МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Смелов

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

**ЗАДАНИЕ**

**к курсовому проектированию по дисциплине**"Защита информации и надежность информационных систем"

Специальность 1-40 05 01 03 «Информационные системы и технологии (издательско-полиграфический комплекс)»

Группа: 1

Студент: Савельев Дмитрий Витальевич

**Тема: «**Реализация и исследование алгоритма хеширования Lyra2»

**1. Срок сдачи студентом законченной работы**: «10» мая 2022г.

**2. Исходные данные к проекту:**

**2.1**. Функционально должны быть выполнены следующие задачи:

* Реализовать методы для хеширования данных с использованием параллелизма, а также без него.
* Разработать программное средство реализации данного метода или алгоритма.
* Составить руководство пользователя.

**2.2. Требования:**

* Необходимо провести аналитический обзор литературы по теме проекта
* Необходимо описать сферу применимости метода
* Программное средство может быть разработано на любом языке
* Архитектура приложения выбирается разработчиком
* Листинги проекта должны содержать комментарии

**3. Содержание расчетно-пояснительной записки:**

* Введение
* Постановка задачи
* Описание метода
* Описание программного средства
* Тестирование программного средства
* Руководство пользователя
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненного курсового проекта:**

* Теоретическая часть курсового проекта должна быть представлена в формате MS Word.
* Оформление записки должно быть согласно правилам.
* Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio или копии экрана (интерфейс).
* Полные листинги проекта представляются в приложении.
* К записке необходимо приложить CD (DVD), который должен содержать: пояснительную записку, листинги и файлы базы данных.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 12.02.2022 |  |
| 2 | Аналитический обзор литературы по теме проекта | 28.02.2022 |  |
| 3 | Разработка метода | 10.03.2022 |  |
| 4 | Разработка прототипа программного средства | 24.03.2022 |  |
| 5 | Разработка программного средства | 31.03.2022 |  |
| 6 | Тестирование программного средства | 14.04.2022 |  |
| 7 | Написание руководства пользователя | 21.04.2022 |  |
| 8 | Оформление пояснительной записки | 28.04.2022 |  |
| 9 | Сдача проекта | 10.05.2022 |  |

**5. Дата выдачи задания** «15» февраля 2022г.

Руководитель *\_\_\_\_В.О. Берников\_\_\_\_\_\_\_\_*

(подпись)

Задание принял к исполнению \_15.03.2022 \_\_\_\_\_\_\_\_\_

(дата и подпись студента)

# Введение

Аутентификация пользователей – это один из самых важных элементов в современной компьютерной безопасности. Даже несмотря на существование механизмов аутентификации, основанных на биометрических данных («что есть пользователь») или физические устройства, такие как смарт-карты («что есть у пользователя»), наиболее широко распространенный способ все равно основывается на секретных паролях («что пользователь знает»). Это происходит из-за того, что аутентификация, основанная на паролях, остается самым удобным и эффективным способом поддержания общего секрета между пользователем и компьютерной системой. К счастью или сожалению, и несмотря на существование множества предложений по их замене, это преобладание паролей в качестве одного и часто единственного способа аутентификации скорее всего не изменится в ближайшем будущем.

Системы, основанные на паролях, обычно используют некоторые криптографические алгоритмы, которые позволяют генерировать псевдослучайные строки бит на основе паролей, так называемая криптографические хеш-функции (англ. password hashing scheme – PHS) или функция формирования ключа (англ. key derivation function – KDF). Обычно, вывод хеш-функции используется одним из двух способов: он может храниться в форме «токена» для будущих проверок пароля или использоваться в качестве секретного ключа для зашифрования и/или аутентификации данных. В любом случае, такие решения используют внутренние односторонние (например, хеш) функции, так что восстановление пароля из вывода криптографической хеш-функции вычислительно неосуществимо.

Несмотря на популярность аутентификации, основанной на пароле, тот факт, что большинство пользователей выбирают довольно коротки и простые строки в качестве паролей, приводит к серьезным проблемам: они обычно имеют гораздо меньшую энтропию нежели рекомендуется для криптографических ключей. Согласно анализу 5 миллионов слитых паролей Gmail, средняя энтропия их составила 21.6 бит, при том, что для соответствия категории «сильный пароль», он должен иметь энтропию более 60 бит, а «очень сильный» - более 128 бит. Исследование же 2007 года на основании 544960 паролей реальных пользователей получила среднюю энтропию порядка 40.54 бит. Такая слабость паролей сильно упрощает множество видов «брутфорс» атак, такие как атака со словарем и полный перебор, позволяя хакерам полностью обходить свойство неинверсивного процесса хеширования. Как пример, хакеры могут применять криптографические хеш-функции для списка наиболее используемых паролей пока результат не совпадет с локально хранимым токеном или действительным ключом шифрования/аутентификации. Осуществимость таких атак зависит в основном от количества ресурсов доступных для хакеров, кто может ускорить процесс распараллеливая множество тестов. Такие атаки обычно получают выгоду от платформ, оснащенных большим количеством процессорных ядер, например современных видеокарт или специализированного оборудования.

Прямолинейный подход для решения этой проблемы: заставлять пользователей выбирать сложные пароли. Однако это плохой подход, так как такие пароли сложнее запоминать, а значит пользователи должны их где-то записать, что полностью перечеркивает весь смысл аутентификации. Для решения этих проблем современные решения хеширования паролей обычно используют механизмы для увеличения «стоимости» брутфорс атак. Такие методы как PBKDF2 и bcrypt, к примеру, включают конфигурируемые параметры, которые контролируют количество осуществляемых итераций, позволяя пользователям изменять время, требуемое для процесса хеширования пароля. Более актуальный алгоритм scrypt позволяет пользователям контролировать помимо времени обработки использование памяти, усложняя тем самым восстановление пароля. Существует, однако, значительный интерес в исследование и развитии новых и лучших алгоритмов хеширования.

# 2. Обзор алгоритма Lyra2

В результате описанных во введении проблем М. Симплисио младший, Л. К. Алмейда, Э. Р. Андраде, П. К. Ф. Сантуш и П. С. Л. М. Баррето из Политехнической школы Университета Сан-Паулу разработали на основе своего алгоритма Lyra улучшенную версию – Lyra2. Lyra2 предлагает безопасность, эффективность и гибкость Lyra, включает: возможность настраивать желаемое количество используемой памяти, время выполнения и использование параллелизма алгоритмом, возможность обеспечения большего использования памяти с временем обработки, аналогичным таковому, полученному с помощью scrypt.

Lyra2 использует в своей реализации Sponge функции. Sponge функции представляют собой хеш-функции, которые могут итеративно обрабатывать входные и выходные данные произвольной длины. Они включают в себя перестановку f фиксированное длины, которая работает с внутренним состоянием представленным последовательностью размером b+c битов, состоящим из битрейта длиной b и емкостью c, в сочетании с входными данными M, разрезанными на b-битные блоки. Sponge функция включает в себя операции absorb, которая заключается в итеративном применении f к внутреннему состоянию после применения операции XOR битрейта с каждым из b-битных входных битов. Операция squeeze, в свою очередь, представляет собой применение f ко всему внутреннему состоянию и последующую выдачу битрейта на выход, эта операции будет выполняться пока заданное пользователем количество битов не будет предоставлено в качестве вывода.

## 2.1.