%B5%D1%81%D1%81_%D0%93%D0%B0%D0%B1%D0%B5%D1%80%D0%B0. Дата доступа: 17.04.2023 г.

2. Катализатор. [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80. Дата доступа: 18.04.2023 г.

3. Получение аммиака. [Электронный ресурс]. Режим доступа: https://obrazovaka.ru/himiya/poluchenie-ammiaka.html. Дата доступа: 19.04.2023 г.

УДК [004.056+003.26](075.8)

Студ. Д.С. Шкабров
Науч. рук. проф. П.П. Урбанович
(Кафедра информационных систем и технологий, БГТУ)

АНАЛИЗ АЛГОРИТМА ХЭШИРОВАНИЯ KECCAK

Безопасность является важнейшей характеристикой современных ИТ. Многие решения проблемы основаны на использовании хэш-преобразований [1, 2]. В докладе анализируется алгоритм Keccak, который генерирует хэши различных длин: 224, 256,384 и 512. Следует отметить, что Keccak является криптографически стойкой хэш-функцией, что означает, что для любой длины хэша, которую вы выберете, вероятность обнаружения двух сообщений с одинаковым хэшем крайне мала.

Следует учитывать, что более длинные хэши обеспечивают более высокий уровень безопасности, но требуют больше ресурсов для вычисления. Кроме того, более длинные хэши обычно занимают больше места в памяти и на диске, что может быть проблемой в некоторых приложениях. Так же, по логике, должно быть, что вычисление хэша с большей длиной будет дольше по времени, чем с меньшей длиной и мы это проверим.

Рисунок 1 – Вывод результата хэширования и времени вычисления

Анализ Keccak с разной длинной хэшейбудет проводится на языке Pythonиспользуя библиотеку *hashlib*. В коде есть цикл, в котором предложениехэшируется всеми длинами хэшей, и засекается время на хэширование предложения (рис. 1). Каждый цикл новое предложение (предложение + число, которое увеличивается с проходом цикла). Как можно заметить, время, затраченное на вычисление крайне маленькое, что даже в нескольких результатах оно нулевое. Все остальные результаты получились примерно за 0.001 секунд (рис.2). Следовательно, время, как критерий для выбора размера хэша, можно не рассматривать, а выбор нужного размера хэша лучше рассматривать по количеству места в памяти и потребуемому уровню безопасности.

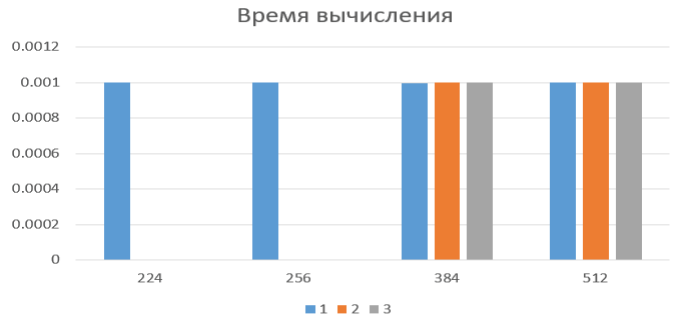

Рисунок 2 – Гистограмма времени вычисления

А вывод тут в том, что выбор длины хэша зависит от конкретного применения и требований к безопасности. Например, для хранения паролей может использоваться хэш длиной 256 бит, а для проверки целостности файла – хэш длиной 512 бит [4].

ЛИТЕРАТУРА

1. Урбанович, П. П. Защита информации и надежность информационных систем: учебное пособие / П. П. Урбанович, Д. В. Шиман. – Минск: БГТУ, 2014. – 90 с.

2. Урбанович, П. П. Лабораторный практикум по дисциплинам «Защита информации и надежность информационных систем» и «Криптографические методы защиты информации». В 2 ч. Ч. 2. Криптографические и стеганографические методы защиты информации: учеб.-метод. пособие для студ. вузов / П. П. Урбанович, Н. П. Шутько. – Минск: БГТУ, 2020. – 226 с.

3. Hashlib–Безопасные хэши и дайджесты сообщений. URL:https://runebook.dev/ru/docs/python/library/hashlib(дата доступа: 04.04.2023г.).

4. Почему Keccak настолько крут и почему его выбрали в качестве нового SHA-3. URL:https://habr.com/ru/articles/168707/(дата доступа: 04.04.2023г.).