УДК 004.056.5 **DOI:10.30837/rt.2019.3.198.05**

О.О. КУЗНЕЦОВ, д-р техн. наук, Ю.І. ГОРБЕНКО, канд. техн. наук,
В.В. ОНОПРІЄНКО, канд. техн. наук, І.В. СТЕЛЬНИК, Д.В. МЯЛКОВСЬКИЙ

ДОСЛІДЖЕННЯ АЛГОРИТМІВ КРИПТОГРАФІЧНОГО ҐЕШУВАННЯ, ЯКІ ЗАСТОСОВУЮТЬСЯ В СУЧАСНИХ БЛОКЧЕЙН-СИСТЕМАХ

Вступ

Стаття є продовженням попередньої роботи «Алгоритми криптографічного ґешування, які застосовуються в сучасних блокчейн-системах».

В цій роботі досліджуються сучасні алгоритми ґешування, які застосовуються або можуть застосовуватися в різних блокчейн-системах. Зокрема, розглядаються найбільш поширені та застосовувані алгоритми криптографічного ґешування, які стандартизовані на міжнародному та національному рівнях, а також алгоритми, які хоча і не стандартизовані, але застосовуються у більшості сучасних децентралізованих системах, побудованих за технологією блокчейн. Зокрема, досліджено наступні функції криптографічного ґешування:

- сімейство криптографічних алгоритмів ARGON (ґеш-функції ARGON2D та ARGON2I);
- алгоритм ґешування BALLOON;
- сімейство криптографічних алгоритмів BLAKE (ґеш-функції BLAKE224, BLAKE256, BLAKE384, BLAKE512);
- сімейство криптографічних алгоритмів BMW (ґеш-функції BMW224; BMW256; BMW384; BMW512);
- сімейство криптографічних алгоритмів CUBEHASH (ґеш-функції CUBEHASH224; CUBEHASH256; CUBEHASH384; CUBEHASH512);
- алгоритм ґешування DJB-2;
- сімейство криптографічних алгоритмів ECHO (ґеш-функції ECHO224; ECHO256; ECHO384; ECHO512);
- алгоритм ґешування ED2K;
- сімейство криптографічних алгоритмів EDONR (ґеш-функції EDONR256; EDONR512);
- алгоритм ґешування DAGGER-HASHIMOTO та його подальший розвиток і вдосконалення – алгоритм ETHASH;
- сімейство криптографічних алгоритмів FUGUE (ґеш-функції FUGUE224; FUGUE256; FUGUE384; FUGUE512);
- алгоритм криптографічного ґешування GOST34.11-94-256;
- сімейство криптографічних алгоритмів GROESTL (ґеш-функції GROESTL224; GROESTL256; GROESTL384; GROESTL512);
- сімейство криптографічних алгоритмів HAMSI (ґеш-функції HAMSI224; HAMSI256; HAMSI384; HAMSI512);
- алгоритм ґешування Has160;
- сімейство криптографічних алгоритмів J-H (ґеш-функції J-H224; J-H256; J-H384; J-H512);
- сімейство криптографічних алгоритмів KECCAK (ґеш-функції KECCAK224; KECCAK256; KECCAK384; KECCAK512, які стандартизовані як SHA3);
- алгоритм ґешування Kupyna (ґеш-функції Kupyna256 та Kupyna512);
- алгоритм ґешування LOSELOSE;
- сімейство криптографічних алгоритмів LUFFA (ґеш-функції LUFFA224; LUFFA256; LUFFA384; LUFFA512);
- сімейство криптографічних алгоритмів LYRA (ґеш-функції LYRA2RE; LYRA2REV2);
- сімейство криптографічних алгоритмів MD (ґеш-функції MD4 та MD5);
- алгоритм криптографічного ґешування PANAMA256;
- алгоритм ґешування PROGPOW;
- алгоритм криптографічного ґешування RIPEMD160;

- алгоритм криптографічного ґешування SCRYPT;
- алгоритм криптографічного ґешування SHA1;
- сімейство криптографічних алгоритмів SHA2 (ґеш-функції SHA2-256 та SHA2-512);
- сімейство криптографічних алгоритмів SHABAL (ґеш-функції SHABAL256 та SHABAL512);
- сімейство криптографічних алгоритмів SHAVITE (ґеш-функції SHAVITE224; SHAVITE256; SHAVITE384; SHAVITE512);
- сімейство криптографічних алгоритмів SIMD (ґеш-функції SIMD224; SIMD256; SIMD384; SIMD512);
- сімейство криптографічних алгоритмів SKEIN (ґеш-функції SKEIN224; SKEIN256; SKEIN384; SKEIN512);
- алгоритм ґешування SNEFRU256;
- алгоритм ґешування STREEBOG (у варіантах STREEBOG256 та STREEBOG512);
- алгоритм ґешування TIGER;
- алгоритм ґешування WHIRLPOOL;
- алгоритми ґешування із сімейства «X» (алгоритми X11; X12; X13; X14; X15; X17).

Розглянуті алгоритми ґешування побудовано за різними схемами обробки інформаційних повідомлень та із застосовуванням різних математичних перетворень блоків даних для обчислення ґеш-кодів. Різна ідеологія побудови зазначених алгоритмів обумовлена міркуваннями авторів-дослідників та їх суб'єктивними уявленнями про раціональну побудову функції ґешування за критеріями стійкість/складність. Нашу увагу зосереджено, перш за все, на можливості застосування досліджених алгоритмів при побудові децентралізованих блокчейн-систем, зокрема, криптовалют, проектів розподілених технологій, смарт-контрактів, тощо.

Особливості побудови алгоритмів ґешування у сучасних блокчейн-системах

Алгоритм криптографічного ґешування ARGON2. Функція формування ключа Argon2 була розроблена Алексом Бірюковим, Даніелем Діну і Дмитром Ховратовичем з Університету Люксембургу в 2015 р. [1, 2].

Це сучасний та простий алгоритм, спрямований на високу швидкість заповнення пам'яті та ефективне використання декількох обчислювальних блоків. Алгоритм випущений під ліцензією Creative Commons.

У 2013 р. був оголошений конкурс Password Hashing Competition для створення нової функції ґешування паролів. До нового алгоритму висувалися вимоги щодо обсягу використовуваної пам'яті, кількості проходів по блоках пам'яті і по стійкості до криптоаналізу.

У 2015 р. Argon2 був оголошений переможцем конкурсу. З того часу алгоритм зазнав чотири серйозні зміни. Виправлені частина описів алгоритмів генерації деяких блоків і помилки, додані рекомендовані параметри.

Існують дві версії алгоритму:

– *Argon2d* – підходить для захисту цифрової валюти та інформаційних систем, що не піддаються атакам по стороннім каналам;
– *Argon2i* – забезпечує високий захист від trade-off атак, але працює повільніше версії d через кілька проходів по пам'яті.

Argon2 оптимізований під x86 – архітектуру і може бути реалізований на Linux, OS X, Windows.

Argon2d призначений для систем, де зловмисник не отримує регулярного доступу до системної пам'яті або процесору. Наприклад, для backend-серверів і кріптомайнерів. При використанні одного ядра на 2-Ghz CPU і 250 Mb оперативної пам'яті з Argon2d (p = 2) кріптомайнінг займає 0,1 с, а при застосуванні 4 ядер і 4 Gb пам'яті (p = 8) автентифікація на backend сервері проходить за 0,5 с.

Argon2i більше підходить для frontend-серверів і шифрування жорсткого диску. Формування ключа для шифрування на 2-Ghz CPU, використовуючи 2 ядра і 6 Gb оперативної пам'яті, з Argon2i (p = 4) займає 3 с, в той час як автентифікація на frontend-сервері, задіявши 2 ядра і 1 Gb пам'яті з Argon2i (p = 4), займає 0,5 с.

Алгоритм ARGON2 використовується в деяких криптовалютах, наприклад у MMXVI [3].

Алгоритм криптографічного ґешування BALLOON. Найбільш детальний опис алгоритму ґешування BALLOON наведено у [4].

Алгоритм Balloon було розроблено для ґешування паролів, він приймає чотири вхідних параметри: пароль, сіль, часовий параметр і параметр простору. Вихідні дані представляють собою бітові рядки фіксованої довжини (наприклад, 256 або 512 біт).

Параметр простору (Buer Size) вказує, скільки блоків робочого простору з фіксованим розміром буде потрібно ґеш-функції під час його обчислення. На високому рівні функція з жорсткими вимогами до пам'яті повинна бути «легкою» для обчислення і повинна бути «жорсткою» для обчислення з набагато меншим простором.

Параметр часу (кількість раундів) визначає кількість "обходів" обчислень. Чим більше параметр часу, тим довше буде обчислення ґешу. На платформах, обмежених пам'яттю, системний адміністратор може збільшити кількість раундів ґешування, щоб збільшити вартість обчислення функції без збільшення вимоги до пам'яті алгоритму.

Великий обсяг пам’яті, необхідний для швидкої роботи алгоритму, дозволяє зменшити переваги більш швидких обчислень ASIC, порівняно з CPU та GPU.

Алгоритм Balloon використовується у криптовалюті Deft.

Алгоритм криптографічного ґешування BLAKE. Ґеш-функція BLAKE запропонована Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.-W. Phan [5]. Зазначені ґеш-функції є "налаштованими" версіями, що подані на фінал конкурсу SHA-3. Оригінальні подані функції були названі BLAKE28, BLAKE-32, BLAKE-48 і BLAKE-64; налаштовані версії – BLAKE-224, BLAKE-256, BLAKE-384 і BLAKE-512.

BLAKE – це сім'я з чотирьох ґеш-функцій: BLAKE-224, BLAKE-256, BLAKE-384 і BLAKE-512. Як і алгоритм SHA-2, ґеш-функція BLAKE має 32-бітну версію (BLAKE-256) і 64-бітну версію (BLAKE-512), з якої інші екземпляри виводяться з використанням різних початкових значень, відмінного заповнення і скороченого виводу.

Реалізація BLAKE вимагає невеликих ресурсів і є швидкою як в програмному, так і в апаратному середовищі [5, 6]. У 180нм ASIC, BLAKE-256 може бути реалізований з близько 13500 gate, і може досягати пропускної здатності більше 4 Гбіт/с; BLAKE-512 може досягати пропускної здатності більше 6 Гбіт/с. На Intel Core 2 Duo, BLAKE-256 може ґешувати близько 15 циклів/байт, коли BLAKE-512 приблизно 10 циклів/байт.

Алгоритм BLAKE використовується в криптовалюті Blakecoin [7]. Крім того, цей алгоритм застосовується як складова в алгоритмах X11, X12, X13, X14, X15 та X17 для майнінгу різних криптовалют та розподілених децентралізованих систем [8].

Алгоритм криптографічного ґешування BMW. Криптографічна ґеш-функція BMW (англ. BMW – Blue Midnight Wish) розроблялася як набагато більш ефективна ґеш-функція, ніж SHA-2, в той же час із таким або кращим рівнем безпеки [9].

Алгоритм BMW працює з повідомленнями, розбиваючи їх на блоки. Блок, в свою чергу, ділиться на слова. Розміри блоків і слів залежать від конкретної реалізації алгоритму.

Алгоритм BMW був представлений у якості кандидата на криптографічний конкурс SHA-3. За результатами досліджень був відхилений конкурсною комісією національного інституту стандартів і технологій США: «Рівень безпеки є нижчим за очікуваний: BMW-256 знижується до 65 біт, BMW-512 – до 128 біт. Витрати пам'яті, необхідні для здійснення цих атак, є несуттєвими» [10].

На сьогодні алгоритм криптографічного ґешування знайшов своє практичне застосування в різних децентралізованих системах типу блокчейн. Зокрема, він застосовується як скла-

дова в алгоритмах X11, X12, X13, X14, X15 та X17 для майнінгу криптовалют [8]: DarkCoin (Dark); DeepOnion (ONION); Cloakcoin (CLOAK); MaruCoin (MARU); Hshare (HSR); Stealthcoin (XST); MarteXcoin (MXT) та інш.

Алгоритм криптографічного ґешування CUBEHASH. Сімейство криптографічних ґеш-функцій CubeHashr / b. CubeHash8 / 1 було запропоновано Деніелом Бернштейном [11] в якості нового стандарту в конкурсі ґеш-функцій SHA-3. Спочатку алгоритм вимагав близько 200 циклів на байт [12]. Після деяких уточнень автор змінив параметри на CubeHash16 / 32, який приблизно в 16 разів швидше, ніж CubeHash8 / 1 і легко наздоганяє SHA-256 і SHA-512 на різних платформах [13, 14].

Ґеш-функція CubeHash пройшла до другого раунду конкурсу SHA-3, але так і не потрапила в п'ятірку фіналістів [10].

Варто відзначити, що CubeHash не поступається в швидкості своїм опонентам. Стійкість цього алгоритму збільшується як при зменшенні b до 1, так і при збільшенні r. Тому CubeHash 8 / 1-512 міцніше (більш безпечніше) ніж CubeHash 1 / 1-512, а CubeHash 1 / 1-512 міцніше ніж CubeHash 1 / 2-512. Найслабший з можливих версій даного алгоритму – це CubeHash 1/128-h. Однак безпека залежить від часу, тобто більш безпечний варіант буде довше обчислювати ґеш-значення.

Ґеш-функція CubeHash застосовується як складова у складі алгоритмів майнінгу криптовалют X11,X12, X13, X14, X15, X17.

Алгоритм ґешування DJB-2. Це дуже проста та швидка ґеш-функція, яка була запропонована Даном Бернштейном багато років тому [15]. Одна з найкращих строкових ґеш-функцій, яка має гарне розподілення та швидкість для безлічі наборів ключів та розмірів таблиць, в порівнянні з іншими алгоритмами, наприклад PJW, LOSELOSE та інш. Існує інша версія алгоритму, яка також схвалена Бернштейном, що використовує операцію xor: hash(i) = hash(i-1)*33^str[i]. Пояснення, чому використані такі константи при обчисленнях, не надаються. Ця функція ґешування не є криптографічною, але вона все ж таки використовується у складі алгоритму майнінгу криптовалют X17 [8].

Алгоритм криптографічного ґешування ECHO. ECHO-256 є кандидатом другого туру конкурсу SHA-3 [10]. Це ґеш-функція на базі алгоритму AES, яка викликала великий інтерес і аналіз. ECHO – ітеративна ґеш-функція [16], функція стиснення ECHO оновлює внутрішній стан, описаний в матриці 16 × 16 елементів GF(2^8), яку також можна розглядати як матрицю 4 × 4 з 16 станів AES. Перетворення на такому великому 2048-бітному стані дуже схожі на AES, основна відмінність полягає в еквівалентному S-Box під назвою BigSubWord, який складається з двох раундів AES. В кінці перестановки операція BigFinal додає поточний стан до вихідного стану (подача вперед) і, в разі ECHO-256, додає чотири стовпці разом, щоб отримати нове значення ланцюжка. В обох раундах AES використовуються два ключа – внутрішній лічильник і сіль відповідно. Вони вводяться в основному для того, щоб зруйнувати існуючі симетрії AES-перестановки без ключа.

Ґеш-функція ECHO застосовується як складова алгоритмів майнінгу криптовалют X11, X12, X13, X14, X15, X17 [8].

Алгоритм криптографічного ґешування ED2K. Ґеш-функція ED2K заснована на алгоритмі MD4, але замість надання одного ґешу всього файлу, вона розбиває файл на частини по 9500 Кбайт і створює остаточний ґеш-код на основі сум MD4 [17].

Ґеш-функція ED2K знайшла застосування у файлообмінних мережах та програмах обміну файлами eDonkey2000 і Overnet [18].

Алгоритм криптографічного ґешування EDONR. Сімейство криптографічних ґеш-функцій EdonR запропоновано у роботі [19]. EdonR є класом ґеш-функцій з змінними довжинами виводу. Він визначається за допомогою квазігрупп і перетворень рядків квазігрупп.

У статті [20] детально описано змінену криптографічну ґеш-функцію Edon-R, яку було представлено в якості кандидата на ґеш-конкурс SHA-3, організований Національним інститутом стандартів і технологій (NIST). Різниця полягає в доданні зворотного зв'язку до вихід-

ної функції стиснення R. Зворотній зв'язок полягає в збереженні виведення функції R з попереднім значенням подвійної труби і значенням поточного блоку повідомлень. Введені зміни роблять недійсними криптоаналітичні зусилля з аналізу квазігрупових операцій, використовуваних в EDON-R, а також його функції R.

Швидкість оптимізованої 32-розрядної версії на певній еталонної платформі з Intel C ++ v 11.0.072 становить 6,71 циклів / байт для n = 224, 256 і 10,74 циклів / байт для n = 384, 512. Швидкість оптимізованої 64-бітної версії на певній еталонної платформі з Intel C ++ v 11.0.072 становить 4,90 циклів / байт для n = 224, 256 і 2,74 циклів [19, 20].

Алгоритм криптографічного ґешування ETHASH. Криптографічна функція Ethash застосовується для перевірки працездатності в заснованих на Ethereum криптовалютах [21]. Вона використовує ґеш-функцію Keccak, стандартизовану як SHA-3. Починаючи з версії 1.0 алгоритм Ethash був розроблений так, щоб бути стійким до ASIC завдяки жорсткості пам'яті (важче реалізувати в спеціальних чіпах ASIC) [21-23]. Він також використовує модифіковану версію більш ранніх ґеш-алгоритмів Dagger і Hashimoto для усунення накладних витрат на обчислення. Ця функція раніше називалася Dagger-Hashimoto [22].

Ethash використовує вихідний набір даних обсягом 1 ГБ, відомий як Ethash DAG, і кеш-пам'ять об'ємом 16 МБ для легких клієнтів. Вони відновлюються кожні 30000 блоків, відомих як епоха. Майнери беруть фрагменти DAG для створення змішаних ґеш, використовуючи дані транзакцій, а також криптографічний одноразовий номер для створення ґешу нижче динамічної цільової складності [21 – 23].

Алгоритм Dagger Hashimoto був попередньою дослідницькою реалізацією і специфікацією алгоритму майнінгу для Ethereum 1.0. Пізніше він був замінений на алгоритм Ethash.

Алгоритм криптографічного ґешування FUGUE. Алгоритм ґешування Fugue був розроблений Shai Halevi , William E. Hall і Charanjit S. Jutla з IBM для конкурсу ґеш-функцій Національного Інституту стандартів і технологій у 2009 р., де пройшов до другого раунду [10]. Однак алгоритм не пройшов до третього раунду конкурсу за недостатньою кількістю криптографічного аналізу та невпевненістю в криптостійкості [24].

Для вхідного повідомлення довжиною від 1 до 2^{64}-1 біт алгоритм генерує 224, 256, 384 або 512-бітове ґеш-значення. Функції для відповідних довжин вихідних даних алгоритм ґешування називається відповідно Fugue-224, Fugue-256, Fugue-384 і Fugue-512 [25]. Автори алгоритму також описали параметризовану версію алгоритму Fugue [25]. Слабозахищена версія Fugue-256, що працює в два рази швидше стандартної версії, також описується через параметризовану версію.

Fugue був спроектований таким чином, щоб зменшити вразливість перед атаками диференціального аналізу [25]. Також, за запевненнями авторів алгоритму, ґеш-функція конкурентоспроможна за ефективністю з SHA ґеш-функціями в програмному і апаратному вигляді, досягаючи продуктивності до 36,2 циклів в байт (CPB) на шостому сімействі процесорів Intel Xeon 5150, і до 25 циклів в байт (CPB) на процесорі Intel Core 2 T7700. На 45 нанометровому чіпі Intel Core 2 T9400 Fugue-256 досягає всього 16 циклів в байт (CPB), використовуючи інструкції SSE 4.1. На процесорах з архітектурою Westmere (32нм), типу Intel Core i5, ґеш для Fugue-256 розраховується зі швидкістю 14 циклів в байт (CPB).

Алгоритм Fugue застосовується як складова в алгоритмах X13, X14, X15 та X17 для майнінгу різних криптовалют [8].

Алгоритм криптографічного ґешування GOST34.11-94. Для обчислення криптографічної функції ґешування в Росії у 1994 р. було введено стандарт ГОСТ Р 34.11-94 [26] (який на цей час вже скасовано) [27]. Згодом цей стандарт було перевидано як міждержавний стандарт СНД ГОСТ 34.311-95 [28]. До 2013 р ЦБ РФ вимагав використовувати ГОСТ Р 34.11-94 для формування та перевірки електронного підпису. З 1 січня 2013 р. був замінений на ГОСТ Р 34.11-2012 «Стрибог» [27].

Розробником стандарту ГОСТ 34.311 є Головне управління безпеки зв'язку (ГУБЗ) Федерального агентства урядового зв'язку та інформації (ФАУЗІ) і Всеросійський науково-

дослідний інститут стандартизації. Цей стандарт є обов'язковим для застосування в якості алгоритму ґешування в державних організаціях РФ і ряді комерційних організацій.

Стандарт визначає алгоритм і процедуру обчислення ґеш-функції для послідовності символів. При обробці блоків використовуються перетворення за алгоритмом криптографічного перетворення ГОСТ 28147-89 [29]. Обробляється блок довжиною 256 біт, вихідне значення теж має довжину 256 біт. Алгоритм визначає контрольну суму, розраховану по всіх блоках вихідного повідомлення, яка є частиною фінального обчислення ґешу, що дещо ускладнює атаки на пошук колізій. Застосовуються також заходи боротьби проти пошуку колізій, що засновані на неповноті останнього блоку. Обробка блоків відбувається за алгоритмом шифрування ГОСТ 28147-89, який містить перетворення на S-блоках, що істотно ускладнює застосування методу диференціального криптоаналізу до пошуку колізій.

Алгоритм криптографічного ґешування GROESTL. Ітеративна криптографічний ґеш-функція GROESTL (правильне написання «Grøstl») [30, 31] є однією з п'яти фіналістів конкурсу SHA-3, організованого NIST [10].

Стискаюча функція Grøstl складається з двох фіксованих перестановок P і Q, структура яких запозичена у шифру AES. Зокрема, використовується такий же S-блок. Результат роботи ґеш-функції може мати довжину від 8 до 512 біт з кроком 8 біт. Варіант, який повертає n біт, називається Grøstl-n.

Алгоритм Grøstl спеціально розроблений для участі в конкурсі криптографічних функцій SHA-3 командою криптографов з Данського технічного університету [10]. Спочатку функція називалася Grøstl-0. Однак для участі в фіналі були збільшені структурні відмінності між перестановками. Були змінені значення ShiftBytes в перестановці Q. Також змінам піддалися раундові константи в P і Q. Оновлена ґеш-функція отримала назву Grøstl. Однак, показавши хорошу криптостійкість, по ряду показників вона поступалася іншим учасникам фінального раунду і не змогла стати переможцем.

Назва алгоритму походить від назви блюда Грёстль австрійської кухні. За рецептом воно дуже близько до страви, яке в США називають «Hash». Буква «ö» в назві функції була замінена на букву «ø» з данського алфавіту, яка має таку ж вимову.

За своєю структурою алгоритм Grøstl є байт-орієнтованою SP-мережею. За своєю будовою цей алгоритм значно відрізняється від алгоритмів сімейства SHA (MD-подібних конструкцій). Багато компонентів ґеш-функції Grøstl також запозичені з шифру AES.

Ця функція ґешування є досить ефективним та безпечним криптопримітивом. Знайшла застосування в деяких блокчейн-системах, зокрема у проекті криптовалюти Verge [32].

Алгоритм криптографічного ґешування HAMSI. Криптографічна ґеш-функція Hamsi була запропонована при проведенні конкурсу SHA-3 [10]. В її основу покладено алгоритми Grindahl [33] і Serpent [34]. Ця ґеш-функція не запатентована і є громадським надбанням. Існують два різновиди алгоритму: Hamsi-256 і Hamsi-512 [10].

В основі алгоритму лежать функція розкладання і циклічна трансформація. Циклічна трансформація працює з чотирма рядками матриці станів. Число стовпців цієї матриці дорівнює 4 для Hamsi-256, 8 для Hamsi-512. Елементами матриці є слова розміром 32 біта.

Ґеш-функція Hamsi була одним з учасників у відкритому конкурсі SHA-3 Національного інституту стандартів і технологій. Однак Hamsi не потрапила в число 5 кандидатів останнього туру, оголошених 10 грудня 2010 [10].

Алгоритм Hamsi застосовується як складова в алгоритмах X12, X13, X14, X15 та X17 для майнінгу різних криптовалют [8].

Алгоритм криптографічного ґешування Has160. HAS-160 – це ґеш-функція, яка призначена для використання з корейським стандартом цифрового підпису (який пов'язаний з DSA, але має деякі відмінності). Він дуже схожий на SHA-1, але має деякі зміни, які, ймовірно, збільшують стійкість алгоритму. Він описаний в корейському стандарті під назвою TTAS.KO-12.0011 / R1 [35], але немає опису англійською мовою.

З огляду на дуже широке застосування SHA-1 (а також і SHA-256 і SHA-512), цілком ймовірно, що алгоритму HAS-160 ніколи не буде приділено стільки ж уваги при аналізі або розгортанні, скільки він заслуговує. HAS-160, однак, значно швидше, ніж SHA-1, і (принаймні, на перший погляд) здається приблизно таким же безпечним. Єдина очевидна можлива атака, яку можливо застосувати до HAS-160, це диференціальна атака, яка зламала SHA-0. HAS-160 в основному можна розглядати як продукт переходу до дизайну, який почався з SHA-1 і закінчився як HAS-V. Він використовується в деяких корейських продуктах, а також в KeyTools Crypto v5.0.1 компанії Baltimore Technologies. Є кілька інших ґешів від тих же розробників, які схожі, але не ідентичні, включаючи PMD-V, PMD-128, PMD-160, PMD-192, PMD-224 і PMD-256 (всі з яких є різними алгоритмами). Ні у одного з них немає специфікацій англійською мовою, є дуже мало інформації про них. Алгоритм в значній мірі описується як різниця між ним і SHA-1.

У [36] наведено реалізацію HAS-160 в C ++ як частина крипто-бібліотеки Botan.

У описі алгоритму з [36] поняття «крок» відноситься до окремого застосування підфункції. Наприклад, SHA-1 (а також HAS-160) використовує 4 раунди по 20 кроків кожен, всього 80 кроків.

Алгоритм криптографічного ґешування J-H. Алгоритм JH – це сімейство з чотирьох криптографічних ґеш-функцій: JH-224, JH-256, JH-384 і JH-512 [37]. Кожна з цих ґеш-функцій відрізняється тільки значенням одного внутрішнього параметра – довжини (в бітах) вихідного значення.

Ґеш-функція JH входить в п'ятірку фіналістів другого туру SHA-3 [10]. В процесі цього конкурсу вона була покращена.

Алгоритм JH розроблений і представлений на конкурс SHA-3 в 2008 р. сінгапурським криптографом Wu Hongjun. Алгоритм базується на конструкції «криптографічна губка». Стиснення виконується бієктивною функцією (блочний шифр з постійним ключем). Прототипом для алгоритму шифрування став AES. AES побудований на основі SP-мережі, вхідні дані представляють собою двовимірний масив. У JH-алгоритмі використовується його модифікація: AES узагальнюється до використання багатовимірних масивів, що дозволило побудувати шифр з великою довжиною блоку на основі більш маленьких компонентів.

У 2011 р. запропонована модифікація алгоритму: кількість раундів блочного шифру збільшилася до 42 для підвищення ефективності реалізації на апаратних платформах і поліпшення характеристик безпеки. Незважаючи на це, в цьому ж році представлена атака на перестановку, яка поширюється на всі 42 раунди, а також атака на функцію стиснення, що дозволила отримати псевдоколізії аж до 37 раунду [38]. У 2013 р. з'явилися інші відомості про ряд атак на ґеш-функції, серед яких опинилася і JH.

Алгоритм JH застосовується як складова в алгоритмах X11, X12, X13, X14, X15 та X17 для майнінгу різних криптовалют [8].

Алгоритм криптографічного ґешування KECCAK. Алгоритм криптографічного ґешування Keccak розроблений групою авторів на чолі з Йоаном Дайменом, співавтором Rijndael, автором шифрів MMB, SHARK, Noekeon, SQUARE і BaseKing [39]. 2 жовтня 2012 р. Keccak став переможцем конкурсу криптографічних алгоритмів SHA-3, що проводився Національним інститутом стандартів і технологій США [10]. 5 серпня 2015 р. алгоритм затверджений і опублікований в якості стандарту FIPS 202 [40].

Стандарт FIPS 202 визначає сімейство криптографічних ґеш-функцій SHA-3:

- SHA3-224 – довжина ґеш-значення 224 бітів;
- SHA3-256 – довжина ґеш-значення 256 бітів;
- SHA3-384 – довжина ґеш-значення 384 бітів;
- SHA3-512 – довжина ґеш-значення 512 бітів,

та дві функції розширення (XOF, Extendable Output Functions) SHAKE128 і SHAKE256, для чого повідомлення необхідно доповнювати «суфіксом» з 2 або 4 біт, в залежності від типу функції:

- функція розширення SHAKE 128 (Secure Hash Algorithm Keccak), результат може бути довільної довжини, забезпечує рівень безпеки 128 бітів;
- функція розширення SHAKE256 (Secure Hash Algorithm Keccak), результат може бути довільної довжини, забезпечує рівень безпеки 256.

У програмній реалізації автори заявляють про 12,5 циклів на байт при реалізації на ПК з процесором Intel Core 2. Однак в апаратних реалізаціях Keccak виявився набагато швидше, ніж всі інші фіналісти [41].

На сьогодні алгоритм SHA-3 є однією із найпоширеніших криптографічних ґеш-функцій, яка застосовується у багатьох криптовалютах, наприклад у Nexus (NXS), SmartCash (SMART), X-Cash (XCASH), MaxCoin (MAX), SecureCoin (SRC), Bitcoin File (BIFI), CreativeCoin, Slothcoin (SLOTH), 365Coin (365), Galleon (GLN), Helix Coin (HXC), CryptoMeth (METH), BitcointalkCoin (TALK) та інш. [42, 43].

Алгоритм криптографічного ґешування Kupyna. Національний стандарт України ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція ґешування» (набрав чинності з 1 квітня 2015 р.) описує ітеративну криптографічну ґеш-функцію «Купина». Цей стандарт прийнятий наказом Мінекономрозвитку від 2 грудня 2014 р. №1431 [44]. Текст стандарту є у вільному доступі [45].

Стандарт розроблено задля поступової заміни міждержавного стандарту ГОСТ 34.311-95 [28] та згідно чинних змін до наказу Держспецзв'язку від 20 серпня 2012 р. №1236/5/453 [46] після 1 січня 2022 р. разом з алгоритмом криптографічного перетворення ДСТУ 7624:2014 є обов'язковим для використання при накладанні та перевірці електронного цифрового підпису за ДСТУ 4145-2002 замість функції ґешування за ГОСТ 34.311-95.

Функція стиснення Купини складається з двох фіксованих перестановок, структура яких запозичена у шифра Калина (докладна специфікація англійською мовою наведена у статті [45]). Зокрема, використовуються чотири таких самих S-блоків. Результат роботи ґеш-функції може мати довжину від 8 до 512 біт. Варіант, який повертає n біт, позначається як Купина-n. Основними режимами роботи функції ґешування, що рекомендуються до застосування, є «Купина-256», «Купина-384» і «Купина-512».

За своєю структурою функція ґешування Купина є подібною алгоритму Groestl [30, 31]. Очікується застосування цієї ґеш-функції для побудови національних розподілених блокчейн систем.

Алгоритм ґешування LOSELOSE. Ґеш-функція запропонована в книзі Брайана Кернігана та Денніса Ритчи «Язык программирования Си» (англ. The C Programming Language. - 1978 також відома як K&R), причому останній з авторів – один з безпосередніх авторів та розробників мови програмування Сі [47].

За оптимістичними оцінками, це: «… *не самий кращій алгоритм, але він дуже простий*. *Алгоритм міг би бути набагато кращим, якщо не був би таким простим*…» [47]. Багато програмістів на Сі використовують цю функцію, фактично не перевіряючи її, наприклад, шляхом *сортування та пошуку* Кнута. В деяких розробках ця проста конструкція використовується як одна з компонент в схемах змішування та кодування.

Функція ґешування LOSELOSE не є криптографічною, але вона застосовується у складі алгоритму майнінгу криптовалют X17 [8].

Алгоритм криптографічного ґешування LUFFA. Криптографічний алгоритм (сімейство алгоритмів) ґешування Luffa [48] змінної розрядності, розроблений Даї Ватанабе (англ. Dai Watanabe), Хісайоші Сато (англ. Hisayoshi Sato) з Hitachi Yokohama Research Laboratory і Крістофом Де Канньере (нід. Christophe De Cannière) з дослідницької групи COSIC Льовенського католицького університету для участі в конкурсі Національного інституту стандартів і технологій США [10]. Luffa є варіантом функції губки, запропонованої Гвідо Бертоні (англ. Guido Bertoni) і співавторами, криптостійкість якої заснована тільки на випадковості перестановки. На відміну від оригінальної функції губки, Luffa використовує множину паралельних перестановок і функції інжекції повідомлень.

В ході другого туру конкурсу SHA-3 алгоритми Luffa-224 і Luffa-256 в первинному варіанті показали низьку криптостійкість, для успішної атаки потрібно лише 2^{216} повідомлень. Після чого, алгоритм був модифікований Даї Ватанабе і отримав назву Luffa v.2. Зміни Luffa v.2 є такими [10]:

- доданий порожній раунд функції завершення для всіх розмірів ґешу;
- змінений S-блок;
- збільшено кількість повторень крокової функції з 7 до 8.

Барт Пренель (Bart Preneel) представив успішну атаку [49] з пошуку колізій для 4 раундів крокової функції Luffa за 2^{90} операцій ґеширования і 2^{224} для 5-раундової схеми, показавши тим самим границю стійкості дизайну до диференціальних атак пошуку колізій.

У 2010 р. Томас Олівіера і Джуліо Лопез провели успішні дослідження [50] можливості збільшення продуктивності оригінальної реалізації Luffa. Оптимізована реалізація алгоритму має 20 % збільшення продуктивності обчислення ґешу Luffa-512 при виконанні в 1 потоці, для Luffa-256/384 приріст продуктивності однопоточної реалізації в різних тестах становить не більше 5 %.

Ґеш-функція Luffa застосовується як складова алгоритмів майнінгу криптовалют X11, X12, X13, X14, X15, X17.

Алгоритм криптографічного ґешування LYRA2RE. Найбільш докладний опис цього алгоритму ґешування наведено у [51].

Функція Lyra2RE – це алгоритм ґешування для майнінгу криптовалют, розроблений командою Vertcoin для заміни алгоритму Scrypt-N. Мета полягала в тому, щоб скоротити споживання електроенергії на майнінг токенов VTC і підтримувати монету недоступною для ASIC-майнінгу. Зміну алгоритму було анонсовано в липні 2014 р. і успішно здійснено розробниками [51, 52].

Lyra2RE – алгоритм зі змінними параметрами, які будуть корисні для зриву майбутніх загроз від ASIC пристроїв (літери RE в назві є абревіатурою «Reduced Efficiency» або «знижена працездатність»). За результатами тестів було визначено, що Lyra2RE споживає на ~ 30 % менше енергії в порівнянні, наприклад, з алгоритмом Scrypt-N і на ~ 17 % менше, ніж алгоритм X11. Lyra2RE дозволяє розробникам Vertcoin самостійно змінювати використання пам'яті і вартість часу [51, 52].

Алгоритм Lyra2RE був змінений розробниками Vertcoin на користь GPU-майнерів через одиничний ботнет на базі CPU, який контролював велику частину ґеш-потужності мережі. В результаті 10 серпня 2015 р. (блок 347000) Vertcoin був модифікований від Lyra2RE до Lyra2REv2 [51, 52].

Lyra2REv2 – це алгоритм ґешування для консенсусу Proof-of-work, створений командою криптовалюти Vertcoin (VTC) [51, 52]. Lyra2REv2 складається з ланцюжка різних ґеш-функцій – Blake, Keccak, Cubehash, Lyra2, Skein та BMW (Blue Midnight Wish). Lyra2REv2, заснований на першій версії Lyra2RE. Цей ланцюговий алгоритм безпечний, надійний і призначений для протистояння ASIC-майнінгу. Різниця між Lyra2RE і Lyra2REv2 полягає в більш пізній версії. Були введені 2 раунди Cubehash для зниження ефективності процесора і для того, щоб пом'якшити ефект від майнінгу ботнетів. Lyra2REv2 також споживає менше енергії, ніж попередня версія. Оскільки алгоритм залежить від пам'яті і через його ланцюгову структуру проектування ASIC буде досить складним. Після запуску Lyra2REv2 в Vertcoin (VTC) багато інших криптовалют стали також використовувати даний алгоритм доказу роботи. Деякі кріптовалютні проекти називають алгоритм ґешування Lyra2REv2, інші як Lyra2v2, однак це один і той же алгоритм. За рахунок того, що ґешування має принципову послідовність, так як кожен новий етап використовує результати попереднього, проводити паралельні потоки обчислень стає неможливим. Такий метод побудови є дуже надійним захистом алгоритму від розробки ASIC. І поки це вдається: на сьогоднішній день інтегральної схеми спеціального призначення для алгоритму Lyra2REv2 не поширені.

Паралелізація обчислень неможлива, а значить, для майнінгу потрібно устаткування, яке краще за інших вміє проводити багатомільйонні однотипні операції швидко і результативно, якими і є відеокарти [51, 52].

До популярних криптовалют, які працюють на основі алгоритму Lyra2v2 відносять наступні [51, 52]: Vertcoin (VTC); MonaCoin (MONA); Rupee (RUP); Straks (STAK); Verge (XVG); Shield (XSH); Galactrum (ORE).

Алгоритм криптографічного ґешування MD4. Криптографічна ґеш-функція MD4 (Message Digest 4) була розроблена професором Массачусетського університету Рональдом Ривестом в 1990 р., і вперше описана в RFC 1186 [53]. Для довільного вхідного повідомлення функція генерує 128-розрядне ґеш-значення. Цей алгоритм використовується в протоколі аутентифікації MS-CHAP, розробленому корпорацією Майкрософт для виконання процедур перевірки достовірності віддалених робочих станцій Windows. Є попередником MD5. Рівень безпеки, які закладаються в MD4, були розраховані на створення досить стійких гібридних систем електронного цифрового підпису, заснованих на MD4 і криптосистем з відкритим ключем. Рональд Ривест вважав, що алгоритм ґешування MD4 можна використовувати і для систем, які потребують сильної криптостойкости. Але в той же час він відзначав, що MD4 створювався насамперед як дуже швидкий алгоритм ґешування, тому він може бути поганий в змісті криптостійкості. Як показали подальші дослідження, він мав рацію, і для додатків, де важлива насамперед криптостійкість, став використовуватися більш надійний алгоритм MD5.

Алгоритм криптографічного ґешування MD5. MD5 (англ. Message Digest 5) це 128-бітний алгоритм ґешування, розроблений професором Рональдом Л. Ривестом в 1991 р. для заміни більш ранньої ґеш-функції MD4, був визначений в 1992 р. як RFC 1321 [54].

Широко застосовувався для перевірки цілісності інформації та зберігання ґешів-паролів. На сьогоднішній день безпека цього алгоритму також є недостатньою [55]. Навіть у 2019 р. MD5 продовжує широко використовуватися, незважаючи на добре документовані недоліки і відсутність підтримки з боку експертів з безпеки [56].

Алгоритм криптографічного ґешування PANAMA256. PANAMA – це криптографічний модуль, який можна використовувати як в якості криптографічної ґеш-функції, так і потокового шифру [57]. Він розроблений, щоб бути дуже ефективним в реалізації програмного забезпечення на 32-бітних архітектурах. Його основні операції виконуються над 32-бітними словами.

Алгоритм «Panama» заснований на машині з кінцевими станами, що складається з двох великих блоків: 544 біта станів і 8192-бітового буферу, який працює за принципом регістра зсуву зі зворотним зв'язком. Зворотній зв'язок забезпечує те, що вхідні біти після входу проходять через декілька ітерацій, що, в свою чергу, забезпечує побітову дифузію. Треба сказати, що подібний буфер застосовується в функції стиснення SHA. Об'єктом роботи «Panama» є 32-бітове слово і стан складається з 17 таких слів, в той час як буфер має 32 осередки, в кожній з яких лежить по 8 таких слів.

Ґеш-функція PANAMA піддавалася успішним атакам двічі. Вже в 2001 р. було показано, що дана ґеш-функція не є крипостійкою, так як були знайдені колізії за 2^{82} операцій. Більш того, можна знайти колізії вже за 2^{6} операцій, а для задоволення параметрами надійності необхідно, щоб колізії перебували хоча б за 2^{128} операцій [58].

Алгоритм криптографічного ґешування PROGPOW. ProgPoW – це алгоритм перевірки роботи, призначений для усунення недоліку ефективності, доступного спеціалізованим ASIC [59]. Він використовує практично всі частини GPU, і поставляється заздалегідь налаштованим для найпоширенішого обладнання, що використовується в мережі Ethereum.

З моменту випуску першого ASIC для майнінга біткоїну було створено багато нових алгоритмів Proof of Work, які були створені з метою бути "стійким до ASIC". Мета «стійкість до ASIC» полягає в тому, щоб протистояти централізації потужностуй майнінгу PoW таким чином, щоб цими монетами не могли так легко маніпулювати кілька гравців.

Дизайнерська мета ProgPoW полягає в тому, щоб вимоги алгоритму відповідали тому, що доступно на графічних процесорах: якщо алгоритм повинен бути реалізований на спеціальній ASIC, то має бути мало можливостей для підвищення ефективності порівняно з раніше застосованими GPU.

Новий алгоритм ProgPoW (Programmatic Proof-of-Work) повинен стати спадкоємцем алгоритму Ethash з посиленим захистом від використання ASIC-Майнер. Одна з найбільш популярних кріптовалют – Ethereum – планує перейти з алгоритму Ethash на ProgPoW, а це може вже незабаром призвести до появи нових монет на даному алгоритмі. Першою криптовалютою на алгоритмі ProgPoW стала Bitcoin Interest (BCI) [60].

Алгоритм криптографічного ґешування EQUIHASH. Цей алгоритм найбільш повно описаний у роботі [61].

Функція ґешування Equihash – це вимогливий до пам'яті алгоритм доказу роботи, запропонований Міжгалузевим центром безпеки, надійності та довіри Люксембургу (SnT) у 2016 р. на симпозіумі Network and Distributed System Security. Алгоритм базується на узагальненні проблеми «Дня народження», яка полягає у знаходженні коллізій ґешів. Він має серйозні часово-просторові компроміси, але уразливий до непередбачених паралельних оптимізацій. Алгоритм розроблений таким чином, щоб паралельні реалізації обмежувалися шириною пропускної здатності пам'яті, намагаючись збільшити витрати на розробку спеціальних реалізацій ASIC. Опір ASIC в Equihash базується на припущенні, що комерційний апарат вже має досить високу пропускну здатність пам'яті, тому вдосконалення, зроблене за допомогою спеціального обладнання, може не коштувати вартості розробки.

Алгоритм Equihash є дуже поширеним, його використовують такі криптовалюти: Zcash (ZEC); ZenCash (ZEN); ZClassic (ZCL); Bitcoin Gold BTG); Bitcoin Private (BTCP); MinexCoin (MNX); BitcoinZ (BTCZ); Komodo (KMD); Hush (HUSH).

Алгоритм криптографічного ґешування RANDOMX. RandomX – це алгоритм перевірки працездатності (PoW), оптимізований для процесорів загального призначення. RandomX використовує випадкове виконання коду (звідси і назва) разом з кількома методами, які займають багато пам'яті, щоб мінімізувати перевагу ефективності спеціалізованого обладнання [62].

RandomX поводиться як ключова ґеш-функція: він приймає ключ і довільний вхід і видає 256-бітний результат. У структурі RandomX використовується віртуальна машина, яка виконує програми в спеціальному наборі команд, який складається з комбінації математики цілих чисел, математики з плаваючою комою і гілок. Ці програми можуть бути перетворені в машинний код процесора «на льоту». Прикладом програми RandomX, перекладеної на збірку x86-64, є program.asm [62].

RandomX є доказом роботи (PoW) алгоритму, який був розроблений, щоб закрити розрив між процесорами загального призначення та спеціалізованим обладнанням. Ядром алгоритму є імітація віртуального процесора.

RandomX був спочатку розроблений як алгоритм PoW для проекту Monero, але на сьогодні застосовується і в інших розподілених системах.

Алгоритм криптографічного ґешування RIPEMD160. Криптографічна ґеш-функція RIPEMD-160 (від англ. RACE Integrity Primitives Evaluation Message Digest) була розроблена в Католицькому університеті Лувена Хансом Доббертіном (Hans Dobbertin), Антоном Босселарсом (Antoon Bosselaers) і Бартом Пренелом (Bart Preneel) [63]. Для довільного вхідного повідомлення функція генерує 160-розрядне ґеш-значення.

RIPEMD-160 є покращеною версією RIPEMD, яка, в свою чергу, використовувала принципи MD4 і за продуктивністю порівняння з більш популярною SHA-1.

Також існують 128, 256 і 320-бітові версії цього алгоритму, які, відповідно, називаються RIPEMD-128, RIPEMD-256 і RIPEMD-320. 128-бітна версія являє собою лише заміну оригінальної RIPEMD, яка також була 128-бітною і в якій були знайдені вразливості [64].

256 і 320-бітові версії відрізняються подвоєною довжиною дайджесту, що зменшує ймовірність колізій, але при цьому функції не є більш крипостійкі.

RIPEMD-160 була розроблена в відкритій академічній спільноті, на відміну від SHA-1 і SHA-2, які були створені NSA. З іншого боку, RIPEMD-160 на практиці використовується не так часто, ніж SHA-1. Використання RIPEMD-160 не обмежене будь-якими патентами.

RIPEMD-160 – це ітеративна ґеш-функція, що працює над 32-бітними словами. Раундова функція приймає на вхід 5-слівну зв'язувальну змінну та 16-слівний блок повідомлення та переводить це в нову зв'язувальну змінну. Усі операції визначено над 32-бітними словами. Заповнення таке саме, як у MD4 [53, 54].

Бітовий розмір результату ґешування та зв'язувальної змінної для RIPEMD-160 збільшено до 160 бітів (п'ять 32-бітних слів), кількість раундів збільшено з трьох до п'яти, між двома рядками зроблено більше відмінностей (змінено не тільки сталі величини, але також булеві функції та порядок слів повідомлення).

RIPEMD-128 і RIPEMD-160 уже мають два паралельні рядки, тому розширення з подвійною довжиною (до 256 та 320 бітів відповідно) можливо побудувати без потреби в двох паралельних екземплярах: достатньо відкинути поєднання двох рядків наприкінці кожного застосування функції стиснення.

Алгоритм RIPEMD є однією із найпоширеніших функцій ґешування, який застосовується у багатьох сучасних криптовалютах [65].

Алгоритм криптографічного ґешування SCRYPT. Найбільш подробно цей алгоритм описаний у роботі [66].

Scrypt (читається ес-крипт [66]) – адаптивна криптографічна функція формування ключа на основі пароля, створена офіцером безпеки FreeBSD Коліном Персивалем для системи зберігання резервних копій Tarsnap. Функція створена таким чином, щоб ускладнити атаку перебором за допомогою ПЛІС. Для її обчислення потрібен значний обсяг пам'яті з випадковим доступом. 17 вересня 2012 р. алгоритм scrypt був опублікований IETF у вигляді Internet Draft для подальшого внесення в RFC. Використовується, наприклад, в якості доказу виконаної роботи в криптовалюті Litecoin.

Scrypt використовується в багатьох криптовалютах як алгоритм перевірки роботи. Алгоритм вперше реалізований для Tenebrix (випущений у вересні 2011 р.) і став основою для Litecoin і Dogecoin. Також алгоритм scrypt використовують ProsperCoin, CashCoin, MonaCoin, Mooncoin та багато інших.

Scrypt важко реалізувати на спеціалізованому апаратному пристрої (ASIC), тому для майнінга зазвичай використовують CPU та GPU з програмною реалізацією алгоритму.

Алгоритм криптографічного ґешування SHA1. Secure Hash Algorithm 1 (SHA1) або алгоритм безпечного ґешування 1 – алгоритм криптографічного ґешування, який опубліковано в RFC 3174 [67]. Довжина вхідних повідомлень дозволена максимум 2^{64} -1 біт (що приблизно дорівнює 2 екзабайта). Алгоритм генерує 160-бітне (20 байт) ґеш-значення, що називається також дайджестом повідомлень, і яке зазвичай відображається як шестнадцятирічне число, довжиною в 40 цифр.

Цей алгоритм використовується у багатьох криптографічних додатках і протоколах. Також його рекомендовано в основному для державних установ в США. Принципи, закладені в основу SHA-1, аналогічні тим, які використовувалися Рональдом Ривестом при проектуванні MD4 [53].

Брюс Шнайер доходить такого висновку: «SHA-1 – це MD4 з додаванням розширювального перетворення, додаткового етапу і поліпшеним лавинним ефектом» [68].

Алгоритм криптографічного ґешування SHA2. Ґеш-функція SHA-2 розроблена Агентством національної безпеки (АНБ) США і опублікована Національним інститутом стандартів і технологій у федеральному стандарті обробки інформації FIPS PUB 180-2 в серпні 2002 р. [69]. В цей стандарт також увійшла ґеш-функція SHA-1, розроблена в 1995 р. У лютому 2004 р. в FIPS PUB 180-2 була додана SHA-224 [70]. У жовтні 2008 р. вийшла нова

редакція стандарту – FIPS PUB 180-3 [71]. У березні 2012 р. вийшла остання на даний момент редакція FIPS PUB 180-4, в якій були додані функції SHA-512/256 і SHA-512/224, засновані на SHA-512 (оскільки на 64-бітних архітектур SHA-512 працює швидше, ніж SHA-256) [72].

Таким чином, SHA-2 (англ. Secure Hash Algorithm Version 2 – безпечний алгоритм ґешування, версія 2) є сімейством криптографічних алгоритмів – односпрямованих ґеш-функцій, що включає в себе алгоритми SHA-224, SHA-256, SHA-384, SHA-512, SHA -512/256 і SHA-512/224.

Ґеш-функції сімейства SHA-2 побудовані на основі структури Меркле – Дамгарда. Повідомлення після доповнення розбивається на блоки, кожен блок містить на 8 слів. Алгоритм пропускає кожен блок повідомлення через цикл з 64-ма або 80-ма ітераціями (раундами). На кожній ітерації 2 слова з восьми перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням ґеш-функції.

Алгоритми ґешування SHA-224, SHA-256, SHA-384 і SHA-512 урядом США допускаються до використання в деяких урядових програмах, включаючи використання в рамках інших криптографічних алгоритмів і протоколів, для захисту інформації, що не мають грифу секретності. Стандарт також допускає використання SHA-2 приватними і комерційними організаціями. Отже алгоритми сімейства SHA-2 є чи не найпоширенішими функціями ґешування, які застосовуються у тому числі в розподілених децентралізованих системах типу блокчейн та величезної кількості криптовалют [73].

Алгоритм криптографічного ґешування SHABAL. Алгоритм криптографічного ґешування SHABAL представлений у роботах [74, 75]. Був одним з учасників конкурсу SHA-3, який проводився Національним інститутом стандартів і технологій (NIST) в 2012 р., не зміг вийти у другий раунд. Він був оцінений як найшвидша функція в конкурсі. Цей алгоритм може бути швидким, але йому не вистачає безпеки.

Був представлений на конкурс дослідницьким проектом «Сапфір», спонсором якого є Французьке дослідницьке агентство (ANR), а головною організацією – France Telecom.

Автори алгоритму: Еммануель Брессон, Анна Кантеоут, Беноїт Шевальє-Мамес, Крістоф Клавьере, Томас Фухр, Аліна Гоуджет, Томас Ікарт, Жен-Франсуа Місарскі, Марія Ная-Пласенкія, Паскаль Пайлер, Томас Порніні, Жан-Рене Рейнхард, Селіна Тьюлльет, Маріон Відеау.

Окремі варіації алгоритму SHABAL називаються SHABAL-512, SHABAL-384, SHABAL-256, SHABAL-224, SHABAL-192 в залежності від довжини одержуваного ґеша, відповідно рівного 512, 384, 256, 224, 192 біт.

Після того як на вхід алгоритму приходить бітова послідовність, вона розбивається на блоки по 512 біт незалежно від використовуваної варіації SHABAL (SHABAL-512, SHABAL-384 і т.д.). Розмір блоку кратний 32. До останнього блоку, якщо його бітова довжина не дорівнює 512 бітам, приписується одна бітова одиниця і необхідне число нулів для досягнення заданого розміру блоку.

Алгоритм SHABAL є чимось середнім між схемою Меркле – Дамгаарда і функцією губки (sponge function). Для стійкості цих схем необхідна криптостойкость перетворення P. Розробники SHABAL прагнули до цієї мети. На жаль P виявилося нестійким, але криптоаналитики прийшли до висновку, що безпека SHABAL від цього не постраждала.

Ґеш-функція застосовується як складова алгоритмів майнінгу криптовалют X14, X15, X17.

Алгоритм криптографічного ґешування SHAVITE. Криптографічна ґеш-функція SHAvite розроблена ізраїльськими криптографами Елі Біхамом (англ. Eli Biham) і Ором Дункельманом (англ. Orr Dunkelman) [76]. Одна з чотирнадцяти учасників другого раунду конкурсу SHA-3, організованого NIST [10].

SHAvite-3 заснована на поєднанні компонентів AES с фреймворком HAIFA. Дана ґеш-функція використовує такі криптографічні примітиви, як мережа Фейстеля і конструкція Девіса-Мейєра.

Сімейство ґеш-функцій SHAvite-3 включає в себе два алгоритму – SHAvite-256 і SHAvite-512 [76].

Функція дійшла до другого раунду конкурсу криптографічних функцій SHA, але до фіналу не була допущена за недостатню захищеність ініціалізації S-блоків, що лежать в основі блочного шифру, що призводило до відносно низького рівня безпеки 512-розрядної версії [10].

SHAvite-3 добре підходить для різних платформ і машин, як і AES. Завдяки байт орієнтованії структурі і будівельним блокам AES, SHAvite-3 стає «нативним» для 8-бітних, 32-бітових, 64-бітних машин і фактично для будь-якої машини, яка вже постачає або використовує AES.

Алгоритм SHAVITE застосовується як складова в алгоритмах X11, X12, X13, X14, X15 та X17 для майнінгу різних криптовалют [8].

Алгоритм криптографічного ґешування SIMD. Ітеративна криптографічна ґеш-функція SIMD була розроблена Gaëtan Leurent, Charles Bouillaguet, Pierre-Alain Fouque. Була висунута як кандидат на конкурс стандарту SHA-3, де пройшла до другого раунда [10].

Існують два варіанти ґеш-функції: SIMD-256 і SIMD-512, що перетворюють повідомлення довільної довжини в 256 або 512-бітове ґеш-значення, зване також дайджестом повідомлення. Крім того можливо визначити ґеш-функції SIMD-n як усічення функцій SIMD-256 і SIMD-512 для n <256 і 256 <n <512 відповідно [10].

Як стверджують розробники алгоритму, головною особливістю ґеш-функції є значне розширення повідомлення, яке дозволяє захиститися від диференціального криптоаналізу [10].

Сімейство ґеш-функцій SIMD засноване на двох функціях SIMD-256 і SIMD-512. Крім того специфікація визначає SIMD-n з $n \le 256$ як усічення SIMD-256 і SIMD-n з $256 < n \le 512$ як усічення SIMD-512.

Кожна функція SIMD-n приймає, як вхідне, повідомлення довільного розміру, і виводить дайджест (ґеш-код) з n бітів.

Ґеш-функція SIMD була відібрана в якості фіналіста конкурсу SHA-3. Експерти конкурсу відзначили, що, хоча ґеш-функція SIMD багато в чому повторює алгоритми сімейств MD / SHA, але поліпшення, зроблені авторами, дійсно дозволили захистити SIMD від багатьох типів атак (наприклад, колізійна атака). Крім того, зміни, проведені для другого раунду, змогли захистити ґеш-функцію SIMD від атаки на основі диференціального криптоаналізу [10]. Однак високі вимоги до RAM і наявності SIMD інструкцій для гарної продуктивності роблять ґеш-функцію поганим кандидатом для реалізації на FPGA [10]. Головним чином з цієї причини ґеш-функція SIMD не потрапила у фінальну стадію конкурсу.

Алгоритм SIMD застосовується як складова в алгоритмах X11, X12, X13, X14, X15 та X17 для майнінгу різних криптовалют [8].

Алгоритм криптографічного ґешування SKEIN. Алгоритм ґешування Skein (англ. Skein) змінної розрядності було розроблено групою авторів на чолі з Брюсом Шнайєром [77].

Ґеш-функція Skein була створена в 2008 р. і увійшла до п'ятірки фіналістів конкурсу SHA-3, однак в 2012 р. у фіналі переможцем був обраний алгоритм Keccak, найбільш продуктивний і нечутливий до вразливостей SHA-2 [78]. Назва ґеш-функції Skein означає «моток пряжі».

Ґеш-функція Skein виконана як універсальний криптографічний примітив, на основі блочного шифру Threefish, що працює в режимі UBI-ґешування. Основною концепцією розробки була оптимізація під мінімальне використання пам'яті, криптографічно безпечне ґешування невеликих повідомлень, стійкість до всіх відомих атак на ґеш-функції, оптимізація під 64-розрядні процесори й активне використання звернень до таблиць.

Skein підтримує розміри внутрішнього стану 256, 512 і 1024 біт і розмір вихідного блоку до 2 64 −1 біт. Автори заявляють про 6.1 такт на байт для будь-якого розміру вихідного блоку на ПК з процесором Intel Core 2 Duo. З числа кандидатів на SHA-3 Skein входить в п'ятірку найшвидших, проте є лідером лише в 64-розрядному варіанті, який перевершує за швидкісними характеристиками 32-розрядний у більш ніж чотири рази. Це пояснюється тим, що автори спочатку орієнтувалися на оптимізацію під 64-розрядні процесори [77].

Skein-512 може бути реалізований з використанням всього 200 байт пам'яті, а спрощена версія Skein-256 – із використанням 100 байт, що оптимально для апаратної реалізації алгоритму в смарт-картах [78].

Як заявляють автори, ґеш-функція Skein на поширених процесорах працює в середньому два рази швидше SHA-512, Threefish в два рази швидше AES.

Skein захищена від нових видів атак на ґеш-функцій – підбору подовжених повідомлень і псевдоколізій.

Threefish, що лежить в основі Skein має дуже просту структуру і може бути використаний для заміни алгоритмів блочного шифрування, будучи швидким і гнучким шифром, що працює в довільному режимі шифрування. Сам Threefish не використовує S-блоки, натомість покладається на комбінації інструкцій XOR, складання і циклічного зсуву.

Область застосування Skein досить широка. Використовуючи повідомлення і ключ в якості відповідних входів, можна обчислити КАП. За допомогою аргументу Nonce використовувати Skein в режимі потокового шифру. Також можливе застосування в якості генератора псевдовипадкових чисел, наприклад в алгоритмах Fortuna і Yarrow, як Key Derivation Function і Password-Based Key Derivation Function (використовуючи аргументи Key і Key Derivation Identifer), в якості ґеш-функції для обчислення електронного підпису (мається на увазі використання аргументу Public Key).

Ґеш-функція Skein застосовується як складова у складі алгоритмів майнінгу криптовалют X11, X12, X13, X14, X15, X17. Одна з відомих криптовалют, яка видобувається на алгоритмі Skein через майнінг, є DigiByte (DGB).

Алгоритм криптографічного ґешування SNEFRU. Криптографічна ґеш-функція Snefru була запропонована Ральфом Меркле (сама назва Snefru, продовжуючи традиції блокових шифрів Khufu і Khafre, також розроблених Ральфом Меркле, являє собою ім'я єгипетського фараона) [79, 80]. Функція Snefru перетворює повідомлення довільної довжини в ґеш довжини m (зазвичай m = 128 або m = 256).

Snefru – це ітеративна ґеш-функція, яка спирається на будову Меркла – Дамгара. Її було спроектовано як таку, яка ґешує повідомлення довільної довжини в 128-бітові значення (також був представлений 256-бітний варіант за тією самою будовою). Snefru використовує схему заповнення, яка завжди додає додатковий блок заповнення з довжиною повідомлення (на відміну від компактнішої схеми заповнення MD4 , яка додає ще один блок тільки за потреби).

Використовуючи засоби диференційного аналізу, Елі Біхам і Аді Шамір показали, що двопрохідні функція Snefru не є стійкою до колізій 1-го роду і 2-го роду [81].

Алгоритм криптографічного ґешування STREEBOG. Чинний російський криптографічний стандарт ГОСТ Р 34.11-2012 «Информационная технология. Криптографическая защита информации. Функция ґеширования» визначає алгоритм і процедуру обчислення ґеш-функції СТРИБОГ. Алгоритм було розроблено Центром захисту інформації та спеціального зв'язку ФСБ Росії за участю ВАТ «ІнфоТеКС» [82] і введений в дію 1 січня 2013 р. [83].

Стандарт визначає алгоритм і процедуру обчислення ґеш-функції для послідовності символів. Цей стандарт розроблений і введений в якості заміни застарілого стандарту ГОСТ Р 34.11-94.

Основними параметрами алгоритму є: розмір ґешу – 256 або 512 біт; розмір блоку вхідних даних – 512 біт.

На конференції Crypto-2015 Алекс Бірюков, Лео Перрін і Олексій Удовенко представили доповідь, в якій говориться про те, що значення S-блоку блокового симетричного шифру Кузнечик і ґеш-функції Стрибог не є (псевдо) випадковими числами, а згенеровані на основі прихованого алгоритму, який доповідачам вдалося відновити методами зворотного проектування [84]. 29 січня 2019 р. було опубліковано дослідження «Partitions in the S-Box of Streebog and Kuznyechik» [85], яке спростовує заяву авторів про випадковий виборі параметрів таблиць замін в алгоритмах Стрибог і Кузнечик [86].

Платформа блокчейна izzz.io з відкритим вихідним кодом і інтелектуальними контрактами з застосовує криптографічні бібліотеки із реалізованим алгоритмом STREEBOG. Цю платформу застосовують компанії BigNet, BitCoen, Buzcoin, Baikalika, NWP Solution, SBS Platform, NS Platform [87].

Алгоритм криптографічного ґешування TIGER. Криптографічна ґеш-функція Tiger була розроблена Росом Андерсоном і Елі Біхамом в 1995 р. [88].

Tiger був призначений для особливо швидкого виконання на 64-розрядних комп'ютерах [89]. Tiger не має патентних обмежень, може використовуватися вільно як з еталонною реалізацією, так і з її модифікаціями.

Розмір значення ґешу – 192 біта (Tiger / 192), хоча є також більш короткі версії для сумісності з SHA-1 (Tiger / 160) і з MD4, MD5, RIPEMD, Snefru (Tiger / 128). Швидкість роботи – 132 Мбіт / с (перевірено на одному процесорі Alpha 7000, модель 660). На сучасних процесорах значно швидше (навіть при тесті на 32-бітному AMD Sempron 3000+ швидкість близько 225 Мбіт / с) [90].

Tiger2 – версія Tiger, яка відрізняється від основної тільки іншим алгоритмом додавання бітів, подібним з MD5 / SHA-1.

Tiger використовується в технології TTH, де ґеш обчислюється в деревовидній формі. TTH, в свою чергу, застосовується в протоколах файлового обміну Gnutella, Gnutella2, Direct Connect, а також в файлообмінниках Phex, BearShare, LimeWire, Shareaza, DC ++ і Valknut.

Алгоритм криптографічного ґешування WHIRLPOOL. Криптографічна ґеш-функція Whirlpool була розроблена Вінсентом Рейменом і Пауло Баррето [91]. Опублікована в листопаді 2000 р. Ґешує вхідне повідомлення з довжиною до 2^{256} бітів. Вихідне значення ґеш-функції Whirlpool становить 512 бітів.

Ґеш-функція Whirlpool названа в честь Галактики Вир (M51) в сузір'ї Гончі Пси – першої відкритої галактики з спіральною структурою.

З моменту створення в 2000 р. Whirlpool двічі модифікувалася.

Перша версія Whirlpool-0 була представлена як кандидат в проекті NESSIE (англ. New European Schemes for Signatures, Integrity and Encryption, Нові Європейські Проекти з Цифрового Підпису, Цілісності та Шифрування).

Модифікація Whirlpool-0, названа Whirlpool-T, в 2003 р. додана до переліку рекомендованих до використання криптографічних функцій NESSIE. Зміни стосувалися блоку підстановки (S-box) Whirlpool: в першій версії структура S-box була описана, і він генерувався довільно, що створювало певні проблеми при апаратній реалізації Whirlpool. У версії Whirlpool-T S-box «придбав» чітку структуру.

Дефект в дифузних матрицях Whirlpool-T, виявлений Тайдзо Сірано і Кёдзі Сібутані, згодом виправлений, і кінцева (третя) версія, названа для стислості просто Whirlpool, прийнята ISO в стандарті ISO / IEC 10118-3 : 2004 у 2004 р.

Ґеш-функція WHIRLPOOL застосовується як складова алгоритмів майнінгу криптовалют X14, X15, X17.

Алгоритми криптографічного ґешування сімейства «X». У 2014 р. був представлений новий тип криптографічного методу ґешування – алгоритм X11, а трохи пізніше його більш досконалі версії X12, X13, X14, X15 і X17. Перша криптовалюта, блокчейн якої побудований на X11, була DarkCoin [92].

Число в назві алгоритму позначає кількість раундів ґешування і видів функцій, які використовує даний алгоритм. Наприклад, алгоритм X13 використовує 13 ґеш-циклів з 13 різними криптографічними функціями, що робить його одним з найбільш надійних в сучасному світі криптовалют [8].

В табл. 1 наведено перелік використовуваних функцій в кожній алгоритмічної версії.

Таблиця 1

Перелік функцій в кожній версії алгоритму майнінга сімейства «X»

Ґеш-функція	X11	X12	X13	X14	X15	X17
BLAKE	+	+	+	+	+	+
BMW	+	+	+	+	+	+
GROESTL	+	+	+	+	+	+
J-H	+	+	+	+	+	+
KECCAK	+	+	+	+	+	+
SKEIN	+	+	+	+	+	+
LUFFA	+	+	+	+	+	+
CUBEHASH	+	+	+	+	+	+
SHAVITE	+	+	+	+	+	+
SIMD	+	+	+	+	+	+
ECHO	+	+	+	+	+	+
HAMSI		+	+	+	+	+
FUGUE			+	+	+	+
SHABAL				+	+	+
WHIRLPOOL					+	+
LOSELOSE						+
DJB-2						+

Початкове завдання X11 – запобігання проблем з централізацією системи Dash [93]. Надмірна простота SHA-256 могла стати причиною різкого цінового обвалу криптовалюти, оскільки велика ймовірність того, що велика частина цифрової валюти буде зосереджена в руках кількох впливових пулів.

X-алгоритми були створені спеціально для роботи на графічних процесорах, де вони забезпечують високу рентабельність і низьке енергоспоживання. Кожен результат підфункції потім передається в наступний під-алгоритм і так відбувається X раз. Таким чином, створення ASIC-обчислювачів для такого методу буде ускладнене, так як апаратне забезпечення повинне буде оптимізоване під кожен алгоритм, що сильно збільшує складність виробництва і вартість обчислювального обладнання. Можливо згодом виробники ASIC-обчислювачів зможуть розробити моделі для алгоритмів серії X (наприклад, для X11 таке обладнання вже є), але доцільність його використання знаходиться під великим питанням. Щоб зламати, наприклад, X13, потрібно знайти вразливість у всіх 13 ґешах, що набагато складніше, ніж для одного алгоритму, наприклад, SHA-256.

На сьогодні алгоритми ґешування сімейства «X» широко застосовуються в сучасних блокчейн-системах, зокрема криптовалютах. Наприклад, алгоритм X11 застосовуються для майнінгу наступних криптовалют: DeepOnion (ONION); Cloakcoin (CLOAK); MaruCoin (MARU); Hshare (HSR); Stealthcoin (XST); MarteXcoin (MXT). Алгоритм X16 використовується як основний засіб майнінгу у криптовалютах: Stone Coin; Ravencoin; Proton Coin; Gravium; HTHCoin; Motion. І цей перелік постійно зростає, отже поширюється практичне застосування алгоритмів криптографічного ґешування сімейства «X».

Проведений аналіз показує, що різні за своєю побудовою алгоритми застосовують різні математичні перетворення та окремі функціональні схеми. Розглянуті алгоритми ґешування застосовуються (або можуть застосовуватися) як основний криптографічний елемент технології блокчейн, тобто вони застосовуються у понад 90 % існуючих проектів децентралізованих систем [8, 93, 94].

Висновки

В роботі розглянуто переважну більшість відомих та широко розповсюджених алгоритмів ґешування, які застосовуються або можуть бути застосовані найближчим часом у протоколах консенсусу сучасних блокчейн-мереж. Проаналізовано як стандартизовані на міжнародному та національному рівні алгоритми ґешування, так і ґеш-функції, які були представлені на різних криптографічних конкурсах на науково-пошукових проектах. Зокрема встановлено, що більшість проектів розподілених мереж використовує надійні та перевірені часом алгоритми криптографічного ґешування (наприклад, алгоритми KECCAK, SHA2, RIPEMD160, тощо). Але останніми роками для захисту від ASIC-майнерів почали застосовуватися і інші ґеш-функції, які хоча і можуть бути навіть швидшими за KECCAK, SHA2 або RIPEMD160, але володіють певними вразливостями стосовно властивостей необоротності. Як приклад можна навести застосування криптографічних алгоритмів MD4, EDONR-256, EDONR-512, ED2K (надійність та безпечність яких на сьогоднішній день є незадовільною), або навіть найпростійших функцій DJB-2 та LOSELOSE (ці алгоритми не є криптографічними, а являють собою, по суті, звичайну контрольну суму). Через простоту обчислення певних показників не можна нехтувати порушенням властивостей необоротності, особливо якщо саме на них базуються основні переваги блокчейн-мереж. Отже перспективним є дослідження як швидкодії криптографічного ґешування, так і безпеки відповідних алгоритмів, що буде розглянуто у наступних роботах.

Список літератури:

1. The password hash Argon2, winner of PHC. Електронний ресурс. Режим доступу: https://github.com/P-H-C/phc-winner-argon2

2. Argon2. By Dmitry Khovratovich. 30 March 2015. Електронний ресурс. Режим доступу: https://www.cryptolux.org/index.php/Argon2

3. Bitcoin Forum. Електронний ресурс. Режим доступу: https://bitcointalk.org/index.php?topic=1318683.0

4. Dan Boneh, Henry Corrigan-Gibbs, and Stuart Schechter. Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks. 12.05.2017. Електронний ресурс. Режим доступу: https://eprint.iacr.org/2016/027.pdf

5. SHA-3 proposal BLAKE. Електронний ресурс. Режим доступу: https://131002.net/blake/

6. BLAKE2 — fast secure hashing. Електронний ресурс. Режим доступу: https://blake2.net/

7. About Blakecoin. Електронний ресурс. Режим доступу: https://blakecoin.org/about-blakecoin/

8. Алгоритм X13 для майнинга на графических процессорах. Александр Марков. 28 мая 2018. Електронний ресурс. Режим доступу: https://miningbitcoinguide.com/mining/sposoby/x13

9. Danilo Gligoroski, Vlastimil Klima, Svein Johan Knapskog, Mohamed El-Hadedy, Jørn Amundsen, Stig Frode Mjølsnes. Blue Midnight Wish. Trondheim, Norway: Norwegian University of Science and Technology, 2008. P. 71.

10. NIST. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register, 72(112), November 2007. Електронний ресурс. Режим доступу: http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

11. CubeHash specification (2.B.1) Daniel J. Bernstein. Електронний ресурс. Режим доступу: http://cubehash.cr.yp.to/submission2/spec.pdf

12. CubeHash efficiency estimates (2.B.2). By Daniel J. Bernstein. Електронний ресурс. Режим доступу: http://cubehash.cr.yp.to/submission/estimates.pdf

13. CubeHash parameter tweak: 16 times faster.By Daniel J. Bernstein. Електронний ресурс. Режим доступу: http://cubehash.cr.yp.to/submission/tweak.pdf

14. Single Block Attacks and Statistical Tests on CubeHash. By Benjamin Bloom, Alan Kaminsky. August 21, 2009. Електронний ресурс. Режим доступу: http://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1986&context=article

15. Bernstein hash djb2. Електронний ресурс. Режим доступу: https://riot-os.org/api/group__sys__hashes__djb2.html

16. ECHO hash function. Електронний ресурс. Режим доступу: https://crypto.orange-labs.fr/echo/

17. Ed2k-hash. 7 May 2005. Електронний ресурс. Режим доступу: https://wiki.anidb.info/w/Ed2k-hash

18. ed2k-tools. Tools for eDonkey2000 and Overnet. Електронний ресурс. Режим доступу:http://ed2k-tools.sourceforge.net/index.shtml

19. Edon–R, An Infinite Family of Cryptographic Hash Functions. Danilo Gligoroski, Smile Markovski and Ljupco Kocarev. May 2009. Електронний ресурс. Режим доступу: https://pdfs.semanticscholar.org/e901/492cbb9d1f8a4365397676da808a9d9415cc.pdf

20. D. Gligoroski et al. Cryptographic hash function Edon-R′ // 2009 Proceedings of the 1st International Workshop on Security and Communication Networks, Trondheim, 2009, pp. 1-9.

21. The Ethereum Wiki Електронний ресурс. Режим доступу: https://github.com/ethereum/wiki

22. Dagger Hashimoto. Електронний ресурс. Режим доступу: Https://github.com/ethereum/wiki/wiki/Dagger-Hashimoto

23. Ethash Design Rationale. Електронний ресурс. Режим доступу: Https://github.com/ethereum/wiki/wiki/Ethash-Design-Rationale

24. NISTIR 7764. Status Report on the Second Round of the SHA-3 Cryptographic Hash Algorithm Competition. Електронний ресурс. Режим доступу: https://csrc.nist.gov/publications/detail/nistir/7764/final

25. Hash Function Fugue. Електронний ресурс. Режим доступу: https://researcher.watson.ibm.com/researcher/view_group.php?id=3302

26. Криптографическая защита информации функция хэширования. ГОСТ Р 34.11-94. Електронний ресурс. Режим доступу: https://pdf.standartgost.ru/catalog/Data2/1/4294824/4294824580.pdf

27. Министерство промышленности и торговли Российской Федерации. Федеральное агентство по техническому регулированию и метрологии. Об утверждении национального стандарта. ПРИКАЗ от 7 августа 2012 года N 216-ст. Електронний ресурс. Режим доступу: http://docs.cntd.ru/document/902368268

28. ГОСТ 34.311-95. Информационная технология. Криптографическая защита информации. Функция хэширования. Дата введения 1995-01-01. Електронний ресурс. Режим доступу: http://docs.cntd.ru/document/gost-34-311-95

29. Системы обработки информации. Защита криптографическая. ГОСТ 28147-89. Дата введения 01.07.90. Електронний ресурс. Режим доступу: https://files.stroyinf.ru/Data2/1/4294826/4294826631.pdf

30. Grøstl a SHA-3 candidate. By Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schlaffer, and Søren S. Thomsen. March 2, 2011. Електронний ресурс. Режим доступу: http://www.groestl.info/Groestl.pdf

31. Grøstl a SHA-3 candidate. Електронний ресурс. Режим доступу: http://www.groestl.info/team.html

32. Verge Currency. Електронний ресурс. Режим доступу: https://vergecurrency.com/

33. Grindahl a family of hash functions. Lars R. Knudsen, Christian Rechberger and Søren S. Thomsen Електронний ресурс. Режим доступу: https://web.archive.org/web/20120915162204/http://www2.mat.dtu.dk/people/Lars.R.Knudsen/grindahl/grindahl.pdf

34. SERPENT. A Candidate Block Cipher for the Advanced Encryption Standard. Електронний ресурс. Режим доступу: https://www.cl.cam.ac.uk/~rja14/serpent.html

35. Telecommunications Technology Association. Hash Function Standard Part 2: Hash Function Algorithm Standard (HAS-160). TTAS.KO-12.0011/R1, December 2000. Електронний ресурс. Режим доступу: https://www.tta.or.kr/include/Download.jsp?filename=stnfile/TTA-0072.pdf

36. A Description of HAS-160. 2002-10-01. Електронний ресурс. Режим доступу: https://www.randombit.net/has160.html

37. JH. Електронний ресурс. Режим доступу: https://ehash.iaik.tugraz.at/wiki/JH

38. María Naya-Plasencia, Deniz Toz, Kerem Varici. Rebound Attack on JH42 // Advances in Cryptology ASIACRYPT 2011, Vol. 7073 of Lecture Notes in Computer Science, 2011, pp. 252-269, Springer, 2011. Електронний ресурс. Режим доступу: https://link.springer.com/chapter/10.1007/978-3-642-25385-0_14

39. The sponge and duplex constructions. By Team Keccak: Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche and Ronny Van Keer. Електронний ресурс. Режим доступу: https://keccak.team/sponge_duplex.html

40. NIST Releases SHA-3 Cryptographic Hash Standard. August 05, 2015. Електронний ресурс. Режим доступу: https://www.nist.gov/news-events/news/2015/08/nist-releases-sha-3-cryptographic-hash-standard

41. NISTIR 7896 Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition. Електронний ресурс. Режим доступу: https://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf

42. SHA-3 Coins. Електронний ресурс. Режим доступу: https://cryptorival.com/algorithms/sha3/

43. Keccak hashing algorithm (SHA-3) Keccak Coins and miner for Keccak. Електронний ресурс. Режим доступу: https://coinguides.org/keccak-algorithm-miner-coins/

44. Наказ «Про прийняття національних стандартів України, гармонізованих з європейськими стандартами, міжнародних стандартів як національних стандартів України, затвердження національних стандартів України, скасування міждержавних стандартів в Україні та внесення зміни до наказу Державного комітету стандартизації, метрології та сертифікації України від 12.06.2002 № 357» Електронний ресурс. Режим доступу: https://zakon.rada.gov.ua/rada/show/v1431731-14

45. A New Standard of Ukraine: The Kupyna Hash Function. Roman Oliynykov1, Ivan Gorbenko, Oleksandr Kazymyrov,Victor Ruzhentsev, Oleksandr Kuznetsov, Yurii Gorbenko, Artem Boiko, Oleksandr Dyrda, Viktor Dolgov, Andrii Pushkaryov. Електронний ресурс. Режим доступу: https://eprint.iacr.org/2015/885.pdf

46. Наказ «Про затвердження вимог до форматів, структури та протоколів, що реалізуються у надійних засобах електронного цифрового підпису». 20 серпня 2012 р. Електронний ресурс. Режим доступу: https://zakon.rada.gov.ua/laws/show/z1398-12

47. The C Programming Language by Brian W. Kernighan (1978-02-22) Paperback, Prentice Hall, 178 p.

48. The Hash Function Family Luffa (Round 2 Archive). Електронний ресурс. Режим доступу: http://www.hitachi.com/rd/yrl/crypto/luffa/index.html

49. Finding Collisions for Reduced Luffa-256 v2. By Bart Preneel, Hirotaka Yoshida, and Dai Watanabe. Електронний ресурс. Режим доступу: http://www.hitachi.com/rd/yrl/crypto/luffa/FindingCollisionsForReducedLuffa-256v2_20101108.pdf

50. mproving the performance of Luffa Hash Algorithm. Thomaz Oliveira1, Julio Lopez. August 19, 2010. Електронний ресурс. Режим доступу: https://eprint.iacr.org/2010/457.pdf

51. Lyra2RE A new PoW algorithm for an ASIC-free future. By Vertcoin Developers. Електронний ресурс. Режим доступу: https://cryptorating.eu/whitepapers/Vertcoin/Vertcoin_Lyra2RE_Paper_11292014.pdf

52. Lyra2REv2. Електронний ресурс. Режим доступу: https://en.bitcoinwiki.org/wiki/Lyra2REv2

53. MD4 Message Digest Algorithm. RFC 1186. Last updated 2013-03-02. Електронний ресурс. Режим доступу: https://datatracker.ietf.org/doc/rfc1186/

54. The MD5 Message-Digest Algorithm. Електронний ресурс. Режим доступу: https://www.ietf.org/rfc/rfc1321.txt

55. MD5 vulnerable to collision attacks. Електронний ресурс. Режим доступу: https://www.kb.cert.org/vuls/id/836068/

56. A quarter of major CMSs use outdated MD5 as the default password hashing scheme. Електронний ресурс. Режим доступу: https://www.zdnet.com/article/a-quarter-of-major-cmss-use-outdated-md5-as-the-default-password-hashing-scheme/

57. The Panama Cryptographic Function. By Joan Daemen and Craig Clapp, December 01, 1998. Електронний ресурс. Режим доступу: http://www.drdobbs.com/security/the-panama-cryptographic-function/184410745

58. Joan Daemen and Craig Clapp. Fast Hashing and Stream Encryption with Panama. Електронний ресурс. Режим доступу: https://link.springer.com/content/pdf/10.1007/3-540-69710-1_5.pdf

59. Обзор «асикоустойчивого» алгоритма ProgPOW для GPU-майнинга / Александр Марков. 10 октября 2018. Електронний ресурс. Режим доступу: https://miningbitcoinguide.com/mining/sposoby/progpow

60. Company Coinmarket. Електронний ресурс. Режим доступу: https://coinmarket.news/2019/01/20/progpow-obzor-svezhih-majnerov-dlya-novogo-algoritma/

61. Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem / Dmitry Khovratovich, Alex Biryukov. Електронний ресурс. Режим доступу: http://orbilu.uni.lu/bitstream/10993/22277/2/946.pdf

62. Proof of work algorithm based on random code execution. RandomX. Електронний ресурс. Режим доступу: https://github.com/tevador/RandomX

63. The hash function RIPEMD-160. Електронний ресурс. Режим доступу: http://homes.esat.kuleuven.be/~bosselae/ripemd160.html

64. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. By Xiaoyun Wang , Dengguo Feng, Xuejia Lai3, Hongbo Yu. August 17, 2004. Електронний ресурс. Режим доступу: http://eprint.iacr.org/2004/199.pdf

65. Cryptography behind top 20 cryptocurrencies. Електронний ресурс. Режим доступу: https://www.susanka.eu/coins-crypto/

66. Colin Percival. Stronger key derivation via sequential memory-hard functions. 2009. Електронний ресурс. Режим доступу: https://en.bitcoinwiki.org/wiki/Scrypt http://www.tarsnap.com/scrypt/scrypt.pdf

67. US Secure Hash Algorithm 1 (SHA1). By P. Jones. September 2001. Електронний ресурс. Режим доступу: https://www.ietf.org/rfc/rfc3174.txt

68. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. Москва : Триумф, 2002. 816 с.

69. Secure Hash Standard. Federal Information. Processing Standards Publication 180-2. 2002 August 1. (FIPS PUB 180-2) Електронний ресурс. Режим доступу: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

70. FIPS Publication 180-2 (with Change Notice 1). Електронний ресурс. Режим доступу: https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2withchangenotice.pdf

71. Secure Hash Standard (SHS). FIPS PUB 180-3. October 2008. Електронний ресурс. Режим доступу: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf

72. Secure Hash Standar. FIPS PUB 180-4. August 2015. Електронний ресурс. Режим доступу: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

73. SHA-256 Coins. Електронний ресурс. Режим доступу: https://cryptorival.com/algorithms/sha256/

74. Shabal, a Submission to NIST's Cryptographic Hash Algorithm Competition. Initiated by the Saphir project. 28.10.2008 Електронний ресурс. Режим доступу: https://www.cs.rit.edu/~ark/20090927/Round2Candidates/Shabal.pdf

75. Status Report on the Second Round of the SHA-3 Cryptographic Hash Algorithm Competition / Meltem Sönmez Turan, Ray Perlner, Lawrence E. Bassham, William Burr, Donghoon Chang, Shu-jen Chang, Morris J.

Dworkin, John M. Kelsey, Souradyuti Paul, Rene Peralta. 12.2011 Електронний ресурс. Режим доступу: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7764.pdf

76. The SHAvite-3 Hash Function. By Eli Biham and Orr Dunkelman. Електронний ресурс. Режим доступу: http://www.cs.technion.ac.il/~orrd/SHAvite-3/Spec.31.10.08.pdf

77. The Skein Hash Function Family Version 1.3 1 Oct 2010. By Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, Jesse Walker. Електронний ресурс. Режим доступу: http://www.skein-hash.info/sites/default/files/skein1.3.pdf

78. NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition. Created October 02, 2012, Updated December 11, 2018 Електронний ресурс. Режим доступу: https://www.nist.gov/news-events/news/2012/10/nist-selects-winner-secure-hash-algorithm-sha-3-competition

79. Ralph C. Merkle. A fast software one-way hash function // Journal of Cryptology. 1990. 3 (1): 43–58.

80. Cryptohash: snefru256. Електронний ресурс. Режим доступу: https://snefru256.cryptohash.net/

81. Eli Biham, Adi Shamir. Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer (Extended Abstract)

82. Company Infotecs. Електронний ресурс. Режим доступу: http://www.infotecs.ru/

83. ГОСТ Р 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хэширования. Дата введения 2013-01-01. Електронний ресурс. Режим доступу: http://docs.cntd.ru/document/gost-r-34-11-2012

84. Конкурс «Streebog». Открытый конкурс научно-исследовательских работ, посвященных анализу криптографических качеств хэш-функции ГОСТ Р 34.11-2012. Електронний ресурс. Режим доступу: http://www.streebog.info/

85. Cryptology ePrint Archive: Report 2019/092 Partitions in the S-Box of Streebog and Kuznyechik. By Léo Perrin. 29 Jan 2019. Електронний ресурс. Режим доступу: https://eprint.iacr.org/2019/092

86. Очередные странности в алгоритмах ГОСТ Кузнечик и Стрибог. 11 февраля 2019. Електронний ресурс. Режим доступу: https://habr.com/ru/company/virgilsecurity/blog/439788/

87. IZZZIO. Електронний ресурс. Режим доступу: https://en.bitcoinwiki.org/wiki/IZZZIO

88. A Tiger Hash Implementation for C#. 10 Mar 2012. Електронний ресурс. Режим доступу: https://www.codeproject.com/Articles/149061/A-Tiger-Hash-Implementation-for-C

89. Електронний ресурс. Режим доступу: http://th.informatik.uni-mannheim.de/People/Lucks/papers/Tiger_FSE_v10.pdf

90. Cryptanalysis of the Tiger Hash Function. Електронний ресурс. Режим доступу: https://online.tugraz.ac.at/tug_online/voe_main2.getvolltext?pDocumentNr=81263

91. LARC Laboratório de Arquitetura e Redes de Computadores. Електронний ресурс. Режим доступу: http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html

92. Company Dash. Електронний ресурс. Режим доступу: http://dash.org/

93. X11 алгоритм добычи криптовалюты с 11 раундами хэширования / Александр Марков. 23 мая 2018 г. Електронний ресурс. Режим доступу: https://miningbitcoinguide.com/mining/sposoby/x11

94. Алгоритмы майнинга криптовалют – таблица 2019 и краткое описание. Електронний ресурс. Режим доступу: https://mining-cryptocurrency.ru/algoritmy-kriptovalyut/.

Харківський національний
університет імені В.Н. Каразіна;
АТ «Інститут інформаційних технологій»;
Адміністрація Державної служби спеціального зв'язку
та захисту інформації України.

Надійшла до редколегії 02.09.2019