Бутенко С.І.

Національний аерокосмічний університет ім. М.Є. Жуковського "Харківський авіаційний інститут", Харків

**АНАЛІЗ ШВИДКОДІЇ ПОПУЛЯРНИХ АЛГОРИТМІВ ГЕШУВАННЯ ДЛЯ ВИКОРИСТАННЯ У ВИСОКОНАВАНТАЖЕНИХ СИСТЕМАХ**

**Вступ**

Майже з початку розвитку комп'ютерних технологій вчені висунули ідею про необхідність створення одно направленої функції перетворення даних. На сьогодні подібну роль відграють функції гешування.

У загальному випадку геш-функція - це математичний алгоритм, який перетворює дані довільного розміру в бітовий масив фіксованого розміру. На даний час до таких алгоритмів висувається ряд вимог, а саме:

* незворотність.
* стійкість до колізій.
* фіксований розмір вихідної послідовності.
* при незначній видозміні вхідного повідомлення значення вихідної послідовності повинно значно змінюватися.

Велика кількість сучасних інформаційних технологій побудована з використанням геш-функцій. Вони широко застосовуються в сферах електронного документообігу(електронний цифровий підпис), при створенні баз даних задля оптимізації зберігання та пошуку даних, для контролю цілісності файлів та/або даних(контрольна сума), в системах контролю доступу для значного ускладнення компрометації паролів та інших даних у разі їх отримання зловмисниками та в багатьох інших сферах[1].

Розглядаючи використання геш-функцій в рамках високонавантажених систем слід звертати увагу не лише на їх швидкодію, а також на крипостійкість. Через це в рамках даного дослідження не розглядаються окремі алгоритми гешування, які визнані застарілими або ті, для яких знайдено метод пошуку колізій за осяжний проміжок часу. Для прикладу, до таких алгоритмів можна віднести: MD4[2], MD5[2] та більш ранні версії цих алгоритмів, SHA-1[3], RIPEMD-128[4]. До даної категорії відносяться майже всі алгоритми з довжиною вихідного повідомлення 128 або менше біт.

Метою даного дослідження є провести порівняльний аналіз популярних алгоритмів гешування, що визнаються більшістю дослідників як достатньо криптостійкі, за параметром швидкодії, та визначити, які з них більш оптимальні для використання в високонавантажених системах.

**Виклад основного матеріал**

По результатам проведеного аналізу наукових публікацій був сформований перелік алгоритмів гешування, що є найбільш розповсюдженими та достатньо крипостійкими. До цього переліку увійшли: SHA-2 та SHA-3, як ті, що визначені державних стандартом США[5], ГОСТ 34.311-95(використовувався до 2014 року) та ДСТУ 7564:2014, як державні стандарти України в різний період часу[6], RIPEMD-160 та RIPEMD-256, як один зі стандартів, що використовується в Європейському союзі[7], SM-3 як державний стандарт Китаю[8], а також Whirlpool[9], що зазначений як один з актуальних алгоритмів гешування в стандарті ISO/IEC 10118-3:2018. Усі алгоритми, окрім окремо зазначених, використовувалися в версіях з довжиною вихідного гешу в 256 біт.

Для проведення аналізу швидкодії встановлений фіксований розміри вхідного повідомлення 512 кілобайт. Також для проведення експерименту була створена віртуальна машина на базі операційної системи Windows Server 2019 задля мінімізації впливу сторонніх програм на хід експерименту.

За результатами проведення експерименту був отриманий відповідний набір даних, що відображений графіками, що подані на рисунку 1:

Рисунок 1 – Порівняння алгоритмів для вхідного повідомлення 512 кілобайт за кількістю операцій гешування в секунду

**Висновки**

Проаналізувавши отримані результати експерименту можна зазначити, що найбільшу продуктивність показав алгоритм RIPEMD256. Розроблені в Україні алгоритми ДСТУ 7564:2014 та ГОСТ 3411-95 мають кардинально нижчу продуктивність. Подібний результат викликаний тим, що в даних алгоритмах використовуються більш складні математичні перетворення, що викликає збільшення часу проведення кожної ітерації. Алгоритм Whirpool показав себе схожим чином, та також має відносно низьку продуктивність. В свою чергу алгоритми SHA2, SHA3, SM3 та RIPEMD160 показали відносно високу продуктивність, але нижчу на RIPEMD256.

Роблячи висновки з проведеного аналізу слід зазначити, що в умовах відсутності законодавчих обмежень по використанню алгоритмів, для побудови високонавантажених систем рекомендується використовувати алгоритм RIPEMD256. В свою чергу використання алгоритмів ДСТУ 7564:2014, ГОСТ 3411-95 та Whirpool не рекомендується через їх відносно низьку продуктивність.

**Література**

1. MILLER S. AN INTRODUCTION TO HASH FUNCTIONS. Department of Mathematics | The University of Chicago. URL: https://math.uchicago.edu/~may/REU2020/REUPapers/Miller.pdf (дата звернення: 06.07.2024).
2. Improved Collision Attacks on MD4 and MD5 / Y. SASAKI та ін. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2007. E90-A, № 1. С. 36–47. URL: https://doi.org/10.1093/ietfec/e90-a.1.36 (дата звернення: 06.07.2024).
3. Announcing the first SHA1 collision. Google Online Security Blog. URL: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html (дата звернення: 06.07.2024).
4. Landelle F., Peyrin T. Cryptanalysis of Full RIPEMD-128. Journal of Cryptology. 2015. Т. 29, № 4. С. 927–951. URL: https://doi.org/10.1007/s00145-015-9213-5 (дата звернення: 06.07.2024).
5. FIPS PUB 180-4. Secure Hash Standard (SHS). Чинний від 2015-08-04. Вид. офіц. Gaithersburg, MD, 2015. 36 с. URL: https://doi.org/10.6028/NIST.FIPS.180-4 (дата звернення: 04.07.2024).
6. ДСТУ 7564:2014. Інформаційні технології. КРИПТОГРАФІЧНИЙ ЗАХИСТ ІНФОРМАЦІЇ.Функція гешування. Чинний від 2015-04-01. Вид. офіц. Київ, 2015. 39 с. URL: https://usts.kiev.ua/wp-content/uploads/2020/07/dstu-7564-2014.pdf (дата звернення: 04.07.2024).
7. B. Preneel, H. Dobbertin and A. Bosselaers, "The new cryptographic hash function RIPEMD-160", Dr. Dobb's Journal 22(1), pp. 24-28, 1997.
8. SM3 Cryptographic Hash Algorithm (Chinese Standard). crypto on x86 and ARM. URL: https://tinycrypt.wordpress.com/2017/02/22/asmcodes-sm3/ (date of access: 04.07.2024).
9. Stallings W. The Whirlpool Secure Hash Function. Cryptologia. 2006. Т. 30, № 1. С. 55–67. URL: https://doi.org/10.1080/01611190500380090 (дата звернення: 06.07.2024).