МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра математического моделирования и анализа данных

ВОЛКОВ
Евгений Сергеевич

ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ПОСТРОЕНИЯ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ

Дипломная работа

Научный руководитель:
кандидат физ.-мат. наук,
доцент М.В. Мальцев

Допущена к защите

«___» _____2023 г.

Зав. кафедрой математического
моделирования и анализа данных
доктор экономических наук,
доцент В.И. Малюгин

Минск, 2023

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Дипломная работа: 44 страницы, 4 главы, 19 рисунков, 2 таблицы, 27 использованных источников, 1 приложение.

Ключевые слова: ХЭШ-ФУНКЦИЯ, НЕЙРОННЫЕ СЕТИ, МАШИННОЕ ОБУЧЕНИЕ, ЧУВСТВИТЕЛЬНОСТЬ, СТОЙКОСТЬ, КРИПТОГРАФИЯ, КРИПТОАНАЛИЗ.

Объект исследования: криптографические алгоритмы, основанные на нейронных сетях.

Цель работы: разработка и исследование криптографических алгоритмов, основанных на нейронных сетях, анализ стойкости и чувствительности, улучшение рассмотренных алгоритмов.

Методы исследования:

1) теоретические: изучение литературы, посвящённой использованию нейронных сетей и машинного обучения в криптографии и криптоанализе; анализ и описание криптографических алгоритмов, основанных на нейронных сетях;
2) практические: разработка и реализация криптографической хэш-функции, основанной на нейронной сети; анализ стойкости и чувствительности хэш-функции; разработка модификаций хэш-функции; проведение сравнительного анализа.

Результат: разработка и реализация криптографической хэш-функции; проведение анализа стойкости к различным атакам и чувствительности к изменению во входных данных; разработка модификаций реализованной хэш-функции и проведение сравнительного анализа; описание преимуществ и недостатков использования разработанной хэш-функции.

Область применения: разработка и анализ криптографических алгоритмов.

РЭФЕРАТ

Дыпломная праца: 44 старонкі, 4 раздзелы, 19 малюнкаў, 2 табліцы, 27 выкарыстаных крыніц, 1 дадатак.

Ключавыя словы: ХЭШ-ФУНКЦЫЯ, НЕЙРОННЫЯ СЕТКІ, МАШИННАЕ НАВУЧАННЕ, АДЧУВАЛЬНАСЦЬ, УСТОЙЛІВАСЦЬ, КРЫПТАГРАФІЯ, КРЫПТААНАЛІЗ.

Аб’ект даследавання: крыптаграфічныя алгарытмы, заснаваныя на нейронных сетках.

Мэта працы: распрацоўка і даследаванне краптаграфічных алгарытмаў, заснаваных на нейронных сетках; аналіз устойлівасці і адчувальнасці, паляпшэнне разгледжаных алгарытмаў.

Метады даследавання:

1) тэарытычныя: аналіз літаратуры, прысвечанай выкарыстанню нейронных сетак і машыннага навучання ў крыптаграфіі і крыптааналізе; аналіз і апісанне крыптаграфічных алгарытмаў, заснаваных на нейронных сетках;
2) практычныя: распрацоўка і рэалізацыя крыптаграфічнай хэш-функцыі, заснаванай на нейроннай сетке; аналіз устойлівасці і адчувальнасці хэш-функцыі; распрацоўка мадыфікацый хэш-функцыі і іх аналіз.

Вынік: распрацоўка і рэалізацыя крыптаграфічнай хэш-функцыі; выкананне аналізу устойлівасці да розных атак і адчувальнасці да змены ва ўваходных дадзеных; распрацоўка мадыфікацый рэалізаванай хэш-функцыі і іх аналіз; апісанне пераваг і недахопаў выкарыстання рэалізаванай хэш-функцыі.

Вобласць ужывання: распрацоўка і аналіз крыптаграфічных алгарытмаў.

ABSTRACT

Diploma thesis: 44 pages, 4 chapters, 19 figures, 2 tables, 27 sources, 1 appendix.

Keywords: HASH FUNCTION, NEURAL NETWORKS, MACHINE LEARNING, SENSITIVITY, RESISTANCE, CRYPTOGRAPHY, CRYPTOANALYSIS.

The object of the study: cryptographic algorithms based on neural networks.

The purpose of the work: development and research of cryptographic algorithms based on neural networks, analysis of security and sensitivity, improvement of the considered algorithms.

Research methods:

1) theoretical: study of the sources on the use of neural networks and machine learning in cryptography and cryptanalysis; analysis and description of cryptographic algorithms based on neural networks;
2) practical: development and implementation of a cryptographic hash function based on a neural network; analysis of the resistance and sensitivity of the hash function; development of hash function modifications and their analysis.

Result: development and implementation of a cryptographic hash function; analysis of resistance to various attacks and sensitivity to changes in the input data; development of hash function modifications and their analysis; description of the advantages and disadvantages of using the developed hash function.

Scope: development and analysis of cryptographic algorithms.

ВВЕДЕНИЕ

За последние десять лет наблюдается повышение интереса к нейронным сетям. Сегодня нейронные сети используются практически в любой сфере человеческой жизни (экономика, медицина, компьютерная безопасность и т.д.). Нейронные сети являются исключительно мощным методом моделирования, ведь они способны находить самые нетривиальные зависимости. Именно поэтому искусственные нейронные сети постепенно внедряются в качестве инструмента для анализа и синтеза криптографических алгоритмов.

Целью работы является исследование практического применения нейронных сетей в криптографии, а именно, для анализа и синтеза криптографических алгоритмов. В ходе работы был разработан алгоритм создания хэш-функции с использованием нейронной сети, а также проведён анализ стойкости и чувствительности полученной функции хэширования.

Прежде чем перейти к рассмотрению алгоритма, нужно ознакомиться с основами нейронных сетей, принципом их работы и изучить связь криптографии и нейронных сетей.

Первая глава представляет собой введение в нейронные сети. В ней рассмотрены структура нейронных сетей, их виды, процесс обучения нейронных сетей, сферы их применения.

Вторая глава содержит в себе обзор литературы, посвящённой применению машинного обучения и нейронных сетей в криптографии (создание синтетических отпечатков пальцев для обучения биометрических систем безопасности, создание криптографических ключей с высокой энтропией на основе биометрических данных, генерация хэш-функций) и криптоанализе (исследование криптографических алгоритмов, проведение атак на криптосистемы, исследование их надёжности и устойчивости).

В третьей главе описывается применение хэш-функций, в том числе и криптографических, а также приведён разработанный алгоритм работы криптографической хэш-функции на основе нейронной сети.

В заключительной четвёртой главе представлены результаты, полученные в ходе анализа реализованной хэш-функции на стойкость к атакам и чувствительность ко входным данным, исследовано влияние параметров нейронной сети на характеристики хэш-функции, а также описаны её преимущества и недостатки.

Глава 1 НЕЙРОННЫЕ СЕТИ И ИХ ПРИЛОЖЕНИЯ

1.1 Определение и структура нейронной сети

Нейронная сеть (искусственная нейронная сеть, ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма [3]. Данное понятие возникло благодаря учёным-биологам, которые занимались изучением человеческого мозга и проходящих в нём процессов, а затем пытались смоделировать их. В результате были разработаны новые алгоритмы обучения, благодаря которым стало возможным решать многие практические задачи: прогнозирование, распознавание, классификация и др.

Биологический нейрон (Рисунок 1) состоит из ядра, дендритов, по которым принимаются импульсы от других нейронов, и единственного аксона, по которому передаётся импульс. Аксон соединён с дендритами других нейронов через синапсы – специальные образования, которые влияют на передаваемый импульс. Большое количество соединённых вместе нейронов образуют нейронную сеть.

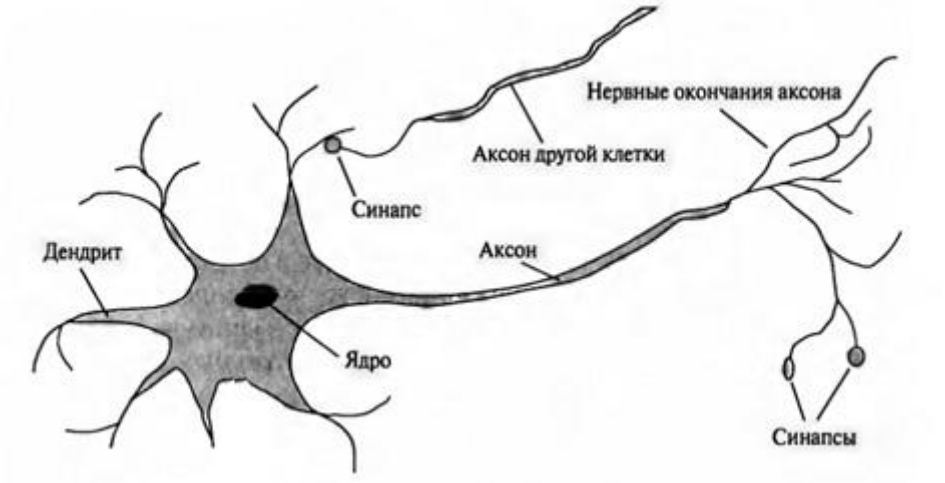

Рисунок 1 – Строение биологического нейрона

В основе искусственных нейронных сетей лежит такой же принцип. Нейроны соединены между собой в один или несколько слоёв, на вход они получают N входных значений x_i, умножают их на соответствующие веса ω_{ij} и добавляют смещение b_j. По своей сути, веса – мера влияния данного входного значения на состояние нейрона. Далее применяется функция активации $f(x)$, которая определяет выходное значение по входным значениям. Пример нейронной сети показан на Рисунке 2, где *Act* – некая функция активации.

Архитектура нейронной сети, т.е. количество слоёв и нейронов в них, определяется особенностями и сложностью конкретных задач. Однако на практике невозможно создать нейронную сеть с огромным количеством нейронов и слоёв ввиду ограниченности ресурсов.

Нейронные сети могут быть полносвязными (Рисунок 2) (каждый нейрон предыдущего слоя соединён с каждым нейроном следующего слоя), свёрточными, рекуррентными (Рисунок 3), трансформерами (комбинация свёрточных и рекуррентных), генеративно-состязательными и др.

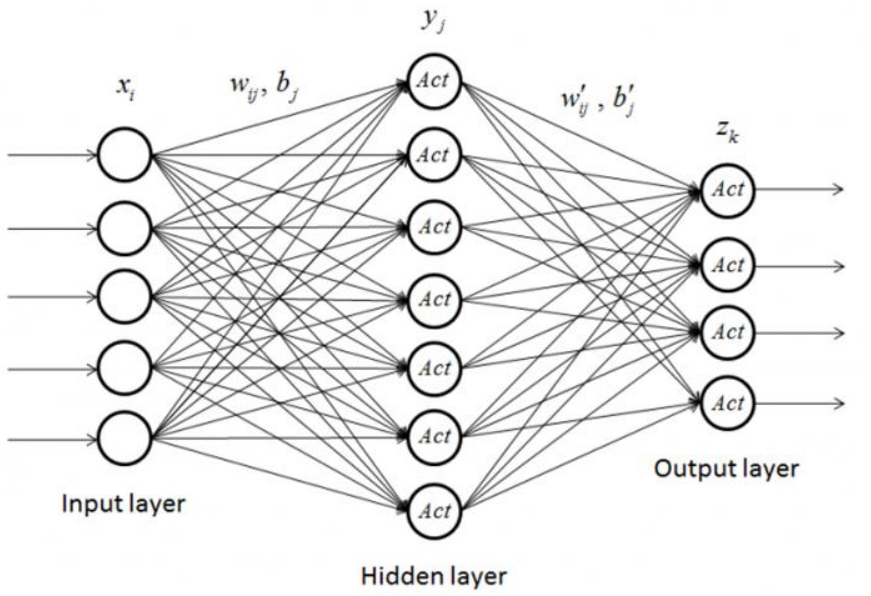

Рисунок 2 – Полносвязная нейронная сеть

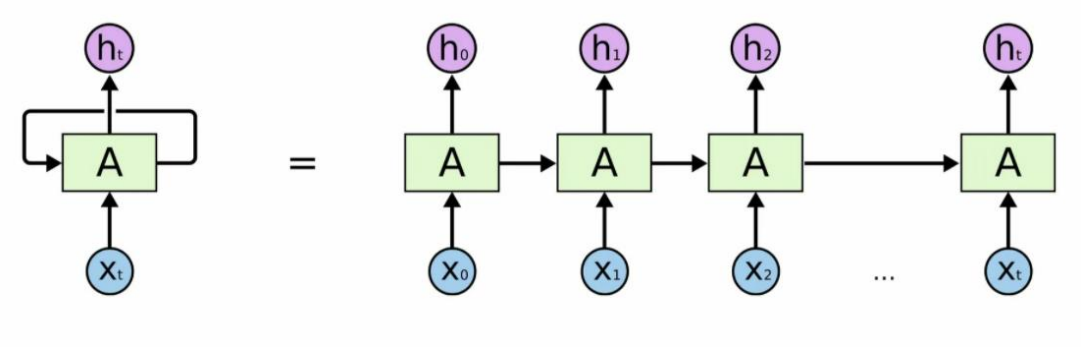

Рисунок 3 – Рекуррентная нейронная сеть

Основной идеей работы искусственных нейросетей является то, что они обучаются, а не программируются. Т.е. во время обучения на некоторых данных основная цель нейронной сети – подстроить свои веса таким образом, чтобы уменьшить ошибки по заранее заданному критерию.

1.2 Функции активации

Рассмотрим некоторые функции активации:

1) *Ступенчатая (пороговая) функция*:

$$f(x) = \begin{cases} 1, x \geq \alpha, \\ 0, x \leq \alpha. \end{cases}$$

Применяется при бинарной классификации (объект принадлежит одному из двух классов: “1” – принадлежит, “0” – не принадлежит, α – порог).

2) *Сигмоида*:

$$f(x) = (1 + e^{-x})^{-1}.$$

Множество значений данной функции – [0,1], т.е. любое значение нейрона на выходе можно отобразить на этот отрезок. Этот факт используют при решении задач классификации.

3) *Softmax*:

$$f_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}.$$

Данная функция преобразует вектор с компонентами x_i в вектор с компонентами $f_i(x)$, значения которых лежат на отрезке [0,1] и сумма которых равна 1. Другими словами, функция показывает вероятность принадлежности каждому из *N* классов.

4) *ReLU*:

$$f(x) = max(0, x).$$

На первый взгляд кажется, что функция линейна, но *ReLU*, как и их комбинация, нелинейна. Это позволяет использовать её в нейронных сетях со множеством слоёв. Ещё один плюс данной функции – разрежённость активации. В силу определения функции, в нейронных сетях с большим количеством нейронов количество используемых нейронов будет меньше, чем при использовании других функций активации.

1.3 Обучение нейронной сети

Выделяют два типа обучения: “с учителем” и “без учителя”.

При обучении “с учителем” на вход нейронной сети подаются данные для обучения вместе с правильными ответами. В этом случае можно задать критерий, по которому будут считаться ошибки предсказания нейронной сети. По выбранному алгоритму идёт изменение весовых коэффициентов до тех пор, пока ошибка не будет на довольно низком уровне.

Математически это можно описать так: в процессе обучения нейронной сетью формируется выходной сигнал y, реализующий некоторую функцию $g(x)$. Пусть решение задачи – функция $y = f(x)$, заданная входными парами (x^i, y^i). Обучение состоит в поиске некоторой функции g, близкой к f, минимизируя ошибку ε. Выбирая функцию потерь, получаем задачу многомерной оптимизации.

Существует много различных как способов задания ошибки (*MSE* – среднеквадратическая ошибка, *SSE* – суммарная квадратическая, *CrossEntropy* и др.), так и методов решения задачи оптимизации (метод градиентного спуска, метод Нестерова, AdaGrad, AdaDelta и др.).

Обучение “без учителя” отличается тем, что на вход подаются данные без ответов (например, при решении задачи кластеризации или оптимизации). Цель алгоритма – подстроить веса так, чтобы при схожих входных данных получались схожие выходные данные. В процессе обучения выделяются статистические особенности входных данных, на основе которых эти данные группируются в классы.

1.4 Применение нейронных сетей

Нейронные сети применяются в тех областях, где неэффективно использовать возможности человека, а иные алгоритмы плохо решают поставленную задачу. Нейронные сети чаще всего применяются для решения задач классификации, прогнозирования, распознавания, аппроксимации нелинейных зависимостей и др. На Рисунке 4 отображены задачи из различных сфер, для которых применяются нейронные сети.

- Проектирование и оптимизация сетей связи

Нейронные сети используются для решения задачи о нахождении оптимального пути трафика между узлами – одной из основных задач из области телекоммуникаций. Особенности задачи заключаются в том, что поиск оптимального решения проходит в реальном времени и что необходимо учитывать не только текущее состояние сети, но и наличие сбойных участков. Также нейронные сети применяются при проектировании новых сетей связи.

- Распознавание речи

Распознавание речи – одна из наиболее популярных областей применения нейронных сетей. Без распознавания речи невозможным становится существование таких функций, как голосовой ввод, голосовой поиск, голосовое управление. Нейронные сети значительно упрощают процесс распознавания речи различных пользователей, благодаря чему данная технология применяется не только в бытовых сферах, но и в медицине, бизнесе, социальной сфере.

- Распознавание символов

Задача распознавания символов очень важна в современном мире. С приходом автоматизации на парковках появилась необходимость автономного определения номера машины, что невозможно без распознавания букв и цифр на полученном снимке. Ещё одной важной задачей является распознавание рукописных текстов для перевода их в электронный формат.

Промышленность		
Управление технологическими процессами	Идентификация химических компонент	Контроль качества артезианских вод
Оценка экологической обстановки	Прогнозирование свойств синтезируемых полимеров	Управление водными ресурсами
Оптимальное планирование	Разработка нефти и газа	Управление работой пресса
Идентификация вида полимеров	Управление ценами и производством	Оптимизация работы моторов
Обнаружение повреждений	Оптимизация закупок сырья	Контроль качества изделий
Приложения аналитической химии	Анализ проблем функционирования заводов и магазинов	Прогнозирование потребления энергии
Высокие технологии		**Оборона**
Проектирование и оптимизация сетей связи	Идентификация и верификация говорящего	Анализ визуальной аэрокосмической информации
Анализ и сжатие изображений	Видеонаблюдение	Отбор целей
Распознавание печатных и рукописных символов	Автоматизированное распознавание речевых команд	Обнаружение наркотиков и взрывчатых веществ
Фальсификации в пищевой и парфюмерной пром-сти	Распознавание слитной речи с (и без) настройки на говорящего	Сличение изображений с криминальной базой данных
Обслуживание кредитных карт	Речевой ввод текста в компьютер	Предсказание целесообразности условного освобожден.
Наука и техника		**Здравоохранение**
Поиск неисправностей в научных приборах	Спектральный анализ и интерпретация спектров	Идентификация микробов и бактерий
Диагностика печатных плат	Интерпретация показаний сенсоров	Диагностика заболеваний
Идентификация продуктов	Моделирование физических систем	Интерпретация ЭКГ
Синтез новых видов стекла	Анализ данных в ботанике	Анализ качества лекарств
Автоматизированное проектирование	Планирование химических экспериментов	Обработка и анализ медицинских тестов
Оптимизация биологических экспериментов	Отбор сенсоров для контроля химических процессов	Прогнозирование результатов применения методов лечения
Геофизические и сейсмологические исследования	Прогноз температурного режима технологических процессов	Оптимизация атлетической подготовки
Распознавание ингредиентов	Диагностика сбоев сигнализации	Диагностика слуха
Бизнес и финансы		
Выбор сбытовой политики	Прогноз прибыли (Cash-flow)	Прогнозирование продаж
Принятие административных решений	Предсказание и расшивка «узких мест»	Анализ целей маркетинговой политики
Предсказания на фондовой бирже	Прогноз эффективности кредитования	Прогнозирование экономических индикаторов
Анализ финансового рынка	Прогнозирование валютного курса	Анализ страховых исков
Исследование фактора спроса	Прогнозирование и анализ цен	Отбор перспективных кадров
Моделирование бизнес-стратегии	Построение макро- и микроэкономических моделей	Стратегии в области юриспруденции
Предсказание наступления финансовых кризисов	Предсказание необходимых трудодней для реализации проекта	Оценка и прогнозирование стоимости недвижимости

Рисунок 4 – Области применения нейронных сетей

- Анализ страховых исков

Основная задача нейронных сетей в данной области – определение подозрительных страховых исков в реальном времени. Для этого на вход нейронная сеть получает различную информацию о страхователе и застрахованном имуществе (возраст, досье, наличие судимостей, стоимость имущества и др.). Нейронная сеть в результате анализа полученных сведений определяет вероятность того, что иск является мошенническим.

- Обслуживание кредитных карт

Нейронные сети применяются для поиска и предотвращения операций, в которых используются украденные банковские карты. Для этого анализируется частота и характер покупок, в результате чего определяются подозрительные сделки. Данная информация передаётся в соответствующую службу банка. Например, система, которая анализировала операции с картами VISA, предотвратила в 1995 г. нелегальные сделки на сумму более 100 млн долларов.

- Медицинская диагностика

В медицине нейронные сети применяются для анализа рентгеновских снимков (КТ, МРТ, флюорография и др.). Зачастую, заболевание на ранних стадиях невозможно определить человеческим глазом, однако эту проблему решают нейронные сети. Несложно привести пример в современных реалиях, для выявления какого заболевания можно применить нейронные сети (*COVID-19*). Нейронные сети также применяются для облегчения работы врачей. Например, для диагностики слуха у грудных детей опытному врачу требуется около часа, в то время как нейронной сети всего несколько минут.

- ChatGPT

GPT (англ. Generative Pre-trained Transformer, рус. Генеративный Предобученный Трансформер) – инновационный алгоритм обработки естественного языка, опубликованный в 2018 году компанией OpenAI. Нейронная сеть способна запоминать и анализировать закономерности во входных данных, что позволяет ей генерировать тексты, трудноотличимые от написанных человеком. Благодаря тому, что используются новые методы обучения, нейронная сеть может анализировать многие языки мира.

ChatGPT – чат-бот, запущенный в конце 2022 года также компанией OpenAI. Функционал программы огромен: от общения в реальном времени и генерации программного кода с его анализом до написания музыки, стихов, эссе.

Нейронные сети применяются также при оценке стоимости недвижимости, исследовании спроса, анализе потребительского рынка и во многих других сферах повседневной жизни.

Глава 2 ПРИМЕНЕНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ В ЗАДАЧАХ ЗАЩИТЫ ИНФОРМАЦИИ

Различные методы машинного обучения представляют собой интересную область исследований с огромным потенциалом применения. Одной из сфер является криптография. Ещё в 1991 году Рональд Ривест (Ronald Rivest), один из создателей криптосистемы *RSA*, в своём докладе на конференции *ASIACYPT'1991* рассказал о связи машинного обучения и криптографии. Это послужило началом для развития данной области.

Можно выделить две основные области применения машинного обучения в криптографии:

1) создание криптосистем на основе машинного обучения,
2) проведение криптоанализа систем, используя машинное обучение.

2.1 Применение машинного обучения в криптографии

Рассмотрим варианты использования машинного обучения в криптографии для создания новых алгоритмов и систем.

В 2005 году Кляйн (E. Klein) представил новое явление, названное взаимным обучением [26]. Явление взаимного обучения может быть использовано в криптографии. Взаимное обучение может помочь двум сторонам коммуникации создать общий секретный ключ через открытый канал. Обе стороны имеют преимущество перед атакующим в том, что атакующий будет обучаться односторонне, что сделает практически невозможным создание такого же секретного ключа, который будет у обеих сторон.

В 2007 году Блум (Blum) обсудил возможности соединения машинного обучения и криптографии [14]. В статье говорится, что на техническом уровне существуют тесные связи между методами машинного обучения и методами, используемыми в криптографии. В качестве примера была рассмотрена техника "*Boosting*", которая является методом машинного обучения, предназначенным для извлечения как можно большей мощности из алгоритма обучения. Это может быть связано с методами усиления криптосистем.

В 2009 году Аль-Шаммари (Al-Shammari) и Зинсир-Хейвуд (Zincir-Heywood) представили новый метод для классификации зашифрованного трафика, который основан на машинном обучении [11]. Работа была сфокусирована на потоке без использования широко используемых характеристик, таких как IP-адреса, номера портов и информация о полезной нагрузке. Результаты исследования показали, что алгоритм обучения C4.5 превзошел другие алгоритмы, такие как, Naive Bayesian, support vector machine и AdaBoost.

В 2014 году Сагар (Sagar) и Кумар (Kumar) представили алгоритм шифрования с симметричным ключом, основанный на сети встречного распространения [25]. Предложенный алгоритм представляет собой симметричный блочный шифр, в котором данные преобразуются в биты и пропускаются через нейронную сеть в качестве текста, а в качестве выхода рассматривается результирующий результат процесса обучения “без учителя”. Несмотря на то, что принцип работы не был четко изложен в статье, это направление может быть изучено в дальнейшем.

В 2015 году Бост (Bost) представил три основных протокола классификации, которые опираются на машинное обучение при классификации зашифрованных данных [21]. В результате работы были получены классификаторы, сохраняющие конфиденциальность. Предложенные классификаторы были протестированы на медицинских наборах данных и доказали, что достигают правильной классификации с удовлетворительной эффективностью.

2.1.1 Генератор псевдослучайных чисел, основанный на обучении с подкреплением

Обучение с подкреплением – метод МО, в котором система (агент, agent) обучается в процессе взаимодействия со средой (environment), получая вознаграждения или наказания.

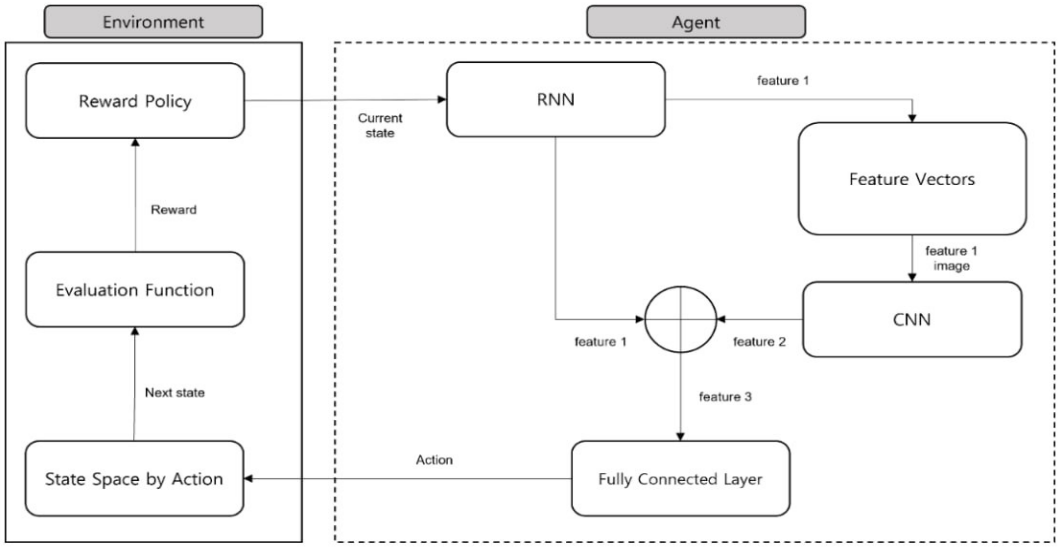

Рисунок 5 – Структура генератора

Рассмотрим структуру генератора (Рисунок 5), которая описана в [23]. Среда состоит из 3 модулей: пространство состояний по действию, функция оценки и политика вознаграждения. Пространство состояний по действию преобразует действия, производимые агентом, в следующее состояние. Функция оценки преобразует случайность следующего состояния (случайное число) в баллы, которые потом используются в качестве награды. Награда состоит из частоты и веса: частота планирует награду, а вес указывает на экспоненциальный рост. Политика вознаграждения увеличивает вес вознаграждения экспоненциально. Агент состоит из RNN, векторов с признаками (вектор-признаки), CNN и

полносвязного слоя. RNN – это слой, на который подаётся текущее значение состояния среды, на выходе получается значение первого признака. Задача модуля с вектор-признаками – это сохранение первых признаков по времени и 6 преобразование их в изображение. CNN – это слой, на который подаются вектор-признаки, на выходе имеем второй признак. Полносвязный слой собирает два полученных признака и преобразует их в действия.

Среда. В среде действия агента преобразуются в следующее состояние через пространство состояний по действию, затем определяется награда с помощью функции оценки. После этого агент обновляет параметры наград. Когда в некоторый момент времени T эпизод заканчивается, время увеличивается: $T = T + 1$, а следующее состояние определяется первым случайным числом. Данные первые случайные числа используются для указания текущего состояния агента.

Агент. В модуле агента случайное число начального состояния подаётся на вход RNN, генерируется первый признак. Полученный первый признак сохраняется по времени в виде изображения с помощью вектор-признаков. Изображения, введённые в CNN, преобразуются во второй признак. Полученные значения первого и второго признака собираются с помощью полносвязного слоя и преобразуются в третий. В качестве RNN используется модель LSTM – структура, состоящая из управляющего элемента и входного элемента, которые полносвязны. Входное значение LSTM – состояние предыдущего случайного числа – является вектором 1×8. Функция активации состоит из *tanh* и *relu.*

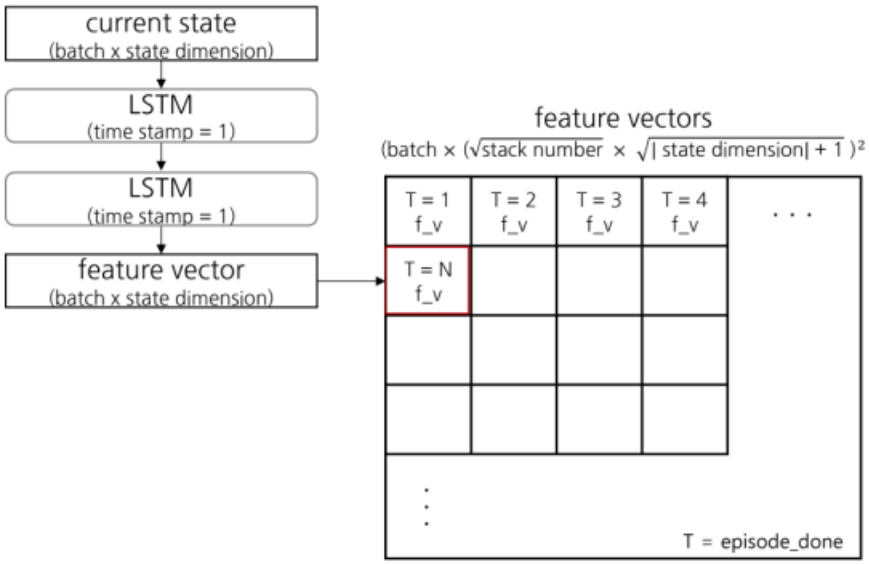

Рисунок 6 – Алгоритм генерации вектор-признаков

Агент генерирует до 8 бит случайных чисел за раз, процесс повторяется 100 раз для получения 800 бит в общей сложности. Как показано на Рисунке 6, используется 2 слоя LSTM для преобразования текущего состояния в значение первого признака, равное 8 бит. Структура модели CNN для долговременной памяти выглядит следующим образом. Входное значение имеет размер 30×30, размер фильтра равен 3×3, значение stride равно 2, функция активации – *relu*.

После генерации второго вектор-признака, оба вектора подаются на вход полносвязной сети для определения дальнейшего действия агента. Сеть состоит из двух слоёв: слой критика и слой актёра. Критик предсказывает награду, затем актёр предсказывает действие. В результате агент выполняет данное действие (меняет своё состояние).

2.1.2 Генератор псевдослучайных чисел, основанный на бинарной рекуррентной нейронной сети

Модель пластичности, зависимая от времени спайка (spike-timing-dependent plasticity, STDP) – это вид синаптической пластичности, который регулирует силу связей в зависимости от относительного времени между активацией нейрона и входными потенциалами действия (спайками на входе) [6]. Согласно этой модели пластичности, если входной спайк следует в среднем непосредственно перед активацией нейрона, то такую связь нужно усилить, а если следует в среднем непосредственно после активации нейрона, то такую связь нужно ослабить. В результате таких действий увеличивается влияние тех входов, которые могут быть причиной активации нейрона, а влияние входов уменьшается, если они не связаны с активацией нейронов.

В предложенной в [17] модели используется anti-STDP – явление, при котором связи между нейронами ослабевают, если входной спайк следует перед активацией нейрона, и наоборот, усиливаются, если вход не влиял на нейрон. Т.е. anti-STDP препятствует установлению причинно-следственной связи между нейронами, которые срабатывают последовательно. Утверждается, что нейронная сеть, работающая в таком режим, переходит в хаотический режим и может обладать чрезвычайно длинными циклами, что позволяет их использовать для генерации случайных чисел.

Рассмотрим рекуррентную сеть, состоящую из N бинарных нейронов. Состояние сети в момент времени t описывается вектором активности $x(t) \in \{0,1\}^N$, где $x(t) = 1$ означает, что нейрон i активен в момент времени t, а $x(t) = 0$ означает, что нейрон неактивен. Связи между нейронами описываются синаптической матрицей W, где $w_{ij} \geq 0$ – связь от нейрона i к нейрону j. Связь от нейрона к самому себе запрещена.

Определим модель $(\mathcal{S}, \mu, f, \mathcal{U}, g)$ следующим образом:

$\mathcal{S}$ - пространство состояний, состоящее из $\frac{\mathrm{N!}}{\mathrm{k!(N-k)!}}$ элементов,

μ - равномерное распределение на (0,1), используется для генерации начального состояния (k активных нейронов),

f - функция перехода, определяет состояние сети в момент времени $t + 1$,

$\mathcal{U} = \{0,1\}$ - выходной алфавит,

$g(x_p) = x_p$ – функция выхода, x_p – нейрон чётности.

Функция перехода. Состояние сети в момент времени t описывается вектором активности x(t). Сеть переходит в x(t+1) двухэтапно.

1. Вычислить $\forall i\ h_i(t+1) = \left(\sum_{j=1}^{N} w_{ij}(t)x_j(t)\right) - T_i(t) - \max\left(x_i(t), x_i(t-1)\right)$, где $T_i(t)$ – порог нейрона i в момент времени t.

2. Вычислить $x(t+1) = kWTA(h(t+1))$, где $kWTA$ – функция, которая выбирает k нейронов с наибольшим значением $h_i(t+1)$ и делает их активными.

Функция пластичности.

$\Delta w_{ij} = -\eta_{anti-STDP}(x_i(t)x_j(t-1) - x_j(t)x_i(t-1))$, где $\eta_{anti-STDP}$ – константа, которая отвечает за снижение связи между нейронами i и j.

$T_i(t+1) = T_i(t) + \eta_{IP}(x_i(t) - \frac{k}{N})$, где η_{IP} – небольшой learning rate. Нейроны, активные в момент времени t, будут увеличивать свой порог, а неактивные – уменьшать.

Для генерации случайного числа используется устройство контроля чётности, которое производит выборку состояния сети для определения чётности числа активных нейронов в произвольном подмножестве сети в момент времени $t-2$. Сначала q нейронов выбираются в блок чётности. Этот блок определяет, было ли как минимум l активных нейронов в момент времени $t-1$ среди всех q нейронов.

Эта информация затем отбирается нейроном чётности с порогом 0 и входным вектором весов [1, −1, …,1, −1] длины q. Задержка в определении чётности группы нейронов в актуальной сети должна быть 2 момента времени, т.к. по одному требуется как для выборки нейронов, чтобы определить, сколько из них было чётных (блок чётности), так и для определения чётности количества активных нейронов в блоке (нейрон чётности). Таким образом, информация, содержащаяся в нейроне чётности, относится к состоянию сети за два момента времени до этого. Входными данными для нейрона чётности являются только нейроны в блоке чётности. Состояние нейрона чётности – выход генератора.

Формула для обновления нейронов в блоке чётности и нейрона чётности:

$$x_{k_l} = \begin{cases} 0, \left(\sum_{n=start}^{end} x_n\right) < l, \\ 1, \left(\sum_{n=start}^{end} x_n\right) \geq l \end{cases} \quad l = 1, \dots, q;$$

$$x_p = \left(\sum_{i=1}^{q} x_{k_i}\right) (mod\ 2).$$

2.1.3 Схема симметричного шифрования, основанная на нейронной сети

Данный алгоритм [15] основан на нейронной сети Хопфилда (Hopfield NN, Рисунок 7) – полносвязной рекуррентной нейронной сети с симметричной матрицей весов (связей).

Состояние нейрона i в момент времени $t+1$ зависит от текущих состояний:

$$S_i(t+1) = sign\left(\sum_{j=0}^{N-1} T_{ij} S_j(t)\right) \in \{0,1\}, i = 0,1, \dots, N-1,$$

где T – симметричная матрица, $t_{ij} \in \{-1,0,1\}$, N − количество нейронов.

Хопфилд доказал, что энергетическая функция данной сети монотонно убывает и сходится к одному из положений равновесия, называемых аттракторами. Также доказано, что при наложении определённых условий на

элементы матрицы связей T связь между начальным значением и результирующим аттрактором теряется, т.е. по начальному значению невозможно определить, в какое положение равновесия оно перейдёт. Положение равновесия (устойчивое состояние) $S(\infty)$ – такое состояние $S(l)$, что $\forall k > l, S(k) = S(l) \neq S(l-1)$.

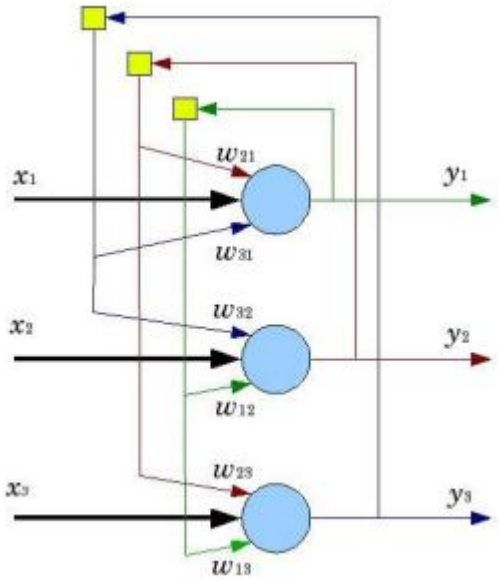

Рисунок 7 – Нейронная сеть Хопфилда (3 нейрона)

Множество всех аттракторов будем обозначать через $\{S^{\mu}\}$, область определения аттрактора (все начальные значения, которые сходятся к данному аттрактору) – через $\{A^{\mu}\}$.

Пусть H – случайная матрица перестановок. Определим новую матрицу связей $\hat{T} = HTH^{T}$. Оказывается, что связь между прежним $\{A^{\mu}\}$ и новым $\{\hat{S}^{\mu}\}$ теряется. Новые множества можно найти по формулам: $\{\hat{S}^{\mu}\} = \{S^{\mu}H^{T}\}$, $\{\hat{A}^{\mu}\} = \{A^{\mu}H^{T}\}$. Сложность нахождения H по известным множествам $\{\hat{A}^{\mu}\}$, $\{A^{\mu}\}$ – $O(N^{N-1})$.

Рассмотрим схему симметричного шифрования, основанную на полученных выше свойствах. Личные ключи: M – матрица кодирования (для представления входных данных в виде нужных векторов), H – матрица перестановок. Данные параметры выбираются случайным образом. Открытые ключи: T – матрица связей, $\{S^{\mu}\}$ – множество всех аттракторов для матрицы T.

Алгоритм зашифрования:

1. Вычислить новую матрицу связей $\hat{T} = HTH^{T}$ и новое множество аттракторов $\{\hat{S}^{\mu}\} = \{S^{\mu}H^{T}\}$.

2. Используя матрицу кодирования M, преобразовать открытый текст X в закодированный открытый текст $X_M \in \{\hat{S}^{\mu}\}$.

3. Используя качественный генератор псевдослучайных чисел, сгенерировать число $r \in \{0,1\}^{N}$ и использовать его как начальное значение $S(0)$ для нейронной сети с матрицей связей $\hat{T}$.

4. Вычислить устойчивое состояние $S(\infty)$ и сравнить с X_M. Если $S(\infty) = X_M$, то начальное значение $S(0)$ принадлежит области определения аттрактора X_M, поэтому шифртекст $Y = S(0)$. Если $S(\infty) \neq X_M$, то нужно вернуться к шагу 3.

Алгоритм расшифрования:

1. Вычислить новую матрицу связей $\hat{T} = HTH^T$.

2. Подать шифртекст на вход нейронной сети с матрицей связей $\hat{T}$. Получить кодированный открытый текст $X_M = S(\infty)$, вычислив устойчивое состояние $S(\infty)$.

3. С помощью матрицы кодирования M декодировать X_M и получить X.

2.1.4 Схема регенерации криптографического ключа на основе мультибиометрии

В данном разделе рассматривается схема регенерации криптографического ключа на основе мультибиометрии [18]. Она представляет собой многоинстанционную систему, объединяющую информацию от левой и правой радужных оболочек глаз человека. Джон Даугман (Daugman) показал, что левая и правая радужки человека не коррелируют и поэтому могут рассматриваться как два независимых источника биометрической информации. Извлекается отличительная информация из изображений радужной оболочки глаза пользователя в виде кодов радужной оболочки. Далее к изображению радужки применяются фильтры Габора, а квантованная фазовая информация используется для построения кода радужки. Затем коды радужки, полученные из изображений обоих глаз пользователя, объединяются, чтобы получить комбинированный мультибиометрический вектор признаков. Более того, предлагается новый подход к объединению информации в области признаков с помощью взвешенной коррекции ошибок, который помогает улучшить производительность. Наконец, вводится ключ перестановки (защищенный паролем), что позволяет повысить точность, конфиденциальность и безопасность системы.

Рассмотрим подробно предлагаемую мультибиометрическую систему регенерации ключей. Основная структура этой схемы аналогична схеме Хао (Hao) [16] (Рисунок 8). Хорошо известно, что коды радужки, полученные из различных изображений радужки одного и того же пользователя, содержат вариации, которые в данной работе называются ошибками. Существует два типа ошибок в кодах радужки:

- фоновые ошибки, вызванные шумом камеры, эффектами захвата изображения и т.д.,
- ошибки всплеска, которые являются результатом зеркальных отражений.

Для борьбы с этими ошибками используются коды с коррекцией ошибок (*ECC*).

Генерируется случайная битовая строка *K*, которая присваивается пользователю и затем кодируется с помощью кодов Рида-Соломона (*RS*), выход

которых далее кодируется кодами Адамара. Коды Адамара исправляют фоновые ошибки, а коды *RS* исправляют ошибки всплеска. Выход кодера называется псевдокодом *S*. Для того чтобы справиться с каскадной структурой двух *ECC*, количество битов в каждом символе *RS* и во входных словах кодов Адамара устанавливается равным 7 ($m = 7$). Эталонный код радужки I_{ref}(или, в общем случае, вектор биометрических признаков в двоичной форме) переписывается с помощью операции *XOR* с *S* для получения заблокированного шаблона кода радужки I_{lock}. На этапе регенерации ключа другой код радужной оболочки I_{test} (тестовый код радужной оболочки) переписывается с помощью операции *XOR* с I_{lock}. Эти операции через *XOR* переносят ошибки в кодах радужки на псевдокод. Если количество ошибок находится в пределах возможностей коррекции ошибок *ECC*, ошибки исправляются частью декодера *ECC* и регенерируется ключ K', который совпадает с *K*. Если ошибок больше, регенерированный ключ $K' \neq K$.

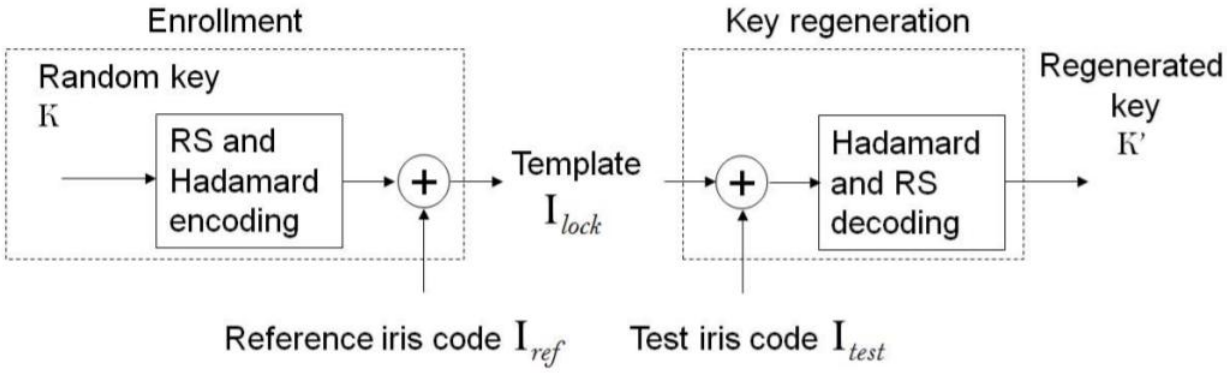

Рисунок 8 – Схема системы регенерации криптографического ключа на основе биометрии

В схеме *ECC* существует два уровня коррекции ошибок: на первом уровне коды Адамара исправляют до $2^{k-1} - 1$ ошибок в 2^k-битном блоке. Если блок имеет больше, чем $2^{k-1} - 1$ ошибок, этот блок декодируется неправильно и приводит к ошибке. Второй уровень *ECC* состоит из *RS*-кодов. Выход этапа декодирования Адамара служит входом для этапа декодера *RS*. *RS*-коды исправляют ошибки, вызванные неправильным декодированием кодами Адамара, и генерируют ключ K'.

2.1.5 Автоматическая генерация синтетических отпечатков пальцев

Отпечатки пальцев человека часто используются в различных приложениях для аутентификации и идентификации, начиная от пропускной системы и заканчивая подтверждением платежей в мобильном банкинге. Для создания и тестирования надёжной системы на основе отпечатков пальцев, необходимо иметь в распоряжении огромную базу данных отпечатков. Однако на практике получение огромного массива изображений отпечатков пальцев

сопряжено с большими затратами. Во многих случаях исследовательские группы, разрабатывающие системы аутентификации по отпечаткам пальцев, не имеют доступа к большой общедоступной базе данных. Поэтому производительность таких систем напрямую зависит от имеющихся данных (от их качества и количества).

Стоит учитывать тот факт, что при сборе большого массива реальных отпечатков пальцев нужно решить проблемы с конфиденциальностью и безопасностью. Собранная база данных отпечатков ни в коем случае не должна попасть в руки злоумышленников, ведь они могут использовать её для обмана иных систем аутентификации. Поэтому задача создания синтетических отпечатков пальцев имеет огромное значение в современном мире.

Синтетические отпечатки пальцев решают проблему доступности, т.к. их можно генерировать в любом количестве. Более того, они генерируются искусственно, поэтому не содержат никакой конфиденциальной информации о человеке.

Предыдущие решения по созданию синтетических отпечатков пальцев были способны либо создавать синтетические шаблоны микроособенностей отпечатков пальцев, либо синтетические изображения реальных отпечатков пальцев в низком разрешении. Решения, основанные на математических моделях отпечатков пальцев, страдают от недостатка энтропии и обобщения до точного распределения вероятностей реальных отпечатков пальцев.

Предыдущие решения, основанные на моделях глубокого обучения, также не могут создавать высококачественные изображения, потому что нет доступа к большому объёму реальных отпечатков пальцев для обучения этих моделей.

Однако была разработана система SYNFI – новая модель для генерации высококачественных синтетических отпечатков пальцев [24]. Данная модель разбивает основную задачу на две подзадачи, решаемые с помощью GAN и суперразрешения (SR). Для корректной работы SYNFI должна удовлетворять следующим требованиям:

- генерируемые изображения должны сохранять миниатюрные характеристики отпечатков пальцев, используемых в системах аутентификации, например, структуру гребней, раздвоения и окончания гребней;
- система должна генерировать полные отпечатки пальцев;
- сгенерированные изображения должно быть трудно отличить от реальных изображений;

- система должна быть полностью автоматизирована, не требуя ручной разработки признаков, и иметь высокую масштабируемость.

Основные этапы работы SYNFI:

1) Генерация синтетических отпечатков низкого качества с помощью GAN. Большинство отпечатков имеют разрешение 256×256. Однако процедура обучения GAN опирается на изображения с меньшим разрешением. Поэтому для обучения GAN используется база данных низкого качества (LQD). Однако модель SR нуждается в базе данных высокого качества (HQD). Поэтому сперва обучается GAN для создания низкокачественных синтетических отпечатков пальцев.
2) Обобщение на высококачественные изображения. На втором этапе SYNFI низкокачественное изображение преобразуется в изображение высокого разрешения с детальной структурой.

2.2 Применение нейронных сетей в криптоанализе

Нейронный криптоанализ – использование глубокого обучения для проведения атак на криптографические структуры (системы, алгоритмы и т.д.).

С увеличением вычислительной мощности нейронный криптоанализ становится более реальным вариантом для проведения атак на более сложные шифры.

2.2.1 Различные атаки с использованием машинного обучения

Наиболее развитой областью нейроприкладного криптоанализа является атака по сторонним каналам. Противник получает физические свойства, такие как уровень мощности сигнала на целевой аппаратной реализации криптосистемы. Глубокое обучение используется для более эффективного анализа таких метрик, как временные характеристики и информация об амплитуде мощности, которые могут быть получены от аппаратного обеспечения.

Методы машинного обучения применяются для проведения атак по сторонним каналам. В 2011 году Господар (Hospodar) представил первое исследование, в котором предлагалось использовать машинное обучение для проведения атак на сторонние каналы. Описанная система основана на алгоритме LS-SVM (метод наименьших квадратов опорных векторов), а в качестве стороннего канала использовалось энергопотребление. Целью атаки была реализация криптосистемы AES. Исследование показало, что выбор параметров алгоритма машинного обучения сильно влияет на результаты [22].

В 2012 году Алани (Alani) представил атаку "известный текст", в которой он обучал нейронную сеть для нахождения открытого текста по шифртексту

без ключа шифрования [10]. Данная атака позволила значительно сократить время и количество необходимых пар "открытый текст-шифртекст". Атаке подвергались криптосистемы DES и Triple-DES.

В 2014 году Лерман (Lerman), Бонтемпи (Bontempi) и Маркович (Markowitch) опубликовали исследование, в котором также проводили атаки на сторонние каналы с использованием машинного обучения [19]. Авторы статьи предложили использовать методы машинного обучения для повышения точности атак на сторонние каналы. Поскольку атаки на сторонние каналы опираются на физические измерения аппаратных реализаций криптосистем, всегда существуют определенные параметрические предположения, на которые опираются эти атаки. Предлагаемое использование машинного обучения позволяет ослабить такие допущения и работать с векторами признаков высокой размерности.

В 2015 году было опубликовано исследование по анализу зашифрованного сетевого трафика Android [12]. Целью анализа являлось определение действий пользователя, которые передаются в зашифрованном виде. Противник перехватывает (прослушивает) зашифрованный трафик, а затем пытается классифицировать его с помощью самых мощных методов машинного обучения. В результате, система верно классифицировала действия пользователя в 95% случаях.

2.2.2 Криптоанализ блочного шифра

Далее рассмотрим несколько сценариев атак на блочные шифры с помощью нейронного криптоанализа, которые описаны в [13].

Большинство атак на блочные шифры осуществляется с точки зрения “чёрного ящика”, т.е. предполагается, что противник знает все спецификации алгоритмов блочных шифров, но не знает секретный ключ и соответствующие тактовые ключи.

Такие атаки называются атаками на восстановление ключа.

- Получив пару открытого текста и шифртекста (p, c), удовлетворяющую $c = Enc(k, p)$, атакующий пытается найти k. В качестве обучающих данных даются тройки из случайного открытого текста, случайного ключа и соответствующего шифртекста (k_i, p_i, c_i) такого, что $c_i = Enc(k_i, p_i)$.
- Атака на восстановление ключа с фиксированным ключом аналогична атаке на восстановление ключа, однако в качестве обучающих данных даются только пары случайных открытых текстов и соответствующих шифртекстов (p_i, c_i), таких, что $c_i = Enc(k, p_i)$. Отдельная модель глубокого обучения не может быть сквозным решением, поскольку ключ не предоставляется в обучающих данных. Поэтому атака обычно

интегрируется с методом анализа, который требует знания алгоритма. Стандартный подход заключается в угадывании ключа и исследовании работы модели для каждого возможного ключа.

Атаки на эмуляцию (подражание) шифра – это попытки имитировать алгоритм шифрования или дешифрования целевого шифра.

- Атака восстановления открытого текста. Учитывая шифртекст c, атакующий пытается угадать биты соответствующего открытого текста p так, чтобы $c = Enc_k(p)$ с неотрицательным преимуществом. Случайные пары открытого текста и соответствующего шифртекста (p_i, c_i) такие, что $\mathrm{c}_i = Enc(k_i, p_i)$, даются в качестве обучающих данных.
- Атака побитового восстановления открытого текста. Атака восстановления открытого текста может быть также развернута по отдельным битам открытого текста. В качестве обучающих данных вместо (p_i, c_i) даются пары $(p_i[k], \mathrm{c}_i)$, где $p_i[k]$- k-й бит открытого текста p_i.
- Атака эмуляции шифрования. Учитывая открытый текст p, атакующий пытается угадать биты соответствующего шифртекста c так, чтобы $c = Enc_k(p)$ с неотрицательным преимуществом. Случайные пары открытого текста и соответствующего шифртекста (p_i, c_i) такие, что $\mathrm{c}_i = Enc(k_i, p_i)$, даются в качестве обучающих данных.

Глава 3 ХЭШ-ФУНКЦИИ И НЕЙРОННЫЕ СЕТИ

В последнее время большое внимание уделяется исследованию различных хэш-функций. Возникает необходимость разработки принципиально новых алгоритмов создания хэш-функций. Одним из таких направлений является использование нейронных сетей. Нейронные сети обладают свойствами перемежения и диффузии, что уже используется для проектирования потоковых и блочных шифров. Фактически, нейронная сеть обладает свойством однонаправленности, т.к. по входам несложно вычислить выход нейронной сети, а по выходу тяжело восстановить все входы нейронной сети. Все эти свойства позволяют использовать нейронную сеть для создания хэш-функции.

В данной главе будет рассмотрено практическое применение нейронных сетей для создания хэш-функций.

3.1 Хэш-функции: применение и свойства

Хэш-функция *h(x)* – это функция, которая принимает в качестве аргумента последовательность (строку) *M* произвольной длины и которая выдаёт в качестве результата последовательность (строку) *h(M)* некоторой фиксированной длины. Полученный результат *h(M)* для последовательности *M* называется хэш-образом (хэш-значением). Длины последовательностей *M* и *h(M)* могут соотноситься по-разному, т.е. любое из условий имеет место:

$$|M| = |h(M)|,\ |M| > |h(M)|,\ |M| < |h(M)|,$$

где $|x|$ – длина последовательности (строки) *x*.

Т.е. *функция хэширования (хэш-функция) – это отображение* $h: \{0,1\}^* \to \{0,1\}^n$*, действие которого задаётся общедоступным полиномиальным алгоритмом* [1].

Можно выделить несколько видов хэш-функций в зависимости от цели и области их применения [1]:

- Контрольные суммы - алгоритмы, используемые для защиты данных от непреднамеренных искажений, в том числе — от ошибок аппаратуры. Преимуществом контрольных сумм является быстрая скорость исполнения и простота реализации. К недостаткам стоит отнести отсутствие стойкости к атакам и коллизиям.
- Геометрическое хэширование – алгоритмы, необходимые для решения задач в двумерном и трехмерном пространствах. Хэш-функция в данном методе обычно получает на вход какое-либо метрическое пространство и разделяет его, создавая сетку из клеток. Геометрическое хэширование применяется в телекоммуникациях при работе с многомерными сигналами.
- Хэш-таблица - структура данных, реализующая ассоциативный массив, позволяющая хранить ключ и значение, выполнять следующие операции: добавление нового ключа и значения, поиска и удаления пары по ключу. Хэш-таблицы применяются с целью ускорения поиска

в базах данных и языковых процессорах для обслуживания таблиц идентификаторов.

- Криптографическая хэш-функция – особый класс хэш-функций, алгоритмы которого удовлетворяют дополнительным требованиям криптостойкости, что позволяет использовать функции этого класса в задачах криптографии.

Любая хэш-функция должна удовлетворять следующим условиям:

- преобразование входных данных любой длины в фиксированную длину;
- однонаправленность (односторонность) – по результату практически невозможно восстановить входные данные;
- стойкость к коллизиям – не должна выдавать одинаковых значений при разных входных данных;
- требует малое количество вычислительных ресурсов;
- должна обладать “лавинным эффектом”, то есть при малейшем изменении входных данных результат должен существенно изменяться.

Любая криптографическая хэш-функция *h(x)* должна удовлетворять следующим требованиям [1]:

- Хэш-значение должно зависеть от всех битов прообраза и от их взаимного расположения;
- При любом изменении входной информации хэш-значение должно изменяться непредсказуемо, иначе говоря, в среднем должна измениться половина битов хэш-значения (каждый бит может измениться с вероятностью *1/2*);
- При заданном значении M задача нахождения хэш-значения *h(M)* должна быть вычислительно разрешима ($\forall M$ легко вычисляется значение *h(M)*);
- По заданному Y = *h(M)* задача нахождения M должна быть вычислительно трудной (свойство однонаправленности (или односторонности));
- Для заданного значения M с хэш-значением *h(M)* задача нахождения значения $M' \neq M$, т. ч. $h(M) = h(M')$ должна быть вычислительно трудной (в таком случае функция h называется свободной от коллизий);
- Задача нахождения двух произвольных сообщений $M_1 \neq M_2$, т. ч. $h(M_1) = h(M_2)$ должна быть вычислительно трудной (в таком случае функция h называется строго свободной от коллизий).

3.2 Создание хэш-функции на основе нейронной сети

3.2.1 Архитектура нейронной сети

В качестве базовой архитектуры нейронной сети для создания хэш-функции воспользуемся описанной в [20] архитектурой (Рисунок 9). Данная

нейронная сеть состоит из трёх слоёв: входной, скрытый и выходной. Они реализуют перемежение, диффузию и сжатие данных. Обозначим входы и выходы слоёв так: $P = [P_0, P_1, \dots, P_{31}], C = [C_0, C_1, \dots, C_7], D = [D_0, D_1, \dots, D_7], H = [H_0, H_1, H_2, H_3]$. Тогда выход нейронной сети определяется уравнением:

$$H = f_2(W_2 D + B_2) = f_2(W_2 f_1(W_1 C + B_1) + B_2)$$
$$= f_2(W_2 f_1(W_1 f_0(W_0 P + B_0) + B_1) + B_2)$$

где f_i, W_i, B_i $(i = 0,1,2)$ – функция активации, весовые коэффициенты и смещение i-го слоя соответственно.

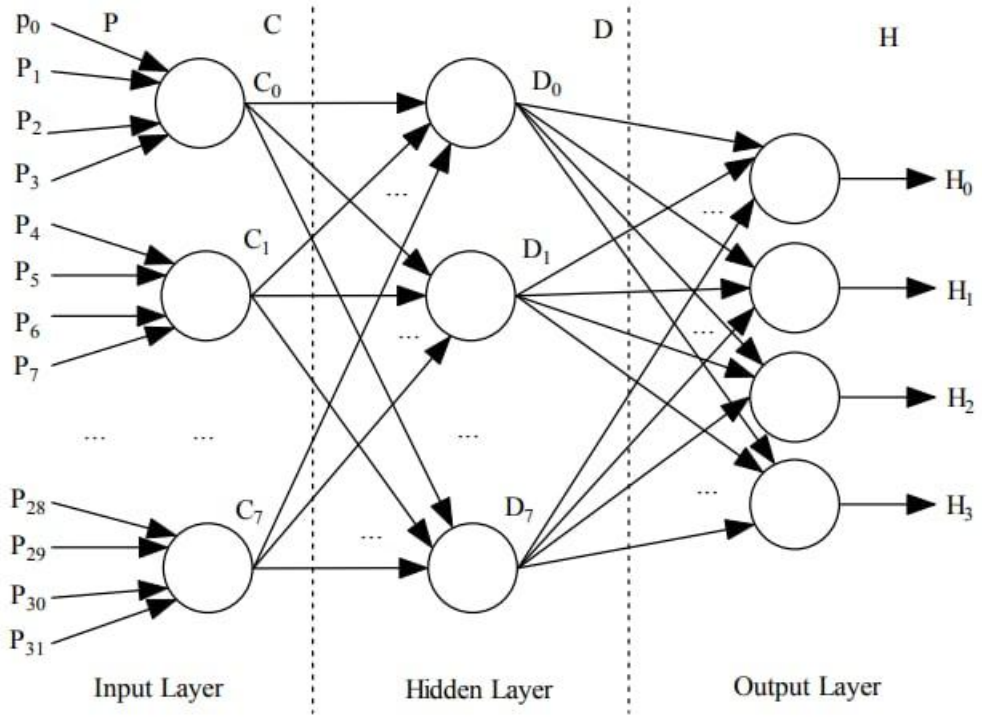

Рисунок 9 – Структура используемой нейронной сети

В качестве f_i используется кусочно-линейное хаотическое отображение, которое определяется следующим образом:

$$Y = f(X, Q) = \begin{cases} X/Q, 0 \le X < Q \\ (X - Q)/(0.5 - Q), Q \le X < 0.5 \\ (1 - X - Q)/(0.5 - Q), 0.5 \le X < 1 - Q \\ (1 - X)/Q, 1 - Q \le X \le 1 \end{cases}$$

где Q – управляющий параметр ($0 < Q < 0.5$). Данное отображение кусочно-линейное, а при условии $0 < Q < 0.5$ ведёт себя хаотически. Таким образом, хаотическое отображение f обладает необходимыми свойствами, подходящими для создания хорошего алгоритма шифрования, такими как чувствительность к

начальным данным и чувствительность к параметру Q. Если данное отображение повторить T (T достаточно велико) раз, то малые изменения в начальном значении X или параметре Q повлекут значительные отличия в итоговом значении Y. В большинстве случаев, после повторения хаотического отображения T ($T \geq 50$) раз получается случайный выход.

Теперь перейдём к рассмотрению слоёв нейронной сети. Вычисление выхода первого слоя (входного) показано на Рисунке 10. Тут $W_0 = [w_{0,0}, \dots, w_{0,31}]$ – матрица весовых коэффициентов входного слоя, B_0 – вектор-столбец смещений входного слоя размерности 8×1.

$$C = f^T\left(\begin{bmatrix} \sum_{i=0}^{3} w_{0,i}P_i + b_{0,0} \\ \sum_{i=4}^{7} w_{0,i}P_i + b_{0,1} \\ \vdots \\ \sum_{i=28}^{31} w_{0,i}P_i + b_{0,7} \end{bmatrix}, Q_0\right) = \begin{bmatrix} f^T(\sum_{i=0}^{3} w_{0,i}P_i + B_{0,0}, Q_0) \\ f^T(\sum_{i=4}^{7} w_{0,i}P_i + B_{0,1}, Q_0) \\ \vdots \\ f^T(\sum_{i=28}^{31} w_{0,i}P_i + B_{0,7}, Q_0) \end{bmatrix} = \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_7 \end{bmatrix}$$

Рисунок 10 – Выход входного слоя

Аналогичным образом можно определить выходы двух других слоёв нейронной сети (Рисунок 11).

$$D = f_1(W_1C + B_1) = f(W_1C + B_1, Q_1) = \begin{bmatrix} f(\sum_{i=0}^{7} w_{1,0,i}C_i + B_{1,0}, Q_1) \\ f(\sum_{i=0}^{7} w_{1,1,i}C_i + B_{1,1}, Q_1) \\ \vdots \\ f(\sum_{i=0}^{7} w_{1,7,i}C_i + B_{1,7}, Q_1) \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ \vdots \\ D_7 \end{bmatrix}.$$

$$H = f_2(W_2D + B_2) = f^T(W_2C + B_2, Q_2) = \begin{bmatrix} f^T(\sum_{i=0}^{7} w_{2,0,i}D_i + B_{2,0}, Q_2) \\ f^T(\sum_{i=0}^{7} w_{2,1,i}D_i + B_{2,1}, Q_2) \\ \vdots \\ f^T(\sum_{i=0}^{7} w_{2,3,i}D_i + B_{2,3}, Q_2) \end{bmatrix} = \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_3 \end{bmatrix}.$$

Рисунок 11 – Выходы скрытого и выходного слоя

Здесь W_1 – матрица весовых коэффициентов скрытого слоя размерности 8×8, B_1 – вектор-столбец смещений скрытого слоя размерности 8×1, W_2 – матрица весовых коэффициентов выходного слоя размерности 4×8, B_2 – вектор-столбец смещений выходного слоя размерности 4×1. Цель скрытого слоя –

диффузия изменений в *C* с изменениями в *D*. Повторение *T* раз отображения *f* увеличивает хаотичность связей между входами и выходами слоёв (а таким образом и между *P* и *H*), что укрепляет криптосистему. Т.к. на вход *f* подаётся значение из [0,1], то все операции сложения производятся по модулю 1 (Рисунок 12).

$$a \bmod 1 = \begin{cases} a, & 0 \le a < 1 \\ a-1, & 1 \le a < 2 \end{cases}.$$

Рисунок 12 – Операция сложения по модулю 1

3.2.2 Хэширование одного блока

Хэш-функция на основе данной нейронной сети, требующая на вход блок фиксированной длины, показана на Рисунке 13. Входной блок *P*, содержащий 1024 бита, преобразуется в хэш-значение *H*, содержащее 128 бит, с использованием пользовательского ключа *K*. Входной блок делится на 32 части по 32 бита, каждая часть преобразуется в целое число, которое потом преобразуется в вещественное число из [0, 1] путём деления на $2^{32} - 1$. В итоге получается 32 вещественных числа, которые являются входом нейронной сети. На выходе из нейронной сети получается 4 вещественных числа, из которых получают по 32 бита, объединяя в итоговое хэш-значение. Таким образом, открытый текст *P*, содержащий 1024 бита, преобразуется в хэш-значение *H*, содержащее 128 бит.

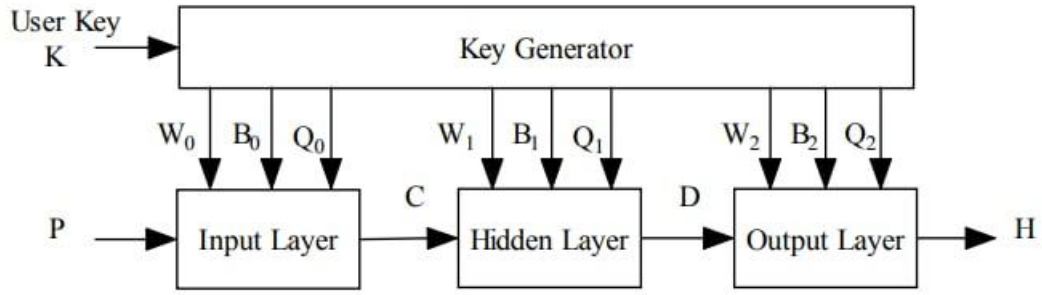

Рисунок 13 – Блочное хэширование (BlockHash)

Ключ пользователя K используется для генерации параметров $W_0, B_0, Q_0, W_1, B_1, Q_1, W_2, B_2, Q_2$. Всего необходимо сгенерировать 151 значение. Для этого разделим ключ $K = k_0 k_1 \dots k_{127}$ на 4 части: $K_0 = k_0 k_1 \dots k_{31}, K_1 = k_{32} k_{33} \dots k_{63}, K_2 = k_{64} k_{65} \dots k_{95}, K_3 = k_{96} k_{97} \dots k_{127}$. Генерировать i-тое значение $p_i, (i = 0,1, \dots, 150)$ будем следующим образом:

$$\begin{cases} X_0(i) = f^{T+i}(K_0, K_1), \\ X_1(i) = f^{T+i}(K_2, K_3), \\ p_i = \big(X_0(i) + X_1(i)\big) \, mod \, 1. \end{cases}$$

3.2.3 Многоблочное хэширование

Для блочного хэширования необходим исходный текст фиксированной длины в 1024 бит. Чтобы получить хэш-значения для входного текста произвольной длины рассмотрим многоблочное хэширование. Исходный текст

M делится на n частей $M_0, \dots, M_{n-1}$ по 1024 бит. Если в последней части не хватает битов, то к ней добавляется один единичный бит и необходимое количество нулевых битов. Получение хэш-значения для всего сообщения показано на Рисунке 14, где *BlockHash* – блочное хэширование.

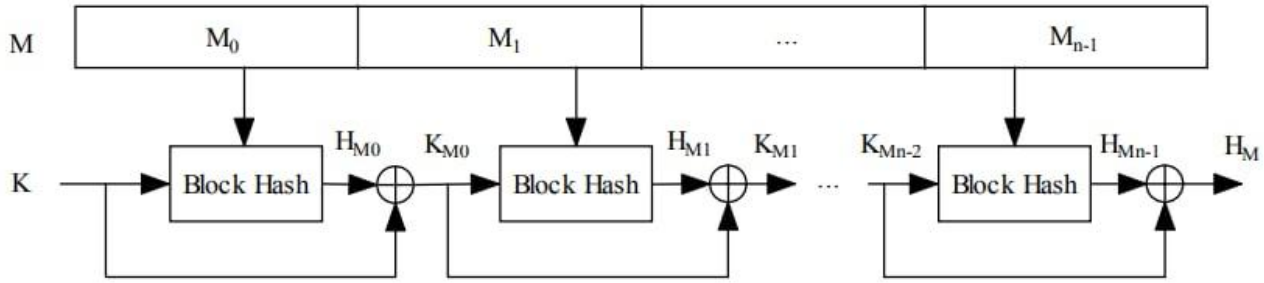

Рисунок 14 – Многоблочное хэширование (Multi-Block Hash)

Для получение итогового хэш-значения всего сообщения применятся операция “круглый плюс” (*XOR*, сложение по модулю 2): $H_M = K_{M_{n-2}} \oplus H_{M_{n-1}} = \left(K_{M_{n-3}} \oplus H_{M_{n-2}}\right) \oplus H_{M_{n-1}} = \dots = \left(K \oplus H_{M_0}\right) \oplus H_{M_1} \oplus \dots \oplus H_{M_{n-1}}$.

Заметим, что возможны два режима работы описанной системы. Если ключ K – секретный параметр, то система работает в режиме имитозащиты, основанной на хэш-функции (*HMAC*), если K – общедоступный параметр (закреплён в стандарте или лежит в открытом доступе), то система работает в режиме криптографической хэш-функции.

Глава 4 АНАЛИЗ СТОЙКОСТИ ХЭШ-ФУНКЦИИ

4.1 Свойство однонаправленности

Однонаправленность – это высокая алгоритмическая сложность построения прообраза хэш-функции по её значению. Согласно формуле (1), в предложенной хэш-функции выходной вектор является легко вычисляемым значением при известном значении входного вектора P и значении ключа K, однако выполнение обратной операции при известном H является вычислительно сложной задачей. Данное свойство обусловлено тем, что задача нахождения множителей при знании только результата произведения не является тривиальной и, также, на всех слоях функцией активации является хаотическое отображение, а на входном и выходном слое оно повторяется T раз.

Рассмотрим выход первого слоя:

$$C_j = f^T\left(\sum_{i=4j}^{4j+3} w_{0,i}P_i + b_{0.j}, Q_0\right) = f^T(Z_j, Q_0), j = \overline{0,7}.$$

Предположим, что нам известны все значения Z_j. Попытаемся определить P_i по известным значениям Z_j при условии, что параметры $w_{0,i}$ и $b_{0.j}$ неизвестны.

Можно применить два метода: атака “грубой силой” и атака при выбранном открытом тексте. Для атаки грубой силой понадобится перебрать 2^{128} вариантов, что на сегодняшний момент займёт значительное время.

Для атаки при выбранном открытом тексте нужно иметь в распоряжении как минимум 5 пар $\{P, Z\}$. Это позволит составить систему сравнений по модулю 1. Но эту систему нужно решить. Т.е. данная атака имеет место при известных Z_j. Однако для этого нужно получить Z_j из C_j. Благодаря использованию хаотического отображения и его повторении T раз, потребуется перебрать как минимум $4^T (\geq 2^{100}$ при $T \geq 50)$ вариантов для получения Z_j из C_j, что является довольно трудной задачей. Для двух других слоёв можно провести аналогичные рассуждения.

Таким образом, можно сделать вывод, что полученная хэш-функция является односторонней.

4.2 Атаки

Можно выделить 2 типа атак на данную хэш-функцию:

1) атаки с целью нахождения ключа;
2) атаки с целью поиска коллизий.

Для атак второго типа знание ключа является необходимым условием, т.к. невозможно проводить поиск коллизий (вычислять хэш-значения для поддельных сообщений), не зная ключа, на котором получен хэш для исходного сообщения.

4.2.1 Атаки с целью нахождения ключа

Атака при известном тексте – противнику известно некоторое количество корректных пар $(X_i, h(X_i, K))$.

Атака при выбранном тексте – противник имеет возможность получать корректные пары $(X_i, h(X_i, K))$ для заранее выбранных X_i.

Атака при выбираемом тексте – противник имеет возможность получать корректные пары $(X_i, h(X_i, K))$ для любых X_i, которые могут выбираться на основе предыдущих запросов.

Основная цель описанных атак – нахождение общего личного ключа сторон либо построение алгоритма хэширования любого нового сообщения X' без знания ключа K по известным результатам $(X_i, h(X_i, K))$.

В настоящее время неизвестно, возможно ли осуществить эти атаки.

Атака "грубой силой" – атака с целью нахождения неизвестного ключа K. Противнику известна как минимум одна пара $(X_i, h(X_i, K))$. Далее он последовательно перебирает ключи и при успешном поиске проводит проверку найденного ключа на нескольких парах $(X_i, h(X_i, K))$. Трудоёмкость атаки – $O(2^{128})$.

4.2.2 Атака "грубой силой"

Данная атака может применяться также с целью нахождения неизвестного сообщения X по известному хэш-значению $h(X, K)$. Противник ищет любое сообщение $X' \neq X$, но $h(X, K) = h(X', K)$. Трудоёмкость атаки – $O(2^{128})$, что является трудной задачей для современных вычислительных устройств.

4.2.3 Атака "дней рождения"

Атака "дней рождения" применяется для нахождения двух различных сообщений с одинаковым хэш-значением. В основе атаки лежит парадокс "дней рождения": вероятность совпадения дня рождения у 23 случайно выбранных людей больше *1/2*. Сложность данной атаки равна $O(\sqrt{n})$, где n – длина хэш-значения.

В нашем случае длина хэш-значения – 128 бит. Поэтому сложность атаки равна $O(2^{64})$, что является потенциальной опасностью в современных реалиях. Поэтому необходимо увеличить длину хэш-значения. Заметим, что длину хэш-значения легко увеличить до 256 или 512 бит, что значительно усложнит атаку. Если увеличить количество нейронов в выходном слое до 8, то хэш-значение будет длиной 256 бит. Если увеличить длину исходного текста до 2048 бит и удвоить количество нейронов на каждом слое, то выходной хэш будет длины 512 бит.

4.2.4 Атака "встреча посередине"

При атаке "встреча посередине" для исходного текста $M = M_0 \dots M_{n-2} M_{n-1}$ ищется коллизия $M' = M_0 \dots M_{n-2} M'_{n-1}$ (Рисунок 15).

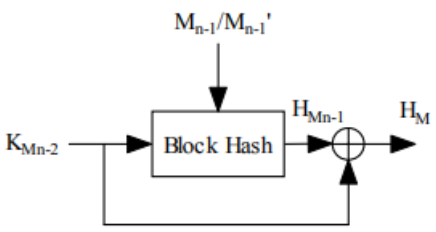

Рисунок 15 – Атака “встреча посередине”

Т.к. ключ $K_{M_{n-2}}$ неизвестен, то также неизвестны весовые коэффициенты, смещения нейронной сети и параметры Q для хаотического отображения, что не позволяет противнику вычислить хэш-значение $H_{M_{n-1}}$ для блока M_{n-1}, для которого ищется коллизия. Поэтому данную атаку можно рассмотреть, если противнику известен общий ключ K. В этом случае ему будут известны промежуточный ключ $K_{M_{n-2}}$ и хэш-значение $H_{M_{n-1}} = h(M_{n-1}, K_{M_{n-2}})$.

На данный момент сложность атаки равна $O(2^{128})$, что сравнимо с полным перебором. Таким образом, хэш-функции с длиной хэш-значения 128 бит и больше устойчивы к атакам “встреча посередине”.

4.2.5 Атака с коррекцией блока

Цель противника при данной атаке – изменить сообщение без изменения хэш-значения. Противник подбирает и добавляет к изменённому сообщению один или несколько блоков, чтобы изначальное хэш-значение не поменялось. Такие блоки называются корректирующими.

Для того, чтобы подобрать корректирующий блок, противник должен уметь вычислять хэш-значение, т.е. ему известен ключ K, соответственно и известен промежуточный ключ $K_{M'_{n-2}}$, необходимый для хэширования последних (корректирующих) блоков. Цель – найти блок, зная его хэш-значение и ключ. Поэтому атака с коррекцией блока равносильна атаке “встреча посередине”. Исходя из вышеописанного анализа этой атаки, хэш-функции с длиной хэш-значения 128 бит и больше устойчивы к атакам с коррекцией блока.

4.2.6 Атака с фиксированной точкой

Фиксированной точкой будем называть такой блок сообщения M_i, что $K_{M_i} = h\left(M_i, K_{M_{i-1}}\right) = K_{M_{i-1}}$. Перепишем равенство по-другому (учитывая схему на Рисунке 16):

$$K_{M_i} = K_{M_{i-1}} \oplus H_{M_i} = K_{M_{i-1}} \Rightarrow H_{M_i} = 0$$

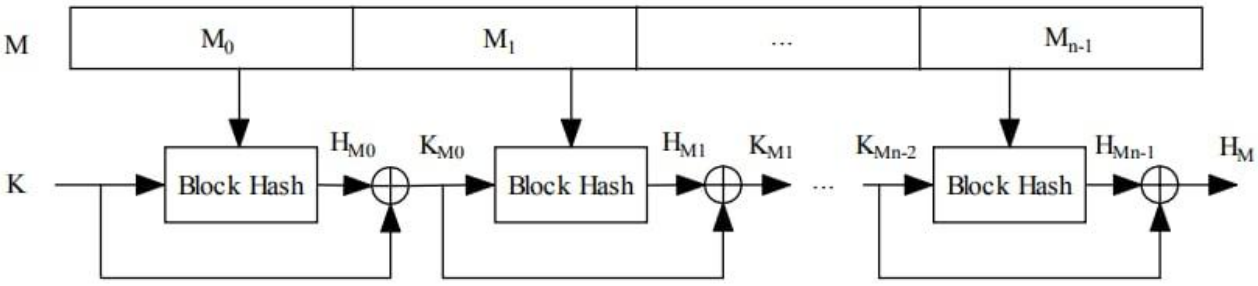

Рисунок 16 – Многоблочное хэширование (Multi-Block Hash)

Т.е. если для некоторого блока сообщения M_i выполняется $h\left(M_i, K_{M_{i-1}}\right) = 0$ (128-битный вектор, состоящий из 0), то такие блоки можно добавлять или удалять сколько угодно раз без изменения итогового хэш-значения. Требуется найти сообщение $M' \neq M_i$, дающее нулевое хэш-значение, при известных

ключе и хэш-значении. Поэтому данная атака сводится к атаке “встреча посередине”, сложность которой $O(2^{128})$.

Таким образом, хэш-функции с длиной хэш-значения 128 бит и больше устойчивы к атакам с фиксированной точкой.

4.3 Анализ чувствительности

Исследуем чувствительность хэш-функции к небольшим изменениям в исходном тексте и ключе. Любая криптографическая хэш-функция должна обладать “лавинным эффектом”: при малейших изменениях во входных данных итоговое хэш-значение должно непредсказуемо меняться, т.е. в среднем должно поменяться ½ бит.

Для исследования чувствительности будем последовательно менять только i-й бит входного текста (ключа) и считать расстояние Хэмминга (количество различных битов в последовательностях) между исходным хэш-значением и изменённым хэш-значением. Ось X – номер изменяемого бита текста (ключа), ось Y – расстояние Хэмминга (в процентах). На Рисунке 17 представлен результат для исходного текста длины 3360 бит. На Рисунке 18 – результат для ключа.

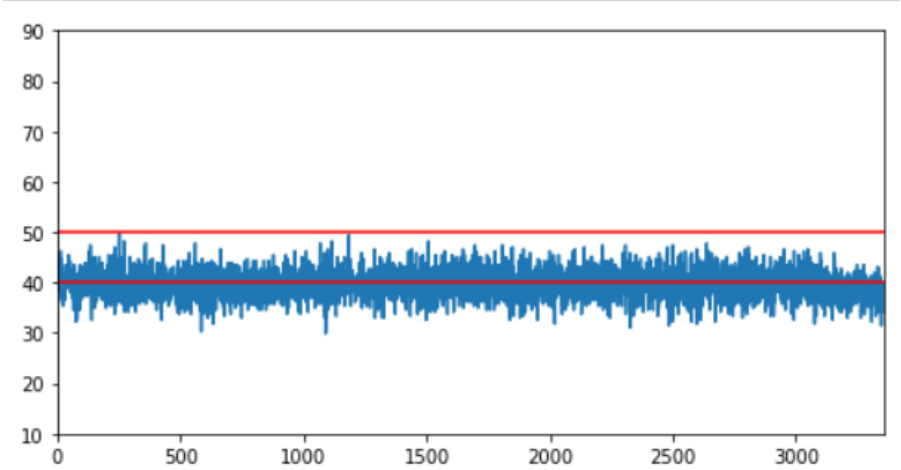

Рисунок 17 – Чувствительность к изменениям в исходном тексте (1 бит)

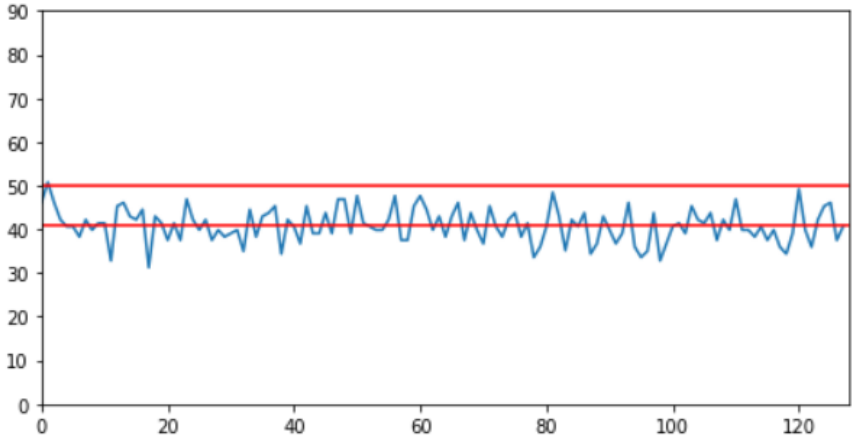

Рисунок 18 – Чувствительность к изменениям в ключе (1 бит)

Из графиков следует, что значения лежат в пределах 41%. Но чувствительность к изменению в ключе можно поднять, немного модифицировав алгоритм выработки параметров нейронной сети по ключу. Например, при использовании нового соотношения

$$p_i = \left(2.5{X_0}^4(i) + 1.5{X_1}^2(i) - 1.8 * X_0(i) * X_1(i) + 0.33\right) mod\ 1,$$

которое получено эмпирически, результат будет следующий (Рисунок 19):

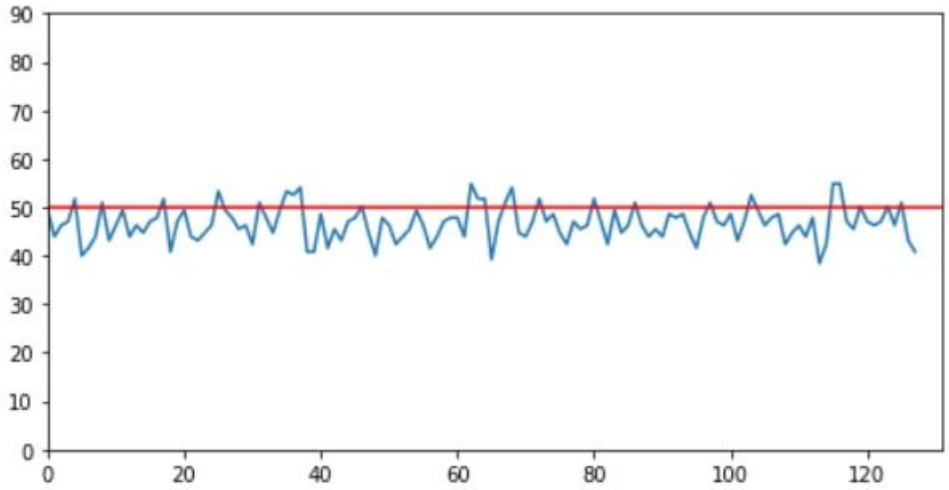

Рисунок 19 – Чувствительность к изменениям в ключе (1 бит), новая функция

Исследуем чувствительность при изменении случайных 5, 10 и 15 бит в ключе и тексте. Результаты отображены в Таблице 1.

Таблица 1 – Анализ чувствительности для 5, 10 и 15 бит

Архитектура НС	Текст			Ключ		
	5 бит	10 бит	15 бит	5 бит	10 бит	15 бит
8-8-4	45%	47%	50%	48%	50%	50%

Далее рассмотрим модификации нейронной сети и исследуем чувствительность новых хэш-функций к изменениям во входных значениях. Результаты отображены в Таблице 2, где в первом столбце представлены архитектуры нейронной сети (например, запись “8-8-4” означает, что нейронная сеть состоит из 3 слоёв, на первых двух слоях по 8 нейронов, на последнем слое 4 нейрона), во втором и третьем столбце представлены результаты эксперимента (эксперимент повторялся по 100 раз для каждого случая), в последнем столбце отображено среднее время выработки одного хэш-значения.

Таблица 2 – Сравнительный анализ модификаций хэш-функции

Архитектура НС	Текст	Ключ	Время, с.
8-8-4	40 %	41 %	< 0.01
8-8-8	41 %	41 %	0.01
16-16-8	42 %	43 %	0.03
16-16-16	43 %	45 %	0.04
8-8-16-8-8	44 %	46 %	0.11

8-16-32-16-8	46 %	47 %	0.29

На основании проведённого анализа можно сделать вывод, что при усложнении структуры нейронной сети результат улучшается, но увеличивается время выработки одного хэш-значения. Т.е. если оптимизировать алгоритм выработки хэш-значения, модифицировать функцию выработки параметров нейронной сети, то хэш-функция со структурой нейронной сети 8-16-32-16-8 будет удовлетворять требованиям стойкости и чувствительности, что позволяет применять её в реальных задачах.

4.4 Преимущества и недостатки реализованной хэш-функции

Преимущества:

- Стойкость: данная хэш-функция построена на основе нейронной сети, что усложняет проведение всех известных атак (многие атаки сводятся к полному перебору). Т.е. хэш-функция является стойкой ко многим известным атакам. Также функция является однонаправленной.
- Легко модифицируема: длину хэш-значения легко увеличить до 256 или 512 бит, что значительно усложнит атаку. Если увеличить количество нейронов в выходном слое до 8, то хэш-значение будет длиной 256 бит. Если увеличить длину исходного текста до 2048 бит и удвоить количество нейронов на каждом слое, то выходной хэш будет длины 512 бит.
- Допускает распараллеливание, что снизит время работы приблизительно в 10 раз.

Недостатки:

- Низкая чувствительность: при использовании трёхслойной нейронной сети хэш-функция имеет низкую чувствительность, что вынуждает использовать более сложные структуры. Однако это значительно увеличивает время работы.

ЗАКЛЮЧЕНИЕ

В работе исследовано применение ИНС для решения задач криптографии.

В первой главе были рассмотрены основные определения и понятия нейронных сетей: структура нейронных сетей, их виды, а также их приложения в повседневной жизни: распознавание речи, медицинская диагностика, распознавание символов и др.

Во второй главе были рассмотрены области применения ИНС и машинного обучения в криптографии и криптоанализе.

В третьей и четвёртой главах были рассмотрены основные свойства хэш-функций и требования для их стойкости к атакам. Был подробно разобран алгоритм работы криптографической функции хэширования на основе нейронной сети с использованием хаотических отображений и проведён её анализ.

Основные результаты работы:

- проведён обзор литературы, посвящённый применению искусственных нейронных сетей в задачах защиты информации;
- разработана архитектура ИНС, реализующей функцию хэширования;
- разработано программное обеспечение, реализующее функцию хэширования на основе ИНС;
- проведён анализ криптографических свойств построенной функции хэширования: стойкость к известным атакам, наличие лавинного эффекта;
- проведён анализ влияния параметров ИНС на криптографические характеристики построенной функции хэширования;
- определены параметры, обеспечивающие наибольшую криптографическую стойкость.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Агиевич, С.В. Криптографические методы [Электронный ресурс]. – Режим доступа : http://apmi.bsu.by/assets/files/agievich/cm.pdf. – Дата доступа : 16.05.22.
2. Галушкин, А. И. Нейронные сети: основы теории / А. И. Галушкин. — Москва: Горячая линия-Телеком, 2017. — 496 с.
3. Нейронная сеть [Электронный ресурс]. - Режим доступа : https://ru.wikipedia.org/wiki/Нейронная_сеть. - Дата доступа : 12.12.2021.
4. Нейронные сети [Электронный ресурс] // Большая российская энциклопедия. - Режим доступа : https://bigenc.ru/technology_and_technique/text/4114009. - Дата доступа : 12.12.2021.
5. Обучение нейронной сети [Электронный ресурс]. - Режим доступа : https://neuronus.com/theory/nn/238-obucheniya-nejronnoi-seti.html. - Дата доступа : 12.12.2021.
6. Синаптическая пластичность [Электронный ресурс]. - Режим доступа : https://ru.wikipedia.org/wiki/Синаптическая_пластичность. - Дата доступа : 12.12.2021.
7. Функции активации нейросети: сигмоида, линейная, ступенчатая, ReLu, tahn [Электронный ресурс]. - Режим доступа : https://neurohive.io/ru/osnovy-data-science/activation-functions. - Дата доступа : 11.12.2022.
8. Элементарное введение в технологию нейронных сетей с примерами программ / Тадеусевич Р. [и др.]; перевод с польск. И.Д. Рудинского. – М.: Горячая линия-Телеком, 2011. – 408 с.
9. Alani, M.M. Applications of Machine Learning in Cryptography: A Survey [Electronic resource]. – Mode of access : https://arxiv.org/pdf/1902.04109.pdf. – Date of access : 18.09.2022.
10. Alani, M.M. Neuro-Cryptanalysis of DES and Triple-DES [Electronic resource]. – Mode of access : https://sci-hub.se/http://dx.doi.org/10.1007/978-3-642-34500-5_75. – Date of access : 16.05.22.
11. Alshammari, R. Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype / R. Alshammari, A.N. Zincir-Heywood [Electronic resource]. – Mode of access : https://web.cs.dal.ca/~zincir/bildiri/cisda09-rn.pdf. – Date of access : 16.05.22.
12. Analyzing Android Encrypted Network Traffic to Identify User Actions / M. Conti [et al.] [Electronic resource]. – Mode of access : https://riki8686.github.io/files/android2015.pdf. – Date of access : 16.05.22.
13. Baek, S. Recent Advances of Neural Attacks against Block Ciphers / S. Baek, K. Kim [Electronic resource]. – Mode of access :

https://caislab.kaist.ac.kr/publication/paper_files/2020/scis2020_SG.pdf. – Date of access : 04.10.2022.
14. Blum, A. Machine Learning Theory [Electronic resource]. – Mode of access : https://www.cs.cmu.edu/~avrim/Talks/mlt.pdf. – Date of access : 16.05.22.
15. Guo, D. A New Symmetric Probabilistic Encryption Scheme Based on Chaotic Attractors of Neural Networks / D. Guo, L.M. Cheng, L.L. Cheng [Electronic resource]. – Mode of access : https://sci-hub.se/http://dx.doi.org/10.1023/A:1008337631906. – Date of access : 16.05.22.
16. Hao, F. Combining Crypto with Biometrics Effectively / F. Hao, R. Anderson, J. Daugman [Electronic resource]. – Mode of access : https://www.cs.umd.edu/~gasarch/TOPICS/cryptoml/crackprng.pdf. – Date of access : 16.05.22.
17. Hughes, J.M. Pseudo-random Number Generation Using Binary Recurrent Neural Networks [Electronic resource]. – Mode of access : https://www.cs.umd.edu/~gasarch/TOPICS/cryptoml/crackprng.pdf. – Date of access : 16.05.22.
18. Kanade, S. Multi-Biometrics Based Cryptographic Key Regeneration Scheme / S. Kanade, D. Petrovska-Delacretaz, B. Dorizzi [Electronic resource]. – Mode of access : https://hal.science/hal-01360794/file/kanade-btas-2009.pdf. – Date of access : 18.09.2022.
19. Lerman, L. Power Analysis Attack: An Approach Based on Machine Learning / L. Lerman, G. Bontempi, O. Markowitch [Electronic resource]. – Mode of access : https://sci-hub.se/http://dx.doi.org/10.1504/IJACT.2014.062722. – Date of access : 16.05.22.
20. Lian, S. One-way Hash Function Based on Neural Network / S. Lian, J. Sun, Z. Wang [Electronic resource]. – Mode of access : https://arxiv.org/ftp/arxiv/papers/0707/0707.4032.pdf. – Date of access : 18.09.2022.
21. Machine Learning Classification over Encrypted Data / R. Bost [et al.] [Electronic resource]. – Mode of access : https://eprint.iacr.org/2014/331.pdf. – Date of access : 16.05.22.
22. Machine Learning in Side-Channel Analysis: A First Study / G. Hospodar [et al.] [Electronic resource]. – Mode of access : https://sci-hub.se/http://dx.doi.org/10.1007/s13389-011-0023-x. – Date of access : 16.05.22.
23. Park, S. Dynamical Pseudo-Random Number Generator Using Reinforcement Learning / S. Park, K. Kim, K. Kim, C. Nam [Electronic resource]. – Mode of access: https://www.mdpi.com/2076-3417/12/7/3377. – Date of access : 16.05.22.
24. Sadegh Riazi, M. SYNFI: Automatic Synthetic Fingerprint Generation / M. Sadegh Riazi, S.M. Chavoshian, F. Koushanfar [Electronic resource]. – Mode of access : https://eprint.iacr.org/2020/217.pdf. – Date of access : 18.09.2022.

25. Sagar, V. A Symmetric Key Cryptographic Algorithm Using Counter Propagation Network (CPN) / V. Sagar, K. Kumar [Electronic resource]. – Mode of access : https://sci-hub.se/http://dx.doi.org/10.1145/2677855.2677906. – Date of access : 16.05.22.
26. Synchronization of Neural Networks by Mutual Learning and Its Application to Cryptography / E. Klein [et al.] // Advances in Neural Information Processing Systems. – 2005. - P. 689–696.
27. Turčaník, M. Hash Function Generation by Neural Network / M. Turčaník, M. Javurek [Electronic resource]. – Mode of access : https://www.researchgate.net/publication/310624366_Hash_function_generation_by_neural_network. – Date of access : 20.09.2022.

ПРИЛОЖЕНИЕ

Реализация хэш-функции версии 8-8-8 на языке Python

```
# Используемые библиотеки
import numpy as np
import math
import time
import bitstring
from scipy.spatial.distance import hamming
from matplotlib import pyplot as plt

# Глобальные значения
L = 151 + 32 + 4 #количество параметров нейронной сети
nbits = 8 #количество битов в одном символе
block_size = 32 #размер одного
max_int = 2**32 - 1 #максимальное целое число
input_size = 1024 #размер одного блока входного сообщения
key_len = 256 #длина ключа
key_parts = 4
hash_len = 256 #длина хеша
n0_input = input_size // block_size
n1_input = 8
n1_input_per_neuron = n0_input // n1_input
n1_output = n2_input = 8
n2_output = n3_input = 8
n3_output = 8
T = 50


# Функция активации
def f(x, q, t):
 for _ in range(t):
 if 0 <= x < q:
 x = x/q
 elif q <= x < 0.5:
 x = (x-q)/(0.5-q)
 elif 0.5 <= x < 1-q:
 x = (1-x-q)/(0.5-q)
 elif 1-q <= x <= 1:
 x = (1-x)/q
 return x

# Конвертация текста в двоичный вид
def text_to_bin(text):
 bin_text = ''
 for byte in text.encode():
 tmp = format(byte, 'b')
 tmp = tmp if len(tmp) == nbits else '0'*(nbits - len(tmp)) + tmp
 bin_text += tmp
 return bin_text

# Конвертация вещественного числа в двоичный вид
def float_to_bin(float_val):
 return bitstring.BitArray(float=float_val, length=32).bin

# Деление текста на блоки
def split_into_blocks(data, l):
 k = len(data) // l
 r = len(data) - k*l
 blocks = [data[i:i+l] for i in range(0, len(data), l)]
 if r:
```

```
 blocks[-1] += '1' + '0'*(l-r-1)
 blocks = [[int(block[block_size*i:block_size*(i+1)], 2) / max_int for i in
range(n0_input)] for block in blocks]
 return blocks

# Генерация параметров НС
def gen_params(key, t):
 key = text_to_bin(key)[:key_len]
 key = [int(key[block_size*i:block_size*(i+1)], 2) / max_int for i in
range(key_len//block_size)]
 key = [(key[0]+key[6]) % 1, (key[3]+key[5]) % 1, (key[2]+key[4]) % 1,
(key[1]+key[7]) % 1 ]
 ks = []
 x0 = f(key[0], key[1], t)
 x1 = f(key[2], key[3], t)
 for k in range(L):
 x_ = x0 + x1
 ks.append(np.fabs(x_))
 x0 = f(x0, key[1], 1)
 x1 = f(x1, key[3], 1)
 ks = np.array(ks, dtype='float32') % 1
 # W0, B0, Q0, W1, B1, Q1, W2, B2, Q2
 return ks[:32], ks[32:40], ks[40], ks[41:105], ks[105:113], ks[113],
ks[114:178], ks[178:186], ks[-1]

# Блочное хэширование
def get_hash_256(p, f_vect, t, w0, b0, q0, w1, b1, q1, w2, b2, q2):
 # input layer
 tmp = np.array([sum([w0[n1_input_per_neuron*j+i]*p[n1_input_per_neuron*j+i]
for i in range(n1_input_per_neuron)]) + b0[j] for j in range(n1_input)])
 tmp = tmp % 1
 c = f_vect(tmp, q0, t)
 # hidden layer
 w1 = w1.reshape(n1_output, n2_input)
 tmp = np.array([sum([w1[j][i]*c[i] for i in range(n2_input)]) + b1[j] for j
in range(n2_output)])
 tmp = tmp % 1
 d = f_vect(tmp, q1, t)
 # output layer
 w2 = w2.reshape(n3_output, n2_output)
 tmp = np.array([sum([w2[j][i]*d[i] for i in range(n3_input)]) + b2[j] for j
in range(n3_output)])
 tmp = tmp % 1
 h = f_vect(tmp, q2, t)
 h = ''.join(float_to_bin(fl_hash) for fl_hash in h)
 return h

# Мультиблочное хэширование
def nn_hash_256(data, t, init_key, mode='', key_mode=''):
 f_vect = np.vectorize(f)
 # разделение data на блоки
 data_bin = data
 if mode != 'bin':
 data_bin = text_to_bin(data)
 blocks = split_into_blocks(data_bin, input_size)

 bin_key = init_key
 if key_mode != 'bin':
 bin_key = text_to_bin(init_key)[:key_len]
 # хэширование каждого блока
 for block in blocks:
 params = gen_params(bin_key, t)
 bin_hash = get_hash_256(block, f_vect, t, *params)
```

```
 bin_key = ''.join([str((ord(a) ^ ord(b))) for a, b in zip(bin_key,
bin_hash)])
 hex_hash = '{:0{}x}'.format(int(bin_key, 2), hash_len//4)
 return hex_hash, bin_key

# Анализ чувствительности (текст)
text = "Every time you're online, you are bombarded by pictures, articles,
links and videos trying to tell their story. Unfortunately, not all of these
stories are true. Sometimes they want you to click on another story or
advertisement at their own site, other times they want to upset people for
political reasons. These days it's so easy to share information. These
stories circulate quickly, and the result is … fake news."
bin_text = text_to_bin(text)
b_len = len(bin_text)
key = '#<02_May_23>,15..40,<BSU,FAMCS>#'
h0 = nn_hash_256(bin_text, T, key, 'bin')[1]
hashes = []
for j in range(len(bin_text)):
 s = str((int(bin_text[j]) + 1) % 2)
 bin_text_new = bin_text[:j] + s + bin_text[j+1:]
 hashes += [nn_hash_256(bin_text_new, T, key, 'bin')[1]]
dist = [hamming(list(h0), list(h))*100 for h in hashes]
m = np.mean(dist)
print(m)
line1 = [[0, b_len], [50, 50]]
line2 = [[0, b_len], [m, m]]
plt.subplots(figsize=(8, 4))
plt.plot(dist)
plt.plot(line1[0], line1[1], color = 'red')
plt.plot(line2[0], line2[1], color = 'red')
plt.ylim(10,90)
plt.xlim(0, b_len)
plt.show()

# Анализ чувствительности (ключ)
text = "Every time you're online, you are bombarded by pictures, articles,
links and videos trying to tell their story. Unfortunately, not all of these
stories are true. Sometimes they want you to click on another story or
advertisement at their own site, other times they want to upset people for
political reasons. These days it's so easy to share information. These
stories circulate quickly, and the result is … fake news."
key = '#<02_May_23>,15..40,<BSU,FAMCS>#'
bin_key = text_to_bin(key)
h0 = nn_hash_256(text, T, key)[1]
hashes = []
for j in range(len(bin_key)):
 s = str((int(bin_key[j]) + 1) % 2)
 bin_key_new = bin_key[:j] + s + bin_key[j+1:]
 hashes += [nn_hash_256(text, T, bin_key_new, key_mode='bin')[1]]
b_len = 256
dist = [hamming(list(h0), list(h))*100 for h in hashes]
m = np.mean(dist)
print(m)
line1 = [[0, b_len], [50, 50]]
line2 = [[0, b_len], [m, m]]
plt.subplots(figsize=(8, 4))
plt.plot(dist)
plt.plot(line1[0], line1[1], color = 'red')
plt.plot(line2[0], line2[1], color = 'red')
plt.ylim(0, 90)
plt.xlim(0, b_len)
plt.show()
```