Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 5, May 2016, pg.531 – 539

A Hash Function System Based on Quadratic Function

Samuel King Opoku
Computer Science Department, Kumasi Polytechnic, Ghana
samuelk.opoku@kpoly.edu.gh, samuel.k.opoku@gmail.com

***Abstract*— A hash function is a function that takes a set of characters called plaintext and converts it to a form that is very difficult to revert back to the original set of characters. Hash functions play very important role in computer security, computer vision, bloom filters and search algorithms. The current cryptographic hash functions such as Hashed Message Authentication Code (HMAC), Message Digest (MD) and Secure Hash Algorithm (SHA) including their various variations generate large values for small data and are therefore not appropriate for encrypting large number of small data – such as passwords of organizational members – that need to be stored and retrieved in a database. This paper presents a robust mechanism for implementing a hash function that returns eight characters irrespective of the length of the input string. The mechanism is based on quadratic function. The model is implemented in JavaSE and available for downloading freely at http://www.api.skopoku.org/QuadraticHash.jar**

***Keywords*— Block size, Computer Security, Cryptography, Digital Signature, Hash, Quadratic root**

I. INTRODUCTION

The ability to secure information has been a predominant concern for system developers and administrators. Many mechanisms have been proposed with each having its strengths and limitations [1]. One of the mechanisms is hash function. A hash function is a function that takes a set of characters called plaintext and converts it to a form that is very difficult to revert back to the original set of characters [2], [3]. The values returned by a hash function are called hash values, hash codes, hash sums or simply hashes. Hash functions may be used with or without a key. If a key is used, both symmetric (single secret key) and asymmetric keys (public/private key pairs) may be used [3]. Hash functions have numerous applications. It can be used to index and retrieve items in a database with a short hashed key [2]. It can also be used to detect altered and forged documents by providing message integrity [3]. It accelerates table lookup by detecting duplicated record in a large file such as finding similar stretches in DNA sequences [4]. Geometric hashing [5] is used in computer vision for the detection of classified objects in arbitrary scenes by selecting a region of interest, computing its normalized hash values and then measure the distance between the multiple hash values. Another area of application of hash function is the implementation of bloom filters which are usually applied in applications such as spell-checker, string matching algorithms, network packet analysis tools and network/internet caches through the use of membership query which uses multiple distinct hash functions by allowing the result of a membership query for the existence of a particular value to have a certain probability of error [6].

Cryptographic hash functions are used to hash user's passwords. The hashed passwords are stored in a system rather than storing the actual password [7], [8]. The current systems of cryptographic hashing are Hashed Message Authentication Code (HMAC), Message Digest (MD) and Secure Hash Algorithm (SHA). Hashed Message Authentication Code (HMAC) combines authentication and a shared secret key in hashing. HMAC is

complicated since it performs hashing twice in order to resist some forms of cryptographic analysis [9]. It is therefore a time wasting mechanism. The basic idea of Message Digest (MD) is to do hashing in block-wise by padding and compression [10]. There have been many variations. MD2 is a byte-oriented system, providing a 128-bit hash value and designed for smart cards. MD4 is similar to MD2, designed specifically for fast processing in software and MD5 is similar to MD4 but slow because the data is manipulated more. MD5 was developed since MD4 was easily broken by collision attack [11]-[13]. Secure Hash Algorithm (SHA) was proposed by the National Institute of Standard and Technology (NIST) for the Secure Hash Standard (SHS). It was modelled after MD4 and produces a 160-bit hash value. SHA is more collision resistant than MD5 and is the most commonly used hash for cryptographic purpose these days [14], [15]. SHA-1 was first implemented but in 2010 cryptographic weaknesses were discovered [14] and SHA-2 was developed in 2012 to replace SHA-1 [14], [16]. These methods described above generate large values for small data and is therefore not appropriate for encrypting large number of small data – such as passwords of organizational members – that need to be stored and retrieved in a database. This paper presents a robust mechanism for implementing a hash function that returns eight characters irrespective of the length of the input string. The mechanism is based on quadratic functions. The model is implemented in Java Platform Standard Edition (JavaSE) and available to download freely at http://www.api.skopoku.org/QuadraticHash.jar

A quadratic function is a function of the form $f(x) = ax^2 + bx + c$ where "a", "b" and "c" are constants and "a" is not equal to zero. The graph of the quadratic function is called a parabola. It is a "U" shaped curve that may open up or down depending on the sign of the coefficient of "a" [17]. Fig. 1 shows three parabolas

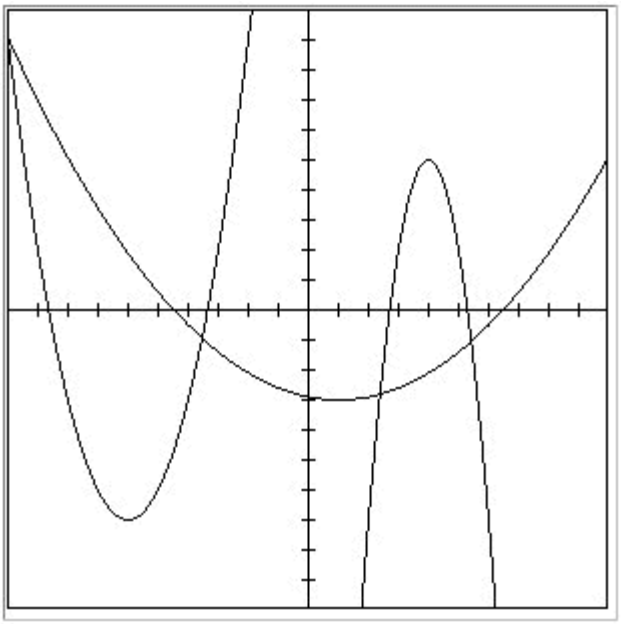

Fig. 1: Three Sample Parabolas

Some of the characteristics of quadratic functions are [18]:

1. Standard form is $y = ax^2 + bx + c$, where $a \neq 0$
2. The graph is a **parabola**, a u-shaped figure
3. The parabola will open upward or downward
4. **A parabola that opens upward** contains a vertex that is a **minimum** point.
5. **A parabola that opens downward** contains a vertex that is a **maximum** point.
6. The **domain** of a quadratic function is all real numbers.
7. An **axis of symmetry** (also known as a **line of symmetry**) will divide the parabola into mirror images. The line of symmetry is always a vertical line of the form $x = n$, where n is a real number.
8. The *x*-**intercepts** are the points at which a parabola intersects the *x*-axis. These points are also known as **zeros**, **roots**, **solutions** and **solution sets**. Each quadratic function will have two, one, or no *x*-intercepts

The x intercepts of the graph of a quadratic function f, given by $f(x) = ax^2 + b\,x + c$ are the real solutions, if they exist, of the quadratic equation when $ax^2 + b\,x + c = 0$. The equation has two real solutions and therefore the graph has x intercepts when the discriminant $D = b^2 - 4 *a* c$ is positive. It also has one repeated solution when D is equal to zero. The solutions are given by the quadratic formula [19]:

$$x_1 = (-b + \sqrt{D})/(2 *a) \text{ and } x_2 = (-b - \sqrt{D})/(2 *a)$$

II. HASH FUNCTION DESIGN

Let L be the length of the string whose hash value is to be computed, such that $\forall L, L \in Z^+ \; and \; L \neq 0$. Given a quadratic equation $ax^2 + bx + c$ where a, b, c are constants and a≠ 0, the roots of the equation is given as $x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ $\forall x_{1,2} \in R \; iff \; b^2 - 4ac \geq 0$. We want to find an expression for the constants a, b, c such that $\forall (b^2 - 4ac) \in z^+ \in$ R and $\forall (b^2 - 4ac) > 0$ and the roots $x_1 \neq x_2$, $|x_1| \in z^+$ and $|x_2| \in z^+$

A. Finding the Constants

The constants of a quadratic equation – a, b, and c – are determined in terms of L, the length of the string whose hash value is to be computed. Thus a, b and c are proportional to L as $a \propto L$, $b \propto L$ and $c \propto L$, $\forall a, b, c \in Z^+$. The constants are picked so that a ≠ b ≠ c to ensure maximum security by obtaining two distinct roots. Let $a = \emptyset L$, $b = (\emptyset + 1)L$ and $c = (\emptyset + 2)L$ where $\forall \emptyset, \emptyset \in Z^+$ and $\emptyset > 0$. The value of $\emptyset$ is determine so that $b^2 - 4ac > 0$. By substitution, $b^2 - 4ac > 0$ implies $[(\emptyset + 1)L]^2 - 4(\emptyset L)(\emptyset + 2)L > 0$ which gives $L^2(\emptyset + 1)^2 - 4L^2\emptyset(\emptyset + 2) > 0$. By simplification, we have $L^2[-3\emptyset^2 - 6\emptyset + 1] > 0$ which can be finalized as

$$-L^2[3\emptyset^2 + 6\emptyset - 1] > 0 \text{ ---------- (1)}$$

Since $\forall \emptyset, \emptyset \in Z^+$, it implies that from equation (1), $(3\emptyset^2 + 6\emptyset) \in Z^+$ and $-L^2[3\emptyset^2 + 6\emptyset - 1] > 0$ if and only if $\emptyset = 0$. But $\emptyset > 0$ and hence when $a = \emptyset L$, $b = (\emptyset + 1)L$ and $c = (\emptyset + 2)L$ it implies $b^2 - 4ac < 0$. Thus the above constants cannot be used.

Varying the value of one of the constants, the significant constant, b, is considered. For $b^2 - 4ac > 0$, it implies $b^2 > 4ac$. Thus if $b > c$ and $b > a$ then $b^2 > ac$ given that $\forall b, b \in Z^+$, $b > 0$. Thus varying the value of b above so that $b > c$ and $b > a$, let $a = \emptyset L$, $b = (\emptyset + 3)L$ and $c = (\emptyset + 2)L$ where $\forall \emptyset, \emptyset \in Z^+$ and $\emptyset > 0$ which implies $(\emptyset + 3)L > (\emptyset + 2)L$ and $(\emptyset + 3)L > \emptyset L$

The new value of $\emptyset$ such that $\forall \emptyset, \emptyset \in Z^+$ and $\emptyset > 0$ is determined so that $b^2 - 4ac > 0$. By substitution, $b^2 - 4ac > 0$ implies $[(\emptyset + 3)L]^2 - 4(\emptyset L)(\emptyset + 2)L > 0$ which gives $L^2(\emptyset + 3)^2 - 4L^2\emptyset(\emptyset + 2) > 0$. By simplification, we have $L^2[-3\emptyset^2 - 2\emptyset + 9] > 0$ which can be finalized as

$$-L^2[3\emptyset^2 + 2\emptyset - 9] > 0 \text{ ---------- (2)}$$

From equation (2), $-L^2[3\emptyset^2 + 2\emptyset - 9] > 0$ while $\emptyset > 0$ if and only if $3\emptyset^2 + 2\emptyset < 9$. If $\emptyset = 1, 3\emptyset^2 + 2\emptyset < 9$ and if $\emptyset > 1, 3\emptyset^2 + 2\emptyset > 9$. Hence $b^2 - 4ac > 0$ if and only if $\emptyset = 1$ which implies $a = L$, $b = 4L$ and $c = 3L$. Thus the roots of the equation such that $|x_1|, |x_2| \in Z^+$ and $|x_1| \neq |x_2|$ can be obtained in terms of L, the length of the string as

$$|x_1| = \frac{-b-\sqrt{b^2-4ac}}{2a} \quad \text{and} \quad |x_2| = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

B. Hash Value Determination

Irrespective of the length of the string, the hash function algorithm which is based on quadratic function returns eight characters as the hash code. The key of the system is the ASCII code for printing characters which has decimal values from 32 to 126. Table 1 below shows the various characters associated with the decimal codes.

TABLE I
PRINTING ASCII CHARACTERS AND THEIR DECIMAL CODES

Decimal Codes	Printing Characters	Decimal Codes	Printing Characters	Decimal Codes	Printing Characters	Decimal Codes	Printing Characters
32		57	9	82	R	107	k
33	!	58	:	83	S	108	l
34	“	59	;	84	T	109	m
35	#	60	<	85	U	110	n
36	$	61	=	86	V	111	o
37	%	62	>	87	W	112	p
38	&	63	?	88	X	113	q
39	‘	64	@	89	Y	114	r
40	(	65	A	90	Z	115	s
41	)	66	B	91	[	116	t
42	*	67	C	92	\	117	u
43	+	68	D	93	]	118	v
44	,	69	E	94	^	119	w
45	-	70	F	95	_	120	x
46	.	71	G	96	`	121	y
47	/	72	H	97	a	122	z
48	0	73	I	98	b	123	{
49	1	74	J	99	c	124	|
50	2	75	K	100	d	125	}
51	3	76	L	101	e	126	~
52	4	77	M	102	f		
53	5	78	N	103	g		
54	6	79	O	104	h		
55	7	80	P	105	i		
56	8	81	Q	106	j		

Using the absolute values of the roots, $|x_1|$ and $|x_2|$ the following six steps are followed to compute the hash value of a string:

- **Step 1 (Length Checking and Padding)**: If the original length is odd append “z” to make it even. The length of the string should be multiples of sixteen (16) otherwise the string is padded with the first decimal coded modal character and a digit from 0 to 6 to get the required length.

- **Step 2 (Block Sizing)**: Block sizing deals with dividing the padded input string into blocks of 16-character size. The 16-character blocks are divided into two blocks with each consisting of 8 characters to form pair blocks which are used in step 3. Hereinafter, a half pair refers to the 8-character size block. First half pair refers to characters from position 1 to 8 and second half pair refers to characters from position 9 to 16. Counting is done from left to right

- **Step 3 (Working with the roots)**: Each block which consists of 16 characters and has been divided into 8-character pair is used. The root $|x_1|$ and its successive integers as $|x_1| + i, \forall i \in Z^+ \; and \; i = 0, 1, 2, \ldots, 7$ is added to the decimal codes of the first half pair consisting of 8 characters starting with the first character. Similarly, the root $|x_2|$ and its successive integers as $|x_2| + i, \forall i \in Z^+ \; and \; i = 7, 6, 5, \ldots, 0$ is added to the decimal codes of the second half pair consisting of 8 characters starting with the first character

- **Step 4 (Evaluating the block pairs)**: Each value of the first half pair obtained in step 3 is added to the corresponding value of the second half pair obtained in step 3. If the summation for each value pair is greater than 126, then subtraction is used for the affected values. If after subtraction, the value obtained is less than 32, then 32 is added to absolute value such that the value of the absolute value is greater than or equal to 32 but less than or equal to 126.

- **Step 5 (Evaluating multiples blocks)**: If the number of 16-character block is more than one, then each block will be handled independently. After each block has been reduced to eight-value sets, the first block set and the second block are picked and evaluated using the procedures described in step 4. The resulting set and the subsequent block are also picked and evaluated. The process continues until all the various block sets are exhausted leaving one result with eight sets of decimal values representing the final eight characters.

- **Step 6 (Final Character set conversion)**: At this stage, the final output values are between 32 and 126 inclusive. Each value is converted to character based on the ASCII decimal coded values in Table 1.

C. Typical Example on How to Use the Design

The example below demonstrates how the model designed above works. Let the string be **Samuel**. Thus L, the length of the string is 6. Finding the roots such that a=6, b=24 and c=18 based on a=L, b=4*L and c=3*L. The roots $x_{1,2} = \frac{-24 \pm \sqrt{24^2 - 4(6)(18)}}{2(6)}$ are $x_1 = -3$ and $x_1 = -1$. Hence the model uses $|x_1| = 3$ and $|x_2| = 1$

- **Step 1 (Length Checking and Padding)**: L = 6 which is even and thus the string – Samuel – is not padded with "z". Each character appears once. Thus the first modal character after sorting based on the ASCII decimal coded values is "S" (That is "S" has coded value of 83 from Table 1 which is less than "a" which has a coded value of 97). Hence padding with "S" and digits, it implies Samuel becomes **Samuel0S1S2S3S4S**

- **Step 2 (Block Sizing)**: Creating the pair with each sub-block consisting of 8 characters, we have
  P1: Samuel0S
  P2: 1S2S3S4S

- **Step 3 (Working with the roots)**: Using $|x_1| = 3$ and $|x_2| = 1$

P1:	S	a	m	u	e	l	0	S
Codes:	83	97	109	117	101	108	48	83
$\|x_1\|$:	3	4	5	6	7	8	9	10
Add:	---	---	---	---	---	---	---	---
Result:	86	101	114	123	108	116	57	93

Similarly

P2:	1	S	2	S	3	S	4	S
Codes:	49	83	50	83	51	83	52	83
$\|x_2\|$:	8	7	6	5	4	3	2	1
Add:	---	---	---	---	---	---	---	---
Result:	57	90	56	88	55	86	54	84

- **Step 4 (Evaluating the block pair)**: Add the various values. If the value is greater than 126, then subtract instead. Below is addition or subtraction section. Valid additions are bolded whiles subtraction is italicized. We are only interested in absolute values.

Result of P1:	86	101	114	123	108	116	57	93
Result of P2:	57	90	56	88	55	86	54	84
Add or subtraction:	---	---	---	---	---	---	---	---
Result:	*29*	*11*	*58*	*35*	*53*	*30*	**111**	*9*

Some of the values are less than 32 and thus 32 is added to them

Result:	*29*	*11*	*58*	*35*	*53*	*30*	**111**	*9*
Add 32:	32	32	-	-	-	32	-	32
	---	---	---	---	---	---	---	---
Final Result:	61	43	58	35	53	62	111	41

- **Step 5 (Evaluating multiples blocks)**: We only have one block and thus step 5 is skipped.

- **Step 6 (Final Character set conversion)**: Looking at Table 1 decimal code 61 corresponds to "=", 43 corresponds to "+" and so forth. Hence the final hash value of the string Samuel is **=+:#5>o)**

III. IMPLEMENTATION AND TESTING

The function is implemented using Java Platform Standard Edition so that the hash function could be tested and analysed.

A. *Algorithm Design*

The algorithm employed in the implementation of the hash function is described as follows:

Begin

1. Get the string
2. Determine the length of the string
3. Find the constants – a, b and c
4. Evaluate the roots, x_1 and x_2
5. Check if the length is odd and then append "z" to the string to make it even
6. If the length of string modulo 16 is not equal to zero
   - 6.1 Check for the modal character
   - 6.2 If the number of modal character is greater than 1, Choose the first modal character when the string is sorted based on the decimal codes of the characters
   - 6.3 Pad the string with one digit from 0 to 6 and the modal character until the length of the string modulo 16 is zero
7. Divide the string into blocks of size 16 characters
8. For each block
   - 8.1 Divide the block into two with each consisting of 8 characters and label them P1 and P2 such that P1 consists of characters from position 0 to 7 and P2 consists of characters from position 8 to 15.
   - 8.2 Evaluate the various blocks using the Java code shown below:

```
int[] eval = new int[8];
ArrayList<String> packing = new ArrayList<String>();
for (int i=0; i < P1.length; i++){
        int num1 = int(P1.charAt(i)) + Math.abs(x1) + i;
        int num2 = int(P2.charAt(i)) + Math.abs(x2) + (P2.length – (i+1));
        int sum = num1 + num2;
        if (sum > 126)
                sum = Math.abs(num1 – num2);
        if (sum < 32)
                sum = sum + 32;
        eval[i] = sum;
}
Packing.add(Arrays.toString(eval)); //place the values in array separated by commas
```

   - 8.3 Get the next block
   - 8.4 If not end of blocks, go to 8.1
9. For the various values stored in the packing Arraylist evaluate as follows:

```
String[] S1 = packing.get(0).split(",");
for (int i=1; i < packing.size; i++){
        String[] S2 = packing.get(i).split(",");
        for (int j=0; j < S1.length; j++){
                int num1 = Integer.parseInt(S1[j]);
                int num2 = Integer.parseInt(S2[j]);
                int sum = num1 + num2;
                if (sum > 126)
                        sum = Math.abs(num1 – num2);
                if (sum < 32)
                        sum = sum + 32;
                S1[j] = Interger.toString(sum);
        }
}
```

10. Convert the final values in S1 array block to characters as the final hash value using the code

```
String output="";
for (int i=0; i < S1.length; i++){
        char c = (char)Integer.parseInt(S1[i]);
        output = output + Character.toString(c);
}
```

End

B. Implementation of Hash Function Application Programming Interface(API)

API is designed and implemented as a class library for Java so that other applications can use. The API consists of two public and static classes in which one extends the other for utility methods. The class library (.jar file) was created using JavaSE and can be downloaded at http://api.skopoku.org.QuadraticHash.jar. Fig. 2 illustrates the relationship between the two public and static classes.

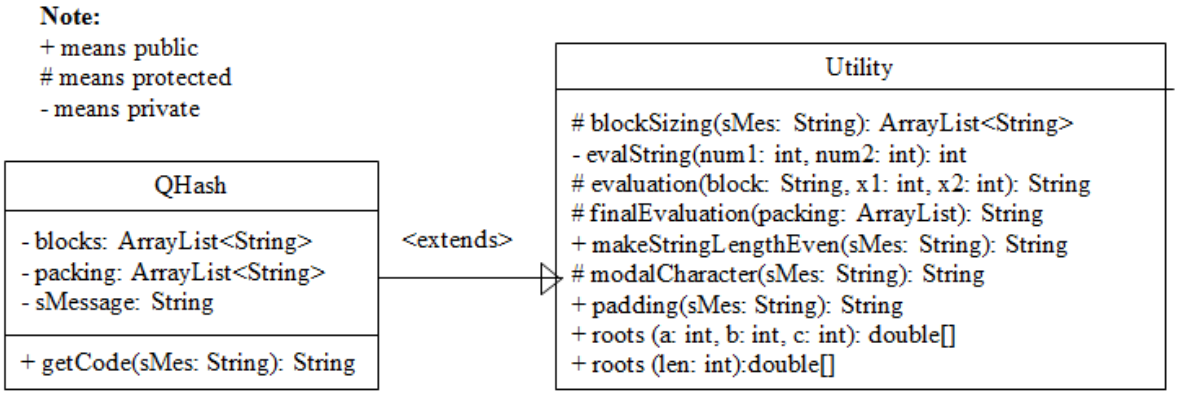

Fig. 2: Class diagram of the Hash Function Classes

A Javadoc file is also available online. Fig. 3 shows the home page section of the Javadoc comments

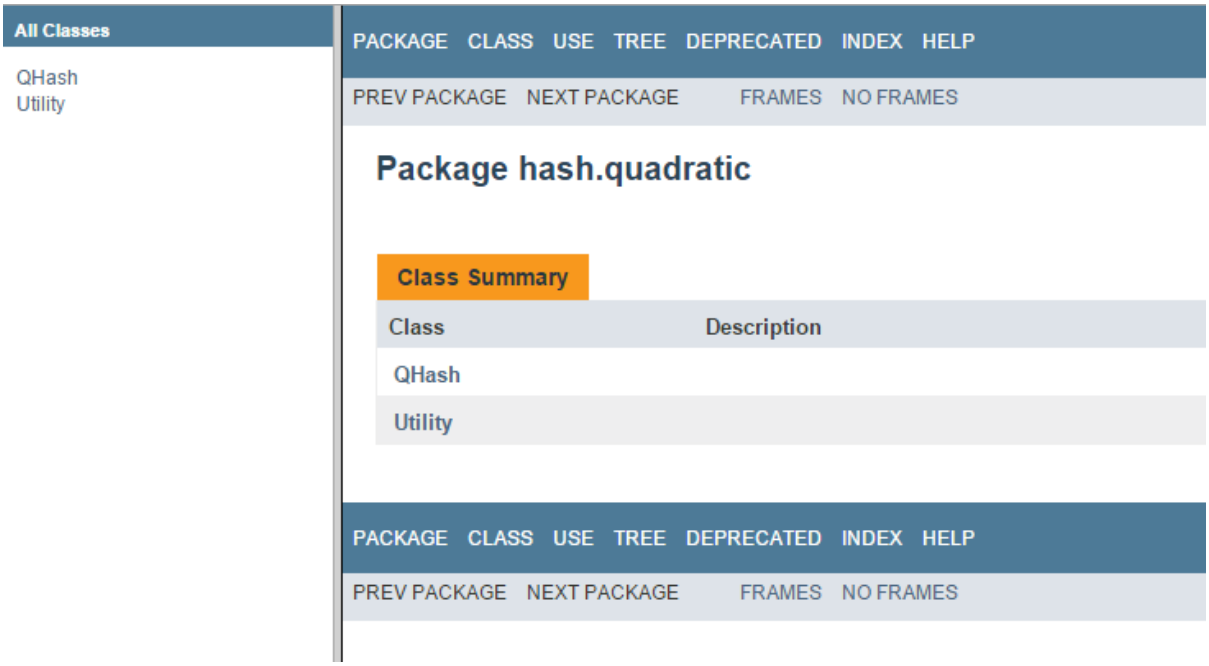

Fig. 3: Home Page of JavaDoc Comments of the Hash Function Library

C. Testing System Development

A testing system is developed to test the API. A form with two textfields is used. The first field accepts plaintext and the second non editable field returns the hash value of the plaintext when compute button is clicked as shown in Fig. 4. To use the hash function library

1. Download the library – free of charge – from *api.skopoku.org* where you can also have access to the description of the various classes; the JavaDoc comments
2. Integrate the library into IDE like NetBeans or Eclipse and add to your project library. OR Add the path of the location of the library to your system's class path based on your operating system settings
3. Add the following import statement to your class file:
   import hash.quadratic.QHash;
4. To get the hash value, pass the plaintext which is a string value to the getCode(String) method as:
   String output = QHash.getCode(s); //where s is a String variable containing the plaintext

Fig. 4 illustrates the outcome of a plaintext during system testing.

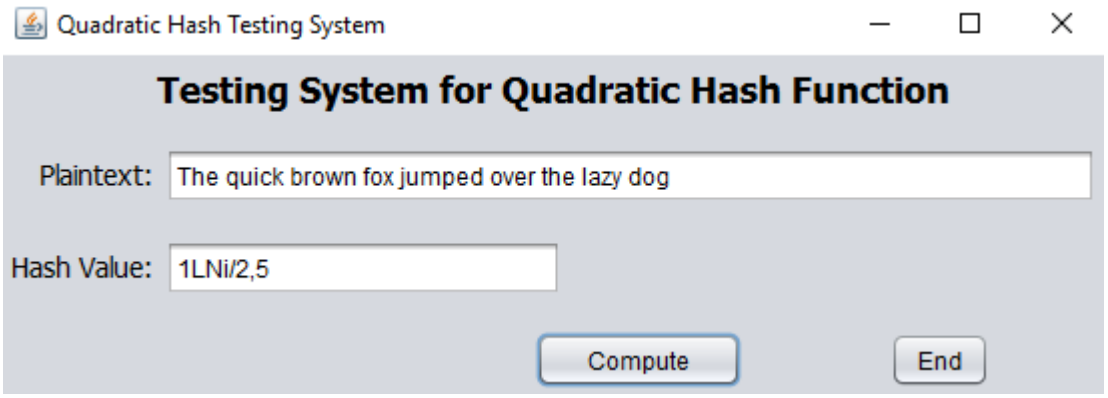

Fig. 4: Testing the Hash Function

IV. CONCLUSIONS

A new hash function mechanism based on quadratic function is implemented. This model opens a door for implementing hash functions that return few bytes of hash codes. The implemented quadratic hash function has the following characteristics:

- The roots x_1 and x_2 are always negative
- Unless there is no character entered, the length of the hash value is always eight, thus memory space is conserved whiles ensuring maximum security.

The quadratic model is very useful for protecting data especially passwords in systems that have numerous users. Future work focuses on enhancing the evaluations of the constants of quadratic equation for this function.

REFERENCES

[1] S. K. Opoku, "A Robust Cryptographic System using Neighbourhood-Generated Keys", *International Journal of Research in Computer Science*, Vol. 2, Issue 5 pages 1-9, September, 2012
[2] (2016) Margaret Rouse, whatis.com [online] Available: *http://searchsqlserver.techtarget.com/definition/hashing*
[3] (2014) Tom sheldon's Linktionary.com website [online] Available: *http://www.linktionary.com/h/hash_function.html*
[4] B. Langmead, C. Trapnell, M. Pop, S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome." *Genome boil*, Vol 10, pp 25, March 2009
[5] H. J. Wolfson and R. Isidore. "Geometric hashing: An overview." *Computing in Science & Engineering*, Vol 4, pp10-21, 1997
[6] B. Andrei and M. Mitzenmacher, "Network applications of bloom filters: A survey", *Internet mathematics*, Vol 1, pp 485-509, April 2004
[7] C. W. Kaufman, "System for controlling access to encrypted data files by a plurality of users." U.S. Patent No. 6,178,508. 23 Jan. 2001.
[8] C. W. Kaufman, R J. Pearlman, and M. Gasser. "System for increasing the difficulty of password guessing attacks in a distributed authentication scheme employing authentication tokens." U.S. Patent No. 5,491,752. 13 Feb. 1996.
[9] H. Krawczyk, C. Ran and B. Mihir. "*HMAC: Keyed-hashing for message authentication.*" (1997).
[10] A. Cheddad, J. Condell, K. Curran, P. McKevitt, "A secure and improved self-embedding algorithm to combat digital document forgery." *Signal Processing*, Vol 89, pp 2324-2332, December 2009

[11] K Burton, "*The MD2 message-digest algorithm*", 1992
[12] X Wang, D Feng, X Lai, H Yu, "Collision for Hash Function MD4, MD5, HAVAL-128 and RIPEMD", IACR Cryptology ePrint Archive pg 199, 2004
[13] Robshaw M. J. B., "On recent results for MD2, MD4 and MD5", RSA Laboratories Bulletin 4, 1996
[14] S. Zhang, R. Tao, and G. Yu,. "Research of Secure Hash Algorithm SHA-1." Computer Security (2010).
[15] Y Kyu Kang, D. W. Kim, T. W. Kwon, J R Choi, "An efficient implementation of hash function processor for IPSEC", Proceedings of third IEEE Asia-Pacific Conference on ASICs, Taipei, Taiwan, 2002
[16] T. Grembowski, R. Lien, K. Gaj, N. Njuyen, P. Bellows, J. Flidr, T. Lehman, B. Schitt, "Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512", *Information Security*, Springer Berlin Heidelberg, pg 75-89, 2002
[17] L. Eduard and J. Wahl, "Quadratic functions and smoothing surface singularities". *Topology*, Vol 25 pp 261-291, March 1986
[18] D. Mumford. "Varieties defined by quadratic equations." Questions on algebraic varieties. Springer Berlin Heidelberg, 2010
[19] (2014) Quadratic Functions [online] Available: *http://www.analyzemath.com/quadraticg/quadraticg.htm*

