УДК 004.45 **ТЕХНИЧЕСКИЕ НАУКИ**

***Стрелец Андрей Иванович**, магистр*
кафедра «Компьютерные системы и технологии»,
Национальный исследовательский ядерный университет «МИФИ»
Россия, г. Москва
***Храпов Александр Сергеевич,** магистр*
кафедры «Компьютерные системы и технологии»,
Национальный исследовательский ядерный университет «МИФИ»
Россия, г. Москва
***Иванников Владислав Сергеевич,** магистр*
кафедры «Компьютерные системы и технологии»,
Национальный исследовательский ядерный университет «МИФИ»
Россия, г. Москва
***Атавина Анастасия Владиславовна,** магистр*
кафедры «Финансовый мониторинг»,
Национальный исследовательский ядерный университет «МИФИ»
Россия, г. Москва

ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ АЛГОРИТМОВ PROOF-OF-WORK

Аннотация: В статье исследуется современные алгоритмы доказательства выполнения работы (Proof-of-work), их достоинства и недостатки.

Ключевые слова: доказательство выполнения работы, информационная безопасность, ассиметричные алгоритмы.

Abstract: This article is about algorithms of Proof-of-work, advantages and disadvantages of these algorithms.

Key words: proof-of-work, information security, asymmetric cryptography.

Введение

Алгоритмы вида Proof-of-Work являются основой для работы криптовалют и сервисов, обеспечивающих защиту от DoS атак. Но пока что требования к быстрой и не ресурсоёмкой проверке результата вычисления даёт преимущество в вычислениях не персональным компьютерам, а многочисленным ботнетам, а также GPU, FPGA и ASIC фермам [1]. Именно для решения этой проблемы был предложен алгоритм Equihash, основанный на математическом парадоксе дней рождения, и использующий алгоритм Вагнера.

Алгоритмы Proof-of-Work

Предложение использовать задачу, в ходе решения которой выполняются интенсивные вычисления, в качестве средства противодействия спаму было впервые предложено в 1992 году. В 2001 году было предложено использовать подобные задачи в виде головоломки клиента TLS в качестве защиты от атак, ориентированных на отказ в обслуживании (DoS) [2]. Объем работы, произведённый клиентом, подтверждается доказательством, которое носит название Proof-of-Work. Это доказательство может получить любой обычный пользователь, но в то же время процесс его получения замедляет скорость запросов от одной машины. Самый простой пример подобного алгоритма — это схема Hashcash используемый в тензорных процессорах [3], Алгоритм использует результат вычисления хеш-функции, которая в адаптированном виде используется в криптовалюте Биткойн [4].

Первая версия алгоритма производила вычисления с привязкой к памяти, путём получения доступа псевдослучайным образом к случайному массиву среднего размера. Позднее было предложено заполнять этот массив путём выполнения функции, активно использующей память при вычислениях, в результате чего объём используемой памяти мог быть уменьшен только при очень больших вычислительных затратах. А поскольку память является намного более дорогим ресурсом с точки зрения площади и амортизированной стоимости чипа, ASIC в данном случае ненамного эффективнее обычного процессора. Однако проблема ботнетов всё ещё остаётся актуальной, хотя

использование огромных объёмов памяти уже гораздо сложнее скрыть. Можно так же предположить, что уменьшение преимуществ ASIC может обеспечить дополнительный стимул для создания ботнетов с использованием нейронных сетей.

Однако предложенные варианты так никогда и не были использованы на практике. В первую очередь из-за того, что предложенные схемы симметричны по использованию памяти. Проверяющий должен использовать то же количество памяти, что и используется для вычисления доказательства. Это противоречит принципу вычислительной асимметричности, используемому, к примеру в биткойне, где проверка корректности доказательства выполняется без использования тех ресурсов, что нужны для получения этого самого доказательства. Таким образом, проверяющий доказательство должен быть почти таким же мощным, как и тот, кто предоставляет доказательства. Поэтому данные алгоритмы достаточно бесполезны в задачах противодействия DoS атак.

Для решения перечисленных выше проблем было разработано семейство быстрых, асимметричных по памяти, не оптимизируемых, ограниченно параллелизуемых Proof-of-Work алгоритмов, каждый из которых основан на вычислительно сложной, но хорошо изученной задаче. Наиболее перспективный из них, Equihash, основан на парадоксе дней рождений, который хорошо изучен как с теоретической стороны, так и с точки зрения практического применения.

Алгоритм Equihash

Equihash основан на алгоритме Вагнера, вызывающим хеш-функцию. Чтобы начать использовать алгоритм, верификатор должен выбрать хеш-функцию H и целые числа n, k, d, которые задают требования ко времени выполнения и памяти следующим образом:

- Память M задаётся как $2^{n/(k+1)+k}$ байт;
- Время T составляет $(k+1)2^{n/(k+1)+d}$ вызовов хеш-функций H;
- Размер решения S задаётся как $2^k(n/(k+1)+1)+160$ бит;
- Стоимость верификации составляет 2^k вызовов H и XOR.

Затем, верификатор выбирает начальное число (которое может быть хэшем транзакции, переменной в цепочке блоков и т. д.), после чего просит

решателя найти 160-битное число и (n/(k+1) + 1) битные x_1, x_2 ,… , x_2^k удовлетворяющее следующим условиям:

- $H(I||V||x_1) \oplus H(I||V||x_2) \oplus \cdots \oplus H(I||V||x_2^k) = 0$,
- $H(I||V||x_1||x_2|| \ldots ||x_2^k)$ где d лидирующих нулей,
- $H(I||V||x_{w2^{l}+1}) \oplus \ldots \oplus H(I||V||x_{w2^{l}+2^{l}})$ где nl/(k+1) первых нулей для w,l,
- $(x_{w2^{l}+1}||x_{w2^{l}+2}|| \ldots ||x_{w2^{l}+2^{l-1}}) < (x_{w2^{l}+2^{l-1}+1}||x_{w2^{l}+2^{l-1}+2}|| \ldots ||x_{w2^{l}+2^{l}})$.

Путём изменения параметров криптосистемы можно достичь широкого диапазона необходимой памяти и время вычисления предложенного алгоритма Proof-of-Work.

В таблице 1 представлены различные экспериментальные характеристики алгоритма Equihash при различных параметрах. В представленной таблице приведены результаты профилирования реализации, написанной на C++ с использованием STL без ассемблерных вставок. Надо учитывать, что для приведённого алгоритма может быть предложен более эффективный метод доступа к памяти, процедура сортировки так же может быть оптимизирована. Это само по себе не противоречит свойству свободы от оптимизации, но указывает на то, что реализации могут не совсем точно соответствовать требованиям к памяти и времени выполнения.

Таблица 1 – Характеристики Equihash при различных переменных

n	**k**	**Пиковая память**	**Время**	**Размер решения**
96	5	$2^{19.2}$	2^{74}	88 байт
128	7	2^{120}	2^{94}	292 байт
160	9	$2^{20.3}$	2^{114}	1.1 кБайт
176	10	$2^{20.4}$	2^{124}	2.2 кБайт
192	11	$2^{20.5}$	2^{134}	4.4 кБайт

Заключение

Описанный здесь общий подход применим для построения произвольного алгоритма вида Proof-of-Work из какой-либо вычислительно сложной задачи. Алгоритм Equihash разрабатывался с учётом таких требований, как устойчивость к ASIC и асимметричность по вычислительной работе и памяти. Его основной является решение проблемы обобщённого парадокса дня рождений. Благодаря наличию переменных величин в алгоритме, потребляемая память и вычислительное время может варьироваться в широких пределах.

Библиографический список:

1. Biryukov, Alex; Khovratovich, Dmitry. "Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem: Open Review". – 2017. – С.27.
2. Dean D., Stubblefield, A. “Using Client Puzzles to Protect TLS.” – 2001. – С. 134.
3. Dwork, C., Naor, M., Wee, H. “Pebbling and Proofs of Work.” Lecture Notes in Computer Science 3621 – 2005. – С. 37–54.
4. Dwork, C., Naor, M. “Pricing via Processing or Combatting Junk Mail.” Lecture Notes in Computer Science. – 1992. – С. 139–147.