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

ОЦЕНКА НАДЁЖНОСТИ БЛОЧНЫХ КРИПТОСИСТЕМ С ПОМОЩЬЮ СРЕДСТВА CASCADA

Курсовая работа

Свирид Полины Дмитриевны
студента 3 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 следить за их устойчивостьюстью ко всем существующим атакам. История показывает, что шифр, разработанный без должной обработки его защищенности от криптоанализа, часто приводит к недостаткам и 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, но в недавних работах было предложено использовать 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.

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 свойства шифра, которое можно использовать, является самой сложной частью атаки.

Хорошо известными примерами атак, включающих этап распознавания, являются разностная, линейная атаки, а также атаки на связанных ключах, 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 праобразов из множества наборов блоков и σ из множества подстановок на $\{0,1\}^n$ выбраны случано, 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 дествия на m-наборе праобразов.

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 криптографиские 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.

Дифференциалом называется разность двух текстов $X, X^{'} \in \{0,1\}^n$. Для криптосистем, подобных DES, дифференциал или разность 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 (α, β, δ), где α — разность текстов на входе блочного алгоритма шифрования, β — разность выходов после r-го такта шифрования и δ — разность ключей. Вероятность дифференциала связанных ключей в r-раунде — это вероятность того, что разность выходов r-го раунда будет равна β, когда разность на входе равна α, разность ключей равна δ, а открытый текст X и ключ K выбираются равномерно случайным образом.

Разностная атака с использованием связанных ключей может работать следующим образом:

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

2. Выбрать случайным образом X и зашифровать под ключом K, чтобы получить зашифрованный текст Y. Вычислить $X^{'} = X + \alpha$ и зашифровать под ключом $K^{'} = K + \delta$ для получения зашифрованного текста $Y^{'}$.

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 β, где $D_K(Y)$ — функция дешифрования 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, 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 алгоритмы.

Пусть F — d -тактовая блочная криптосистема.

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

$$\alpha \cdot X \oplus \beta \cdot Y = \gamma \cdot K,$$

который выполняется с некоторой вероятностью p. В отличие от разностного криптоанализа, в котором большое значение вероятности гарантирует успех атаки, в линейном криптоанализе успех криптоанализа 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 ненулевими 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.

Рассматриваем 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 $(y, y^{'}) = (x \lll \eta, (x \lll (\gamma \oplus \eta)) \oplus (\delta \lll \eta))$.

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 $X^{'}$:

$$\Delta_{\gamma}(X, X^{'}) = X^{'} \oplus (X \lll \gamma) = 0.$$

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 ограниченииий, 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 отличительных атак, а именно разностный и линейный криптоанализ в CryptoSMT и RX-атака, атаки по связвнным 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 знаний и усилий, 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 Основные понятия

Пропозициоая формула ϕ может быть пропозиционной переменной p, отрицанием $\neg\phi_0$, конъюкцией $\phi_0 \wedge \phi_1$, дизъюнкцией $\phi_0 \vee \phi_1$, импликацией $\phi_0 \Rightarrow \phi_1$ или эквиваленцией $\phi_0 \Leftrightarrow \phi_1$ более мелких формул. Привсвоение истинности M для формулы ϕ отображает пропозиционные переменные, вхо-

дящие в ϕ, в $\{true, false\}$. Присваивание истинности M удовлетворяет ϕ ($M \models \phi$), если M заставляет ϕ принимать значение истина при обычной реализации таблицы истинности. Например, пусть ϕ — это следующая формула:

$$p \vee (\neg q \wedge r),$$

тогда присваивание следующей истинности удовлетворяет ϕ:

$$M = \{p \mapsto false, q \mapsto false, r \mapsto true\}.$$

Формула ϕ выполнима, если существует M такое, что $M \models \phi$, и ϕ тождественна, если для всех M выполняется $M \models \phi$. Мы говорим что, ϕ_1 и ϕ_2 равносильны, если ϕ_1 выполнима тогда и только тогда, когда ϕ_2 выполнима.

Литерал — это либо пропозиционная переменная p, либо ее отрицание ¬p. Предложение — это дизъюнкция литералов:

$$l_1 \vee \ldots \vee l_n.$$

Формула находится в конъюнктивной нормальной форме (КНФ), если она является конъюнкцией дизъюнкций:

$$C_1 \wedge \ldots \wedge C_m.$$

Будем писать формулы КНФ в виде набора дизъюнкций. Любая формула может быть преобразована в КНФ, за линейное время, путём введения новых переменных для каждой составной подформулы и добавления подходящих предложений.

2.2 Логика первого порядка

В то время как язык решателей SAT основан на булевой логике, язык решателей SMT — это логика первого порядка. Язык включает в себя операции булевой логики, но вместо пропозиционных переменных используются более сложные выражения, включающие символы констант, функций и предикатов. Как и в логике высказываний, выражения в логике первого порядка состоят из последовательностей символов. Символы делятся на логические символы и нелогические символы или параметры.

Логические символы:

- Скобки: (,)

- Логические операции: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- Переменные: υ_1, υ_2, ...
- Кванторы: $\forall$, $\exists$

Параметры (нелогические символы):

- Символ равенства (необязательно): $=$
- Предикатные символы: $p(x)$, $x < y$
- Константные символы: 0, π
- Функциональные символы: $f(x)$, $x + y$

Каждый символ предиката и функции имеет соответствующую арность — натуральное число, указывающее, сколько аргументов он принимает. Равенство — это специальный символ арности 2. Постоянные символы также можно рассматривать как функции, арность которых равно 0.

2.3 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 теорий 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.

2.4 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 доказатть 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 разнастного, так и для линейного криптоанализа, можно получить верхнюю границу вероятности наилучшей характеристики.

2.5 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:

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

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

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

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.

Дополнительное линейное уравнение добавляется, чтобы гарантировать, что по крайней мере один S-блок активен: это позволяет избежать тривиального решения, когда минимальное количество активных S-блоков равно нулю. 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, но только x-переменные, соответствующие 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.

2.6 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:

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

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 сножества 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, невозможных дифференциалах 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:

- Объединение 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 проблемой без кванторов, и она равноразрешима той же задаче SMT с кванторами существования. С другой стороны, 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:

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

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

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

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 $k \in \mathcal{K}$:

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

Поскольку пара RX-разностей (α, β) эквивалентна дифференциалу (α, β), определяемому как

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

будем называть RX-разностные пары RX-дифференциалами, а дифференциалы с $\bigtriangledown = \oplus = \bigtriangleup$ будем называть XOR-дифференциалами.

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}.

На практике для утверждения, что блочные шифры защищены от атак распознавания, необходимо показать, что никакие высоковероятные 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 задачи SMT с битовым вектором, вероятности распространения характеристик и свойств должны быть закодированы 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 достоверности, ограничением вероятности-один и ограничением веса.

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

- Ограничение вероятности-один 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.

3.4 Модель бит-векторных характеристик

Назовем характеристику $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_r)$ функции $f = f_{r-1} \circ f_{r-2} \circ \ldots \circ f_0$ действительной, 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 достоверности, ограничение вероятности-один и ограничение веса .

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

- Ограничение вероятности-один 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 достоверности, вероятности-один 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 маловесовых характеристи

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 целочисленные 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 ограничение суммы веса распространения Γ_{KS} и веса распространения Γ_{E_k} до целевого 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 невыполнимо, 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 по связвнным ключам и 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 модуль битового вектора, модуль примитивов, модули свойств и модуль SMT, так что каждый модуль можно использовать и расширять независимо.

3.8 Бит-векторный модуль

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 запоми-

нания представляет собой компромисс между пространством и временем, с помощью составления таблиц, где промежуточные результаты сохраняются в таблице, чтобы их можно было извлечь, когда те же самые входы происходят снова.

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.

3.9 Модуль примитивов

Модуль примтивов предоставляет типы данных для представления функций шифрования и блочных шифров. В то время как функции расписания ключей могут быть реализованы непосредственно как бит-векторные функции, тип функции шифрования определяет бит-векторную функцию и список тактовых ключей, а тип блочного шифрования определяет функцию расписания ключей и функцию шифрования.

Модуль примитивов реализует множество криптографических примитивов, а именно AES, CHAM, Chaskey, FEAL, HIGHT, Simeck, Simon, TEA, XTEA и другие. Список примитивов продолжает пополняться.

3.10 Модули свойств

Модули свойств CASCADA состоят 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 представления 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 разницы XOR и разницы RX. Он реализует тривиальные

модели XOR и RX многих $\oplus$-линейных и детерминированных по распространению операций, то есть операций, которые распространяют входное свойство на уникальное выходное свойство с вероятностью единица.

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 $\vee$ путем экстраполяции ограничений для 1-битных входных данных.

3.11 SMT модуль

Модуль SMT реализует автоматизированные методы, описанные ранее. Для решения базовых проблем 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 бит-вектора SMT, можно настроить многие параметры автоматизированных методов, реализованных в модуле SMT, в том числе тип ограничений (например, ограничения достоверности и веса или ограничения только для вероятности-один), дополнительные ограничения, уровень детализации или необходимость фильтрации характеристик с использованием эмпирического веса. Если последняя опция включена, после того, как характеристика получена как решение задачи SMT, вычисляется эмпирический вес характеристики, а характеристики с большими ошибками аппроксимации между их весами распространения и их эмпирическими весами отбрасываются.

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 авторы продолжают его дополнять.

Основной целью практической части курсовой работы было дополнение инструмента CASCADA государственным стандартом симметричного шифрования и контроля целостности Республики Беларусь BelT.

Заключение

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

2. Бабепко Л.К., Ищукова Е.А. Криптографическая защита информации: симметричное шифрование // Учебное пособие – Изд-во: Южный федеральный университет, 2015.

3. Агибалов Г.П., Панкратова И.А. Элементы теории статистических аналогов дискретных функций с применением в криптоанализе итеративных блочных шифров – Изд-во: Томский государственный университет, 2010.

4. Derkach, O. Verification of Model Assumptions in Cryptanalysis of Arx-Ciphers – National Technical University of Ukraine "Igor Sikorsy Kyiv Polytechnic Institute".

5. Daum, M. "Cryptanalysis of hash functions of the MD4-Family," Ph.D. thesis , Ruhr-Universität Bochum, May 2005.

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

7. Tomer Ashur and Yunwen Liu. 'Rotational Cryptanalysis in the Presence of Constants'. In: IACR Trans. Symmetric Cryptol. 2016.1 (2016).

8. Ralph Ankele and Eik List. 'Differential Cryptanalysis of RoundReduced Sparx-64/128'. In: ACNS. Vol. 10892. Lecture Notes in Computer Science. Springer, 2018.

9. Eli Biham. 'New Types of Cryptanalytic Attacks Using Related Keys'. In: J. Cryptol. 7.4 (1994).

10. Nocedal, J. Numerical Optimization / J. Nocedal, S.J. Wright. – Springer Series in Operations Research, Springer Verlag, 2006.

11. Eli Biham and Adi Shamir. Differential Cryptanalysis of the Data Encryption Standard. Springer, 1993.

12. Jinyu Lu et al. 'Rotational-XOR Cryptanalysis of Simon-Like Block Ciphers'. In: ACISP. Vol. 12248. Lecture Notes in Computer Science. Springer, 2020.

13. Yunwen Liu, Qingju Wang and Vincent Rijmen. 'Automatic Search of Linear Trails in ARX with Applications to SPECK and Chaskey'. In: ACNS. Vol. 9696. Lecture Notes in Computer Science. Springer, 2016.

14. Mitsuru Matsui. ‘Linear Cryptanalysis Method for DES Cipher’. In: EUROCRYPT. Vol. 765. Lecture Notes in Computer Science. Springer, 1993.

15. Nicky Mouha and Bart Preneel. ‘A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis’. In: IACR Cryptol. ePrint Arch., 2013.

16. Ling Song, Zhangjie Huang and Qianqian Yang. ‘Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA’. In: ACISP (2). Vol. 9723. Lecture Notes in Computer Science. Springer, 2016.