Collision Free Cichelli Perfect Hash Function

[1]Rajeev Ranjan Kumar Tripathi, [2]Deepak Singh Dikhit, [3]Rajesh Kumar Singh

[1,2] ITM, GIDA Gorakhpur Uttar Pradesh, India ,
[3] KIPM-College of Engineering GIDA ,
Gorakhpur, Uttar Pradesh, India

***Abstract*: Hashing is used to improve searching. In hashing, address of a key is calculated by hash function and key is stored at that address. Next time when search is performed, address is computed and referred to get the key. Generally collision takes place in hash functions. A hash function must be free from collision and it should utilize the available addresses and such hash function is referred as perfect hash function. Cichelli proposed a hash function that is known as Minimal Perfect Hash Function. This hash function also shows collision in certain cases. In this paper, we are proposing a technique of code spreading to eliminate the collision. Basically in this proposed scheme we have two sets of symbols, in given key, first and last words are substituted by the symbols from lists. This selection of symbols is done randomly. Hash function is also used in maintaining the message integrity. The randomness used in selection of symbols, creates confusion to the attacker. Every symbol is assigned a unique weight.**

***Keywords*—hash function, perfect hash function, collision, message integrity**

I. INTRODUCTION

Search operation is categorized into two ways: Linear Search and Binary Search. Searching time is directly proportional to search space. In binary search, search space is reduced in every unsuccessful comparison [2,3,5]. That's why binary search is more efficient than linear search where we search an item linearly. To improve searching, hashing is used. Key idea of hashing is placing a record to an address which is dependent on the record. Let us consider a key *n* and we a hash function H. Then the generated address *x* is H (*n*). At address *x*, *n* is stored. When we search the key *n*, we again use the hash function H and compute the address, and directly go to that address. In this way we save the time consumed in the search operation. Let us consider two keys n_1 and n_2.

$$X=H(n_1) \quad (1)$$

$$Y=H(n_2) \quad (2)$$

Where, X and Y are computed addresses.

If n_1 and n_2 are not equal and X=Y this case is referred as collision. Probing techniques are used to resolve this issue. A hash function which is free from collision is known as perfect hash function [2, 3,5,9,10,11,12,13,14,15].

II.PERFECT HASH FUNCTION

Two desired characteristics of hash function are as:

A. *Uniqueness of Addresses*

Let n_1 and n_2 be two keys and n_1 and n_2 are not same. Then $H(n_1)\neq H(n_2)$. Here H is a hash function. This property ensures that unique key has always unique address. It avoids collision.

B. *Effective Utilization of Space*

Records are stored into table. A table has a starting and ending address. A_{min} is the starting address of table and A_{max} is the ending address of the table. Suppose we want to store *n* records into the table then

$$A_{min} \leq H(R_i) \leq A_{max} \quad (3)$$

Where, R_i is the i^{th} record.

Perfect hash function has both the above properties [2,3,5].

III.HASH FUNCTION IN CRYTOGRAPHY

Hash functions can be used to generate hash codes of a given message. Let us consider the message *m* and its hash code as H(*m*). This hash code is appended with the original message *m* as *m*||H(*m*)(|| is used as concatenation symbol). This message is transmitted over channel. At receiver end, message *m* is again supplied to the hash function H. Let the computed hash code at receiver end be $H^1(m)$. This $H^1(m)$ is compared with the received H(*m*) which is the part of received message. If $H^1(m)$ and H(*m*) matches , it is declared that message is unaltered else message is altered and discarded.

Hash code attached with the message appears as cryptographic text. If attacker changes the message only without changing the hash code, the change becomes detectable. To make the changes undetectable, attacker must have exact idea of hash function. During handshake phase, receiver and sender agree to use a hash function and this information is kept secret. Birthday paradox attack is possible in this approach. Many variants of this approach are used in communication [4, 6, 7,8,9,10]

IV.BIRTHDAY PARADOX

The probability that, in a set of *n* randomly chosen people, some pair of them will have the same birthday. By the pigeon-hole principle, the probability reaches 100% when the number of people reaches 367 (since there are 366 possible birthdays, including February 29). However, 99% probability is reached with just 57

people and 50% probability with 23 people [4,6,7,14,15].

A one-way hash function is secure and the best way to attack it is by using brute force. It produces an *m*-bit output. Finding a message that hashes to a given hash value would require hashing 2^m random messages. Finding two messages that hash to the same value would only require $2^{m/2}$ random messages.

V.CICHELLI HASH FUNCTION

The Cichelli's method is used primarily when it is necessary to hash a relatively small collection of keys, such as the set of reserved words for a programming language.
The basic formula is:

$$h(S) = S.length() + g(S[0]) + g(S[S.length()-1]) \quad (4)$$

Where g() is constructed using Cichelli's algorithm so that h() will return a different hash value for each word in the set[1].

VI.COLLISION IN CICHELLI HASH FUNCTION

Consider the abbreviated list of months: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, and DEC. Every string has same length 3. Now consider JAN and JUN. Here first and last alphabets of JAN and JUN are same and hence collision occurs. The g() method returns same values for J and N in JAN and JUN.
One solution is given for such type of problem is to consider second alphabets of each string [1].

VII.BASIC IDEA

Shannon theory of diffusion and confusion is the core idea behind the proposed scheme. The words which are prone to collision are selected and the first and last letters of that word are being replaced by the symbol of the list. We have assumed that size of list is *n*. For one letter we have a list having *n* symbols and we have to replace a letter from a symbol out of n symbols. There are *n* different ways of this substitution. In other words we say that one letter is spread over *n* symbols. This spreading is creating sufficient confusion to avoid attacks against the communicated text.
Size of list may be variable also. Confusion directly depends on the size of list. A list having more symbols will create more confusion.

VIII.PROPOSED SOLUTION

Here we use two lists of symbols. List may contain mathematical symbols, currency symbols or Latin character set.

For example we have a list L_1 as {Ẽ,ẽ,ą,Ó,≠,€} for J and second list L_2 as {©,®,≤,±,Σ,÷}for N. Now we will assign unique weight to the symbols of L_1 and L_2. Let weight of L_1 starts from 102 and ends at 107 and L_2 starts from 501 and ends at 506 respectively.

Now ≠ has weight 106 and € has weight 107. The symbol ≠ is selected for J in JAN and € is selected for J in JUN. Again the symbol © has weight 501 and Σ has weight 505. The symbol © is selected for N in JAN and Σ is selected for N in JUN.

Now JAN appears as ≠A© and JUN appears as €UΣ. Now the hash code for JAN will be 106+501+3=610 and for JUN will be 107+505+3=615.

The next section of this paper is shedding light on the agreement.

IX.AGREEMENT

When communication is started an agreement is required on the set of symbols. Collision in Cichelli method is dependent on input text. If given text is prone to zero collision we can use Cichelli method else sender has to find out the words which are generating collision. Now sender has to provide the lists of symbols to be used in substitution to receiver.
During this handshake phase content negotiation may take place. There is a chance that some characters proposed by sender are not available to the receiver machine. In this case receiver may either suggest new characters or sender may select different character set along with weights. After this agreement message exchange starts. At receiver end, decoding starts. This process of decoding is same as it takes place in substitution cipher.

X.COMPLEXITY

Let size of both lists be *n* (for both the symbols: starting and ending symbol) At sender end when encoding takes place, we have to select one symbol from each list. Time complexity for selecting one symbol from the list is O(*1*). We have to select two symbols for each word the total complexity is O(*1*)+O(*1*)=O(*2*). Let in the message to be transmitted, there are *m* words that are prone to collision then total time complexity to encode the message using this text is *m**O(*2*) i.e. O(*2m*).
At receiver end linear search is used. For a symbol on an average O(*n*) comparisons are required. Encoded word has two symbols from the list i.e. O(*2n*) comparisons are required. As there are *m* words in the message, total complexity is O(*2nm*).

XI.CONCLUSION

In this approach we can assign weights to the symbols to effectively utilize the available address. Security in terms of confusion is directly dependent on the size of set of symbols. A rich set of symbols creates more confusion to the attackers. Again supports in avoiding collision to the words which have same starting and ending letters. This approach preserves the original spirit of the proposed method of Cichelli. Sender can wisely assign the weights to both the symbols and original characters (for those words which are not prone to collision) to utilize the available space of given table. If this approach is being use to maintain security of message then weights can be ignored. Further this approach can be used to store data in secure

manner in web application. Generally secure information like login and password are placed in to DBMS either by using cryptographic encryption schemes or by using hash functions. These techniques are fairly good to secure the information but they are computationally more costly than the proposed scheme for the message having smaller size.

REFERENCES

[1] Cichelli, "Minimal Perfect Hash Functions Made Simple,"Communications of the ACM , Vol.23, No.1, Jan 1980.

[2] "An Introduction to Data Structures with Applications" by Trembley and Sorenson, TMH , 2nd edition , 1984.

[3] "Introduction to Algorithms" by Coreman, Leiserson,Rivest and Stein, PHI,2nd edition, 2005.

[4] "Cryptography and Network Security: Principles and Practice" by William Stallings, Pearson, 3rd edition,2004.

[5] "Fundamentals of Data Structures" by Horowitz and Sahni, Computer Science Press,1983.

[6] "Cryptography and Network Security" by Forouzan , TMH, Special Indian Edition,2007.

[7] "Cryptography and Network Security" by Atul Kahate, TMH, 2nd edition,2008.

[8] I.B.Damgard ,"A Desihgn Principle for Hash Functions," CRYPTO'89,LNCS435,pp.416-427,1990.

[9] I.B.Damgard ,"Collision Free Hash Functions and Public Key Signature Schemes," CRYPTO'87,Springer-Verlag,1998.

[10] Bellare,Keelveedhi and Ristenpart, "Message-Locked Encryption and Secure Deduplication," Eurocrypt2013.

[11] Cramer and Shoup, "Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption,"Eurocrypt2002, LNCS 2332,pp.45-64,2002.

[12] Shafi Goldwasser,Micali, and Ronald L. Rivest , "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," SIAM J. Comput., pp 281–308.,13 July 2006.

[13] Jung Yeon Hwang1, Doo Ho Choi, Hyunsook Cho1 and Boyeon Song, "New efficient batch verification for an identity-based signature scheme,"Security and Communication Networks,Volume 8, Issue 15, pages 2524–2535, October 2015.

[14] J. Rompel, "One-way functions are necessary and sufficient for secure signatures," STOC '90 Proceedings of the twenty-second annual ACM symposium on Theory of computing, Pages 387-394,ACM New York, NY, USA,1990.

[15] "Applied Cryptography" by Bruice Schneir, Wiley Student Edition, 2nd Edition,2006.