UDC 681.3.06.

DYNAMIC PROCESSES OF HASH FUNCTION FORMATION IN A SYSTEM OF FINITE FIELDS

G. Vostrov, O. Ponomarenko

Odessa national polytechnic university

Abstract. *The primary purpose of this article is the hash function formation based on the theory of finite fields. Standard cryptographic hash functions for digital signature algorithms were considered. The methods of hashing different initial data types were analyzed. A method for hashing text files by applying irreducible polynomials was proposed.*

Key words: *hash function, hash table, collision, cryptosystem, electronic digital signature, prime number, finite field, irreducible polynomial.*

Introduction

The exponential nature of the growth information volume entails the need of creation of new protecting information and authentication systems as well as tools that provide the ability to reduce the amount of memory required for the convenience of storing and transferring large files. Hash function is a such tool. These functions make it possible to convert an input data sequence of arbitrary length into a certain fixed length sequence, by means of a certain algorithm. The algorithm and length of the output sequence must be accurately determined by the importance of the data being converted. The initial data is called "message" or "key" and the result of the hash function is "hash code", "hash" or "convolution". The result of a hash function can also be defined as a checksum that uniquely verifies the integrity of the data that was digitally transmitted. In this paper, we investigate methods for cryptographic hash functions constructing and their use in electronic digital signature algorithms and also as a means of accelerating search in various data structures.

The of hash functions vulnerability should be taken into account when developing such systems. The powers of the M input sequence set and the set of the convolution $H(M)$ possible values can be found in any relation. As a rule, the set of input data M has a larger dimension than the set of all possible values of the convolution $H(M)$, it can lead to the transformation of different messages into equals convolution value. Such a case is called "collision" and it is one of the important factors that must be taken into account when constructing hashing algorithms. The lower the probability of collisions, the more reliable the algorithm.

Collisions are the main vulnerabilities of hash functions, this fact should be taken into account when developing various hashing algorithms in cryptographic systems, as well as in some data structures, such as hash tables. Hash-tables are developed to speed up the search in databases by storing a pair of "key" and "address" (hash) values. The appearance of collisions is inevitable in algorithms, where the variable length of input data are hashed into fixed length values. Collisions make it difficult to perform operations in tables, because different keys will correspond to the same value, which is inconvenient for working with large data, especially when the number of collisions is large. Based on this, there is a need of methods that can resolve collisions in hash tables.

The main instruments for resolving collisions in hash tables are the separate chaining and open addressing methods. The method of separate chaining involves combining keys with the same address in a linked list, and these lists can be ordered or unordered [1]. The choice of the type of linked lists depends on the purposes of the implementation, because when trying to organize keys with the same hashes in the ordered lists, it can perform the data operations at a faster rate. However, the use of unordered lists is more advisable in terms of the device memory saving. It should also be taken into account that the effectiveness of using separate chaining depends on the average number of items in the list. The larger it is, the slower the method works. Unlike this method, public lists do not use linked lists. One of the open addressing methods is the method of linear probing. The idea of this method is to search for an element with a key that is equals to the search key and to resolve collisions by placing the element in an empty space in the table. In this method, lists are not used, it helps to save memory, but the speed of finding an empty space in the table depends on its sparsity [1]. If the table is heavily loaded, a large number of soundings may be required. If the table is completely filled, it will lead to an infinite number of soundings, which makes this method unsuitable for resolving collisions.

The main task of this article is to analyze the existing hashing algorithms and to construct a method based on finite fields theory.

Cryptographic hash functions in digital signature algorithms

Hash functions are widely used in various cryptosystems. A cryptographic hash function must satisfy a number of requirements specific to cryptographic methods of information protection, namely [2]:

- resistance to the search of the first prototype. This means that it is impossible to create an effective polynomial algorithm that, in real time, restores the message M by its convolution $H(M)$;

- resistance to the second prototype search. It should be impossible to find such a message N as $N \neq M$ and $H(M) = H(N)$ knowing M and $H(M)$;

- resistance to collisions of the second kind. It is must be impossible to select a pair of different messages $M \neq N$ for which the values of the hash functions would be the same $H(M) = H(N)$;

- with a slight change in the initial message, the result of the hash function must differ to a large extent, that is, has an avalanche effect.

These properties are independent and should be considered only in the aggregate. The east feature allows you to use hash functions in cryptography, authentication systems, various data structures for searching, as well as in computer graphics and computational geometry.

At the moment, the existence of an irreversible hash function remains an unproven fact and is an unsolved problem in computer science. Usually finding an inverse value is a computationally complex task.

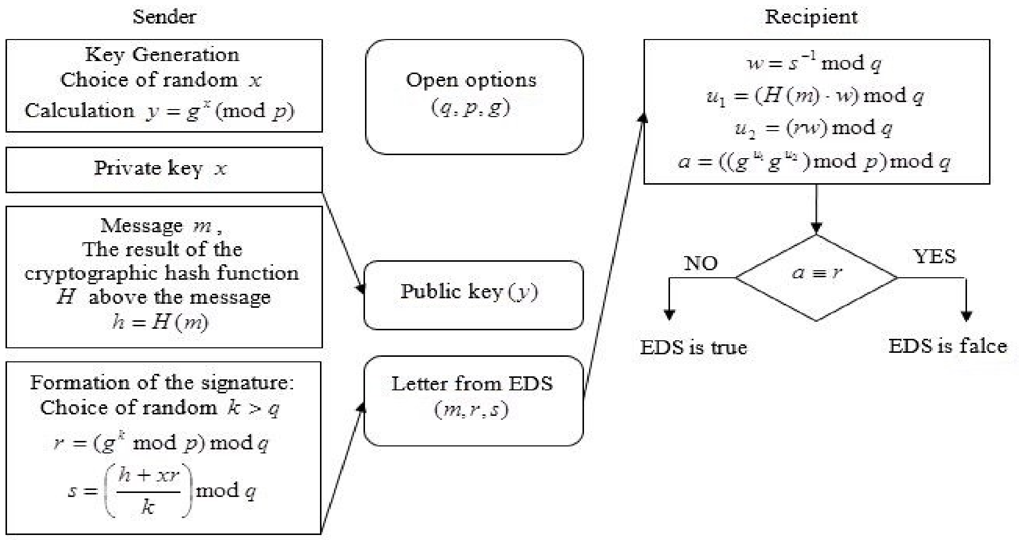
Cryptographic hash functions are widely used in electronic digital signature algorithms. The use of hash functions is not necessary, but it is a convenient tool for converting raw data into a fixed-length sequence, which significantly reduces the computational complexity of applying a digital signature over such a sequence.

Electronic digital signature serves as a means of authentication and not a means of protecting information. The following requirements apply to digital signatures [3]:

1. Guarantee of the information integrity in the processes of its transmission and storage.
2. Providing access to information exclusively to its legitimate users.
3. Absolute authentication of information in accordance to the selected forms of its presentation.
4. Impossibility of authorship refusal.
5. Guarantee of non-traceability of information by providing a consistent analysis of the dynamics of the information messages formation in full.

The above requirements make it possible to use the electronic digital signature tool in economic obligations and in other spheres, as well as in systems that require unambiguous identification of the participants. Such documents may contain several electronic signatures that determine the consent of all involved in the signing process participants.

The scheme of the algorithm for creating and verifying an electronic signature is as follows [2]:


Sender
Recipient
Key Generation
Choice of random x
Calculation $y = g^x (\mathrm{mod}\ p)$
Open options (q, p, g)
Private key x
Message m,
The result of the cryptographic hash function H above the message
$h = H(m)$
Public key (y)
Formation of the signature:
Choice of random $k > q$
$r = (g^k \bmod p) \bmod q$
$s = \left(\frac{h + xr}{k}\right) \bmod q$
Letter from EDS (m, r, s)
$w = s^{-1} \bmod q$
$u_1 = (H(m) \cdot w) \bmod q$
$u_2 = (rw) \bmod q$
$a = ((g^{u_1} g^{u_2}) \bmod p) \bmod q$
$a \equiv r$
NO
YES
EDS is true
EDS is falce

Img.1. The scheme of the algorithm for creating and verifying an electronic signature.

From this scheme it is clear that the formation of an electronic signature does not sign the document itself or the message, but its convolution, that is, the result of applying the cryptographic hash function. Such a function must to follow the requirements for cryptographic hash functions. There is no need to work with a large amount of data each time, but only with their convolution.

In developing cryptographic hash functions algorithms for digital signature systems, the great part of attention is paid to the last property due to the fact that it guarantees the integrity of information. The recipient must receive the message exactly as the sender wanted to send, excluding the possibility of intercepting and changing the message. In case of any accidental or deliberate modification of the document, the electronic signature will be considered as an invalid, since it is calculated according to a special algorithm based on the original document and corresponds only to it. As a consequence, forging documents becomes inappropriate in most cases.

In electronic digital signature algorithms, the hash function is a separate component and can be selected depending on the individual requirements of the system and can also be changed in the process of its improvement or to ensure a dynamic change in the protection system when it is subjected to an unauthorized access.

Such hash functions as MD5, RIPMD-160 and SHA-1 are widely used in digital signature algorithms. Each of them is a generalization of the MD4 algorithm and is iterative in nature [2].

A digital signature must have a certain stability, otherwise its usefulness will rise a question. The MD5 algorithm is one of many algorithms that guarantees the necessary level of protection. This algorithm is more resistant to collision attacks than MD4, but it is not invulnerable, as it has been shown that it is possible to create a couple of different messages with the same hash. In SHA-1 the avalanche effect is improved. The longer hash can be obtained with the help of SHA-1 algorithm and its resistance to similar attacks makes this algorithm more secure than MD5, but it is inferior in terms of performance. The RIPMD-160 function is comparable to SHA-1 by performance.

It is a table of comparing the performance of the above algorithms below [4].

Table 1

The table of comparing the performance hash algorithms

Algorithm	The number of cycles	Speed Mbit / s	Relative performance
MD4	241	191.2	1.00
MD5	337	136.7	0.72
SHA-1	837	55.1	0.29
RIPEMD-160	1013	45.5	0.24

The algorithms mentioned above are belong to the specialized hash functions class. There are also algorithms based on block ciphers (MDC-2, MDC-4) and modular arithmetic (MASH-1, MASH-2). Such algorithms have large computational complexity and low speed. The algorithms based on modular arithmetic don’t have a wide spread in cryptography due to specific requirements for software and hardware.

Specialized methods are easy to implement, therefore they are widely used not only in electronic digital signatures, but also in other information security and authentication systems. It should be noted, that such methods are more vulnerable and could be attacked by cryptanalysis methods.

When using cryptographic hash functions in the electronic digital signature algorithm, the important thing is the bit length of the output sequence, since it must correspond the standards proposed for documents of different significance level. So the optimal size is the output sequence within 160 - 256 bits. It should be taken into account that these numbers will be corrected to a larger extent with the increasing capabilities of technology.

Various methods of constructing hash functions will be considered further in the paper depending on the type of the initial data.

Hash functions for different types of initial data

The methods that will be discussed in this section are intended to convert arrays of original integer data. Such methods are applicable in cases when it is necessary to map one numerical set to another one or if the original data of another type is pre-transformed into integers by certain algorithms.

One of the most famous methods is the division method [1]. The main operation is modulo division and hash is calculated as the remainder of dividing by the number of all possible values of the hash function:

$$H(k) = k(\mathrm{mod}\, M)$$

Where k - is a key (input data), M - the number of all possible hash codes. In order to avoid a large number of collisions, it is better to choose a prime number as a value of M and to avoid the choice of numbers by a degree with base a 2. This function is called modular. This method is effectual because the hash distribution is evenly in the gap $[0; M-1]$. Such a method has a high computational complexity for large values of M, which makes it unsuitable for use.

Another method based on division is the finding of a hash by using polynomial coefficients that are obtained from the remainder of the division of the original data K that represented as a polynomial $K(x)$ by a pre-selected polynomial $P(x)$ by modulo 2 [5]:

$$K(x) \,\mathrm{mod}\, P(x) = h_{m-1}x^{m-1} + \ldots + h_1 x + h_0$$

$$H(x) = h_{m-1} \ldots h_1 h_0$$

Where h_i - is the remainders of the polynomial division, $i = \overline{0, m-1}$.

Just like in the previous case, the correct choice of the polynomial $P(x)$ minimizes the probability of collisions between almost identical input data. In this case it is expedient to use irreducible polynomials as a polynomial. They are the analogous of primes in the theory of finite fields. The theory of irreducible polynomials will be analyzed in this paper. Methods based on the modular arithmetic have large computational complexity, especially with big initial data.

The multiplication method is used in addition to the methods based on division. To compute a hash code using the multiplication method, it is needed to use a hash function[5]:

$$H(k) = \lfloor M * (\{K * A\}) \rfloor$$

Where A - is a rational number in the range $[0,1]$, M - is the number of all possible hash codes, $\{\ \}$ - is the operation of taking the fractional part, $(\ )$- is the priority brackets, $\lfloor\ \rfloor$ - is the operation of taking the whole part of a number.

It is advisable to choose the value of M the power of two because it is a shift to digits in the computer and can be calculated quickly. For an uniform distribution of hash codes, it is important to select the appropriate value of A. Most often selected $\mathrm{A} \approx 0{,}61803398\ 87$, this choice is based on the properties of the golden section [6].

The convolution method is also used as a hash function. The input data is divided into several parts and after operations of addition or non-identity are performed. If the values of two digits are equal to (0 or 1), then the non-identity operation yields 0, otherwise -1 [6].

There are many other hash functions, the scope of which depends on the set of hashed keys. It is impossible to uniquely determine which of the functions to choose without regard to the nature of the input values. When you select a hash function, it is important to calculate it efficiently, since it will not be more efficient to search for an object in one attempt if more time is expended on this attempt than several attempts with the alternative method. It is necessary to take into account the fact that compression of information in protection systems is an integral part of the system and must be performed in a reasonable amount of time while ensuring proper reliability.

These methods are widely used when compressing documents that will be signed or transmitted over the network. If the initial data is not a number, then it must be converted to integers before applying the hash functions described above. For example, for a character string as a binary number, the internal binary representation of the code for each character can be interpreted.

The disadvantage of such methods is that for most computers the binary representations of all letters or digits are very similar to each other. To avoid such situations, a merge method and a weighting method for the string input data were created.

In the merge method to the sequence number in the ANSI sequence of each letter is used create an integer [6]. When there is some integer representation of a string of characters, then to convince it to an acceptable size, the convolution method or the middle of the square method that were given earlier in the work, can be used.

The weighting method uses the position value of each symbol to avoid collisions when using anagrams (permutation of the letters of a certain word or phrase, which results in another word or phrase) as keys [6]. The hash function method based on the theory of finite fields will be proposed in the article.

Methods for constructing hash functions based on the theory of finite fields

Finite fields are widely used in cryptosystems. A great number of cryptographic protocols and cryptosystems are based on the theory of finite fields. Algorithms based on elliptic curves, which are one of the key objects of study in modern cryptography, also use finite fields.

A finite field is a finite set on which arbitrary operations, called addition, multiplication, subtraction and division, are defined (except division by 0). The finite field is usually denoted as F_q, where q is the order of the field, which is always the power of a prime number, it is called the field characteristic [7]. The most common field is $F_p = Z_p$, consisting of all the remainders $\{0,1,\dots,p-1\}$ of division by modulus of a prime number p. Operations in such fields are consistent with the rules of modular arithmetic.

The finite fields satisfy a number of following properties[8]:

1) The characteristic of a finite field F is different from zero and is a prime number.

2) For any two elements a and b of a finite field with a characteristic p, the equality is fulfilled:

$$(a+b)^p = a^p + b^p .$$

3) Any finite field contains p^k elements, where p is the field characteristic, where k is some positive integer.

4) For given p and k there exists a unique field of p^k elements up to isomorphism, which is denoted by F_{p^k}.

5) The multiplicative group $F^*{}_{p^k}$ of nonzero elements of the field F_{p^k} is entirely generated by the powers of some element of the field (it is cyclic) .The study of multiplicative groups has applied value in cryptography, as well as in the tasks of raising to a power or extracting roots [8].

The field F_{p^k} can be constructed as a quotient ring $F_p\,[x]/f(x)$, where the $f(x)$ is irreducible polynomial is over the field. To construct a field from elements it suffices to find an irreducible polynomial of degree k over a field F_p and such a polynomial always exists[8].

A polynomial $f(x) \in F[x]$ is irreducible over a field F or in a ring $F[x]$ if it has a positive degree and equality $f(x) = g(x)h(x)$ is fulfilled, where $g(x), h(x) \in F[x]$ is satisfied only if $g(x)$ or $h(x)$ is a constant polynomial (polynomial degree ≤ 0) [8].

It is possible to construct hashing methods basing on the theory of irreducible polynomials. The result of this method is as follows. An information sequence is fed into the algorithm, which is divided into structural elements - symbols. Each character in the ANSI sequence has a sequence number that is represented as a numeric. A simple number p is chosen depending on the number of possible symbols k in the text that is fed to the input, it satisfies the condition $p \geq k$. Further, for the transformation of these symbols, it is expedient to use a polynomial that is irreducible on the field of characteristics p. Hash of each character can be obtained by substituting its ordinal value (in the ANSI sequence) as the argument of an irreducible polynomial and then dividing by a prime number p modulo. To get the hash value of the whole text, it is needed to sum the hash of each character.

This method is effective for converting and compressing the original text data into a specified range of values. The result of this transformation is the integer value of the convolution, which makes it easy to use it later.

The use of irreducible polynomials as a hash function minimizes the probability of collisions. This is based on the fact that an irreducible polynomial does not have roots over a given field and when the argument value is substituted, the function will not be zero. It is possible to select the same numbers that will be the roots of this polynomial and will give a zero value using a reducible polynomial over a given field. This means that some characters can be replaced or completely deleted, it will not change the final hash value. Thus, it is expedient to apply exactly irreducible polynomials in such algorithms.

However, such methods are expensive in terms of the time operation of the algorithm and its computational complexity for large volumes of input data. In addition, a separate problem is the selection of an irreducible polynomial over a field that depends on the dimension of the original data. The larger the dimension of the original data, the more difficult the problem of finding an irreducible polynomial over the field of a given prime number characteristic.

A separate issue is the problem of finding a prime number from a set of primes. Main problem is constructing an algorithm for the factorization of the prime factors for definition of the simplicity of a given number. There must be specific requirements to such a number and the search for a number with given properties and dimension that satisfies cryptographic systems is another difficult task[9].

Conclusion

As it was noted before, hash functions can be used in information protection and authentication systems as well as in forming of optimal data structures. In the course of this work the collisions resolving in the data tables were analyzed. The methods of hashing integer and string initial data were considered with defining of their advantages and disadvantages.

There was also proposed a method of constructing the hash function that based on the theory of finite fields by using irreducible polynomial over chosen finite field of prime number characteristic. It was proved that this method is stable to collisions. But at the same time it has a drawback in the sense of computational complexity due to operations with a big input data arrays. Also must be constructed effective method for finding prime number of given dimension. In future work it is planed to define a method of computer realization of the hash function that is constructed by using the theory of irreducible polynomials over a finite field.

References

1. Sedgewick, R. (2001), Fundamental algorithms on C++, - Kyiv: Publishing house "DiaSoft", 688 p.

2. Smart, N. (2005), Cryptography, - Moscow: Techno sphere, 528 p.

3. Vostrov, G., Bezrukova, Yu.(2017), Modeling of dynamic data ptotection processes based on a discrete logarithm, - ELTECS.

4. https://ru.wikipedia.org/wiki/RIPEMD-160

5. https://ru.wikipedia.org/wiki/Хеширование.

6. https://studopedia.ru/2_80095_funktsii-heshirovaniya.html

7. Lidl, R., Niederreiter, G.(1988), Finite fields: In 2 volumes, Transl. from English, - Moscow: Publishing house "Mir", 430 p.

8. Crandall, R., Pomerance, K. (2011), Prime numbers: cryptographic and computational aspects, Transl. from English / Ed. and with a preface by V. Chubarikova, - Moscow: URSS: Book House "LIBROKOM", 664 p.

9. Vostrov, G., Khrinenko, A. (2017), Computer modeling of the processes of chaos formation in nonlinear dynamic maps, - ELTECS.

ДИНАМІЧНІ ПРОЦЕСИ ФОРМУВАННЯ ХЕШ-ФУНКЦІЙ В СИСТЕМІ КІНЦЕВИХ ПОЛІВ

Востров Г. М., Пономаренко О. Ю.

Одеський національний політехнічний університет, Одеса, Україна

Анотація. *У даній роботі описані основні методи побудови хеш-функцій та проблеми, які виникають при їх реалізації, що є важливим аспектом при використанні в системах захисту інформації та автентифікації, а також в структурах даних. Основною проблемою в використанні хеш-функцій є виникнення колізій, пов’язане з великим об’ємом масивів даних, які підлягають обробці. Знаходження колізій може використовуватись для несанкціонованих доступу або підробки даних, які підлягають захисту. Виникнення колізій в структурах даних ускладнює процес обробки та зберігання великих масивів даних. Таким чином, виникає необхідність у формуванні алгоритмів, які б мінімізували можливість виникнення колізій і виконувались за раціональну кількість часу. Були розглянуті основні методи пошуку колізій в хеш-таблицях та методи їх вирішення, проаналізовані їх можливі недоліки. В ході роботи були розглянуті стандартні криптографічні алгоритми хеш-функцій, котрі широко використовуються в алгоритмах електронного цифрового підпису, їх переваги і недоліки. Основною ціллю даної статті є моделювання алгоритму хеш-функції на основі теорії кінцевих полів з використанням незвідних многочленів. Запропонований метод обробки текстових файлів за допомогою хеш-функції на основі незвідного многочлену. Даний метод виявився стійким до колізій, однак має високу обчислювальну складність, зумовлену знаходженням простого числа такого, щоб кількість різних символів тексту не перевищувала б значення цього числа. Ще однією проблемою є знаходження незвідного многочлена над кінцевим полем характеристики обраного простого числа. Такий метод може використовуватись в системах захисту інформації та автентифікації, а також в структурах даних, де завчасно можливо виявити кількість різноманітних символів у тексті. Запропонований метод потребує використання ефективних методів розкладення числа на прості множники для визначення простоти числа, а також подальшої модифікації задля зменшення обчислювальних витрат.*

Ключові слова: *хеш-функція, хеш-таблиця, колізія, криптосистема, електронний цифровий підпис, просте число, кінцеве поле, незвідний многочлен.*

ДИНАМИЧЕСКИЕ ПРОЦЕССЫ ФОРМИРОВАНИЯ ХЕШ-ФУНКЦИЙ В СИСТЕМЕ КОНЕЧНЫХ ПОЛЕЙ

Востров Г. Н., Пономаренко Е. Ю.

Одесский национальный политехнический университет, Одесса, Украина

Аннотация. *В данной работе описаны основные методы построения хеш-функций. Рассмотрена проблема возникновения коллизий. В ходе работы были рассмотрены стандартные криптографические алгоритмы хеш-функций, которые широко используются в алгоритмах электронной цифровой подписи. Предложен метод обработки текстовых файлов с помощью хеш-функции на основе теории конечных полей с использованием неприводимых многочленов.*

Ключевые слова: *хэш-функция, хеш-таблица, коллизия, криптосистема, электронная цифровая подпись, простое число, конечное поле, неприводимый многочлен.*

Received 05.04.2018

George Vostrov, Ph. D. of Technical Sciences, Associate Professor of the Department of Applied Mathematics and Information Technologies, Odessa National Polytechnic University. Shevchenko ave., 1, Odessa, Ukraine.
E-mail: vostrov@gmail.com, mob. +380503168776

Востров Георгій Миколайович, кандидат технічних наук, доцент кафедри прикладної математики та інформаційних технологій Одеського національного політехнічного університету. Проспект Шевченко, 1, Одеса, Україна.
E-mail: vostrov@gmail.com, тел. +380503168776

ORCID ID: 0000-0003-3856-5392

Olena Ponomarenko, Student of the Department of Applied Mathematics and Information Technologies, Odessa National Polytechnic University. Shevchenko ave., 1, Odessa, Ukraine.
E-mail: ponomarenkoelena1997@gmail.com, mob. +380934321669

Пономаренко Олена Юріївна, студент кафедри прикладної математики та інформаційних технологій, Одеського національного політехнічного університету. Проспект Шевченко, 1, Одеса, Україна.
E-mail: ponomarenkoelena1997@gmail.com, тел. +380934321669

ORCID ID: 0000-0003-1585-4706