УДК 004.056.53

ПОБУДОВА ТА УЗАГАЛЬНЕННЯ МОДЕЛІ СТІЙКОЇ ДО ASIC-АТАК ХЕШ-ФУНКЦІЇ VERTHASH

Д. В. Молдован[1], Н. В. Кучинська[1]

[1] Навчально-науковий Фізико-технічний інститут

Анотація

У даній роботі розглянуто концепцію побудови хеш-функцій які використовуються в алгоритмах консенсусу блокчейн, стійких до ASIC-атак. Ключовими об'єктами дослідження стали функції хешування, побудовані за принципом Dagger-Hashimoto. Визначено загальні принципи побудови алгоритмів такого типу та побудовано математичну модель роботи хеш-функції `verthash`. Проведено порівняльний аналіз безпеки алгоритмів типу Dagger-Hashimoto, стійких до атак з застосуванням ASIC-схем.

Ключові слова: блокчейн, хеш-функція, verthash, ASIC, Proof of Work

Вступ

За останні два десятиліття блокчейн та, зокрема, криптовалюти пройшли шлях від теоретичного концепту «на папері» до зручного інструменту, що дозволив побудувати фінансову систему з великою капіталізацією, який все більше інтегрується в життя людей на побутовому рівні.

На даний час, усі обчислення, необхідні для функціонування блокчейну Bitcoin [1] виконуються великими майнінг фермами, що являють собою обчислювальні датацентри з великою кількістю потужних спеціально розроблених та з'єднаних між собою в один пул апаратних пристроїв (ASIC). Тобто на практиці така система залежить від обмеженого кола користувачів, що уможливлюють функціонування всієї системи.

Саме тому була запропонована криптовалюта Vertcoin, яка у своїй внутрішній структурі базується на принципово відмінному механізмі. Головним в такому механізмі є алгоритм хешування Verthash [2], який декларує місію залишатись доступним для обчислень на GPU (відеокартах), використовуючи підхід, що вимагає багаторазового доступу до пам'яті постійно оновлюючись для протидії реалізації обчислень на ASIC.

1. Постановка проблеми та актуальність дослідження

Постановка проблеми в загальному вигляді

Рішення до наявних проблем хешування за допомогою ASIC-схем існують протягом певного часу і довели свою ефективність на практиці. Однак, для багатьох схем, що є стійкими до ASIC-атак бракує формального опису математичної моделі за якою вони працюють. Зокрема, практична реалізація алгоритму `verthash` відкрита у загальному доступі, але не має створеної аналітичної моделі, що описувала б клас подібних їй функцій хешування. Дослідження криптографічних безпекових властивостей даної моделі варто розглянути як окрему задачу.

Мета дослідження

Для даного класу функцій хешування не повною мірою описана математична модель їх роботи, що ускладнює аналіз їх криптографічних властивостей.

Відсутність повною мірою опису роботи аналітичної моделі функцій гешування типу Dagger-Hashimoto зі складністю, що опирається на введення/виведення даних з пам'яті.

2. Огляд джерел за тематикою дослідження

Функція хешування `ethash`

Функція хешування `ethash` [3] стала першою успішною реалізацією стійких до ASIC-атак хеш-функцій. Вона була випущена як головна функція протоколу Proof of Work криптовалюти Ethereum. Наявна проблема використання ASIC-схем для майнінгу блоків системи блокчейн криптовалюти Bitcoin збільшувала ефективність проведених обчислень у сотні разів, в той час, як для проведення аналогічних атак у блокчейні системи Ethereum, необхідно було спочатку побудувати апаратну складову ASIC-схем нової конфігурації, яка зрештою давала перевагу лише у 2-3 рази, в порівнянні з виконанням обчислень за допомогою графічних процесорів GPU.

2.1. Принципи роботи алгоритму `ethash`

Одним із підходів до розв'язання проблеми централізації майнінгу, що виникла в мережі Bitcoin,

стала розробка алгоритму `ethash`, запропонованого в рамках криптовалюти Ethereum. Даний алгоритм є модифікацією підходу Dagger-Hashimoto та був створений з метою обмеження ефективності спеціалізованих обчислювальних пристроїв. На відміну від класичних алгоритмів Proof-of-Work, `ethash` орієнтований не на обчислювальну складність, а на необхідність інтенсивного доступу до пам'яті.

Функціонування алгоритму базується на використанні великого псевдовипадкового набору даних – DAG (Directed Acyclic Graph), який генерується на основі висоти блокчейну та оновлюється через фіксовані проміжки часу (епохи). У процесі майнінгу на основі заголовка блоку та значення nonce обчислюється початкове значення mix за допомогою хеш-функції типу Keccak. Отримане значення використовується для визначення адрес доступу до елементів DAG, що забезпечує псевдовипадковий характер звернень до пам'яті.

На кожній ітерації алгоритму виконується операція читання даних з DAG та їх поєднання з поточним значенням mix за допомогою функції змішування, зокрема FNV. Даний процес повторюється багаторазово (типово 64 ітерації), після чого отриманий результат стискається до компактного значення (mix digest) і порівнюється з цільовим значенням складності.

Ключовою характеристикою алгоритму є його memory-hard властивість. Для обчислення одного хешу необхідно виконати серію псевдовипадкових звернень до пам'яті загальним обсягом близько 8 КБ, що відповідає приблизно 64 операціям доступу до DAG. Такий підхід унеможливлює ефективне кешування даних і робить продуктивність алгоритму залежною від пропускної здатності пам'яті, а не від обчислювальної потужності пристрою.

Згідно з практичними оцінками, сучасні графічні процесори демонструють продуктивність, близьку до теоретичної межі, визначеної пропускною здатністю пам'яті. Це свідчить про те, що вузьким місцем алгоритму є саме операції доступу до пам'яті. Така властивість суттєво обмежує можливості оптимізації за допомогою ASIC-пристроїв, хоча повністю виключити їх використання не вдалося, і з часом спеціалізовані рішення для `ethash` все ж були створені.

2.2. Схема роботи алгоритму `verthash`

Ключовим елементом алгоритму є файл `verthash.dat`, що являє собою таблицю великого розміру (приблизно 1–1.2 Гб), яка однакова для всіх учасників мережі. Генерація цієї таблиці відбувається детерміновано на основі фіксованого початкового значення (seed), що задається як 16-байтовий масив:

$$S = \texttt{VERTHASHDATSEED}$$

Для кожного індексу i формується вхідний вектор:

$$\text{Input}_i = S \parallel \text{LE32}(i)$$

після чого обчислюється значення:

$$T[i] = \text{SHA3-256}(\text{Input}_i)$$

Отримані значення послідовно записуються у таблицю. Завдяки цьому всі вузли мережі мають ідентичний набір даних, що виключає можливість прихованих оптимізацій та забезпечує узгодженість обчислень.

Процес майнінгу в алгоритмі Verthash базується на інтенсивному використанні цієї таблиці. Нехай H – заголовок блоку розміром 80 байтів, який включає значення nonce. На першому етапі формується набір псевдовипадкових значень за допомогою криптографічної хеш-функції SHA3-512:

$$P_k = \text{SHA3-512}(\text{MutateFirstByte}(H, k)), \quad k = 0, \dots, 7$$

де функція MutateFirstByte змінює перший байт заголовка.

Отримані значення конкатенуються у масив P довжиною 512 байт, який інтерпретується як набір індексів:

$$\text{Index}_j = \text{LE32}(P[4j : 4j + 3]) \bmod N$$

Далі виконується ітеративний процес змішування, в якому використовується акумулятор:

$$A_0 = 0^{32}$$

та на кожній ітерації виконується операція:

$$A_{j+1} = A_j \oplus T[\text{Index}_j]$$

Після завершення всіх ітерацій фінальне значення акумулятора:

$$\text{PoW hash} = A_{final}$$

використовується як результат доказу виконаної роботи. Блок вважається валідним у випадку виконання умови:

$$\text{PoW hash} < \text{Target}$$

Важливою особливістю алгоритму [2] є те, що він передбачає велику кількість випадкових доступів до пам'яті. У процесі майнінгу виконується тисячі операцій читання з таблиці `verthash.dat`, причому індекси цих звернень залежать від значення nonce та заголовка блоку. Це унеможливлює ефективне кешування даних та виключає можливість попереднього обчислення.

3. Математична модель роботи алгоритму `verthash`

3.1. Генерація таблиці даних

Алгоритм `verthash` став досконалішою версією алгоритму `ethash`, покращуючи його роботу в тих місцях, де це можливо. Одним з них є процес генерації таблиці даних, необхідних для роботи алгоритму. На подальших кроках роботи алгоритму, ця таблиця бере участь у процесі майнінгу.

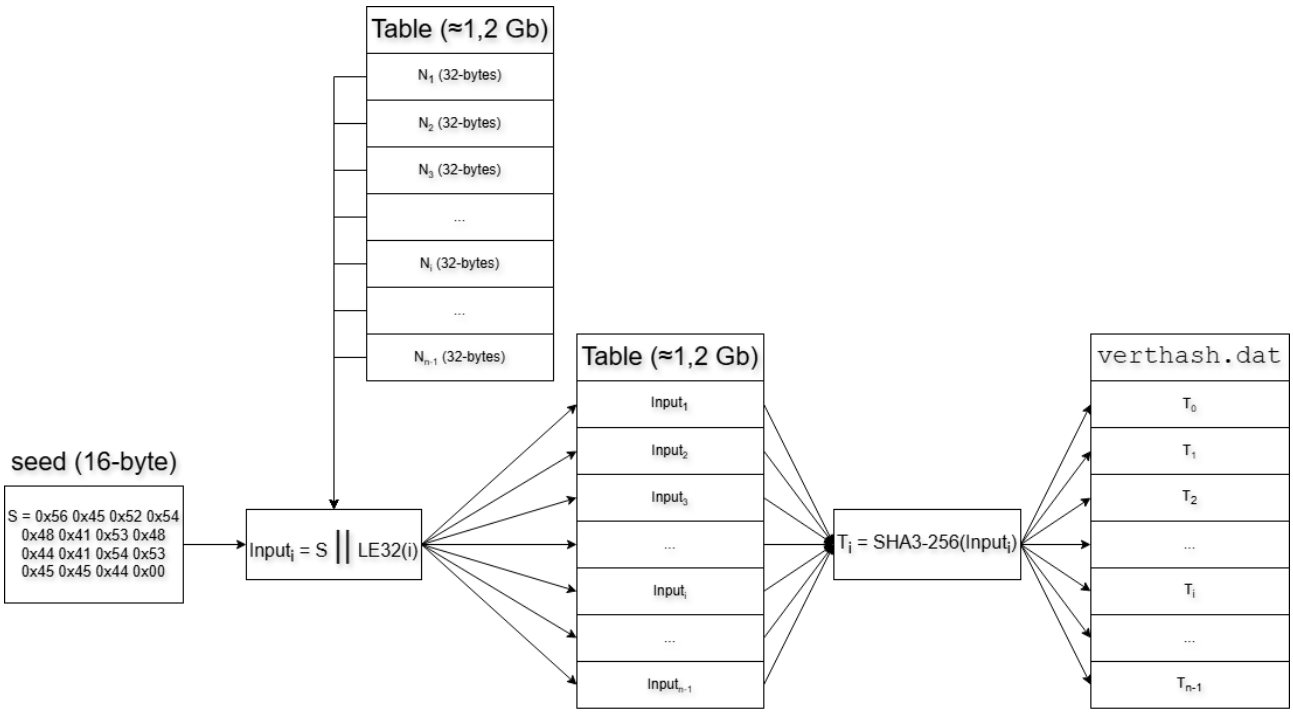

Рис. 1. Генерація табличних даних

3.2. Схема майнінгу за допомогою алгоритму `verthash`

Схема майнінгу криптовалюти vertcoin комбінує в собі як підходи до побудови хеш-функцій за Меркле-Дамгардом [4], так і схеми хешування типу Sponge [4]. Хешування за допомогою алгоритму verthash відбувається у декілька етапів. Спочатку генерується набір індексів, за допомогою алгоритму хешування SHA3-256, який є представником схем хешування типу Sponge. Процес створення набору індексів є рандомізованим, а тому не може бути обчисленим наперед. Тобто процедура доступу до табличних даних, що зберігається у пам'яті забезпечує складність роботи схеми відносно операцій з пам'яттю (memory hard).

Завершальним етапом роботи схеми побудови нового блоку монети vertcoin є міксінг. Даний процес є ітеративним (128 ітерацій), у ньому кожне наступне входження залежить від обрахунку попереднього значення, що безпосередньо відповідає принципам побудови хеш-функцій за схемою Меркле-Дамгарда.

3.3. Порівняння алгоритмів `hashimoto`, `ethash` та `verthash`

Одним з перших варіантів розв'язання проблеми застосування ASIC-схем для майнінгу був алгоритм `hashimoto` [5]. У схемі даного алгоритму вперше була реалізована ідея використання датасету утвореного з складових блокчейну в якості великого набору даних для своєї роботи. Обмежувальним фактором, що запобігає проведенню ASIC-атак є процес зчитування датасету з пам'яті.

Процес роботи даного алгоритму потребує повного доступу до датасету, оскільки схема роботи `hashimoto` передбачає псевдовипадкове зчитування його частин. Це означає що розподілення обсягу роботи даного алгоритму між нодами буде мало ефективним. В такому випадку кожній з них доведеться локально тримати в пам'яті даний датасет.

У схемах роботи даного алгоритму можна виділити два основних етапи: етап створення масиву вказівників для роботи з датасетом та етап міксингу. Саме у внутрішніх процесах даних складових алгоритму і криється відмінність їх роботи. Наприклад, алгоритм `ethash` розширює підхід `hashimoto` до створення вказівників, використовуючи SHA3-подібну функцію хешування та складніші механізми доступу до пам'яті. Використання таких підходів допомагає підвищити загальну ASIC-стійкість схеми.

Алгоритм `verthash` став оновленою версією алгоритму `ethash`, покращивши ASIC-стійкість його схеми. Зокрема, `ethash` використовує 64 операції читання з пам'яті, у межах яких обробляються блоки даних розміром 128 байтів, тоді як Verthash використовує 128 ітерацій із блоками по 256 байтів. Тобто `ethash` використовує 8 Кб пам'яті для обчислення одного хешу, а `verthash` - 32 Кб (Таблиця 1).

Висновки

У межах даного дослідження побудовано формалізовану модель стійкої до ASIC-атак хеш-функції verthash та побудовано відповідні схематичні представлення. Проведений аналіз та узагальнення властивостей функцій, які фокусуються на обмеженні швидкості через доступ до даних (I/O-bound або Dataset-bound), може стати основою для подальших досліджень та визначення шляхів розв'язання проблем, пов'язаних з застосуванням ASIC-атак на алгоритми хешування.

Таблиця 1. Порівняння схеми алгоритмів `hashimoto`, `ethash` та `verthash`

Характеристика	`Hashimoto`	`ethash`	`verthash`
Конфігурація	I/O-bound PoW	Memory-hard PoW	Memory-hard PoW з фіксованим датасетом
Внутрішня хеш-функція	SHA3 (Keccak)	SHA3 + FNV	SHA3-256 / SHA3-512
Розмір датасету	Залежить від реалізації	Зростаючий	Фіксований (1–1.2 ГБ)
Доступ до пам'яті	Псевдовипадковий	Псевдовипадковий	Псевдовипадковий
Ітераційність	Обмежена	64 ітерації	128 ітерацій
Практична цінність	Демонстрація концепції, поштовх до подальших досліджень	GPU-орієнтований майнінг	Орієнтованість на ASIC-resistance

Перелік використаних джерел

1. *Nakamoto S.* Bitcoin: A Peer-to-Peer Electronic Cash System. — SSRN Electronic Journal, 2008. — 9 p. — URL: https://ssrn.com/abstract=3440802.
2. *Developers.* Vertcoin: Algorithmic Adaptation and the Pursuit of Equitable Proof-of-Work. — 2025. — URL: https://vertcoinproject.org/vertcoin_whitepaper.pdf.
3. *Buterin V.* Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. — 2014. — 36 p. — URL: https://ethereum.org/whitepaper/.
4. *Dan B., Victor S.* A Graduate Course in Applied Cryptography. — 2017. — 818 p. — URL: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf.
5. *Dryja T.* Hashimoto: I/O bound proof of work. — 2014. — 5 p. — URL: https://diyhpl.us/~bryan/papers2/bitcoin/meh/hashimoto.pdf.