МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ
И ИНФОРМАТИКИ
Кафедра математического моделирования и анализа данных

АВТОМАТИЧЕСКИЕ СРЕДСТВА ОЦЕНКИ НАДЁЖНОСТИ БЛОЧНЫХ КРИПТОСИСТЕМ

Курсовой проект

Свирид Полины Дмитриевны
студента 4 курса,
специальность «компьютерная безопасность»

Научный руководитель:
канд. физ.-мат. наук
доцент С.В. Агиевич

Минск, 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Ещё на этапе разработки криптографических систем важно оценить их стойкость ко всем существующим атакам. История показывает, что шифр, разработанный без должной обработки его защищенности от криптоанализа, часто имеет недостатки и приводит к атакам со стороны других исследователей.

Практически все формы криптоанализа блочных криптосистем используют тот факт, что некоторые характерные особенности структуры открытого текста могут сохраняться при шифровании и проявляться в соответствующих особенностях структур шифртекста.

Нахождение и анализ свойств криптосистемы происходит на этапе распознавания. Традиционно эти свойства искались вручную, что, в общем случае, является сложной задачей, успех которой зависит от опыта и квалификации криптоаналитика. В последнее десятилетие широкое распространение при оценке безопасности криптографических алгоритмов, особенно блочных шифров получили автоматизированные инструменты, основанные на задачах удовлетворения ограничений. К сожалению, большинство автоматизированных методов, опубликованных в литературе, не имеют программной реализации или предоставляют узкие реализации, специфичные для шифра и атаки распознавания, а распространение этих реализаций на другие шифры или атаки требует значительных усилий и опыта.

Заметным исключением является CASCADA (Characteristic Automated Search of Cryptographic Algorithms for Distinctive Attacks) — библиотека Python с открытым исходным кодом для оценки защищенности блочных шифров и других криптографических примитивов от множества атак распознавания путем поиска уязвимых свойств с использованием бит-векторных SMT-решателей.

Библиотека с открытым исходным кодом CASCADA является результатом огромных инженерных усилий, направленных на предоставление современного инструмента для оценки широкого класса криптографических алгоритмов против множества атак распознавания. С этой целью CASCADA имеет модульную структуру, обширную документацию и набор тестов, так что CASCADA может использоваться и дополняться разработчиками и криптоаналитиками.

ГЛАВА 1
АТАКИ РАСПОЗНАВАНИЯ НА БЛОЧНЫЕ КРИПТОСИСТЕМЫ

При разработке и криптоанализе блочного шифра его надёжность оценивается путем проверки того, что никакие известные атаки не могут эффективно вскрыть криптосистему. Большинство криптографических атак включают первоначальный этап распознавания, в котором неслучайное свойство шифра используется для того, чтобы отличить шифр от случайной перестановки. За этапом распознавания следует этап восстановления ключа, но нахождение свойств шифра, которые можно использовать, является самой сложной частью атаки.

Хорошо известными примерами атак, включающих этап распознавания, являются разностная, линейная атаки, а также атаки на связанных ключах, по невозможным дифференциалам, с нулевой корреляцией и RX-атака.

1.1 Основные понятия

Представим блочную криптосистему, действующую на множестве блоков $\{0,1\}^n$ длины n, как семейство подстановок, зависящих от ключей:

$$F = \{F_K : K \in \mathcal{K}\},$$

где $\mathcal{K}$ — множество ключей. Здесь и далее будем предполагать, что ключ K из множества $\mathcal{K}$, $X \in (\{0,1\}^n)^m$ — некоторый набор прообразов из множества наборов блоков и σ из множества подстановок на $\{0,1\}^n$ выбраны случайно, равновероятно.

Свойство криптосистемы F — это пара предикатов $b, b' : (\{0,1\}^n)^m \to \{0,1\}$ такая, что вероятность

$$p = P\{b'(Y) = 1 \mid b(X) = 1, Y = F_k(X)\}$$

существенно отличается от вероятности

$$q = P\{b'(Y) = 1 \mid b(X) = 1, Y = \sigma(X)\}.$$

Иными словами, свойство подстановки — это особенность её действия на m-наборе прообразов.

Используемые на практике криптосистемы F являются блочно-итерационными. В них подстановки F_K представляют собой композиции структурно и алгоритмически простых тактовых подстановок $\Sigma_{k_1}, \Sigma_{k_2}, \ldots, \Sigma_{k_d}$:

$$F_K = \sum_{k_d} \circ \ldots \circ \sum_{k_2} \circ \sum_{k_1},$$

где k_i — тактовые ключи, которые строятся по исходному ключу K с помощью определенного алгоритма.

В современной криптографии любое отличительное свойство F считается слабостью криптосистемы. При разработке криптосистемы стремятся, чтобы её преобразования не обладали свойствами, нехарактерными для случайных подстановок.

Рассмотрим подробнее, какие свойства блочных криптосистем используют различные криптографические атаки.

1.2 Разностная атака

Разностный криптоанализ был предложен израильскими специалистами Эли Бихамом и Ади Шамиром в 1990 году. Используя данный метод, они нашли более эффективный способ вскрытия DES чем метод "грубой силы".

Разностный криптоанализ — метод, основанный на анализе распределения разностей в парах шифртекстов, при фиксированной разности пар открытых текстов. Полученная неоднородность в распределении может быть использована для присвоения вероятностей возможным ключам.

Разность двух текстов $X, X^{'} \in \{0,1\}^n$ для криптосистем, подобных DES представляет собой операцию XOR:

$$\Delta X = X \oplus X^{'}.$$

Пусть F — d-тактовая блочно-итерационная криптосистема, где $Y(i)$, $Y^{'}(i)$ — результаты выполнения i-ых тактов преобразования F_K над X и $X^{'}$ соответственно. Тогда для неё имеем следующую последовательность разностей:

$$\Delta Y(1) = Y(1) \oplus Y^{'}(1) \ , \ \ldots \ , \ \Delta Y(d) = Y(d) \oplus Y^{'}(d).$$

Последовательность $(\alpha, \beta(1), \ldots, \beta(r))$ возможных значений разностей $(\Delta X, \Delta Y(1), \ldots, \Delta Y(r))$ называется r-тактовой характеристикой, а пара $(\Delta X, \Delta Y(r))$ — r-тактовым дифференциалом.

Введём вероятность r-тактового дифференциала с учётом предположения о равномерном распределении на $\{0,1\}^n$ независимых текстов:

$$p_r = P\{\Delta Y(r) = \beta(r) | \Delta X = \alpha\}.$$

Вероятность характеристики определяется аналогично:

$$P\{\Delta Y(1) = \beta(1), \ldots, \Delta Y(r) = \beta(r) | \Delta X = \alpha\}.$$

В общем алгоритме разностной атаки первым этапом является нахождение (d - 1)-тактового дифференциала (α, β(d - 1)) с максимальной вероятностью p_{d-1} ($p_{d-1} \gg 2^{-n}$). Далее полученные результаты могут быть использованы для восстановления ключа.

Таким образом, в разностном криптоанализе используется свойство размерности $m = 2$. Слова-блоки интерпретируются как векторы $\mathbb{F}_2^n$, предикаты задаются ненулевыми векторами α и β:

$$b(X, X') = I\{X \oplus X' = \alpha\}\ ,\ b'(Y, Y') = I\{Y \oplus Y' = \beta\},$$

где $I\{\mathcal{E}\}$ — индикатор наступления события $\mathcal{E}$. Вероятность q близка к 0, а p сильно отличается от неё.

1.3 Атака по невозможным дифференциалам

Метод разностного криптоанализа по невозможным дифференциалам был предложен в 1999 году Эли Бихамом, Ади Шамиром и Алексом Бирюковым. Сам принцип появился несколько раньше, во время популярности разностных атак, но формально определен ещё не был.

Данный метод во многом основан на, уже рассмотренном нами, разностном криптоанализе. В отличие от основной идеи разностной атаки, которая заключается в нахождении дифференциалов с высокой вероятностью, принцип атаки по невозможным дифференциалам заключается в нахождении дифференциалов с нулевой вероятностью:

$$p_r = P\{\Delta Y(r) = \beta(r) | \Delta X = \alpha\} = 0.$$

Следовательно, в разностной атаке по невозможным дифференциалам используется свойство размерности $m = 2$, а предикаты задаются аналогично разностной атаке.

1.4 Разностная атака на связанных ключах

Данный тип атаки впервые был предложен израильским учёным Эли Бихамом в 1992 году. В своей статье он показал эффективность криптоанализа на связанных ключах против LOKI89, LOKI91 и Lucifer.

В криптографии атака с использованием связанного ключа — это любая форма криптоанализа, при которой злоумышленник может наблюдать за работой шифра с использованием нескольких разных ключей, значения которых изначально неизвестны, но при этом злоумышленнику известны некоторые математические отношения, связывающие ключи.

Разностный криптоанализ использует распространение разностей, когда пара различных открытых текстов зашифровывается с использованием одного и того же ключа. Разностный криптоанализ на связанных ключах использует свойства распространения разностей, когда открытые тексты X и $X^{'}$, которые могут быть равными, зашифровываются с разными ключами K и $K^{'}$ соответственно.

Формально r-тактовый дифференциал связанных ключей представляет собой тройку $(\Delta X, \Delta Y, \Delta K)$, где ΔX — разность текстов на входе блочного алгоритма шифрования, ΔY — разность выходов после r-го такта шифрования и ΔK — разность ключей. Вероятность дифференциала связанных ключей в r-раунде — это вероятность того, что разность выходов r-го раунда будет равна ΔY, когда разность на входе равна ΔX, разность ключей равна ΔK, а открытый текст X и ключ K выбираются равномерно случайным образом.

Разностная атака с использованием связанных ключей может происходить по следующему алгоритму:

1. Найти высоковероятностную $(r-1)$-тактовую разность связанных ключей, где r — количество тактов блочного шифра.

2. Выбрать случайным образом X и зашифровать под ключом K, чтобы получить шифртекст Y. Вычислить $X^{'} = X + \Delta X$ и зашифровать под ключом $K^{'} = K + \Delta K$ для получения шифртекста $Y^{'}$.

3. Найти все возможные пары ключей последнего раунда $(K_r, K_r^{'})$, такие, что разница между $D_{K_r}(Y)$ и $D_{K_r^{'}}(Y^{'})$ равна ΔY, где $D_K(Y)$ — функция расшифрования для входа Y и тактового ключа K. Добавить один к счетчику, который соответствует одной из ранее вычисленных пар ключей.

4. Повторять предыдущие два шага, пока одна или несколько пар ключей последнего раунда не будут встречаться значительно больше, чем другие.

Разностная атака на связанных ключах не обязательно должна следовать алгоритму, описанному выше. В общем случае, любую атаку, использующую разность связанных ключей, можно назвать атакой с использованием связанного ключа.

1.5 Линейная атака

Метод линейного криптоанализа впервые был предложен в 1993 году японским учёным Матсуи. В своей работе он показал, как можно осуществить атаку на алгоритм шифрования DES, сократив сложность анализа до 2^{47}. Впоследствии линейный криптоанализ был распространён и на другие криптографические примитивы.

Пусть F — d -тактовая блочная криптосистема.

Матсуи показал, что существует возможность заменить функцию шифрования линейным статистическим аналогом:

$$\alpha \cdot X \oplus \beta \cdot Y,$$

который выполняется с некоторой вероятностью p. В отличие от разностного криптоанализа, в котором большое значение вероятности гарантирует успех атаки, в линейном криптоанализе успех может быть обеспечен как уравнениями с очень высокой вероятностью, так и уравнениями с очень низкой вероятностью. Для того чтобы понять, какое из возможных уравнений лучше всего использовать для анализа, используют понятие отклонения:

$$\eta = |1 - 2p| \,.$$

Чем больше величина вероятности отклонения, тем эффективнее линейный криптоанализ с меньшим количеством открытых текстов, необходимых для атаки. Фактически, отклонение показывает насколько вероятность статистического аналога отдалена от значения $p = 1/2$.

Найти аппроксимацию с наибольшим отклонением для одного такта относительно несложно, возникает проблема с вычислением вероятности отклонения полнотактового шифра. Вероятность выполнения общей линейной аппроксимации можно определить с помощью леммы о набегании знаков:

$$P = \frac{1}{2} - 2^{d-1} \prod_{i=1}^{d} \left(q_i - \frac{1}{2} \right),$$

где q_i — вероятности того, что линейная аппроксимация i-ого такта равна 0.

В линейном криптоанализе используется свойство размерности $m = 1$. Слова-блоки интерпретируются как векторы $\mathbb{F}_2^n$, а предикаты задаются ненулевыми векторами α и β:

$$b(X) = \alpha \cdot X,\ b'(Y) = \beta \cdot Y.$$

Вероятность q близка к $1/2$, а p сильно отличается от неё.

1.6 Атака с нулевой корреляцией

Атака с нулевой корреляцией — это метод криптоанализа блочных шифров, разработанный Андреем Богдановым и Винсентом Рэйменом. Она использует линейные приближения с нулевой корреляцией и является расширением линейной атаки.

Пусть (α, β) — линейная аппроксимация, тогда её корреляция определяется следующим образом:

$$C(\beta Y, \alpha X) = 2P\{\alpha X \oplus \beta Y = 0\} - 1.$$

В рассматриваемой атаке используются линейные приближения с нулевой корреляцией, в то время как классическая линейная атака использует линейные аппроксимации с корреляцией, максимально отдаленной от нуля.

Самый простой способ оценить корреляцию по всем 2^n парам входных/-выходных текстов:

$$C(\beta Y, \alpha X) = \frac{|\{(X, Y) | \beta Y \oplus \alpha X = 0\}|}{2^{n-1}} - 1.$$

Для любой нетривиальной линейной аппроксимации (α, β) значение корреляции C можно оценивать более эффективно с 2^{n-1} пар входных/выходных текстов:

$$C(\beta Y, \alpha X) = \frac{|\{(x, y) | \beta Y = 0 \text{ and } \alpha X = 0\}|}{2^{n-2}} - 1 =$$

$$= \frac{|\{(x, y) | \beta Y = 1 \text{ and } \alpha X = 1\}|}{2^{n-2}} - 1.$$

Для случайной подстановки корреляция C нетривиальной линейной аппроксимации может быть описана как стохастическая переменная со следующими распределением:

$$P\{C = \omega \cdot 2^{2-n}\} = \frac{\binom{2^{n-1}}{2^{n-2}+\omega}^2}{\binom{2^n}{2^{n-1}}}.$$

Вероятность того, что значение корреляции C равно 0 для нетривиальной линейной аппроксимации случайной подстановки при $n \geq 5$ может быть аппроксимирована следующим выражением:

$$P\{C = 0\} = \frac{1}{\sqrt{2\pi}} 2^{\frac{4-n}{2}}.$$

Таким образом, чтобы отличить блочный шифр, заменённый линейной аппроксимацией с нулевой корреляцией, от случайной подстановки, криптоаналитик должен собрать 2^{n-1} пар открытых/шифртекстов, полученных на некотором ключе. Для линейной аппроксимации (α, β) с нулевой корреляцией криптоаналитик оценивает корреляцию $C(\beta Y, \alpha X)$. Для случайной подстановки $C(\beta Y, \alpha X)$ будет отклоняться от 0 с вероятностью, приведённой ранее. Вероятность ошибки распознавания незначительна практически для всех размеров блоков ($n \geq 32$).

Аналогично классической линейной атаке для атаки с нулевой корреляцией используется свойство размерности $m = 1$. Однако вместо линейных аппроксимаций с высокой или низкой вероятность, ищутся аппроксимации с вероятностью $p = \frac{1}{2}$.

1.7 RX-атака

RX-атака впервые была предложена Дмитрием Ховратовичем и Ивица Николич в 2010 году. Это общая криптографическая атака на алгоритмы, основанные на трех операциях: сложение по модулю, циклический сдвиг и XOR — ARX для краткости.

Основная идея ARX-криптоанализа заключается в том, что циклическая связь между двумя входными парами текстов сохраняется с некоторой вероятностью за счет операций ARX. Таким образом, связанные циклическим сдвигом входные пары могут использоваться, чтобы видеть корреляции через

такты, состоящие из операций ARX. Эти свойства затем можно использовать для взлома шифра способом, аналогичным разностному криптоанализу.

Рассмотрим пары открытых текстов X и $X^{'} = (X \lll \gamma) \oplus \delta$. Аналогично дифференциалу в разностной атаке, RX-разность для X и $X^{'}$ определяется следующим образом:

$$\Delta_{\gamma}(X, X^{'}) = X^{'} \oplus (X \lll \gamma).$$

Распространение RX-разностей через линейные операции структуры ARX происходят по следующим правилам:

- **XOR**. Для двух RX-пар $(x_1, (x_1 \lll \gamma) \oplus \delta_1)$ и $(x_2, (x_2 \lll \gamma) \oplus \delta_2)$ их XOR это следующая RX-пара $(y, y^{'}) = (x_1 \oplus x_2, ((x_1 \oplus x_2) \lll \gamma) \oplus \delta_1 \oplus \delta_2)$.

- **Циклический сдвиг на η бит**. Циклический сдвиг RX-пары $(x, (x \lll \gamma) \oplus \delta)$ на η бит представляет собой следующую RX-пару $(y, y^{'}) = (x \lll \eta, (x \lll (\gamma + \eta)) \oplus (\delta \lll \eta))$.

Следовательно, при применении циклического сдвига и XOR входные RX-разности распространяются и на выходные RX-разности. Рассмотрим распространение RX-разностей через оставшуюся операцию ARX, сложения по модулю.

Как правило, для ARX-криптоанализа используются пары $(X, X^{'})$, где $X^{'} = X \lll \gamma$. Тогда RX-разность для X и $X^{'}$:

$$\Delta_{\gamma}(X, X^{'}) = X^{'} \oplus (X \lll \gamma).$$

Вероятность распространения входных RX-разностей $(X_1, X_1^{'})$ и $(X_2, X_2^{'})$ определяется следующим образом:

$$P\{(X_1 \lll \gamma) \boxplus (X_2 \lll \gamma) = (X_1 \boxplus X_2) \lll \gamma\} = \frac{1}{4}(1 + \frac{1}{2^{\gamma}} + \frac{1}{2^{n-\gamma}} + \frac{1}{2^n}).$$

Таким образом в ARX-криптоанализе используется свойство размерности $m = 2$. Слова-блоки интерпретируются как векторы $\mathbb{F}_2^n$, предикаты задаются следующим образом:

$$b(X, X^{'}) = I\{\Delta_{\gamma}(X, X^{'}) = 0\}\ ,\ b^{'}(Y, Y^{'}) = I\{\Delta_{\gamma}(Y, Y^{'}) = 0\}.$$

ГЛАВА 2
АВТОМАТИЧЕСКИЕ СРЕДСТВА ОЦЕНКИ НАДЁЖНОСТИ

Традиционно свойства криптосистем искали вручную, но в недавних работах было предложено использовать автоматизированные инструменты, основанные на задачах удовлетворения ограничений, таких как SMT (теория выполнимости по модулю) и MILP (смешанное целочисленное линейное программирование). Автоматизированные методы моделируют поиск как проблему удовлетворения ограничений и решают её с помощью мощных готовых решателей, освобождая криптоаналитиков от усилий по реализации и оптимизации поиска.

К сожалению, большинство автоматизированных методов, опубликованных в литературе, не имеют программной реализации или предоставляют узкие реализации, специфичные для шифра и атаки распознавания, а распространение этих реализаций на другие шифры или атаки требует значительных усилий и опыта.

Заметными исключениями являются основанные на SMT инструменты CryptoSMT и ArxPy. Обе библиотеки поддерживают множество блочных шифров и несколько атак распознавания, а именно разностный и линейный криптоанализ в CryptoSMT и RX-атака, атаки по связанным ключам и невозможным дифференциалам в ArxPy. Однако эти две библиотеки имеют серьезные ограничения. Например, в CryptoSMT отсутствует документация по коду и тесты, а добавление нового шифра в CryptoSMT требует значительных усилий и знаний, поскольку необходимо реализовать разностную и линейную SMT модели шифра. Инструмент ArxPy не страдает от этих ограничений, но он поддерживает только шифры Addition-Rotation-XOR (ARX) и не поддерживает линейный криптоанализ и криптоанализ с нулевой корреляцией.

2.1 Задачи удовлетворения ограничений

Задача удовлетворения ограничений или CSP (Constraint Satisfaction Problem) представляет собой обобщенное понятие системы алгебраических уравнений (или геометрических ограничений) в случае произвольных ограничений над областями значений. Достаточно много задач можно сформули-

ровать как проблемы удовлетворения ограничений, например, кроссворд. В этой задаче требуется только, чтобы слова, пересекающиеся друг с другом, имели одинаковую букву в месте пересечения.

Определим формально задачу удовлетворения ограничений:

$$x_1, x_2, \ldots, x_T$$

— множество переменных,

$$C_1, C_2, \ldots, C_T$$

— множество ограничений. Каждая переменная x_i имеет непустую область определения. Каждое ограничение C_i включает некоторое подмножество переменных и задаёт допустимые комбинации значений для этого подмножества. Состояние задачи определяется путем присваивания значений некоторым или всем этим переменным, $\{x_1 = v_1, x_2 = v_2, \ldots\}$. Присваивание, которое не нарушает никаких ограничений, называется совместимым, или допустимым присваиванием. Полным называется присваивание, в котором каждой переменной присвоено значение. Следовательно, решением задачи CSP является полное присваивание, которое удовлетворяет всем ограничением.

Задачу CSP удобно представлять в виде графа, узлы которого соответствуют переменным задачи, а дуги — ограничениям. Рассматривая некоторую задачу в виде задачи CSP, можно достичь нескольких важных преимуществ. Представление задачи CSP соответствует некоторому стандартному шаблону (т.е. выражается в виде множества переменных с присвоенными значениями), поэтому параметры задачи можно записать в универсальной форме, применимой ко всем задачам CSP. Более того, могут быть разработаны эффективные, универсальные эвристические функции, для создания которых не требуются дополнительные знания о конкретной проблемной области. Также для упрощения процесса решения может использоваться структура графа, что позволяет в некоторых случаях добиться экспоненциального уменьшения сложности. Это представление задачи CSP является простейшим из ряда схем представления.

Задачи CSP простейшего вида характеризуются тем, что в них используются переменные, которые являются дискретными и имеют конечные области определения. К такому виду относится задача раскраски карты. К категории задач CSP с конечной областью определения относятся булевы задачи CSP, в которых переменные могут иметь значения либо true, либо false. Булевы задачи CSP включают в качестве частных случаев некоторые NP-полные задачи, такие как 3SAT. Поэтому в худшем случае нельзя рассчитывать на то,

что мы сможем решать задачи CSP с конечной областью определения за время меньше экспоненциального. Но в большинстве практических приложений алгоритмы CSP общего назначения позволяют решать задачи, на несколько порядков величины более крупные по сравнению с теми, которые могут быть решены с помощью алгоритмов поиска общего назначения.

Дискретные переменные могут также иметь бесконечные области определения, например, такие, как множество всех целых чисел или множество всех строк. При решении задач с бесконечными областями определения больше нет возможности описывать ограничения, перечисляя все допустимые комбинации значений. Вместо этого должен использоваться язык ограничений. Например, в виде алгебраических неравенств. Кроме того, больше нет возможности решать такие ограничения, просто перечисляя все возможные присваивания, поскольку количество подобных возможных присваиваний бесконечно велико. Можно доказать, что не существует алгоритмов решения общих нелинейных ограничений с целочисленными переменными. В некоторых случаях задачи с целочисленными ограничениями можно свести к задачам с конечной областью определения, устанавливая пределы значений всех этих переменных.

В реальном мире очень часто встречаются задачи удовлетворения ограничений с непрерывными областями определения, и эти задачи интенсивно изучаются в области исследования операций. Одной из широко известных категорий задач CSP с непрерывной областью определения являются задачи линейного программирования, в которых ограничения должны представлять собой линейные неравенства, образующие выпуклую область. Задачи линейного программирования могут быть решены за время, которое зависит полиномиально от количества переменных. Кроме того, проводились исследования задач с другими типами ограничений и целевых функций, например, задачи квадратичного программирования и т.д.

Кроме исследования типов переменных, которые могут присутствовать в задачах CSP, полезно заняться изучением типов ограничений. Простейшим типом ограничения является унарное ограничение, которое ограничивает значение единственной переменной. Каждое унарное ограничение можно устранить, выполняя предварительную обработку области определения соответствующей переменной, чтобы удалить любое значение, нарушающее это ограничение. Бинарное ограничение связывает между собой две переменные. Бинарной задачей CSP называется задача, в которой предусмотрены только бинарные ограничения; она может быть представлена в виде графа.

В ограничениях высокого порядка участвуют три или больше переменных. Одним из известных примеров таких задач являются криптоарифмети-

ческие головоломки, называемые также числовыми ребусами. Ограничения высокого порядка могут быть представлены в виде гиперграфа. Каждое ограничение высокого порядка с конечной областью определения можно свести к множеству бинарных ограничений, введя достаточное количество вспомогательных переменных.

Все ограничения, рассмотренные нами выше, были абсолютными ограничениями, нарушение которых равносильно тому, что какое-то потенциальное решение больше не рассматривается как таковое. С другой стороны, во многих реальных задачах CSP применяются ограничения предпочтения, которые указывают, какие решения являются предпочтительными. Ограничения предпочтения часто можно закодировать как стоимости присваиваний отдельных переменных. При использовании такой формулировки задачи CSP с предпочтениями можно решать, используя методы поиска с оптимизацией, либо основанные на поиске пути, либо локальные.

Задача выполнимости булевой формулы (SAT), теории выполнимости по модулю (SMT), смешанное целочисленное программирование (MIP) и программирование набора ответов (ASP) — это все области исследований, сосредоточенные на решении конкретных форм проблем удовлетворения ограничений.

2.2 SAT и SMT

Выполнимость — одна из фундаментальных проблем теоретической информатики, а именно проблема определения того, имеет ли решение формула, выражающая ограничение. Проблемы удовлетворения ограничений возникают во многих различных областях, включая проверку программного и аппаратного обеспечения, расширенную статическую проверку, генерацию тестовых данных, планирование, задачи с графами и другие. Самая известная проблема удовлетворения ограничений — это пропозиционная выполнимость SAT, где цель состоит в том, чтобы решить, может ли формула над булевыми переменными, образованная с использованием логических связок, стать истинной путем выбора значений $true/false$ для ее переменных.

В последнее время особый интерес представляет теория выполнимости по модулю (SMT), где интерпретация некоторых символов ограничена фоновой теорией. Например, теория арифметики ограничивается интерпретацией таких символов, как: $+$, $\leq$, 0 и 1.

SMT опирается на самые многочисленные проблемы символической логики прошлого века: проблема решения, полнота и неполнота логических тео-

рий и, наконец, теория сложности. Вычислительная сложность большинства задач SMT очень высока. Теория линейной арифметики, являющаяся основой линейного программирования, является одной из известных теорий, которая полезна во многих приложениях. Алгоритмы линейного программирования можно использовать для проверки выполнимости конъюнкций линейных арифметических неравенств, но они не применяются непосредственно для булевых комбинаций. Решатели SMT отличаются тем, что обрабатывают такие комбинации.

Из-за высокой вычислительной сложности невозможно построить процедуру, которая может решить произвольные задачи SMT. Поэтому большинство процедур сосредоточено на более реалистичной цели эффективного решения проблем, возникающих на практике. Они полагаются на предположение о том, что, несмотря на то, что формулы могут быть громоздкими, большинство из них поверхностны. То есть лишь небольшая часть формулы действительно важна для установления выполнимости.

В последние годы произошел огромный прогресс в масштабе проблем, которые можно решить, благодаря инновациям в алгоритмах, структурах данных, эвристике и вниманию к деталям реализации. Современные процедуры SAT могут проверять формулы с сотнями тысяч переменных и миллионами предложений. Аналогичный прогресс наблюдается в процедурах SMT для наиболее часто встречающихся теорий.

Наиболее успешные решатели SAT основаны на подходе, называемом систематическим поиском. Пространство поиска представляет собой дерево, каждая вершина которого представляет пропозиционную переменную, а рёбра представляют два варианта выбора (т. е. $true$ и $false$) для этой переменной. Для формулы, содержащей n переменных, в этом дереве 2^n листьев. Каждый путь от корня к листу соответствует назначению истины. Для заданной формулы ϕ процедура, основанная на систематическом поиске, ищет в дереве истинное назначение M, удовлетворяющее ϕ. Большинство решателей SAT на основе поиска основаны на подходе DPLL. Учитывая формулу КНФ, алгоритм DPLL пытается построить удовлетворительное значение истинности, используя три основные операции: решение, распространение и возврат. Операция «решить» эвристически выбирает неназначенную пропозиционную переменную и присваивает ей значение $true$ или $false$. Эта операция также называется ветвлением или разделением регистра. Операция распространения выводит последствия присваивания частичной истины, используя правила вывода. Наиболее широко используемым правилом вывода является правило единичного предложения, которое гласит, что если в предложении всем литералам, кроме одного, присвоено значение false, а оставшемуся литералу l

не назначено, то единственный способ, которым это предложение может быть оценено как true, — это присвоить l значение true. Пусть C будет предложением:

$$p \vee \neg q \vee \neg r,$$

а M — присвоением частичной истины:

$$\{p \mapsto false, r \mapsto true\},$$

тогда единственный способ для C получить истинное значение — это присвоить q ложному.

Доступность такого мощного инструмента побудила многих исследователей предложить использование решателей SAT и SMT для криптоанализа. Например, решатели SAT используются в линейном и разностном криптоанализе блочных шифров. Решатели SAT и SMT становятся все более важным инструментом в наборе инструментов практического криптоанализа.

Известные инструменты на основе SAT, которые были разработаны специально для криптоанализа (по крайней мере, на начальном этапе), включают CryptLogVer и Transalg, которые представляют собой инструменты для кодирования криптографических функций в SAT, CryptoMiniSat, который включает рассуждение для операции XOR, а также CryptoSAT и CryptoSMT, которые предоставляют языки более высокого уровня. Для решения системы алгебраических уравнений обычно используются решатели SAT и SMT. Но другие типы решателей также оказались полезными. H. Муха и др. используют решатели смешанного целочисленного линейного программирования (MILP), чтобы найти границы безопасности в блочных шифрах.

2.3 MILP

Линейное программирование — это исследование оптимизации (минимизации или максимизации) линейной целевой функции $f(x_1, x_2, \ldots, x_n)$ с учетом линейных неравенств, включающие переменные решения x_i, $1 \leq i \leq n$. Для многих таких задач оптимизации необходимо ограничить некоторые переменные решения целыми значениями, т.е. для некоторых значений i, мы требуем, чтобы $x_i \in \mathbb{Z}$. Методы формулировки и решения таких программ называются смешанным целочисленным линейным программированием (MILP). Если все переменные решения x_i должны быть целым числом, используется термин (чистое) целочисленное линейное программирование (ILP).

MILP методы нашли множество практических применений в области экономики и бизнеса, но их применение в криптографии до сих пор ограничено. В контексте криптографических атак он позволяет доказать границы безопасности как для разностного, так и для линейного криптоанализа. Метод значительно снижает нагрузку на криптоаналитиков, потому что он включает в себя только написание простых уравнений, которые вводятся в решатель MILP. Поскольку требуется очень мало программирования, время, затрачиваемое на криптоанализ, и возможность человеческих ошибок значительно сокращаются.

Устойчивость к линейному и разностному криптоанализу является стандартным критерием разработки новых шифров. Доказав нижнюю границу количества активных S-блоков как для разностного, так и для линейного криптоанализа, можно получить верхнюю границу вероятности наилучшей характеристики.

2.4 Применение MILP для разностной атаки

Рассмотрим усеченные разности, то есть каждый байт в анализе может иметь либо нулевую, либо ненулевую разность. Более формально мы определяем следующий разностный вектор:

Пусть Δ — строка, состоящая из n байтов $\Delta = (\Delta_0, \Delta_1, \ldots, \Delta_{n-1})$. Тогда разностный вектор $x = (x_0, x_1, \ldots, x_{n-1})$, соответствующий Δ, определяется как

$$x_i = \begin{cases} 0 & \text{если} \quad \Delta_i = 0, \\ 1 & \text{иначе.} \end{cases}$$

Для любой $\phi \in \mathbb{F}_2^n$ дифференциальный номер ветви обозначается через $\mathcal{B}_\mathcal{D}$ и определяется следующим образом:

$$\mathcal{B}_\mathcal{D}(\phi) =: \min_{x \neq x'} \{w(x + x') + w(\phi(x) + \phi(x'))\},$$

где $w(x)$ — вес Хэмминга, количество ненулевых компонент в x:

$$w(x) = \#\{x_j | x_j \neq 0, 1 \leq j \leq n\}.$$

Пусть $(x_0, \ldots, x_{n-1})$ и $(y_0, \ldots, y_{n-1})$ — входной и выходной векторы разности линейного преобразования L. Учитывая дифференциальный номер ветви $\mathcal{B}_\mathcal{D}(L)$, линейное преобразование описывается следующим образом:

$$\begin{cases} \sum_{k=0}^{n-1} x_k + \sum_{k=0}^{n-1} y_k \leq \mathcal{B}_{\mathcal{D}} d, \\ d \leq x_k, \\ d \leq y_k, \end{cases}$$

где d — фиктивная переменная, принимающая значения из $\{0, 1\}$.

Целевая функция, которую необходимо минимизировать, — это количество активных S-блоков. Эта функция равна сумме всех переменных, соответствующих входам S-блока.

Если все d-переменные и все x-переменные ограничены двоичными данными, результирующая программа представляет собой чистую задачу ILP (целочисленное линейное программирование). Если все d-переменные ограничены бинарностью, а x-переменные только те, что соответствуют входным данным (открытый текст), уравнения гарантируют, что оптимальное решение для всех других x-переменных также будет двоичным и приводит к задаче MILP (смешанного целочисленного линейного программирования), которую можно решить быстрее.

С помощью приведенной выше модели строится модель MILP для вычисления нижней границы количества активных S-блоков для блочного шифра.

2.5 Применение MILP для линейной атаки

На двойственность между разностным и линейным криптоанализом указывал еще Мацуи. Уравнения, описывающие линейную функцию, такие же как и в случае разностного криптоанализа, однако дифференциальный номер ветвления $\mathcal{B}_{\mathcal{D}}$ заменяется на линейный номер ветвления $\mathcal{B}_{\mathcal{L}}$.

Номер линейной ветви — это минимальное количество ненулевых линейных аппроксимаций для входа и выхода функции, исключая случай всех нулей. Более формально для любой $\phi \in \mathbb{F}_2^n$ линейный номер ветви обозначается через $\mathcal{B}_{\mathcal{L}}$ и определяется следующим образом:

$$\mathcal{B}_{\mathcal{L}}(\phi) =: \min_{C(\beta\phi(x),\alpha x)\neq 0} \{w(\alpha) + w(\beta)\}.$$

ГЛАВА 3
СРЕДСТВО CASCADA

CASCADA (Characteristic Auto mated Search of Cryptographic Algorithms for Distinguishing Attacks) — библиотека Python с открытым исходным кодом для оценки безопасности блочных шифров против множества атак распознавания путем поиска уязвимых свойств с использованием бит-векторных решателей SMT.

Библиотека CASCADA основана на ArxPy, но только треть исходного кода CASCADA происходит от ArxPy. CASCADA реализует автоматизированные методы атак распознавания, поддерживает более широкий класс блочных шифров, а также улучшает код, документацию и тесты.

Инструмент CASCADA реализует поиск дифференциалов, пар RX-разностей и линейных аппроксимаций, которые будут использоваться в разностной, линейной и RX-атаках, атаках на связанных ключах, по невозможным дифференциалам и с нулевой корреляцией.

3.1 SMT на основе бит-векторной теории

Бит-векторное выражение — это константа, переменная или операция с бит-векторами в качестве входных данных. Константы битового вектора интерпретируются как целые числа без знака по основанию 2; n-битный вектор $x = b_{n-1} \dots b_1 b_0$ обозначает целое неотрицательное число:

$$b_0 + 2b_1 + \ldots + 2^{n-1} b_{n-1}.$$

i-ый бит x, b_i, также обозначается как $x[i]$, $x[0] = b_0$ называется младшим значащим битом (LSB), а $x[n-1] = b_{n-1}$ обозначает старший значащий бит (MSB). Рассмотрим следующие бит-векторные операции и их обозначения:

- Конкатенация и извлечение битовых векторов.

- Побитовые логические операции: отрицание $\neg$, конъюнкция $\wedge$, дизъюнкция $\vee$ и исключающее или (XOR) $\oplus$.

- Операции сдвига: сдвиг влево $\ll$, сдвиг вправо $\gg$, циклический сдвиг влево $\lll$ и циклический сдвиг право $\ggg$.

- Арифметические операции: модульное сложение $\boxplus$, модульное вычитание $\boxminus$, модульное умножение $\boxtimes$, операция усеченного деления без знака $\div$ и операция остаток по модулю $\%$.

- Операции сравнения: $=$, $<$, $>$, $\leq$ и $\geq$.

- Оператор if-then-else $Ite(b, x, y)$, возвращающий x, если b равен биту 0 и в противном случае возвращает y.

Задача SMT, определенная в теории битовых векторов, или просто задача SMT битового вектора, задается списком переменных битового вектора, каждая из которых связана с квантором существования $\exists$ или всеобщности $\forall$, и списком ограничений битового вектора, включая эти переменные. Задача SMT, в которой кванторы не указаны, называется задачей без кванторов, и она равно разрешима той же задаче SMT с кванторами существования. Задачи SMT, сочетающие кванторы всеобщности и существования, называются количественными задачами, и их гораздо сложнее решить.

3.2 Свойства и характеристики битового вектора

Свойство бит-вектора над функцией f — это пара бит-векторов (α, β) с соответствующей вероятностью распространения $PP_f(\alpha, \beta) \in [0, 1] \subseteq \mathbb{R}$.

Для функции f_k, зависящей от внешних значений $k = (k_1, k_2, \ldots)$, не входных, но неизвестных фиксированных значений, таких как раундовые ключи, свойство битового вектора над f_k может включать дополнительное значение битового вектора κ, так что вероятность распространения зависит не только от входных и выходных свойств (α, β), но и от внешнего свойства κ.

Разностное свойство (α, β) над функцией f определяется как свойство битового вектора (α, β) над f, где вероятность распространения определяется уравнением в общем виде:

$$\#\{x : f(x \bigtriangleup \alpha) \bigtriangledown f(x) = \beta\}/2^n,$$

и усредняется по $\mathcal{K}$, если f содержит внешние значения $k \in \mathcal{K}$:

$$p = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \#\{x : f(x \bigtriangleup \alpha) \bigtriangledown f(x) = \beta\}/2^n,$$

где как правило $\bigtriangleup = \oplus = \bigtriangledown$.

Свойство линейной аппроксимации (α, β) над f — это свойство битового вектора (α, β) над f, где вероятность распространения определяется абсолютным значением корреляции $C_f(\alpha, \beta)$, определяемой уравнением:

$$C_f(\alpha, \beta) = 2 \times (\#\{x : \langle x, \alpha \rangle = \langle \beta, f(x) \rangle\}/2^n) - 1$$

и усредняется по $\mathcal{K}$, если f содержит внешние значения $k \in \mathcal{K}$:

$$p = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} C_f(\alpha, \beta)^2.$$

Пары RX-разностей (α, β) с $\{\triangle, \bigtriangledown\}$, определяемыми как

$$x' \bigtriangledown x = x' \oplus (x \lll 1), \quad x \triangle \alpha = (x \ggg 1) \oplus \alpha,$$

эквивалентны дифференциалу (α, β). Поэтому будем называть RX-разностные пары RX-дифференциалами, а дифференциалы с $\bigtriangledown = \oplus = \triangle$ будем называть XOR-дифференциалами. Вероятность их распределения:

$$\sum \#\{x : (f(x) \lll \gamma) \oplus f((x \lll \gamma) \oplus \Delta_\gamma) = \Delta'_\gamma\}/2^n,$$

В то время как для атак распознавания требуется только глобальное свойство (α, β) и его вероятность распространения, вычисление вероятности распространения является сложной задачей для сложных функций, таких как блочные шифры. Основной подход к атакам распознавания заключается в анализе локальных вероятностей распространения функций раунда, получении следа локальных свойств и оценке глобальной вероятности распространения как произведения локальных вероятностей распространения.

Бит-векторная характеристика Γ над $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$ — это след свойств $(\gamma_0, \gamma_1, \ldots, \gamma_r)$, где (γ_i, γ_{i+1}) — свойство битового вектора над f_i. Бит-векторы (γ_0, γ_r) также называются входным и выходным свойствами соответственно. Вероятность распространения Γ определяется как:

$$PP_f(\Gamma) = PP_{f_0}(\gamma_0, \gamma_1) \times PP_{f_1}(\gamma_1, \gamma_2) \times \ldots \times PP_{f_{r-1}}(\gamma_{r-1}, \gamma_r).$$

Дифференциальная характеристика связанного ключа над блочным шифром — это пара дифференциальных характеристик $(\Gamma_{KS}, \Gamma_{E_k})$, где Γ_{KS} определена над функцией расписания ключа KS, а Γ_{E_k} над функцией шифрования E_k, такая что свойства раундового ключа Γ_{E_k} задаются свойствами Γ_{KS}.

В зависимости от функции и свойств, вероятность распространения характеристики $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_r)$ может неточно аппроксимировать вероят-

ность распространения глобального свойства (γ_0, γ_r). Тем не менее, это приближение широко используется при разработке и криптоанализе блочных шифров (особенно шифров ARX) из-за отсутствия других систематических подходов.

На практике для утверждения, что блочные шифры защищены от атак распознавания, необходимо показать, что никакие высоко вероятные характеристики и глобальные свойства с нулевыми вероятностями не могут быть найдены. Таким образом, систематические методы поиска этих объектов имеют решающее значение для разработки и анализа блочных шифров.

Чтобы искать характеристики с высокой вероятностью и глобальные свойства с нулевой вероятностью, используя бит-векторные задачи SMT, вероятности распространения характеристик и свойств должны быть интерпретированы как ограничения битового вектора.

3.3 Модель бит-векторных свойств

Свойство (α, β) над функцией f верно, если вероятность его распространения отлична от нуля. В этом случае можно определить вес распространения (α, β) как отрицательный двоичный логарифм вероятности его распространения, то есть:

$$PW_f(\alpha, \beta) = -\log_2(PP_f(\alpha, \beta)).$$

Модель свойств битового вектора f представляет собой набор ограничений битового вектора, который моделирует вес распространения свойств над f. Модель свойств f задается тремя ограничениями битового вектора: ограничением допустимости, ограничением вероятности-1 и ограничением веса.

- Ограничение допустимости с входными данными (α, β) истинно тогда и только тогда, когда свойство (α, β) допустимо.

- Ограничение вероятности-1 с входными данными (α, β) истинно тогда и только тогда, когда вероятность распространения свойства (α, β) равна 1.

- Весовое ограничение с входными параметрами (ω, α, β) истинно тогда и только тогда, когда битовый вектор ω равен весу распространения свойства (α, β).

Весовое ограничение определено только для входов (α, β) с ненулевой вероятностью распространения; истинностное значение весового ограничения для недопустимых (α, β) не имеет значения.

В задачах SMT с битовым вектором умножение $\boxtimes$ обходится дороже, чем сложение $\boxplus$. Чтобы избежать моделирования вероятности распространения $g \circ f$ как произведения локальных вероятностей f и g, модель свойств включает ограничение веса, а не ограничение вероятности (ограничение с входными данными (p, α, β) истинно, если p равна вероятности распространения (α, β)). Таким образом, вес распространения $g \circ f$ можно эффективно смоделировать как сумму локальных весов распространения f и g.

По умолчанию n_ω-битные входные данные ω весового ограничения интерпретируются как неотрицательное целое число:

$$\omega[0] + 2\omega[1] + \ldots + 2^{n_\omega - 1}\omega[n_\omega - 1].$$

Однако, поскольку вес распространения может быть нецелым значением для некоторых свойств и функций, мы рассматриваем ограничения веса, когда входные данные ω интерпретируются как рациональное значение:

$$2^{-l}(\omega[0] + 2\omega[1] + \ldots + 2^{n_\omega - 1}\omega[n_\omega - 1]),$$

для заданного фиксированного числа l дробных битов. Более того, мы также рассматриваем весовые ограничения, которые верны тогда и только тогда, когда:

$$|\omega - PW_f(\alpha, \beta)| < \epsilon.$$

для фиксированной границы ошибки ϵ.

Слабая модель упрощает вероятность распространения путем рассмотрения только четырех возможных вероятностей распространения в зависимости от того, равны ли α или β нулю или отличны от нуля. Модель на основе ветвей похожа на слабую модель, но с дополнительным правилом, согласно которому ненулевое свойство (α, β) считается недействительным, если количество ненулевых слов в α и β строго меньше заданного фиксированного числа B. Эти упрощенные модели первоначально использовались для разности XOR и свойств линейной аппроксимации, где слабые модели использовались для S-блоков, а модели на основе ветвлений использовались для линейных слоев.

3.4 Модель бит-векторных характеристик

Назовем характеристику $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_r)$ функции $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$ допустимой, если вероятность распространения Γ отлична от нуля, при этом случае мы определяем вес распространения Γ как:

$$PW_f(\Gamma) = -log_2(PP_f(\Gamma)) = PW_{f_0}(\gamma_0, \gamma_1) + \ldots + PW_{f_{r-1}}(\gamma_{r-1}, \gamma_r).$$

Модель бит-векторной характеристики $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$ представляет собой набор ограничений бит-вектора, который моделирует вес распространения характеристик по $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$. Характеристическая модель задается тремя ограничениями битового вектора: ограничение допустимости, ограничение вероятности-1 и ограничение веса.

- Ограничение допустимости с входными данными $(\gamma_0, \gamma_1, \ldots, \gamma_r)$ истинно тогда и только тогда, когда характеристика $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_r)$ допустима.

- Ограничение вероятности-1 с входными данными $(\gamma_0, \gamma_1, \ldots, \gamma_r)$ истинно тогда и только тогда, когда вероятность распространения характеристики $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_r)$ равна 1.

- Весовое ограничение с входными параметрами $(\omega, \gamma_0, \gamma_1, \ldots, \gamma_r)$ истинно тогда и только тогда, когда битовый вектор ω равен весу распространения действительной характеристики $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_r)$.

Даны модели свойств функций $f_0, f_1, \ldots$ и f_{r-1}, ограничения характеристической модели получаются следующим образом. Пусть VC_i, POC_i и WC_i обозначают ограничения допустимости, вероятности-1 и весовое ограничение, соответственно, модели свойств f_i. Тогда ограничения VC, POC и WC характеристической модели задаются выражениями:

$$\begin{gathered}
\mathrm{VC}(\gamma_0, \ldots, \gamma_r) = \mathrm{VC}_0(\gamma_0, \gamma_1) \wedge \ldots \wedge \mathrm{VC}_{r-1}(\gamma_{r-1}, \gamma_r). \\
\mathrm{POC}(\gamma_0, \ldots, \gamma_r) = \mathrm{POC}_0(\gamma_0, \gamma_1) \wedge \ldots \wedge \mathrm{POC}_{r-1}(\gamma_{r-1}, \gamma_r). \\
\mathrm{WC}(\omega, \gamma_0, \ldots, \gamma_r) = \exists \omega_0, \ldots, \omega_{r-1} : (\omega = \omega_0 \boxplus \ldots \boxplus \omega_{r-1}) \wedge \\
\mathrm{WC}_0(\omega_0, \gamma_0, \gamma_1) \wedge \ldots \wedge \mathrm{WC}_{r-1}(\omega_r, \gamma_{r-1}, \gamma_r)
\end{gathered}$$

3.5 Поиск маловесных характеристик

Пусть (VC, POC, WC) — ограничения характеристической модели функции $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$. Найти характеристику с целым весом ω можно, решив задачу бит-векторного SMT с помощью SMT решателя:

$$\begin{gathered}\exists \gamma_0, \gamma_1, \ldots, \gamma_r, \omega, \omega' : \\ \mathrm{VC}(\gamma_0, \gamma_1, \ldots, \gamma_r) \\ \mathrm{WC}(\omega', \gamma_0, \gamma_1, \ldots, \gamma_r) \\ \omega = Truncate(\omega', l),\end{gathered}$$

где $Truncate(\omega', l)$ игнорирует l младших значащих битов, извлекая $n'_\omega - l$ значащих битов.

Для поиска характеристики с наименьшим целочисленным весом предыдущая подпрограмма просто повторяется, начиная с целочисленного веса $\omega = 0$ и увеличивая целочисленный вес, если текущая задача SMT неразрешима. Если граница погрешности ϵ характеристической модели равна нулю, то первая разрешимая задача приводит к оптимальной характеристике в том смысле, что нет характеристик с целым весом строго меньшим, и поиск завершается.

В противном случае пусть $\hat{\omega}$ — целочисленный вес первой полученной характеристики. Поиск завершается после получения всех характеристик с целыми весами в интервале $[\hat{\omega}, \hat{\omega} + \epsilon]$ и возвращается та, у которой наименьший вес (оптимальная характеристика).

На практике поиск можно ускорить, если сначала найти оптимальную характеристику Γ_0 над простой функцией f_0, а затем с помощью целочисленного веса Γ_0 как начальный вес поиска по $f_1 \circ f_0$; этот процесс итеративно повторяется до тех пор, пока $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$. Этот процесс использует тот факт, что если все задачи SMT для $f_i \circ f_{i-1} \circ \ldots \circ f_0$ и для целых весов $\{0, 1, \ldots, \omega\}$ оказались неудовлетворительными, то все задачи SMT для $f_{i+1} \circ f_i \circ f_{i-1} \circ \ldots \circ f_0$ и для целочисленного веса $\{0, 1, \ldots, \omega\}$ также невыполнимы, так как характеристический вес определяется как сумма неотрицательных локальных весов распространения.

Этот автоматизированный метод можно использовать для поиска дифференциальных или линейных характеристики блочного шифра, просто установив в качестве f функции шифрования E_k. Дифференциальные характеристики со связанными ключами также можно искать, просто расширяя подпрограмму выше для пары характеристических моделей $(\Gamma_{KS}, \Gamma_{E_k})$, и огра-

ничение суммы веса распространения Γ_{KS} и веса распространения Γ_{E_k} до целочисленного веса ω.

3.6 Поиск недействительных свойств

Искать глобальные свойства с нулевой вероятностью, можно с помощью трёх методов SMT на основе битовых векторов: метод грубой силы обобщает автоматический поиск невозможных дифференциалов шифров с небольшими S-блоками, метод промаха посередине обобщает поиск невозможных дифференциалов со связанными ключами шифров ARX, и метод количественного определения представляет собой новый автоматизированный метод, основанный на количественном определении бит-векторных задач SMT.

Метод грубой силы. Пусть VC будет ограничением достоверности характеристической модели функции $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$ и пусть (α, β) будет свойством f для некоторых констант битового вектора α и β. Методы грубой силы и промаха посередине основаны на том факте, что если задача битового вектора SMT:

$$\exists \gamma_0, \gamma_1, \ldots, \gamma_r : \mathrm{VC}(\gamma_0, \gamma_1, \ldots, \gamma_r) \wedge (\alpha = \gamma_0) \wedge (\beta = \gamma_r)$$

невыполнима, то (α, β) имеет нулевую вероятность распространения. Отличие этих двух методов заключается в выборе свойств (α, β). Метод грубой силы просто выбирает подмножество свойств с большим количеством нулевых битов и проверяет, является ли проблема SMT, заданная уравнением выше невыполнимой для каждого свойства.

Этот выбор обусловлен тем фактом, что для некоторых функций большинство невозможных дифференциалов, найденных до сих пор, имеют много нулевых битов.

Метод промах посередине. Идея метода промаха посередине состоит в том, чтобы найти невозможный дифференциал, построенный из двух характеристик с вероятностью-один Γ_0 и Γ_2, где характеристика Γ_0 (соответственно Γ_2) покрывает первую (соответственно вторую) половину шифра, такую, что выходная разность α алгоритма Γ_0 не совпадает с входной разностью β Γ_2 в середине шифра.

Как и метод грубой силы, метод промаха посередине основан на невыполнимость задач SMT, но где свойства (α, β) выбираются как выходы и входы характеристик с вероятностью-один покрывая первую и последнюю часть шифра соответственно.

Количественный метод. В отличие от метода грубой силы и промаха посередине, количественный метод основан на решении выполнимой бит-векторной SMT проблемы, которая сочетает в себе кванторы существования и всеобщности. Учитывая ограничение достоверности VC характеристической модели $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$ рассмотрим количественную задачу SMT с бит-вектором:

$$\exists \gamma_0, \gamma_r, \forall \gamma_1, \gamma_2, \ldots, \gamma_{r-1} : \mathrm{VC}(\gamma_0, \gamma_1, \ldots, \gamma_r) = \mathrm{False}.$$

В случае выполнимости решением этой задач является такое задание переменных (γ_0, γ_r), что характеристика $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_r)$ неверна для всех промежуточных свойств $(\gamma_1, \gamma_2, \ldots, \gamma_{r-1})$. Иными словами, решение этой проблемы есть свойство (γ_0, γ_r) функции f с нулевой вероятностью распространения.

Таким образом, свойства нулевой вероятности f могут быть получены путем решения задачи решателем SMT, поддерживающим количественные формулы бит-вектора, такие, как Boolector или Z3.

Бесквантворные задачи, решаемые методами грубой силы и промаха посередине, могут быть решены намного быстрее, чем количественные задачи, однако любое свойство с нулевой вероятностью, найденное с помощью грубой силы или промаха посередине может быть найдено количественным методом. Обратное в общем случае неверно: последний метод может найти свойства с нулевой вероятностью, недостижимые с помощью грубой силы и промаха посередине.

Все три метода являются надежными, но не полными. Любое свойство, обнаруженное этими методами, имеет нулевую вероятность распространения, но некоторые свойства с нулевой вероятностью могут быть не обнаружены этими методами. Другими словами, если проблемы SMT, заданные выше невыполнимы, то (α, β) имеет нулевую вероятность распространения, а наоборот, вообще говоря, не выполняется.

3.7 Инструмент CASCADA

Инструмент CASCADA основан на ArxPy, инструменте для поиска дифференциальных характеристик и невозможных дифференциалов шифров ARX. Однако в то время как ArxPy ограничивается RX-атакой, атаками на связанным ключам и по невозможным дифференциалам, CASCADA реализует структуру свойств битового вектора, новые автоматизированные методы, множество новых функций и улучшений.

По сравнению с ArxPy интерфейс в CASCADA был улучшен для поддержки не только шифров ARX, но также и других примитивов и шифров, а документация была расширена, так что большинство функций и классе Python содержат документацию с примерами.

Библиотека CASCADA имеет модульную конструкцию со слабой связью, разделенную на несколько модулей, а именно модули битового вектора, примитивов, свойств и SMT, так что каждый из них можно использовать, дополнять и расширять независимо. Рассмотрим каждый модуль поподробнее.

Бит-векторный модуль обрабатывает создание, оценку, символьную обработку и представление выражений и функций битового вектора. С этой целью он предоставляет типы данных для создания бит-векторных констант, переменных, операций, выражений и функций. Он использует SymPy (библиотека Python с открытым исходным кодом для символьных вычислений) для символьных манипуляций с бит-векторами, и он обеспечивает несколько представлений типов данных битового вектора, включая представление исполняемой строки, представление кода C или представление DOT (язык описания графов).

Модуль битового вектора также предоставляет несколько функциональностей для модификации, создания, оценки и обработки выражений битового вектора. Например, функция упрощения определяет, следует ли упрощать выражения, применяя правила булевой алгебры, а функция запоминания представляет собой компромисс между пространством и временем, с помощью составления таблиц, где промежуточные результаты сохраняются в таблице, чтобы их можно было извлечь, когда на вход подаются те же самые значения.

Модуль битового вектора не зависит от других модулей CASCADA, и поэтому его можно использовать независимо в приложениях, требующих символьной обработки выражений или функций битового вектора.

Далее рассмотрим модуль примитивов. Он предоставляет типы данных для представления функций шифрования и блочных шифров. В модуле примитивов реализовано множество криптографических примитивов, а именно AES, CHAM, Chaskey, FEAL, HIGHT, Simeck, Simon, TEA, XTEA и другие. Список примитивов продолжает расти и пополняться новыми примитивами.

Модуль свойств CASCADA состоит из модуля абстрактных свойств, предоставляющего интерфейс для реализации свойств битового вектора, а также дифференциального и линейного модулей, которые конкретизируют модуль абстрактных свойств для разностных и линейных свойств аппроксимации соответственно.

Модуль абстрактных свойств предоставляет типы данных для представ-

ления свойств битового вектора, моделей свойств, характеристик и моделей характеристик.

Модуль абстрактных свойств также реализует генерацию характеристических моделей, декомпозицию характеристик и характеристических моделей функций на основе округления, представление характеристик и характеристических моделей в формате DOT и вычисление эмпирических весов.

Дифференциальный модуль создает экземпляр модуля абстрактных свойств для свойств разности XOR и разности RX. Он реализует тривиальные модели XOR и RX многих $\oplus$-линейных и детерминированных по распространению операций, то есть операций, которые распространяют входное свойства на уникальное выходное свойство с вероятностью 1.

Линейный модуль создает экземпляр модуля абстрактного свойства для свойства линейной аппроксимации. Помимо тривиальных моделей $\oplus$-линейных и детерминированных по распространению операций, он реализует нетривиальные модели $\boxplus$, $\boxminus$(из модели $\boxplus$, тождества $\neg(x \boxminus y) = \neg x \boxplus y$), $\wedge$ и $\vee$.

Модуль SMT реализует автоматизированные методы поиска характеристик, описанные ранее. Для решения базовых задач SMT модуль использует PySMT, API Python с открытым исходным кодом для решателей SMT. В результате модуль SMT поддерживает любой из решателей SMT для битовых векторов, изначально поддерживаемых PySMT (например, Boolector, CVC4, MathSAT, Z3 и Yices), и он также может использовать другие решатели через интерфейс PySMT.

Помимо выбора решателя SMT, можно настроить многие параметры автоматизированных методов, реализованных в модуле SMT, в том числе тип ограничений (например, ограничения допустимости и веса или ограничения только для вероятности-1), дополнительные ограничения, уровень детализации или необходимость фильтрации характеристик с использованием эмпирического веса.

ГЛАВА 4
ЭКСПЕРИМЕНТЫ С BelT

Библиотека с открытым исходным кодом CASCADA — инструмент для оценки широкого класса криптографических примитивов против множества атак распознавания. Модуль примитивов CASCADA состоит из большого числа криптографических примитивов и шифров, к тому же автор и другие исследователи продолжают его дополнять.

Основная цель практической части дынной работы — дополнить инструмент CASCADA государственным стандартом симметричного шифрования Республики Беларусь BelT, а также исследовать его на наличие характеристик, представляющих возможность реализации атак, описанных ранее.

4.1 Belt: вспомогательные преобразования и переменные

Подстановка $H : \{0,1\}^8 \to \{0,1\}^8$ задаётся таблицей 4.1. В таблице входы (прообразы) и выходы (образы) H записываются в шестнадцатеричном виде. Для входного октета $u = IJ_{16}$ соответствующий выходной октет $H(u)$ находится на пересечении строки I и столбца J. Например, $H(A2_{16}) = 9B_{16}$.

Преобразование G_r $(r = 5, 13, 21)$: $\{0,1\}^{32} \to \{0,1\}^{32}$ ставит в соответствие слову $u = u_1 \parallel u_2 \parallel u_3 \parallel u_4, u_i \in \{0,1\}^8$, слово

$$G_r(u) = RotHi^r(H(u_1) \parallel H(u_2) \parallel H(u_3) \parallel H(u_4)),$$

где $RotHi$ — циклический сдвиг влево.

Переменные: $a, b, c, d, e \in \{0,1\}^{32}$.

Входными данными являются блок $X \in \{0,1\}^{128}$ и ключ $K \in \{0,1\}^{256}$.

Выходными данными является зашифрованный блок $Y \in \{0,1\}^{128}$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	B1	94	BA	C8	0A	08	F5	3B	36	6D	00	8E	58	4A	5D	E4
1	85	04	FA	9D	1B	B6	C7	AC	25	2E	72	C2	02	FD	CE	0D
2	5B	E3	D6	12	17	B9	61	81	FE	67	86	AD	71	6B	89	0B
3	5C	B0	C0	FF	33	C3	56	B8	35	C4	05	AE	D8	EO	7F	99
4	E1	2B	DC	1A	E2	82	57	EC	70	3F	CC	F0	95	EE	8D	F1
5	C1	AB	76	38	9F	E6	78	CA	F7	C6	F8	60	D5	BB	9C	4F
6	F3	3C	65	7B	63	7C	30	6A	DD	4E	A7	79	9E	B2	3D	31
7	3E	98	B5	6E	27	D3	BC	CF	59	1E	18	1F	4C	5A	B7	93
8	E9	DE	E7	2C	8F	0C	0F	A6	2D	DB	49	F4	6F	73	96	47
9	06	07	53	16	ED	24	7A	37	39	CB	A3	83	03	A9	8B	F6
A	92	BD	9B	1C	E5	D1	41	01	54	45	FB	C9	5E	4D	0E	F2
B	68	20	80	AA	22	7D	64	2F	26	87	F9	34	90	40	55	11
C	BE	32	97	13	43	FC	9A	48	A0	2A	88	5F	19	4B	09	A1
D	7E	CD	A4	D0	15	44	AF	8C	A5	84	50	BF	66	D2	E8	8A
E	A2	D7	46	52	42	A8	DF	B3	69	74	C5	51	EB	23	29	21
F	D4	EF	D9	B4	3A	62	28	75	91	14	10	EA	77	6C	DA	1D

Таблица 4.1: Подстановка H

4.2 Belt: алгоритм зашифрования

1. Определить $(X_1, X_2, X_3, X_4) = \mathbf{Split}(X, 32)$.
2. Определить $(K_1, K_2, \ldots, K_8) = \mathbf{Split}(K, 32)$.
3. Обозначить $k[i] = K_{(i-1)mod8+1}, i = 1, 2, \ldots, 56$.
4. Установить $a \leftarrow X_1, b \leftarrow X_2, c \leftarrow X_3, d \leftarrow X_4$.
5. Для $i = 1, 2, \ldots, 8$ выполнить (Рис. 4.1):
   - 5.1 $b \leftarrow b \oplus G_5(a \boxplus k[7i-6])$;
   - 5.2 $c \leftarrow c \oplus G_{21}(d \boxplus k[7i-5])$;
   - 5.3 $a \leftarrow a \boxminus G_{13}(b \boxplus k[7i-4])$;
   - 5.4 $e \leftarrow G_{21}(b \boxplus c \boxplus k[7i-3]) \oplus \langle i \rangle_{32}$;
   - 5.5 $b \leftarrow b \boxplus e$;
   - 5.6 $c \leftarrow c \boxminus e$;
   - 5.7 $d \leftarrow d \boxplus G_{13}(c \boxplus k[7i-2])$;
   - 5.8 $b \leftarrow b \oplus G_{21}(a \boxplus k[7i-1])$;

5.9 $c \leftarrow c \oplus G_5(d \boxplus k[7i])$;

5.10 $a \leftrightarrow b$;

5.11 $c \leftrightarrow d$;

5.12 $b \leftrightarrow c$.

6. Установить $Y \leftarrow b \parallel d \parallel a \parallel c$.

7. Возвратить Y.

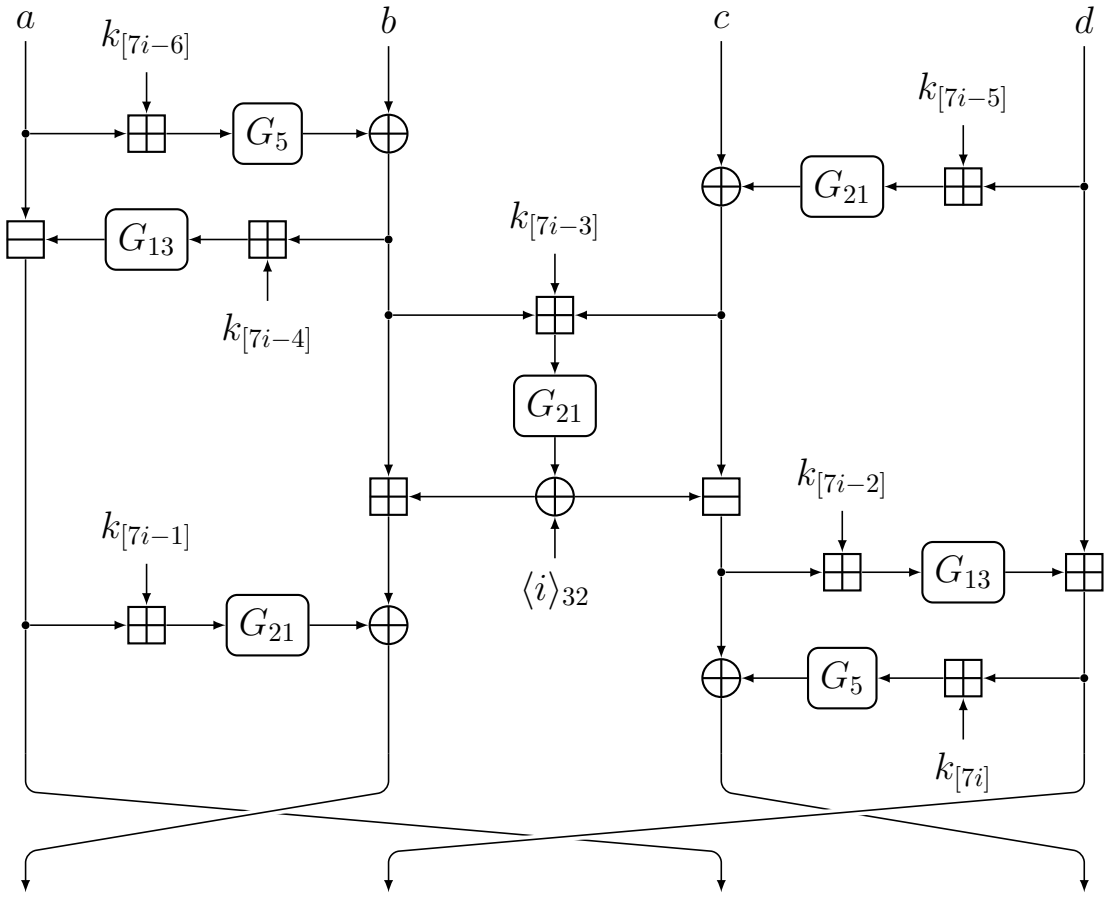

Рис. 4.1: Вычисления на i-ом такте зашифрования

4.3 Результаты добавления примитива Belt в CASCADA

На первом этапе для исследования криптографического примитива с помощью библиотеки CASCADA необходимо реализовать шифр в модуле при-

митивов, если его нет в списке уже реализованных. В документации к проекту отмечено, что добавление нового примитива — задача нетривиальная, поэтому в качестве основы для нового шифра следует взять уже реализованный в CASCADA примитив наиболее похожий на исследуемый, и адаптировать его. Для реализации BelT в качестве примера были взяты такие примитивы как Speck и AES. Для корректной работы решателей, используемых в CASCADA, исследуемый примитив должен быть реализован в бит-векторной конструкции и состоять только из бит-векторных операций, определённых в бит-векторном модуле CASCADA.

CASCADA предоставляет множество способов поиска характеристик, необходимых для различных атак распознавания. Часть способов требует лишь реализованный в бит-векторной конструкции примитив, для остальных же необходимо строить дополнительные модели (XorModelBlockCipher, RXModelBlockCipher и др.)

В ходе практической части курсовой работы в модуль примитивов CASCADA был добавлен шифр BelT, реализованный в бит-векторной структуре, поддерживаемой CASCADA для поиска характеристик. Также были исследованы несколько первых раундов BelT, а именно поиск недействительных характеристик линейных аппроксимаций и дифференциалов.

При исследовании BelT возникла проблема, связанная с её тактовой функцией. Мощности CASCADA не хватает для обработки шифра в том виде, в котором он определён изначально, так как такт шифрования BelT состоит из набора преобразований и является достаточно массивным и сложным по сравнению с другими примитивами, например, Speck. Для решения данной проблемы мы рассматривали такт шифрования BelT как 7 тактов. В итоге 8 тактов шифрования BelT разбились на 56 более мелких тактов.

После проверки корректности реализации, на GitHub был сделать запрос с добавлением нового примитива Belt в официальный репозиторий CASCADA. С одобрения автора Адриана Ранеа Belt был добавлен в список примитивов и сейчас доступен в составе инструмента CASCADA.

4.4 Результаты поиска характеристик Belt с помощью CASCADA

В курсовом проекте помимо реализации примитива Belt также были предприняты попытки его исследования на недействительные характеристики, однако методы поиска CASCADA, использовавшиеся на данном этапе

работы, дали результаты лишь на 5 тактах из 56.

Результаты поиска дифференциалов на 5 тактах шифрования BelT:

```
1  from cascada.differential.difference import XorDiff
2  from cascada.smt.invalidpropsearch import round_based_invalidprop_search
3  from cascada.primitives import belt
4
5  Belt = belt.BeltCipher
6  iterator = round_based_invalidprop_search(Belt, 5, 5, XorDiff, "btor")
7  tuple_rounds, tuple_chs = next(iterator)
8  print(tuple_rounds, ":", ', '.join([ch.srepr() for ch in tuple_chs]))
9
10 (0, 2, 1, 2) : Ch(w=0, id=00000000 00000001 00000001 00000000, od=01000000
       00000000 01000000 00000000), Ch(w=Infinity, id=01000000 00000000 01000000
        00000000, od=00000000 00000000 00000000 00000001), Ch(w=0, id=00000000
       00000000 00000000 00000001, od=00000000 00000000 01000000 00000000)
```

Листинг 4.1: Поиск невозможных дифференциалов

Строка номер 10 является выводом функции поиска характеристик *round_based_invalidprop_search* с заданным параметром XorDiff, что значит поиск дифференциалов.

Результаты поиска характеристик линейных аппроксимаций на 5 тактах шифрования BelT:

```
1  from cascada.linear.mask import LinearMask
2  from cascada.smt.invalidpropsearch import round_based_invalidprop_search
3  from cascada.primitives import belt
4
5  Belt = belt.BeltCipher
6  iterator = round_based_invalidprop_search(Belt, 5, 5, LinearMask, "btor")
7  tuple_rounds, tuple_chs = next(iterator)
8  print(tuple_rounds, ":", ', '.join([ch.srepr() for ch in tuple_chs]))
9
10 (0, 2, 1, 2) : Ch(w=0, id=08000000 00000000 00000000 80000000, od=00000000
       00000008 00000000 00000080), Ch(w=Infinity, id=00000000 00000008 00000000
        00000080, od=00000000 00000000 00000000 00000000 80000000), Ch(w=0, id
       =00000000 00000000 00000000 00000000 80000000, od=00000000 00000000
       00000080 00000000)
```

Листинг 4.2: Поиск линейных аппроксимаций с нулевой корреляцией

Аналогично строка номер 10 является выводом функции поиска характеристик *round_based_invalidprop_search* с заданным параметром LinearMask, что значит поиск линейных аппроксимаций.

В этом семестре мы продолжили работу по оценке надёжности шифра Belt, однако сосредоточились на поиске обычных характеристик вместо недействительных. Первым инструментом для исследований стал метод *round_based_ch_search*, который при заданном параметре XorDiff/RXDiff/LinearMask ищет выбранную характеристику в указанном диапазоне тактов шифра. Также можно указать дополнительные параметры:

начальный вес, решатель, поддерживающий количественные формулы, и др. Задав в качестве искомой характеристики XorDiff были получены следующие результаты:

```
/*************************************************************************/
13 : Ch(w=10, id=00004000 00100000 00040000 80000000, od=80000000 00100000
   00000000 80000000)
14 : Ch(w=11, id=00008000 00008000 00100000 80000000, od=00100000 80000000
   80000000 00000010)
15 : Ch(w=13, id=00001000 80010000 00000000 00000400, od=00000000 80000000
   00000000 00000010)
16 : Ch(w=14, id=00004000 00002000 00008000 80000000, od=00002000 80100000
   80100000 80000000)
17 : Ch(w=15, id=80000000 00100001 00f00000 00000000, od=00000000 00000000
   00000000 00000000)
18 : Ch(w=15, id=80000000 e0100000 00f00000 00000000, od=00000000 00000000
   00000000 00000000)
19 : Ch(w=15, id=80000000 0010000b 00f00000 00000000, od=00000000 00000000
   00000000 00000000)
/*************************************************************************/
```

Листинг 4.3: Поиск дифференциалов

Данный результат заведомо неверный, т.к. после 16 такта происходит аннигиляция, т.е. выходная разность обнуляется. Для решения этой проблемы было предложено добавить условие на ненулевую выходную разность, т.е. характеристика с такой выходной разностью игнорируется, и поиск продолжается. При таком подходе тот же метод даёт следующие результаты:

```
/*************************************************************************/
56 : Ch(w=308, id=00001000 80004000 a0000000 00001000, od=00000000 00000000
   00100000 80000000)
/*************************************************************************/
```

Листинг 4.4: Поиск дифференциалов (с условием на ненулевую od)

Этот результат даёт слишком хорошие показатели и также является неверным, т.к. на 56 тактах число активных S-блоков не больше 56 * 4 = 224 (число активных S-блоков не больше общего числа задействованных S-блоков).

4.5 Проблема аннигиляции

Рассмотрим центральную часть схемы зашифрования Belt.

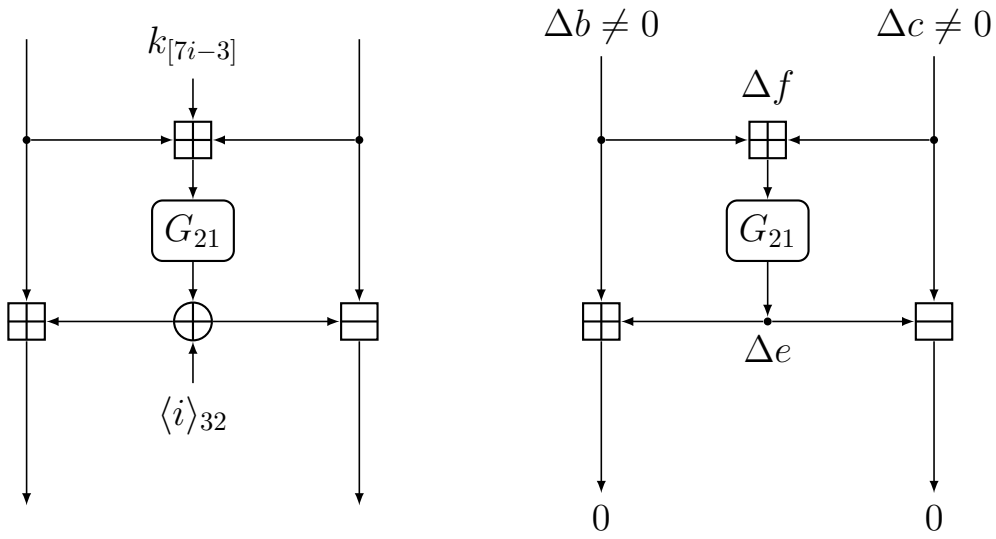

Рис. 4.2: Центральный блок зашифрования Belt в исходном виде и в упрощенном с аннигиляцией

Именно в этом месте происходит аннигиляция, которой, как видно из упрощенной схемы на Рис. 4.2, на самом деле не может произойти. Первой версией причины возникновения аннигиляции стал S-блок в преобразовании G_{21} центральной части шифра. Однако предположение не подтвердилось, при удалении S-блока, т.е. подстановки H из функции G_{21} аннигиляция не исчезла, но появилась уже в 28 такте.

Версия, рассматриваемая в данный момент это использование формул Липма для модульного сложения при поиске характеристик. Рассмотрим данные формулы поподробней. Пусть $x, y \in \{0,1\}^n$, тогда

$$x \boxplus y = x \oplus y \oplus carry(x, y),$$

где $carry(x, y)$ определяется рекурсивно следующим образом:

$$carry(x, y) := c \in \{0,1\}^n:$$

1) $c_0 = 0$;
2) $c_{i+1} := (x_i \wedge y_i) \oplus (x_i \wedge c_i) \oplus (y_i \wedge c_i)$ для $i \geq 0$.

Данные формулы уже давно известны и активно используются в криптографии, однако в литературе уже был исследован тот факт, что формулы Липма не работают, когда две и более модульные операции выполняются подряд, что собственно и происходит в Belt. Поэтому при исследовании конкретно этого шифра использовать данные формулы неверно.

Заключение

В курсовом проекте была рассмотрена тема оценки надёжности блочных криптосистем используя автоматизированные средства, с помощью которых можно находить свойства криптосистем необходимые для множества атак распознавания. Большинство автоматизированных методов криптоанализа, ранее опубликованных в литературе, зачастую не имеют программной реализации или предоставляют узкие реализации, специфичные для шифра и конкретной атаки распознавания. На фоне этого CASCADA, современный инструмент для оценки широкого класса криптографических алгоритмов против множества атак распознавания, является заметным исключением.

В первой главе были рассмотрены классические атаки распознавания, в которых используются характеристики. Именно эти атаки и реализованы в библиотеке CASCADA: разностная атака, разностная атака на связанных ключах, атака по невозможным дифференциалам, линейная атака, атака с нулевой корреляцией и RX-атака.

Вторая глава посвящена автоматизированным средствам оценки надёжности криптографических примитивов и теоретическим основам, на которых основаны данные средства, а именно на задачах удовлетворения ограничений, таких как SMT (теория выполнимости по модулю) и MILP (смешанное целочисленное линейное программирование). Мы изучили основные принципы решений задач SMT и MILP, а также их приложение в криптографии.

Далее в третьей главе, мы подробно рассмотрели инструмент CASCADA: его составные модули, теоретические основы и методы поиска, реализованные в библиотеке.

Четвёртая глава посвящена практической части работы с автоматическим инструментом CASCADA. В ней подробно описан алгоритм шифрования Belt и приведены результаты по реализации шифра в бит-векторной структуре и поиска характеристик с помощью методов CASCADA. Код размещен на площадке GitHub, ссылка на которую приведена `https://github.com/SviridPolina/CASCADA-BelT`. А также ссылка на оригинальный репозиторий CASCADA, в модуль примитивов которого уже добавлен Belt `https://github.com/ranea/CASCADA`.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Агибалов Г.П., Панкратова И.А. Элементы теории статистических аналогов дискретных функций с применением в криптоанализе итеративных блочных шифров. – Изд-во Томского гос. ун-та, 2010.

2. Бабепко Л.К., Ищукова Е.А. Криптографическая защита информации: симметричное шифрование // Учебное пособие. – Изд-во Южного федерального ун-та, 2015.

3. СТБ 34.101.31-2020. Информационные технологии и безопасность. Алгоритмы шифрования и контроля целостности // Государственный стандарт Республики Беларусь.

4. Харин Ю. С., Агиевич С. В., Васильев Д. В., Матвеев Г. В. Криптология. — Изд-во Белорус. гос. ун-та, 2013.

5. Barrett C., Sebastiani R., Seshia S. A., Tinelli C. Satisfiability Modulo Theories. // Handbook of Satisfiability. — IOS Press, 2008.

6. Bellare M., Kohno T. A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. —Dept. of Computer Science & Engineering, University of California at San Diego.

7. Biham, E. New Types of Cryptanalytic Attacks Using Related Keys. —Computer Science Department Technion - Israel Institute of Technology Haifa, 1994.

8. Biham E., Shamir A. Differential Cryptanalysis of the Data Encryption Standard. // Journal of Cryptology, 1991 International Association for Cryptologic Research. — Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, 1993.

9. Biham E., Biryukov A., Shamir A. Cryptanalysis of Skip jack Reduced to 31 Rounds using Impossible Differetials. — Technion - Computer Science Department. // Technical Report CS0947, 1998.

10. Bogdanov A., Rijmen V. Linear Hulls with Correlation Zero and Linear Cryptanalysis of Block Ciphers. —KU Leuven, ESAT/COSIC and IBBT.

11. Daemen J., Rijmen V. Probability distributions of correlation and differentials in block ciphers. // J. Math. Crypt. 1, 2007.

12. Derkach, O. Verification of Model Assumptions in Cryptanalysis of Arx-Ciphers – Department of Mathematical Methods of Information Security. Institute of Physics and TechnologyNational Technical University of Ukraine “Igor Sikorsy Kyiv Polytechnic Institute”.

13. Jakimoski G., Desmedt Y. Related-Key Differential Cryptanalysis of 192-bit Key AES Variants. — Computer Science Department, Florida State University Tallahassee.

14. Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer (2002).

15. Lu J., Liu Y., Ashur T., Sun B., Li C. Rotational-XOR Cryptanalysis of Simon-Like Block Ciphers. —Department of Mathematics, National University of Defense Technology. // Lecture Notes in Computer Science, 2020.

16. Mouha N., Wang Q., Gu D., Preneel B. Differential and Linear Cryptanalysis using Mixed-Integer Linear Programming. — Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit Leuven. Interdisciplinary Institute for BroadBand Technology (IBBT). Department of Computer Science and Engineering, Shanghai Jiao Tong University.

17. Moura L., Bjorner N. Satisfiability Modulo Theories: An Appetizer. // Microsoft Research, One Microsoft Way.

18. Nejati, S. CDCL(Crypto) and Machine Learning based SAT Solvers for Cryptanalysis. // A thesis degree of Doctor of Philosophy in Electrical and Computer Engineering, 2020.

19. Ranea A., Rijmen V. Characteristic Automated Search of Cryptographic Algorithms for Distinguishing Attacks (CASCADA), 2020.

20. Sahu H. K., Pillai N. R., Gupta I., Sharm R. K. SMT solvers based Cryptanalysis of Block Ciphers. — Department of Mathematics, IIT Delhi.

21. Soleimany H., Nyberg K. Zero-Correlation Linear Cryptanalysis of Reduced-Round LBlock. — Aalto University School of Science, Department of Information and Computer Science.

22. Stanek, M. Experimenting with Shule Block Cipher and SMT Solvers. — Department of Computer Science Comenius University.

23. Wu S., Wang M. Security Evaluation against Differential Cryptanalysis for Block Cipher Structures. — State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences. Graduate School of Chinese Academy of Sciences.

24. Zhou C., Zhang W., Ding T., Xiang Z. Improving the MILP-based Security Evaluation Algorithm against Differential/Linear Cryptanalysis Using A Divide-and-Conquer Approach. // IACR Transactions on Symmetric Cryptology.