ANALYSIS OF HASH ALGORITHMS*

Mihaela D. Todorova

ABSTRACT: *A hash algorithm is a function that converts a data string into a numeric string output of fixed length. The output string is generally much smaller than the original data. Hash algorithms are designed to be resistant ,with hight level of security. This paper discusses some of the main properties, a study of which shows how hash function satisfies the performance requirements of security .*

KEYWORDS: *Hash Function, Statistical Analysis*

ИЗСЛЕДВАНЕ НА ХЕШИРАЩИ АЛГОРИТМИ*

Михаела Д. Тодорова

1 Въведение

Хеш функция е всеки алгоритъм, който може да се използва за преобразуване на входни данни с произволен размер, към данни с фиксиран размер. Стойностите, върнати от хеш функцията, се наричат хеш стойности, хеш кодове, извлечения или просто хешове. Криптографските хеш функции са въведени от Дифи и Хелман за употреба в инфраструктури с публични ключове [2].

Хеш функциите често се използват в комбинация с хеш таблица, често срещана структура от данните, използвана в компютърния софтуер [16], с цел бързо търсене на данни. Хеш алгоритмите, подобряват времето за търсене на информация в таблици или бази данни [3], [11], като откриват повтарящи се записи в голям файл. Те са особено полезни и в криптографията. Криптографската хеш функция позволява лесно да се провери дали дадени входни данни отговарят на съответна хеш стойност. В случаите, когато входните данни не са известни, е изключително трудно да се възстановят, като се знае единствено запазената хеш стойност.

Основните приложения на хеш функциите са за удостоверяване на съобщения и запазване на паролите [1] и целостта на данните [9]. Те се използват при работа с контролни суми, проверка на пръстови отпечатъци и при работа с кодове за коригиране на грешки. При конструирането на нови хеш функции е възможно използването на различни псевдослучайни генератори [7], [12], [13], [14], [15], [17], които да внесат допълнителна нелинейност при компресирането.

В този доклад са представени някои от основните показатели, чиито изследване изяснява дали хеш функцията е подходяща за използване в софтуерни приложения.

2 Изследване на хеш функции

Хеш функциите се характеризират с някои от следните свойства [5], [6], [10]:

*Partially supported by Scientific Research Grant RD-08-121/06.02.2018 of Konstantin Preslavski University of Shumen, Bulgaria.

- равномерно разпределение
- чувствителност
- конфузионни и дифузионни свойства и
- устойчивост по отношение на колизиите.

2.1 Анализ на разпределение на стойностите

Типично свойство на хеш-стойността е да бъде равномерно разпределена в обхвата на компресията. Имайки в предвид дължината на хеш стойността, при този тест се извършват симулационни експерименти с даден параграф от съобщението.При избраното входно съобщение се изчислява хеш стойността. Генерира се друго входно съобщение със същата дължина, състоящо се от празни интервали. След изчисляването на хеш стойността на второто съобщения, двете стойности се съпоставят и се извършва проверка дали има равномерно разпределение дори при граничните случаи. Читателят може да се запознае с примери за равномерно разпределени хеш стойности от Ref. [5], Фиг. 3 и Ref. [6], Фиг. 4.

2.2 Анализ на чувствителността спрямо оригиналното съобщение и входния ключ

Чрез този анализ се разглежда чувствителността на хеш функцията по отношение на леки промени във входното съобщение или в секретния ключ. Първата стъпка от анализа е изборът на входно съобщение и изчисляването на хеш стойността му. На базата на входното съобщение се дефинират няколко на брой нови съобщения с леки промени, след което хеш стойностите им, изчислени при същите входни параметри и условия, се сравняват с хеш стойността на оригиналното съобщение. Също така се извършва сравнение на хеш стойността, изчислена за оригиналното съобщение, при начални условия с лека промяна.

Целта на този анализ е да се покаже, че всяка малка промяна на входното съобщение или секретния ключ, води до съществена промяна на хеш стойността. Примери могат да се намерят в Ref. [5], Фиг.4 и Ref. [10], Фиг. 2.

2.3 Статистически анализ на конфузия и дифузия

От историческа гледна точка, Шанън, с публикуване през 1949 г. на неговия доклад "Теория на комуникациите в секретните системи" [4], въвежда идеята за два метода за статистически анализ на криптиращите алгоритми: конфузия и дифузия. Целта на дифузията е да се направят изходящите битове силно зависими от входните битове. В идеалния случай, малка промяна във входния блок от данни води до 50% промяна на изходния блок. Целта на конфузията е да направи връзката между входните битове и изходните битове възможно най-сложна и зависима. Този вид анализ се състои от няколко експеримента, които показват способност на хеш функцията за конфузия и дифузия. За тази цел обикновено се използват следните шест статистики:

Минимално разстояние по Хеминг: $B_{min} = min(\{B_i\}_{i=1}^{N})$

Максимално разстояние по Хеминг: $B_{max} = max(\{B_i\}_{i=1}^{N})$

Средно разстояние по Хеминг: $\bar{B} = \frac{1}{N}\sum_{i=1}^{N} B_i$

Средна вероятност: $P = \frac{\bar{B}}{n} \times 100\%$

Стандартно отклонение на броя на променените битове:

$\Delta B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i - \bar{B})^2}$

Стандартно отклонение: $\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\frac{B_i}{n} - P)^2} \times 100\%$,
където N е броя на тестовете и B_i обозначава разстоянието по Хеминг(брой различни битове) между стойностите на хеш, получени в i-тото изпитване.

Тези експерименти са разделени на два вида тестове, тестове от тип A и тестове от тип B.

При тестове от тип A, се генерира произволно съобщение, наричано оригинално съобщение, с размер L = 50n и неговата съответна n- битова стойност на хеш се изчислява. След това се генерира ново съобщение, като на случаен принцип се избира един бит от оригиналното съобщение и обръща. След това n-битовата хеш стойност на новото съобщение се сравнява с тази на оригиналното съобщение и изчисленото разстояние по Хеминг между двете хеш стойности се записват като Bi. Тази операция се повтаря N пъти, където всеки път се избира ново оригинално съобщение и един от неговите битове,избран произволно, се обръща. Тестовете обикновено се извършват със стойности за големината на хеш функцията n = 128; 160; 256; 512; 1024. Например, за 128 битови хеш стойности и $N = 10000$ е необходимо да се постигнат следните подобни резултати [8]: $B_{min} = 44$, $B_{max} = 84$, $B = 63.95$, $P(\%) = 49.96$, $\Delta B = 5.62$ и $\Delta P(\%) = 4.39$.

При тестове от тип B се генерира произволно оригинално съобщение с дължина L = 50n и съответната му n-битова хеш стойността се изчислява. След това се избира и обръща единичен случаен бит от първоначалното съобщение и се изчислява стойността на хеш функцията от модифицираното съобщение. Двете хеш стойности се сравняват и броят на променените битове се изчислява и записва като Bi. За разлика от преходния тест, в този тип тест се използва едно и също входно съобщение за всички N итерации. Тестът се извършва при стойности за големината на хеш функцията равни на n = 128, 160, 256, 512, 1024 и различни стойности на N. Например, за 128 битови хеш стойности и $N = 2048$ е необходимо да се постигнат следните подобни резултати [6]: $B_{min} = 48$, $B_{max} = 83$, $B = 64.22$, $P(\%) = 50.17$, $\Delta B = 5.65$ и $\Delta P(\%) = 4.42$.

Целта на този вид анализ е да покаже дали хеш функцията има силна способност за конфузия и дифузия и дали ще бъде надеждна срещу този вид атаки.

2.4 Тест за устойчивост на колизии

При този анализ хеш функцията се изследва въз основа на тестовете за колизии, предложени в [5]. За да се извърши количествено проучване на анализа на колизия, се извършват следните два вида тестове. При тестове от тип A се генерира входно съобщение с размер L = 50n и съответната му n-битова хеш стойност се изчислява и съхранява във формат ASCII. След това се генерира ново съобщение, като се избира случайно единичен бит от входното съобщение и се променя на 0, ако е 1 или 1, ако е 0. n-битовата хеш стойността на новото съобщение се изчислява и се съхранява в ASCII формат. Двете хеш стойности се сравняват и се преброяват ASCII символите с една и съща стойност в една и съща позиция. Освен това, абсолютната разлика D между двете хеш стойности се изчислява по следната формула

$D = \sum_{i=1}^{n/8} |dec(e_i) - dec(e_i^{'})|$,

където e_i и $e_i^{'}$ ще бъдат i-тия запис на хеш стойностите на първоначалното и модифицираното съобщение,като съответно функцията $dec()$ преобразува записите в техните еквивалентни десетични стойности.

Тестът от тип A се повтаря N = 10 000 пъти и се записват експериментите за минималната, максималната и средната стойност от D ,за различните хеш стойности с размер n = 128,

160, 256 и 512. Например, за 128 битови хеш кодове е необходимо да се постигнат подобни или по-добри от следните стойности [18]: минимална стойност 655, максимална стойност 2064 и средна стойност 1367.

При тестовете тип B се създава произволно входно съобщение с фиксиран размер L = 50n бита и се изчислява неговата съответна n-битова хеш стойност. След това се избира един бит от входното съобщение, променен на 0, ако е 1 или 1, ако е 0, и се изчислява хеш стойността на модифицираното съобщение. Експериментът се извършва за хеш стойности с размер n = 128, 160, 256 и 512, генерирани при тестове от тип B. Същото въведено съобщение се използва за всички N итерации. Например, за 128 битови хешове и $N = 6400$ е необходимо да се постигнат подобни или по-добри от следните стойности [6]: минимална стойност 661, максимална стойност 2294 и средна стойност 1360.

В допълнение към горните експерименти, тестовете на тип B се повтарят за много къси входни низове, състоящи се от един n-битов блок.

След получаване на резултатите от този вид тест се изяснява, дали хеш функцията има устойчивост по отношение на колизиите.

Следващите стъпки на екипа са конструиране на нов хеш алгоритъм базиран на някои от следните псевдослучайни генератори [12], [13], [14] и [17] и неговото изследване чрез описаните по-горе статистически тестове.

ЛИТЕРАТУРА:

[1] Eunice, A. B., Umoren, A. M., Markson, I. (2017). Design of Web-Based Customer Relation Management Application for Power Distribution Company: A Case Study of PHCN Owerri Business Unit, Mathematical and Software Engineering, 3(1), 108-123.

[2] W. Diffie, M.E. Hellman, “New directions in cryptography”, IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, 1976.

[3] Jude, O. O., Jimoh, A. J., Eunice, A. B. (2016). Software for Fresnel-Kirchoff Single Knife-Edge Diffraction Loss Model, Mathematical and Software Engineering, 2(2), 76-84.

[4] C.E. Shannon, “Communication Theory of Secrecy Systems”, . Bell System Technical Journal, vol. 27, no. 4, pp. 656-715, 1949

[5] A. Kanso and M. Ghebleh, “A Fast and Efficient Chaos-Based Keyed Hash Function”, Communications in Nonlinear Science and Numerical Simulation, vol. 18, no. 1, pp. 109-123, 2013.

[6] A. Kanso and M. Ghebleh, “A Structure-Based Chaotic Hashing Scheme”, *Nonlinear Dynamics*, vol. 81, no. 1-2, pp. 27-40, 2015.

[7] Kordov, K. (2015). Signature Attractor Based Pseudorandom Generation Algorithm, Advanced Studies in Theoretical Physics, 9(6), 287-293.

[8] Z. Lin, S. Yu, and J. Lü, “A Novel Approach for Constructing One-Way Hash Function Based on a Message Block Controlled 8D Hyperchaotic Map”, *International Journal of Bifurcation and Chaos*, vol. 27, no. 7, 1750106, 2017.

[9] Nachev, A., Reapy, T. (2015). Predictive models for post-operative life expectancy after thoracic surgery. Mathematical and Software Engineering, 1(1), 1-5.

[10] H. Ren, Y. Wang, Q. Xie, and H. Yang, “A Novel Method for One-Way Hash Function Construction based on Spatiotemporal Chaos”, *Chaos, Solitons and Fractals*, vol. 42, pp. 2014-2022, 2009.

[11] Stefanov, T. (2015). System for information extraction from news sites, Mathematical and Software Engineering, 1(1), 25-30.

[12] Stoyanov, B. (2014). Self-shrinking bit generation algorithm based on feedback with carry shift register, Advanced Studies in Theoretical Physics, 8(24), 1057-1061.

[13] Stoyanov, B., Kordov, K. (2014). Novel zaslavsky map based pseudorandom bit generation scheme. Applied Mathematical Sciences, 8(178), 8883-8887.

[14] Stoyanov, B., Kordov, K. (2013) Pseudorandom Bit Generator with Parallel Implementation. In International Conference on Large-Scale Scientific Computing (pp. 557-564). Springer, Berlin, Heidelberg.

[15] Stoyanov, B., Milev, A., Nachev, A. (2010) Research on the self-shrinking 2-adic cryptographic generator. Journal of Communication and Computer, 7(11), 67-71.

[16] Stoyanov, B., Strahilov, S. (2015) Web Based Information System for Heat Supply Monitoring, Mathematical and Software Engineering, 1(2), 37-42.

[17] B. Stoyanov, K. Szczypiorski, and K. Kordov, "Yet Another Pseudorandom Number Generator", *International Journal of Electronics and Telecommunications*, vol. 63, no. 2, pp. 195-199, 2017.

[18] Y. Wang, K-W. Wong, and Di Xiao, "Parallel hash function construction based on coupled map lattices", *Communications in Nonlinear Science and Numerical Simulation*, vol. 16, pp. 2810-2821, 2011.

Михаела Тодорова
Шуменски университет "Еп. Константин Преславски"
E-mail: mihaela.todorova@shu.bg