{a,b} \approx 2^{-t}$.

В **третьей главе** приводится описание табличной реализации некоторых классов отображений, входящих в состав современных алгоритмов защиты информации. Излагается созданная методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа. Приводится оценка стойкости полученных запутанных программных реализаций алгоритмов защиты информации к методу разностного анализа по ошибкам вычислений.

Предложенная автором работы методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа, использует табличную реализацию отображений. В диссертационной работе предложены описания табличных реализаций некоторых классов отображений, наиболее часто встречающихся в алгоритмах защиты информации, основанных на отображениях регистрового типа, а именно:

– аффинные отображения;
– отображения узлов замены S-бокс;
– сложение с константой в кольце вычетов целых чисел;
– отображение управляемого циклического сдвига координат двоичного вектора.

Аффинное отображение $F_A: V_n \to V_m,\ F_A(x) = Mx + b$, где M – матрица над $GF(2)$ размера $n \times m,\ b \in V_m$. Пусть $x = (x_1, \ldots, x_v) \in V_n, x_i \in V_s;\ b = (b_1, \ldots, b_u) \in V_m,\ b_i \in V_t;\ M = (m_{ij}),\ m_{ij}$ – подматрица размера $s \times t$ матрицы M. Тогда $F_A(x) = (y_1, \ldots, y_u), y_j = \sum_{i=1}^{s} m_{ij}x_i + b_j$. В результате табличная реализация отображения F_A описывается кортежем $(\mathbb{T}_A, \mathbb{G}_A, IN_A, OUT_A)$, где:

– $IN_A(x) = (x_1, \ldots, x_v)$, где $x = (x_1, \ldots, x_v) \in V_n, x_i \in V_s$ – входное отображение;
– $OUT_A(y_1, \ldots, y_u) = (y_1, \ldots, y_u)$ – выходное отображение;
– $\mathbb{T}_A = \{\mathbb{T}_{A.1}, \mathbb{T}_{A.2}\}$ – таблицы замены двух типов. Таблицы замены первого типа $\mathbb{T}_{A.1} = \{\mathbb{T}_{1,(1,1)}, \ldots, \mathbb{T}_{1,(1,v)}, \ldots, \mathbb{T}_{1,(u,1)}, \ldots, \mathbb{T}_{1,(u,v)}\}$ реализуют отображение $\mathbb{T}_{1,(i,j)}: V_s \to V_t;\ \mathbb{T}_{1,(i,j)}(x) = m_{ij}x$. Таблицы замены второго типа $\mathbb{T}_{A.2} = \{\mathbb{T}_{2,(1,1)}, \ldots, \mathbb{T}_{2,(1,v-1)}, \ldots, \mathbb{T}_{2,(u,1)}, \ldots, \mathbb{T}_{2,(u,v-1)}\}$ реализуют операцию сложения двух векторов $\mathbb{T}_{2,(i,j)}: V_t \times V_t \to V_t;\ \mathbb{T}_{2,(i,j)}(x, y) = x \oplus y$;
– $\mathbb{G}_A = (X, U)$ – граф табличного представления отображения F_A изображен на рисунке 1.

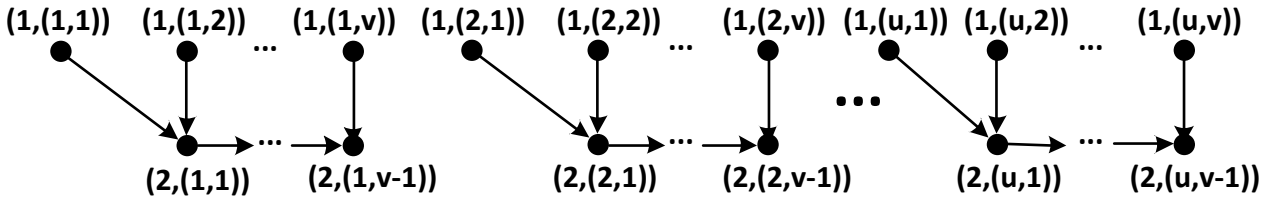

Рисунок 1 – Граф табличной реализации аффинного отображения.

Отображение сложения с константой $F_{\boxplus}: Z_{2^n} \times Z_{2^n} \to Z_{2^n}$ в кольце вычетов целых чисел $Z_{2^n}(+,\cdot)$ входит в состав функции усложнения таких алгоритмов как ГОСТ 28147-89, IDEA, RC5, RC6. Данное отображение определяется соотношением (8):

$$\mathrm{F}_{\boxplus}(\mathrm{x},\mathrm{k}) = [\mathrm{x}_1, \dots, \mathrm{x}_\mathrm{n}]_2 \ + \ \mathrm{k} \ (\mathrm{mod}\ 2^\mathrm{n}), \tag{8}$$

$x \in Z_{2^n}, x = [x_1, \dots, x_n]_2$ – целое число двоичное представление, которого имеет вид $(x_1, \dots, x_n)$, $+$ арифметическая операция сложения.

Табличная реализация отображений (8) описывается кортежем $(\mathbb{T}_{\boxplus}, \mathbb{G}_{\boxplus}, IN_{\boxplus}, OUT_{\boxplus})$:

– $IN(x) = (x_1, \dots, x_t)$ – входное отображение, где $x_i \in V_s,\ s \cdot t = n$;

– $OUT_{\boxplus}(x_1, \dots, x_t) = (x_1, \dots, , x_t)$ – выходное отображение;

– $\mathbb{T}_{\boxplus} = \{\mathbb{T}_{1,1}, \dots, \mathbb{T}_{1,t}, \dots, \mathbb{T}_{t,1}, \dots, \mathbb{T}_{t,t}\}$ – множество таблиц замены $\mathbb{T}_{i,j}: V_{s+1} \times V_s \to V_{s+1}$

$$\mathbb{T}_{i,j}(x,y) = \begin{cases} y \boxplus k_i \boxplus c, \text{если } i \le j \\ x, \text{если } i > j \end{cases}, \tag{9}$$

где c – старший бит числа $x, k = (k_1, \dots, k_t), k_i \in V_s$;

– $\mathbb{G}_{\boxplus} = (X, U)$ – граф табличного представления изображен на рисунке 2.

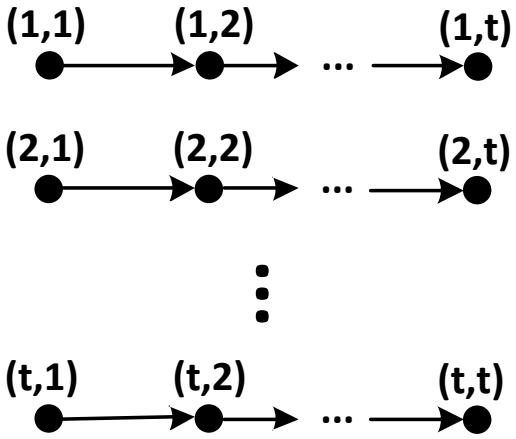

Рисунок 2– Граф табличной реализации отображения сложения с константой в кольце вычетов Z_{2^n}

Отображение управляемого циклического сдвига координат двоичного вектора $F_{\ll}: V_n \times Z_n \to V_n$ имеет вид (10):

$$\mathrm{F}_{\ll}((\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{n}), \mathrm{z}) = (\mathrm{x}_{\mathrm{z}+1}, \mathrm{x}_{\mathrm{z}+2}, \dots, \mathrm{x}_\mathrm{n}, \mathrm{x}_1, \dots, \mathrm{x}_{\mathrm{z}-1}, \mathrm{x}_\mathrm{z}), \quad (10)$$

Положим $x = (x_1, \dots, x_t),\ x_i \in V_s,\ s \cdot t = n;\ z = z_1 + z_2 2^s + \cdots + z_p 2^{ps},\ z_i \in Z_{2^s};\ p = \lceil \frac{\log n}{s} \rceil$. Пусть M_v – матрица размера $n \times n$ над $GF(2)$, осуществляющая циклический сдвиг координат двоичного вектора на v позиций. Тогда, отображение (10) может быть представлено в виде (11):

$$\mathrm{F}_{\ll}((\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{n}), \mathrm{z}) = \mathrm{M}_{\mathrm{z}_\mathrm{p}} \circ \mathrm{M}_{\mathrm{z}_{\mathrm{p}-1}} \circ \dots \circ \mathrm{M}_{\mathrm{z}_1}(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{n})^\mathrm{T}, \quad (11)$$

Таким образом, отображение (10) является суперпозицией p параметризованных линейных преобразований V_n. Положим $M_{z_k} = (m_{ij}^{z_k})$, где $m_{ij}^{z_k}$– матрицы размера $s \times t,\ i = \{1,2,\dots,u\},\ j = \{1,2,\dots,v\}, t \cdot u = m,\ s \cdot v = n.$ Отсюда табличная реализация отображения циклического сдвига в V_n описывается кортежем ($\mathbb{T}_{\ll}$, $G_{\ll}$, $IN_{\ll}, OUT_{\ll}$):

– $IN_{\ll}(x, z) = (x_1, \dots, x_t, z_1, \dots, z_p), x_i \in V_s, z = z_1 + z_2 2^s + \cdots + z_p 2^{ps}, z_i \in Z_{2^s}$;

– $OUT_{\ll}(x_1, \dots, x_t, z_1, \dots, z_p) = (x_1, \dots, x_t)$ – выходное отображение;

– $\mathbb{T}_{\ll} = \{\mathbb{T}_{\ll,1}, \mathbb{T}_{\ll,2}\}$ – таблицы замены двух типов. Таблицы замены первого типа $\mathbb{T}_{\ll,1} = \{\mathbb{T}_{\ll,1,z_1,(1,1)}, \dots, \mathbb{T}_{\ll,1,z_1,(1,v)}, \dots, \mathbb{T}_{\ll,1,z_1,(u,1)}, \dots, \mathbb{T}_{\ll,1,z_1,(u,v)}, \dots, \mathbb{T}_{\ll,1,z_p,(u,v)}\}$ реализуют отображение $\mathbb{T}_{\ll,1,z_k,(i,j)}: V_s \times V_s \to V_t;\ \mathbb{T}_{\ll,1,z_k,(i,j)}(x_i, z_k) = m_{ij}^{z_k} x_i$, а таблицы второго типа $\mathbb{T}_{\ll,2} = \{\mathbb{T}_{\ll,2,z_1,(1,1)}, \dots, \mathbb{T}_{\ll,2,z_1,(1,v-1)}, \dots, \mathbb{T}_{\ll,2,z_1,(u,1)}, \dots, \mathbb{T}_{\ll,2,z_1,(u,v-1)}, \dots, \mathbb{T}_{\ll,2,z_p,(u,v-1)}\}$ реализуют операция сложения двух векторов $\mathbb{T}_{\ll,2,z_k,(i,j)}(x, y) = x \oplus y$;

– $G_{\ll} = (X, U)$ – граф табличного представления отображения (10), приведен на рисунке 3.

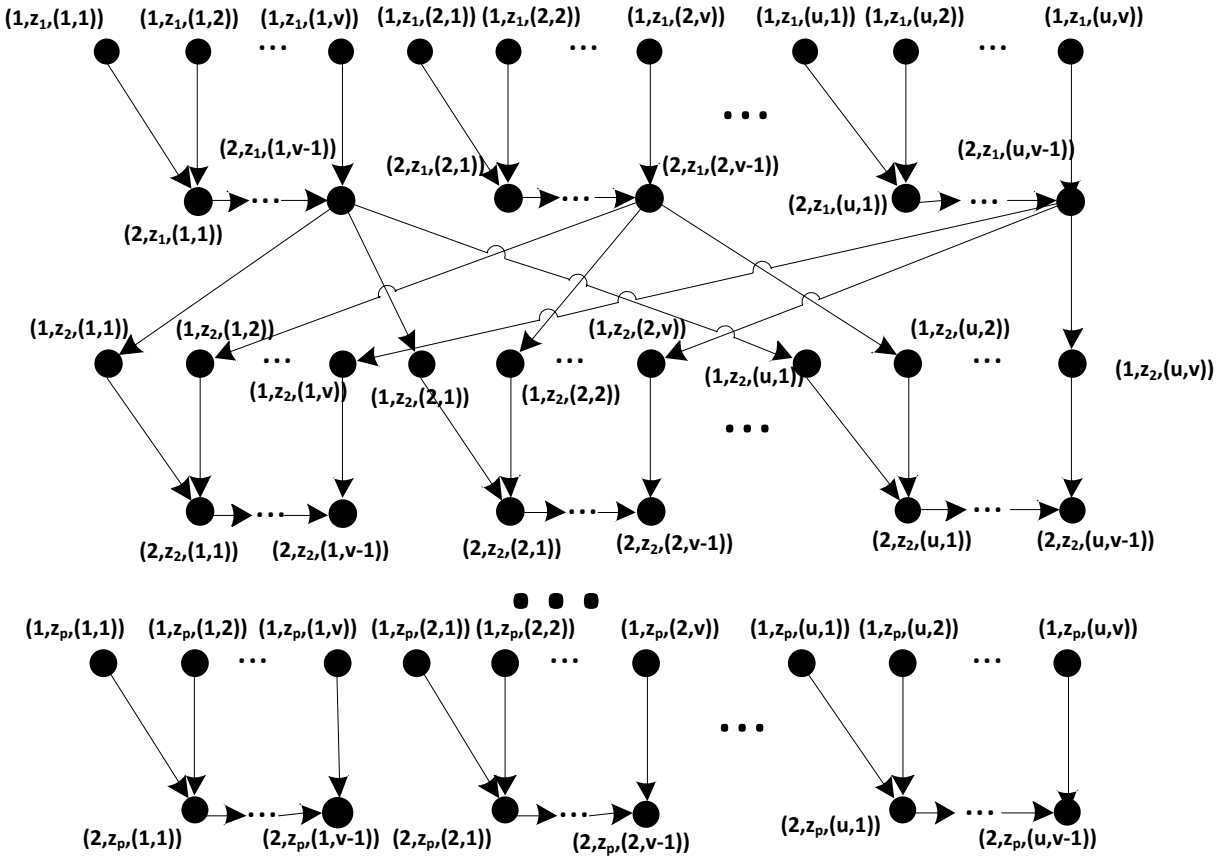

Рисунок 3 – Граф табличной реализации отображения циклического сдвига координат двоичного вектора.

Отображения узлов замены S-бокс имеют тривиальную табличную реализацию.

Запутывание программной реализации алгоритмов защиты информации, основанных на преобразовании регистрового типа. Алгоритмы защиты информации, основанные на преобразованиях регистрового типа, имеют итерационную структуру, в основе которой лежит цикловая функция, представимая в общем виде (12):

$$y(x_1, x_2, \dots, x_m) = (x_2, x_3 \oplus \psi_2(x_2, \dots, x_m), \dots, x_m \oplus \psi_{m-1}(x_2, \dots, x_m), \\ x_1 \oplus \psi_m(x_2, \dots, x_m)), \qquad (12)$$

где $x_i \in V_n$; $\psi_j : V_n \times \dots \times V_n \to V_n$ – функция усложнения, реализующая нелинейное отображение, $i = 1, \dots, m$; $j = 2, \dots, m$. Большинство современных алгоритмов защиты информации, основанных на преобразовании регистрового типа, используют в качестве нелинейных отображений:

- сложение с константой в кольце вычетов Z_{2^n};
- отображение заданное с помощью узлов замены S-бокс;
- управляемый циклический сдвиг координат двоичного вектора.

Предложенная методика запутывания программной реализации алгоритма защиты информации, основанного на преобразовании регистрового типа, реализуется за счет следующей последовательности шагов:

- осуществление декомпозиции отображений входящих в цикловую функцию алгоритма защиты информации: выделение линейных и нелинейных отображений;
- реализация декомпозированных отображений в виде таблиц замены;

– маскирование линейной составляющей декомпозированного отображения путем умножения на перемешивающую матрицу;

– маскирование таблиц замены, с помощью случайных биективных преобразований.

Декомпозиция отображений, входящих в алгоритм защиты информации. Цикловая функция алгоритма защиты информации, основанного на преобразованиях регистрового типа, может быть представлена в виде (13):

$$y(x) = M(\psi(x)) \oplus N(x), \tag{13}$$

где $x = (x_1, \dots, x_m) \in V_{mn}$; $M, N: V_{mn} \to V_{mn}$ – линейные отображения, $\psi: V_{mn} \to V_{mn}$ – нелинейное отображение. Отображения M, N осуществляют циклический сдвиг и побитовое сложение двоичных векторов:

$$\begin{gathered} M(x_1, \dots, x_{m-1}) = (0, x_1, \dots, x_{m-1}), \\ N(x_1, \dots, x_m) = (x_2, \dots, x_m, x_1). \end{gathered} \tag{14}$$

Нелинейное отображение ψ имеет вид (15):

$$\psi(x_1, \dots, x_{m-1}) = \big(\psi_2(x_1, \dots, x_{m-1}), \dots, \psi_m(x_1, \dots, x_{m-1})\big). \tag{15}$$

Реализация декомпозированных отображений в виде таблиц замены. На данном шаге предложенной методики, осуществляется реализация декомпозированных отображений в виде таблиц замены. Описание табличной реализации некоторых отображений, входящих в состав функции усложнения современных алгоритмах защиты информации приведены в данной главе.

Маскирование линейной составляющей отображения. Для того чтобы сокрыть информацию о структуре отображения и сделать распространение ошибок, вносимых в процессе анализа, равномерным, осуществляется маскирование линейного слоя отображения цикловой функции запутываемого алгоритма путем умножения на перемешивающую матрицу.

Пусть $\bar{x} = (x_1, \dots, x_k) \in V_{w_1} \times \dots \times V_{w_k}$ и $\bar{y} = \big(y_1, \dots, y_j\big) \in V_{v_1} \times \dots \times V_{v_j}$ соответственно входное и выходное состояния табличной реализации цикловой функции, запутываемого алгоритма защиты информации. Обозначим $\overline{\mathbb{L}} = (y_{i_1}, \dots, y_{i_s})$ как вектор, состоящий из элементов выходного состояния $\bar{y}$, содержащий биты не подверженные воздействию функции усложнения ψ, а $\overline{\mathbb{R}} = (y_{l_1}, \dots, y_{l_t})$ – как вектор, состоящий из остальных элементов $\bar{y}$. Пусть $\widetilde{\mathbb{L}} = \big(\tilde{y}_{i_1}, \dots, \tilde{y}_{i_s}\big) = \Psi \circ \bar{y}$, где Ψ – случайное, невырожденное линейное отображение. Тогда в результате маскирования линейного слоя (14) выходное состояние табличной реализации цикловой функции имеет вид:

$$\tilde{y} = (y_{l_1} \parallel \tilde{y}_{i_1}, \dots, y_{l_t} \parallel \tilde{y}_{i_s}), \tag{16}$$

где $\parallel$ – конкатенация векторов.

Маскирование линейного слоя позволяет уменьшить преобладание разностной характеристики табличной реализации цикловой функции (13), в результате чего,

значительно возрастает трудоемкость метода разностного анализа по ошибкам вычислений.

Маскирование таблиц замены. Для того чтобы скрыть ключевую информацию содержащуюся в таблицах замены, полученных на предыдущем шаге методики, к ним применяются маскирующие преобразования.

Пусть $(\mathbb{T}_F, \mathbb{G}_F = (X, U), IN_F, OUT_F)$ – табличная реализация отображения $F: V_n \to V_m$. Для исходной таблицы замены $\mathbb{T}_\alpha \in \mathbb{T}_F$; $\mathbb{T}_\alpha: V_{n_{1,\alpha}} \times \ldots \times V_{n_{i,\alpha}} \to V_{m_\alpha}$ маскированная таблица $\mathbb{T}'_\alpha$ имеет вид (17):

$$\mathbb{T}'_\alpha = \Phi_\beta^{-1} \circ \mathbb{T}_\alpha \circ \Phi_\alpha, \tag{17}$$

где $\Phi_\alpha = (\Phi_{\alpha,1} || \Phi_{\alpha,2} || \ldots || \Phi_{\alpha,i})$; $\Phi_{\alpha,j}: V_{n_{\mathrm{j},\alpha}} \to V_{n_{\mathrm{j},\alpha}}$, $\Phi_\beta: V_{m_\alpha} \to V_{m_\alpha}$ – случайные биективные преобразования, $||$ – операция конкатенации отображений, $j = 1, \ldots, i$. Если из вершины $\alpha \in X$ не выходит ни одной дуги, то $\Phi_\beta = E$, если в вершину α не входит ни одна дуга, то $\Phi_\alpha = E$, где E – тождественное преобразование.

В **<u>четвертой главе</u>** приведено описание запутанной программной реализации алгоритма ГОСТ 28147-89 в режиме простой замены и дано обоснование выбранных параметров запутанной реализации. Приводятся примеры практического применения разработанной автором методики запутывания алгоритмов защиты информации, основанных на преобразовании регистрового типа.

С помощью предложенной методики запутывания алгоритмов защиты информации, основанных на преобразованиях регистрового типа, в диссертационной работе была получена запутанная программная реализация алгоритма ГОСТ 28147-89 в режиме простой замены. Отображение одного раунда алгоритма ГОСТ 28147-89 $F: V_{64} \to V_{64}$ может быть представлено в виде (17):

$$y = F(x) = M(SB(x \boxplus k)) + N(x), \tag{18}$$

где $SB: V_{64} \to V_{64}, SB = SB_1 \parallel \cdots \parallel SB_8 \parallel SB_9 \parallel \cdots \parallel SB_{16}$ – преобразование множества V_{64}, заданное в виде узлов замены $SB_j: V_4 \to V_4$, при этом $SB_9, \ldots, SB_{16}$ – тождественные отображения; $\boxplus: V_{32} \to V_{32}$ – бинарная операция сложения в кольце вычетов $Z_{2^{32}}$, $M: V_{64} \to V_{64}, N: V_{64} \to V_{64}$ – линейные преобразования, осуществляющие операцию циклического сдвига двоичного вектора. Пусть $y = (y_1, \ldots, y_v), y_i \in V_t; x = (x_1, \ldots, x_u), x_j \in V_s$, тогда:

$$y_i(x_j) = \sum_{j=0}^{t-1} M_{i,j}\left(SB_j(x_j \boxplus k_j \boxplus c_{j-1} \ldots \boxplus c_1)\right) + \sum_{j=0}^{t-1} N_{i,j} x_j, \tag{19}$$

где $j = \overline{1,u}; i = \overline{1,v}; t = \lceil 64/s \rceil$; $c_j \in \{0,1\}$ – индикатор переполнения возникающего при сложении x_j и k_j по модулю 2^s.

Определим отображение $T_{i,j}(x,z): V_{l+1} \times V_s \to V_{l+1}$ как

$$\begin{gathered} T_{i,j}(x,z) = M_{i,j}\left(SB_j(z \boxplus k_j \boxplus x_{l+1})\right) + N_{i,j} z, j = \overline{2,t} \\ T_{i,1}(x,z) = M_{i,1}(SB_1(z \boxplus k_1)) + N_{i,1} z, \end{gathered} \tag{20}$$

Тогда отображение (19) описывается соотношением (21):

$$y_i(x) = T_{i,t}\left(T_{i,t-1}\left(\dots T_{i,2}\left(T_{i,1}(x_1), x_2\right)\right), x_t\right) \tag{21}$$

В работе произведена оценка емкостной и временной сложности запутанной программной реализации алгоритма ГОСТ 28147-89:

$$\begin{gathered}\eta_{\text{гост}} = O\left((l+1)2^{l+1+s}\right), \\ \tau_{\text{гост}} = O((s \cdot l)^{-1}).\end{gathered} \tag{22}$$

В таблице 3 приведены значения размера и скорости преобразования входных данных запутанной программной реализации алгоритма ГОСТ 28147-89 в режиме простой замены при различных значениях параметров s и l. Тестирование производилось на ЭВМ со следующими характеристиками: Intel Core 2 Duo, 4 Гб ОЗУ с ОС Windows XP SP3.

Таблица 3 – Результаты тестирования запутанной программной реализации алгоритма ГОСТ 28147-89 в режиме простой замены.

Значение параметров s и l	Размер программной реализации алгоритма, Кб	Скорость преобразования входных данных, Кб/с
$s = 8, l = 4$	521	660
$s = 8, l = 8$	8202	1040
$s = 16, l = 8$	1572873	1470

Для того чтобы противостоять рассмотренным в диссертационной работе методам анализа программных реализаций алгоритмов защиты информации, рекомендуется использовать следующие параметры защищенной реализации алгоритма ГОСТ 28147-89: $s = 8, l = 4$.

Методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа, была использована при разработке программно-аппаратного комплекса защиты приложений от обратного исследования и несанкционированного копирования компании «Актив», для обеспечения безопасности протокола взаимодействия внешнего аппаратного модуля с защищаемым приложением.

Полученная в диссертационной работе запутанная программная реализация алгоритма ГОСТ 28147-89 была использована при разработке программного комплекса «Модуль Контроля Действий Пользователей» в Банке «Возрождение» ОАО, предназначенного для предотвращения утечки конфиденциальной информации из корпоративной информационной системы. Использование запутанной программной реализации алгоритма ГОСТ 28147-89 позволило обеспечить конфиденциальность и целостность журнала аудита и конфигурационной информации, хранимой на АРМ пользователей корпоративной информационной системы Банка.

Результаты исследования существующих методов анализа и методика запутывания программных реализаций алгоритмов защиты информации внедрены в учебный курс «Защита Программного Обеспечения» на кафедре «Криптология и Дискретная Математика» НИЯУ МИФИ. В результате внедрения созданы

лабораторные работы, позволяющие слушателям учебного курса получить практические навыки защиты программных реализаций алгоритмов, основанных на преобразованиях регистрового типа, от анализа в недоверенных средах. Информация о методах анализа программных реализаций алгоритмов защиты информации была включена в курс лекций данного курса.

В **заключении** приведены основные результаты диссертационной работы, представлены выводы, полученные в ходе выполнения работы, а так же рассмотрены пути дальнейшего развития темы исследования.

ОСНОВНЫЕ ВЫВОДЫ И РЕЗУЛЬТАТЫ РАБОТЫ

1. Проведен анализ существующих методов обеспечения безопасности программных реализаций алгоритмов защиты информации от анализа в недоверенных средах. Обоснована необходимость разработки новой методики для запутывания алгоритмов защиты информации, основанных на преобразованиях регистрового типа.
2. Построена модель злоумышленника, реализующего атаки направленные на извлечение ключевой информации из программной реализации алгоритмов защиты информации, основанных на преобразованиях регистрового типа. Построенная модель нарушителя позволила учесть возможные атаки на запутанные программные реализации алгоритмов защиты информации.
3. Проведено исследование известных методов анализа программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа. Определены условия для успешного осуществления изученных атак.
4. Предложена математическая модель табличной реализации отображений, позволяющая описать отображения с помощью совокупности таблиц замены. Предложенная модель позволила получить описание основных классов отображений, входящих в состав современных алгоритмов защиты информации, основанных на преобразованиях регистрового типа.
5. Создана методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа. Предложенная методика позволяет получить запутанные программные реализации алгоритмов информации, стойкие к методу разностного анализа по ошибкам вычислений в недоверенных средах.
6. С помощью созданной методики получена запутанная программная реализация алгоритма ГОСТ 28147-89 в режиме простой замены. Произведен анализ емкостной и временной сложности запутанной программной реализации алгоритма ГОСТ 28147-89.
7. Приведены рекомендации по выбору параметров запутанной программной реализации алгоритма ГОСТ 28147-89 режиме простой замены и проведена оценка ее стойкости к известным методам анализа в недоверенных средах.
8. Созданная методика запутывания программных реализаций алгоритмов защиты информации, основанных на преобразованиях регистрового типа, была использована при разработке системы защиты приложений от несанкционированного копирования в компании «Актив».
9. Полученная запутанная программная реализация алгоритма ГОСТ 28147-89 была использована при разработке системы предотвращения утечки конфиденциальной информации из корпоративной информационной системы в

программном комплексе «Модуль Контроля Действий Пользователей» в ОАО Банк «Возрождение».

10. Результаты исследования существующих методов анализа и запутывания программных реализаций алгоритмов защиты информации были использованы при чтении лекций и проведении лабораторных работ в учебном курсе «Защита Программного Обеспечения» кафедры «Криптология и Дискретная Математика» НИЯУ МИФИ.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Родионов Е. Ю. Применение запутывающих преобразований в криптографических целях. [Электронный ресурс] «РусКрипто'2008». Режим доступа к ресурсу: http://www.ruscrypto.ru/netcat_files/File/ruscrypto.2008.032.zip;

2. Родионов Е. Ю. Применение запутывающих преобразований в криптографии // Безопасность Информационных технологий. 2009. №2. С. 50-52. (Журнал входит в перечень ВАК);

3. Родионов Е. Ю. Преобразования для запутывания симметричных блочных шифров // Безопасность Информационных Технологий. 2009. №3. С. 80-82. (Журнал входит в перечень ВАК);

4. Родионов Е. Ю., Сеник М. В. Предотвращение несанкционированного запуска ПО: подходы к идентификации приложений. // Безопасность Информационных Технологий. 2010. №1. С. 102-105. (Журнал входит в перечень ВАК);

5. Родионов Е. Ю. Контроль доступа к периферийным устройствам ввода/вывода на АРМ пользователей // Безопасность Информационных Технологий. 2010. №1. (Журнал входит в перечень ВАК);

6. Родионов Е. Ю. Разработка системы управления политикой информационной безопасности на предприятии // Научная сессия НИЯУ МИФИ-2010. Т. 3. М.: НИЯУ МИФИ, 2010. С. 159;

7. Родионов Е. Ю. Система контроля действий пользователей в корпоративной информационной системе // Труды научной сессии МИФИ 2010 — 2010. — С. 212–214;

8. Родионов Е. Ю., Варфоломеев А. А. О реализации криптографических примитивов, устойчивых к анализу методом «белый ящик». – Научная сессия МИФИ-2011. XIV Московская международная телекоммуникационная конференция студентов и молодых ученых «Молодежь и Наука». Тезисы докладов в 3-х частях. Ч. 3. М.:МИФИ, 2010. – с. 257-259;

9. Родионов Е. Ю., Варфоломеев А. А. О выборе параметров реализаций симметричных алгоритмов шифрования, устойчивых к анализу методом «белый ящик». – Научная сессия МИФИ-2011. XIV Московская международная телекоммуникационная конференция студентов и молодых ученых «Молодежь и Наука». Тезисы докладов в 3-х частях. Ч. 3. М.:МИФИ, 2010. – с. 255-257;

10. Rodionov E., Matrosov A., Rooting about in TDSS // Virus bulletin. October 2010;

11. Rodionov E. «White-Box Implementation of Symmetric Block Ciphers» [Электронный ресурс]: Microsoft Research Summer School 2011. Режим доступа к ресурсу: http://blogs.msdn.com/b/msr_er/archive/2011/07/12/top-students-descend-on-microsoft-research-cambridge.aspx;

12. Rodionov E., Matrosov A., Harley D., When I'm x64: Bootkit Threat Evolution in 2011 // HAKIN9. Vol №7-02, 2012.