КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (3), 2019
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2019.3.112121
УДК 004.627

Пасєка Микола Степанович
кандидат технічних наук, доцент, доцент кафедри інженерії програмного забезпечення
Івано-франківський національний технічний університет нафти і газу, Івано-Франківськ, Україна
OrcID 0000-0002-3058-6650
pms.mykola@gmail.com

Пасєка Надія Мирославівна
кандидат технічних наук, доцент,
Пикарпатський національний університет ім. В. Стефаника, Івано-Франківськ, Україна
OrcID 0000-0002-4824-2370
pasyekanm@gmail.com

Бестильний Михайло Ярославович
аспірант кафедри інженерії програмного забезпечення
Івано-Франківський національний технічний університет нафти і газу, Івано-Франківськ, Україна
OrcID: 0000-0001-9957-8854
pz@nung.edu.ua

Шекета Василь Іванович
доктор технічних наук, доцент, завідувач кафедри інженерії програмного забезпечення
Івано-Франківський національний технічний університет нафти і газу, Івано-Франківськ, Україна
OrcID: 0000-0002-1318-4895
vasylsheketa@gmail.com

АНАЛІЗ ВИКОРИСТАННЯ ВИСОКОЕФЕКТИВНОЇ РЕАЛІЗАЦІЇ ФУНКЦІЙ ХЕШУВАННЯ SHA-512 ДЛЯ РОЗРОБКИ ПРОГРАМНИХ СИСТЕМ

Анотація. Функції хешування відіграють прикладну та фундаментальну роль у сучасному захисті програм та даних методами криптографії. Як правило такі функції захисту передають дані кінцевої довжини у той-же час виробляючи незначний та фіксованого розміру сигнал. Поряд із лавиноподібним зростаючим обсягом даних, які потребують швидкої перевірки, пропускна властивість хеш-функцій стає ключовим фактором. Згідно з науковими дослідженнями опублікованими у даний час в технічній літературі, одна із найбільш швидких реалізацій SHA-512 а це варіант реалізації SHA-2, який забезпечує пропускну здатність алгоритму понад 1550 Мбіт/с проте є і швидші такі як, Whirlpool де пропускна здатності понад 4896 Мбіт/с. На даний час було опубліковано багато робіт, що обговорюють апаратні реалізації SHA-512. Усі розглянуті реалізації, як правило, спрямовані на високу пропускну здатність або ефективне використання обчислювальних ресурсів. Взагалі неможливо завчасно знати, який вибір функціонального дизайну для даного компонента буде найкращим у досягненні специфічної мети дизайну. Після реалізації та виконання алгоритму з різними компонентами можна було провести системний аналіз та прокоментувати якість даної реалізації, оскільки мета стосується досягнення високої пропускної здатності або низької загальної обчислювальної потужності. Ми систематизували результати усіх проведених обчислень та провели аналіз кожної реалізації окремо. Детально сформували опис етапів розширення та стиснення повідомлень. Аналогічно на різних етапах і згадується стадія оновлення хешу, однак її реалізація не завжди чітко визначена. Однією з причин пропускати подробиці попереднього етапу і етапу оновлення хешу є те, що він передбачає, що ці етапи будуть реалізовані таким чином, щоб мінімізувати негативний вплив на нього. Розглянута у статті функція перемішування даних не претендує на найвищу пропускну здатність алгоритму, проте вона виявилась достатньо стійкою для стороннього декодування. Підсумовуючи наші наукові дослідження в області криптографічного захисту різними методами ми можемо стверджувати, що розроблені на основі алгоритму SHA-512 прикладне програмне забезпечення відповідає наступним технічним параметрам, а саме верифікацію цілісності програм та даних і достатньо надійний алгоритм автентифікації.

ISSN 2663 - 4023

Ключові слова: криптографія, алгоритм, хеш-функція, програмне забезпечення, захист програмних систем.

1. ВСТУП

Прикладне використання криптографічних хеш-функцій тісно пов'язане із загальними хеш-функціями, які мають додаткові властивості, а загальні хеш-функції не обов'язково їх мають. Такі функції як правило є відображення кінцевого входу у незначний вихід фіксованого розміру, який ще іноді називають дайджестом. У той же час як хороша функція хешування має попередній опір до відображення, другий опір до відображення і стійкість до декодування.

Під стійкістю до попереднього відображення мається на увазі, що вона повинна бути алгоритмічно непридатною для пошуку будь-якого входу, який продукує дайджест та є рівний наперед заданому виходу. Друге попереднє відображення означає, що запропонований алгоритм повинен мати характеристики, які не давали б можливості знайти інший другий вхід, який має ідентичний вихід, як і будь-який запропонований вхід. Під поняттям стійкості до зіткнень ми розуміємо, такі умови при яких неможливо знайти обчислювальний ресурс для двох різноманітних входів, які мають один і той же вихід. Розглянуті функції хешування можна розділити на дві групи, а саме: keyed and un-keyed. Функції хеш шифрування використовуються у алгоритмах автентифікації повідомлень. Ці хеш-функції беруть в якості вхідних потоків даних тільки хешовані, а також 2-й параметр, що називається ключем шифрування. Хеш алгоритм використовує потоки фактичних даних разом з наданим шифрувальним ключем для генерування вихідного хешу.

Функція un-keyed зазвичай використовуються для перевірки цілісності даних. Вихідні потоки даних, які пройшли через хеш-функцію та отримали дайджест (статус) збережено. На наступному кроці алгоритму копія потоку даних хешуються і порівнюються два хеші. Якщо результат позитивний то всі виміри оригінальні дані, а копія ідентична. Однак, якщо вони відрізняються, то цілісності копії не можна довіряти у повній мірі, оскільки дані можуть бути змінені у результаті некоректної передачі (помилки), або, ще гірший варіант, вони можуть бути змінені зовнішнім впливом для нанесення шкідливих наслідків. Більш ґрунтовний опис та системний аналіз хеш-функцій можна знайти у [1], [2], [6], [7], [10].

При розгляді функцій хешування ми побачили, що вони вимагають значних обчислювальних ресурсів. Внаслідок ітераційного характеру таких функцій, неможливо скоротити реалізацію цих алгоритмів таких інноваційними методами, як паралелізм із метою скорочення часу на опрацювання та підвищення продуктивності. Використання процесорів загального призначення не є ефективним підходом для ефективного виконання основних операцій алгоритму, необхідних для заданого перемішування потоків даних. Проте для забезпечення підвищення продуктивності апаратних реалізацій науковці та практики невпинно працюють над вирішенням даної проблеми. При появі нових алгоритмів хеш-функцій повинні також існувати способи порівняння їх. Поряд з тим важливо розрізняти особливості реалізації прикладного та апаратного забезпечення при виконанні функцій порівнянь. Реалізація прикладного програмного забезпечення, що виконуються на процесорах для загального призначення, використовуються у багатьох протоколах автентифікації та автономних програмах. Поряд із тим апаратна реалізація, яка створена за допомогою спеціалізованих мікро схем, дуже використовується у технічних пристроях, де практично неможливо

використати прикладні програмну реалізацію, таку як вбудовані програмні системи або високошвидкісні розподілені мережеві пристрої. Час затрачений на виконання та використання оперативної пам'яті - це є дві метрики, які використовуються для порівняння результату роботи прикладного програмного забезпечення. Отже чим менше витрачено часу та виконання та використання оперативної пам'яті, тим кращий алгоритм функції хешування. В апаратно-обчислювальних комплексах час витрачений на виконання, а також загальна потужність є двома ключовими показниками, які постійно необхідно враховувати. Для такої реалізації алгоритму хешування можна стверджувати, що час виконання функції може бути значно зменшено за рахунок збільшення кластерної потужності обчислювального комплексу. Обчислювальна потужність можна розрахувати із погляду на фізичний розмір даних, отриманих у результаті виконання та за кількістю логічних вентилів, що використовуються для хеш функції при її реалізації. Обчислювальні пристрої, такі як FPGA, забезпечують високо оптимізовані операції, які виконують алгоритми хешування. У результаті така апаратна реалізація значно перевершує прикладна програмне забезпечення на кілька порядків.

2. ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

Хеш-функцій, досліджених в даній науковій роботі, дають 512 біт хеш-значень. Проте SHA-512 працює на 1024-бітних блоках даних. Хеш-функція SHA-512 є одним з п'яти варіантів, що належать до сімейства SHA-функцій хешування та вона працює на повідомленнях довжиною менше 2128 біт або більших 1025 терабайт. В основі хеш-функції SHA-512 є ітераційний алгоритм, що складається з чотирьох кроків, а саме додавання повідомлень, розширення повідомлень, стиснення повідомлень і оновлення хешу. Для опрацювання одного блоку даних виконуються перший крок стиснення повідомлень або раунд. На (рис. 1) відображена блок-схема високого рівня кроків, що беруть участь у алгоритмі хешування SHA-512. Більш детальна діаграма показана на (рис. 2). Реалізація алгоритму SHA-512 вимагає послідовної оцінки шести логічних функцій кожного кроку. Загалом алгоритм цих функції складаються з простих операцій, таких як зміщення і повернення.

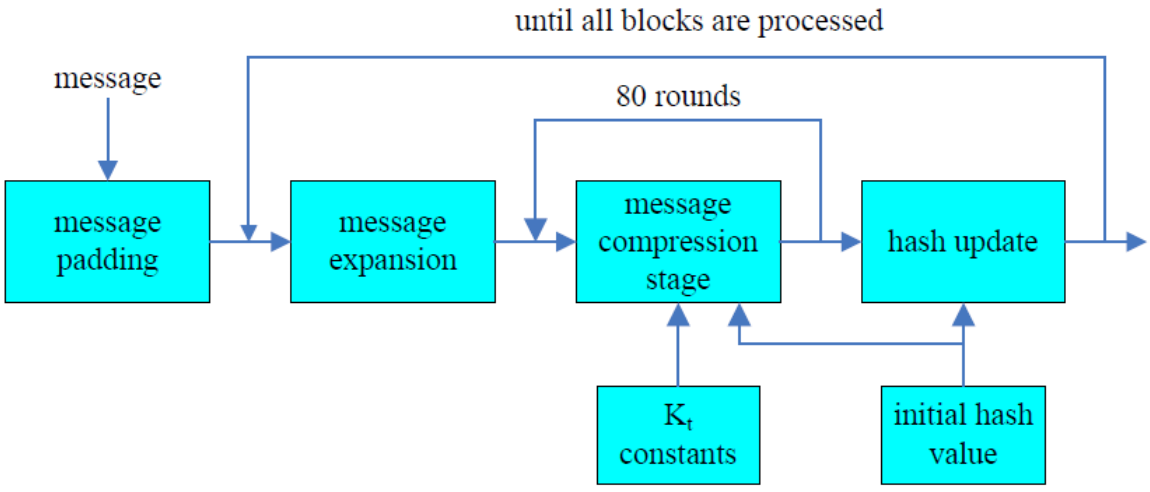


Рис. 1. Структурна схема високого рівня, що відображає структуру алгоритму хешування SHA-512

ISSN 2663 - 4023

Алгоритм також вимагає двох наборів констант: початкового значення хешу і матриці Kt, що складається з перших 64 біт дробових частин коренів куба перших чисел 80. Kt – константи використовуються на стадії стиснення повідомлень алгоритму. Спочатку до кінця інформаційного повідомлення додається один біт "1", за яким слідом йде змінна кількість бітів "0". Кількість доданих бітів "0" є такою величиною, що довжина інформаційного повідомлення стає кратною 896. Отже до довжини початкового інформаційного повідомлення, перш ніж буде виконано будь-яке додавання, додано ціле значення 128 біт без знаку. Як наслідок, алгоритм видає повідомлення, яке є точно кратним 1024 біт.

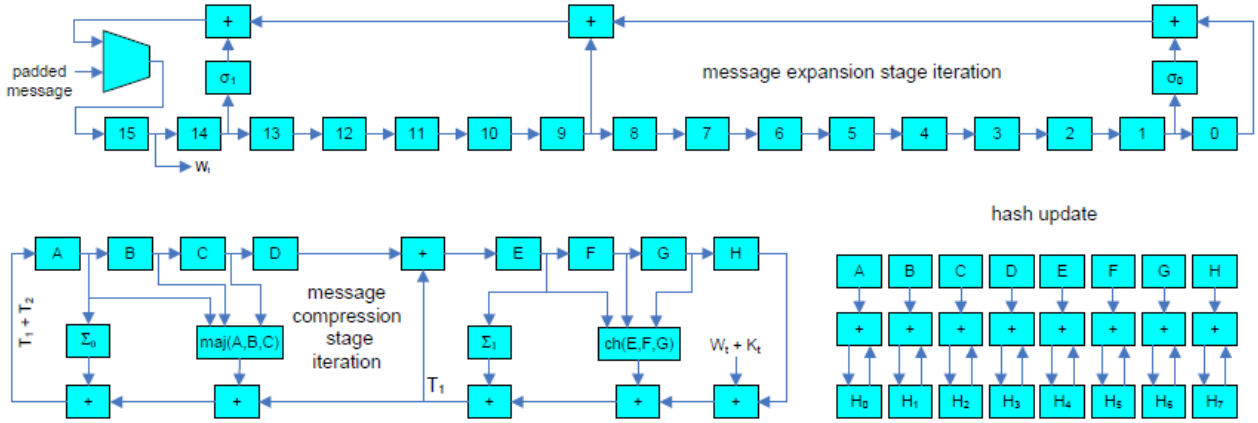


Рис. 2. Детальна блок-схема, що відображає структуру алгоритму хешування SHA-512

Етап розширення повідомлення генерує графік повідомлень, виконуючи ряд операцій з інформаційними даними, що містяться в поточному 512-бітовому блоці. Ця інформація відображається у графіку повідомлень, що складається з 5120 біт даних, оскільки кожна Wt має довжину 64 біти, поряд з тим існує 80 таких Wt, по одному для кожного кроку ітерації графіка повідомлень. Ми повинні також припускати, що перші 1024 біти графіка повідомлень складаються просто з блоку даних, який у даний час експлуатується. На стадії стиснення інформаційного повідомлення, яке займає 5120 біт, та сформоване графіком повідомлення, у результаті роботи алгоритму стискає його до 512 біт. У той же час ці біти використовуються для представлення хеш-значення поточного інформаційного блоку. Під час першої ітерації робоча змінна, була ініціалізована відповідним значеннями. Подібно до етапу розширення повідомлення, цей етап складається з 80 ітерацій. Після закінчення 80 ітерацій завершуються значення, які використовуються в хеш-версії.

3. ПОСТАНОВКА ПРОБЛЕМИ ДОСЛІДЖЕННЯ

I. Реалізувати додаток з графічним інтерфейсом, який дозволяє виконувати наступні завдання:
   1. Обраховувати значення заданої у варіанті хеш-функції:
      1) Текст повідомлення повинен зчитуватися із файлу;

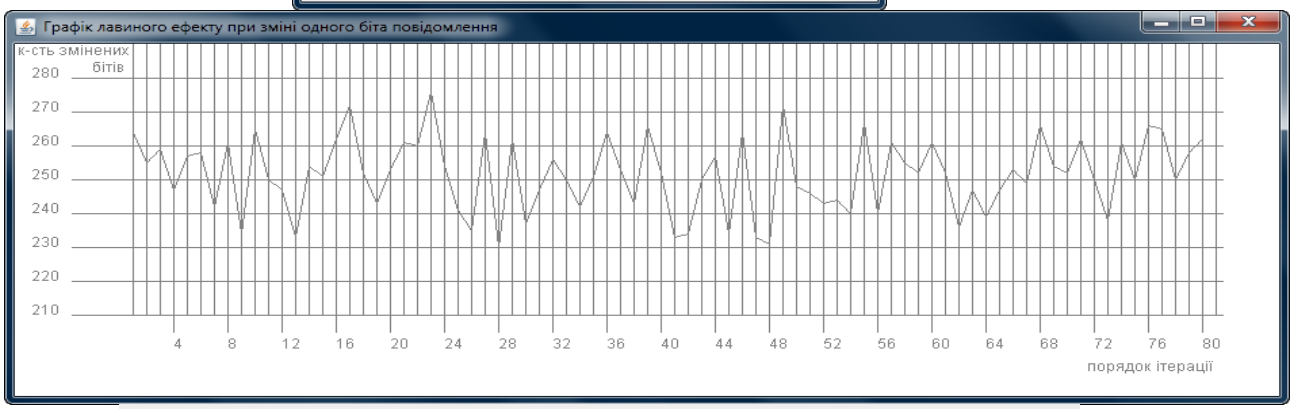
ISSN 2663 - 4023

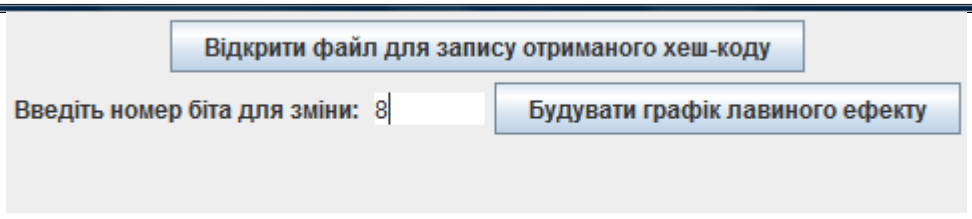
2) Отримане значення повинне зберігатися у шістнадцятковому вигляді і записуватися у файл;
3) Під час роботи програми повинна бути можливість перегляду та зміни повідомлення та хешу функції.

2. Вивчати лавинний ефект для повідомлень, які складаються з одного блоку:
1) Дати можливість вказувати позицію біта, який буде змінюватись;
2) Додаток повинен після кожної ітерації циклу рахувати кількість біт, які змінилися при зміні одного біту;
3) Додаток повинен вміти будувати графіки для зображення лавинного ефекту.

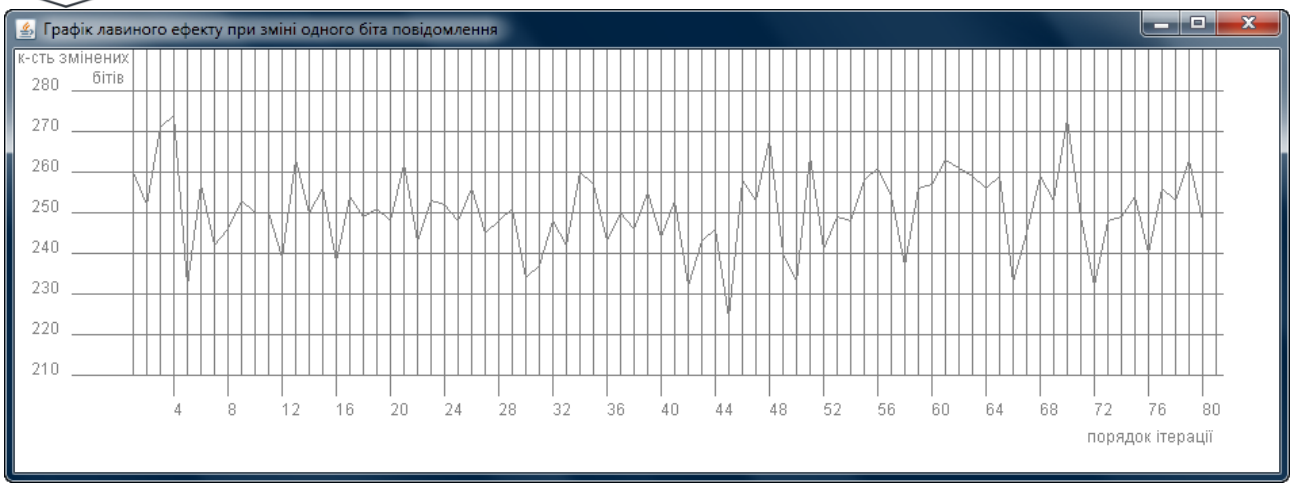
II. За допомогою розробленого додатку.
1. Протестувати правильність роботи програмної реалізації алгоритму.
2. Дослідити лавинний ефект.

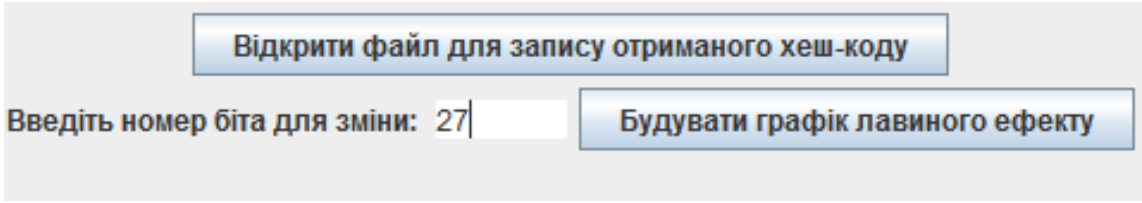
4. ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ SHA-512

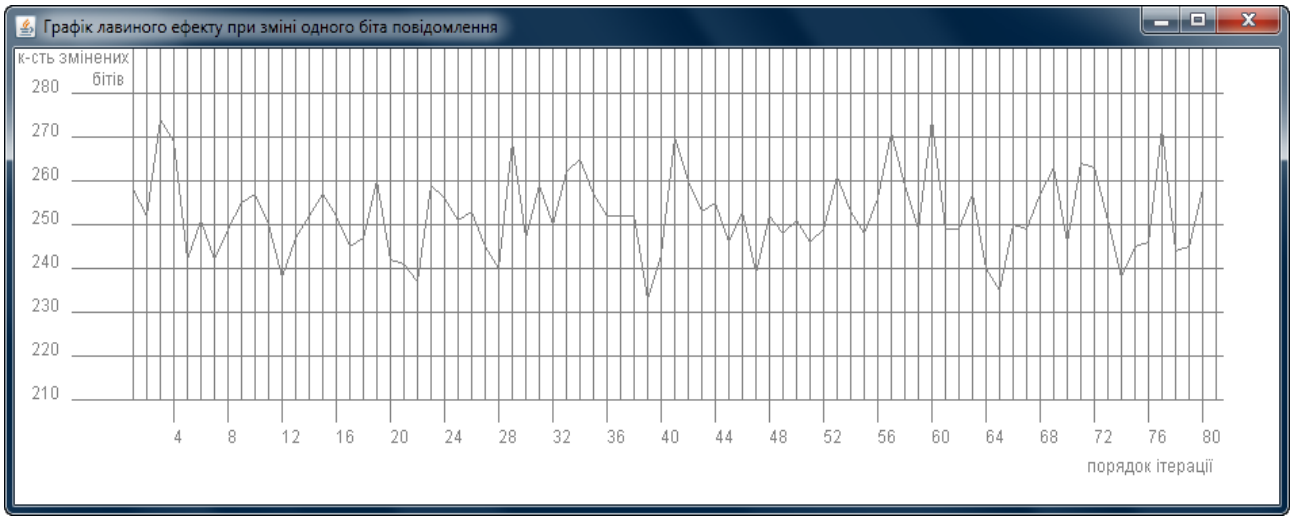




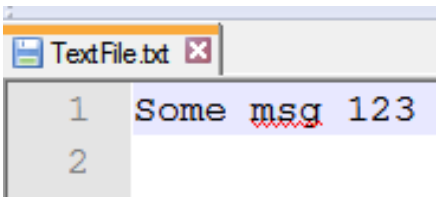
ISSN 2663 - 4023

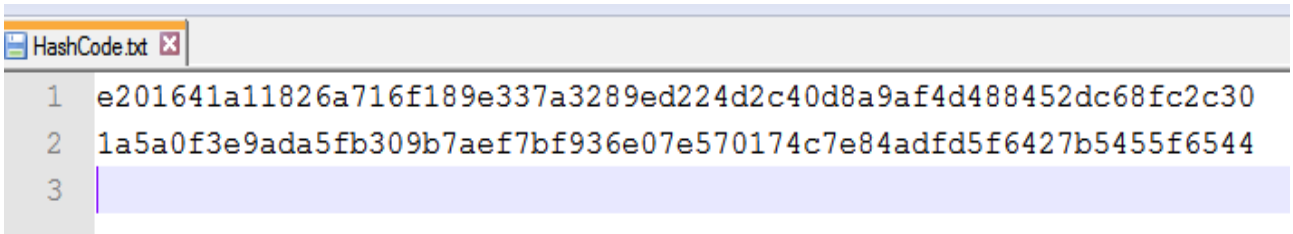






Вміст файлів:





ISSN 2663 - 4023

5. АПАРАТНА РЕАЛІЗАЦІЯ АЛГОРИТМУ SHA-512

На даний час було опубліковано багато робіт, що обговорюють апаратні реалізації SHA-512, [3], [4], [5], [8], [9]. Усі розглянуті реалізації, як правило, спрямовані на високу пропускну здатність або ефективне використання обчислювальних ресурсів. Взагалі неможливо завчасно знати, який вибір функціонального дизайну для даного компонента буде найкращим у досягненні специфічної мети дизайну. По можливості було представлено кілька описів і реалізацій конкретних апаратних компонентів. Після реалізації та виконання алгоритму з різними компонентами можна було провести системний аналіз та прокоментувати якість даної реалізації, оскільки мета стосується досягнення високої пропускної здатності або низької загальної обчислювальної потужності. Ми систематизували результати усіх проведених обчислень та провели аналіз кожної реалізації окремо. Детально сформували опис етапів розширення та стиснення повідомлень. Більшість реалізацій визнають існування попереднього етапу, але не включають і деталізацію. Аналогічно на різних етапах і згадується стадія оновлення хешу, однак її реалізація не завжди чітко визначена. Однією з причин пропускати подробиці попереднього етапу і етапу оновлення хешу є те, що він передбачає, що ці етапи будуть реалізовані таким чином, щоб мінімізувати негативний вплив на нього.

6. ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Розглянута у статті функція перемішування даних SHA-512 та її програмна реалізація засвідчила, про доцільність використання у розробці прикладних програмних систем. Дана хеш функція не претендує на найвищу пропускну здатність алгоритму, проте вона виявилась достатньо стійкою для стороннього декодування.

Підсумовуючи наші наукові дослідження в області криптографічного захисту різними методами ми можемо стверджувати, що розроблені на основі алгоритму SHA-512 прикладне програмне забезпечення відповідає наступним технічним параметрам, а саме верифікацію цілісності програм та даних і достатньо надійний алгоритм автентифікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] I. Ahmad and A. S. Das. Hardware implementation analysis of SHA-256 and SHA- 512 algorithms on FPGAs. Computers and Electrical Engineering, 31(6):345 – 360, 2005.

[2] F. Aisopos, K. Aisopos, D. Schinianakis, H. Michail, and A. P. Kakarountas. A novel high-throughput implementation of a partially unrolled SHA-512. Proceedings of the Mediterranean Electrotechnical Conference - MELECON, 2006:61 – 65, 2006.

[3] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis. Improving SHA-2 hardware implementations. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4249 NCS:298 – 310, 2006.

[6] L. Dadda, M. Macchetti, and J. Owen. An ASIC design for a high speed implementation of the hash function SHA-256 (384, 512). Proceedings of the ACM Great Lakes Symposium on VLSI, pages 421 – 425, 2004.

[7] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman, and B. Schott. Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512. Information Security. 5th International Conference ISC 2002. Proceedings (Lecture Notes in Computer Science Vol.2433), pages 75 – 89, 2002.

ISSN 2663 - 4023

[8] R. P. McEvoy, F. M. Crowe, C. C. Murphy, and W. P. Marnane. Optimisation of the SHA-2 family of hash functions on FPGAs. Proceedings - IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures 2006, 2006:317 – 322, 2006.

[9] M. McLoone and J. V. McCanny. Efficient single-chip implementation of SHA-384 and SHA-512. 2002 IEEE International Conference on Field-Programmable Technology (FPT). Proceedings (Cat. No.02EX603), pages 311 – 14, 2002.

[10] N. Sklavos and O. Koufopavlou. On the hardware implementations of the SHA-2 (256, 384, 512) hash functions. Proceedings - IEEE International Symposium on Circuits and Systems, 5:153 – 156, 2003.

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (3), 2019
ISSN 2663 - 4023

Pasyeka Mykola
Candidate of Technical Sciences, Associate Professor, assistant professor of software engineering department
Ivano-Frankivsk National Technical University of Oil and Gas
Ivano-Frankivsk, Ukraine
OrcID 0000-0002-3058-6650
pms.mykola@gmail.com

Pasieka Nadiia
Candidate of Technical Sciences, Associate Professor, Associate Professor
Vasyl Stefanyk Precarpathian National University
Ivano-Frankivsk, Ukraine
OrcID 0000-0002-4824-2370
pasyekanm@gmail.com

Bestylnyy Mykhaylo
post-graduate student of the department of software engineering
Ivano-Frankivsk National Technical University of Oil and Gas
Ivano-Frankivsk, Ukraine
OrcID 0000-0001-9957-8854
pz@nung.edu.ua

Vasyl Sheketa
Doctor of Technical Sciences, Associate Professor, Head of the Department of Software Engineering
Ivano-Frankivsk National Technical University of Oil and Gas
Ivano-Frankivsk, Ukraine
OrcID 0000-0002-1318-4895
vasylsheketa@gmail.com

ANALYSIS OF THE USE OF THE HIGHLY EFFECTIVE IMPLEMENTATION OF THE SHA-512 HASH FUNCTIONS FOR THE DEVELOPMENT OF SOFTWARE SYSTEMS

Abstract. Hashing functions play an applied and fundamental role in the current protection of programs and data by cryptography techniques. Typically, these security features transmit latency data at the same time, producing a small and fixed-size signal. Along with an avalanche-like growing volume of data requiring quick validation, the hash function throughput is becoming a key factor. According to scientific research published in the technical literature, one of the fastest implementations of SHA-512 is the SHA-2 implementation, which provides bandwidth of the algorithm over 1550 Mbps, but is also faster such as the Whirlpool where bandwidths exceed 4896 Mbps At present, many papers have been published discussing the hardware implementation of the SHA-512. All considered implementations are usually aimed at high bandwidth or efficient use of computing resources. In general, it is impossible to know in advance which choice of functional design for this component will be the best in achieving the specific design purpose. After implementation and implementation of the algorithm with different components, it was possible to carry out system analysis and comment on the quality of this implementation, since the goal relates to the achievement of high bandwidth or low overall computing power. We systematized the results of all the calculations performed and analyzed each implementation separately. A detailed description of the stages of the expansion and compression of messages. Similarly, at different stages and refers to the stage of update hash, but its implementation is not always clearly defined. One of the reasons to skip the details of the previous stage and the stage of the hash update is that it assumes that these steps will be implemented in such a way as to minimize the negative impact on it. The data mixing function in the article does not claim to be the highest bandwidth of the algorithm, but it proved to be sufficiently stable for third-party decoding. Summarizing our research in the field of cryptographic protection by various methods, we can state that the application developed on the basis of the SHA-512 algorithm application software corresponds to the following technical parameters, namely verification of the integrity of programs and data and a sufficiently reliable authentication algorithm.
Key words: cryptography, algorithm, hash function, software, software system protection.

ISSN 2663 - 4023

REFERENCES

[1] I. Ahmad and A. S. Das. The hardware implementation analysis of SHA-256 and SHA-512 algorithms is in FPGAs. Computers and Electrical Engineering, 31 (6): 345-360, 2005.
[2] F. Aisopoulos, K. Aisopoulos, D. Schiniaakis, H. Michail, and A. P. Pecator. A novel high-throughput implementation of a partially unrolled SHA-512. Proceedings of the Mediterranean Electrotechnical Conference - MELECON, 2006: 61 - 65, 2006.
[3] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis. Improving SHA-2 hardware implementations. Lecture Notes in Computer Science (including subseries in Artificial Intelligence and Lecture Notes in Bioinformatics), 4249 NCS: 298-310, 2006.
[6] L. Dadda, M. Macchetti, and J. Owen. An ASIC design for a high-speed implementation of the hash function SHA-256 (384, 512). Proceedings of the ACM Great Lakes Symposium on VLSI, pages 421-425, 2004.
[7] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman, and B. Schott. Comparative analysis of the hardware implementations of SHA-1 and SHA-512 hash functions. Information Security. 5th International Conference ISC 2002. Proceedings (Lecture Notes in Computer Science Vol.2433), pages 75 - 89, 2002.
[8] R.P. McEvoy, F.M. Crowe, C.C. Murphy, and W. P. Marnane. Optimization of the SHA-2 family of hash functions in FPGAs. Proceedings - IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures 2006, 2006: 317 - 322, 2006.
[9] M. McLoone and J. V. McCanny. Efficient single-chip implementation of SHA-384 and SHA-512. 2002 IEEE International Conference on Field-Programmable Technology (FPT). Proceedings (Cat. 02EX603), pages 311-14, 2002.
[10] N. Sklava and O. Koufopavlou. On the hardware implementations of the SHA-2 (256, 384, 512) hash functions. Proceedings - IEEE International Symposium on Circuits and Systems, 5: 153 - 156, 2003.