

Landau-Ramanujan keyed hash functions for message authentication

A. Suganya*

N. Vijayarangan†

SETS

No. 21, Mangadu Swamy street

Nungambakkam

Chennai 600 034

India

Abstract

An algorithm is newly developed for keyed hash functions using Landau-Ramanujan constant. It is tested well for message authentication and digital signatures. The security analysis on this algorithm is compared with [1] and then the algorithm passes validation tests.

Keywords : HMAC, Hash functions, keying hash functions, Landau-Ramanujan constant.

1. Introduction

When two parties exchange information each other over insecure channel, keyed hash function used to avoid for tampering information. For that, many algorithms like keyed hash functions [1], hash based message authentication codes (HMAC) [2], keyed-MD5 [8], Keyed/Unkeyed RIPEMD [13], etc., are used for message authentication. Apart from these algorithms, in this paper a new keyed hash function is based on Landau-Ramanujan constant [9] and hyperbolic function which are efficient in security analysis. For validation, the proposed algorithm passes 3 primary

*E-mail: asuganya@sets.org.in

†E-mail: vijayarangan.n@rediffmail.com, vijayarangan.2005@yahoo.com

tests such as short messages, long messages and pseudo randomly generated messages. Further, test vectors are also provided in this paper.

2. Landau-Ramanujan keyed hash functions

The keyed SHA-2 uses the key and the message for data integrity or authentication. In addition to the input message M in SHA-2, the key is also used as another input. It uses the key to construct the index table (IT). If the length of the key input is less than 88 bytes as required in IT, there are two methods to do: one is to generate sub-keys and the other is to append the key with itself until the length of the result is long enough. But the later method, appending the key with itself will cause the redundancy. So to avoid the redundancy the sub-keys are generated with the available key length. The expansion of key is also explained in this paper. On the contrary, if the key is too long, we only retrieve the front part. Keyed SHA-2 can convert an arbitrary length of message M to 256 bits of message digest. The algorithm is shown as follows:

3. Initialize Parameter Table (PT)

Put 256 values each of 32-bit into array PT (*Parameter Table*). In SHA-256, the first 64 values of PT ($PT[0 \dots 63]$) are from the first 640 digits of Landau-Ramanujan [12] constant calculated by Philippe Flajolet INRIA Paris and Paul Zimmermann (640 digits are divided in to 64 10-digit numbers and each of them is multiplied with 2^{32} to obtain the 32-bit constant). The remained 192 values ($PT[64 \dots 255]$) are $2^{32} \times \text{abs}(\sinh(i))$, where $i = 0.001$ to 0.064 and i is measured in radians. The functions that are used in the keyed SHA-256 are as follows:

$$\text{Ch}(x, y, z) = (x \text{ and } y) \text{ xor } (\text{not } (x) \text{ and } z)$$

$$\text{Maj}(x, y, z) = (x \text{ and } y) \text{ xor } (x \text{ and } z) \text{ xor } (y \text{ and } z)$$

$$\Sigma_{1=\text{ROTR}}^2(x) \text{ xor } \text{ROTR}^2(x) \text{ xor } \text{ROTR}^2(x)$$

$$\Sigma_{2=\text{ROTR}}^2(x) \text{ xor } \text{ROTR}^2(x) \text{ xor } \text{ROTR}^2(x)$$

$$\sigma_{1=\text{ROTR}}^2(x) \text{ xor } \text{ROTR}^2(x) \text{ xor } \text{ROTR}^2(x)$$

$$\sigma_{2=\text{ROTR}}^2(x) \text{ xor } \text{ROTR}^2(x) \text{ xor } \text{ROTR}^2(x)$$

Note. $\text{ROTR}^Y(x)$ is rotate x right by y times. $\text{SHR}^Y(x)$ is shift x right by y times. The operations and, xor and not are the bitwise operations.

Algorithm Keyed SHA-256

Input: M, key

Output: H (256 bits)

Begin

/ Pad the message */*

The message is “padded” so that its length (in bits) is congruent to 448, modulo 512. Padding is performed as follows: a single “1” bit is appended to the message, and then “0” bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. A 64-bit representation of the length of the message (before the padding bits were added), is appended to the result of the previous step. These bits are appended with the low-order word first. At this point, the resulting message has a length that is an exact multiple of 512 bits.

/ Set the initial hash value */*

Set the initial hash value

$$\begin{aligned} H_0^{(0)} &= PT[IT[80]]; & H_1^{(0)} &= PT[IT[81]]; & H_2^{(0)} &= PT[IT[82]]; \\ H_3^{(0)} &= PT[IT[83]]; & H_4^{(0)} &= PT[IT[84]]; & H_5^{(0)} &= PT[IT[85]]; \\ H_6^{(0)} &= PT[IT[86]]; & H_7^{(0)} &= PT[IT[87]]. \end{aligned}$$

/ Main procedure */*

For $i = 1$ to N (Number of blocks)

{ */* Prepare the message schedule */*

$$W_t = M_t \quad 0 \leq t \leq 15$$

$$W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W^{t-15}) + W^{t-16} \quad 16 \leq t \leq 63$$

/ Set the working variables */*

$$a = H_0^{(i-1)}$$

$$b = H_1^{(i-1)}$$

$$c = H_2^{(i-1)}$$

$$\begin{aligned}
d &= H_3^{(i-1)} \\
e &= H_4^{(i-1)} \\
f &= H_5^{(i-1)} \\
g &= H_6^{(i-1)} \\
h &= H_7^{(i-1)}
\end{aligned}$$

/* Mix with parameters */

For $t = 0$ to 63

$$\begin{aligned}
T_1 &= h + \Sigma_1(e) + \text{Ch}(e, f, g) + PT[IT[t]] + W_t \\
T_2 &= \Sigma_2(a) + \text{Maj}(a, b, c) \\
h &= g \\
g &= f \\
f &= e \\
e &= d + T_1 \\
d &= c \\
c &= b \\
b &= a \\
a &= T_1 + T_2
\end{aligned}$$

/* End of for loop "t" */

/* Compute i th intermediate hash value */

$$\begin{aligned}
H_0^{(i)} &= a + H_0^{(i-1)}, & H_4^{(i)} &= e + H_4^{(i-1)} \\
H_1^{(i)} &= b + H_1^{(i-1)}, & H_5^{(i)} &= f + H_5^{(i-1)} \\
H_2^{(i)} &= c + H_2^{(i-1)}, & H_6^{(i)} &= g + H_6^{(i-1)} \\
H_3^{(i)} &= d + H_3^{(i-1)}, & H_7^{(i)} &= h + H_7^{(i-1)}
\end{aligned}$$

/* End of for loop "i" */

/* The resulting hashing value */

$$H_0^{(N)} \parallel H_1^{(N)} \parallel H_2^{(N)} \parallel H_3^{(N)} \parallel H_4^{(N)} \parallel H_5^{(N)} \parallel H_6^{(N)} \parallel H_7^{(N)}$$

End /* End of Begin*/

The block diagram for the keyed hash algorithm is as follows:

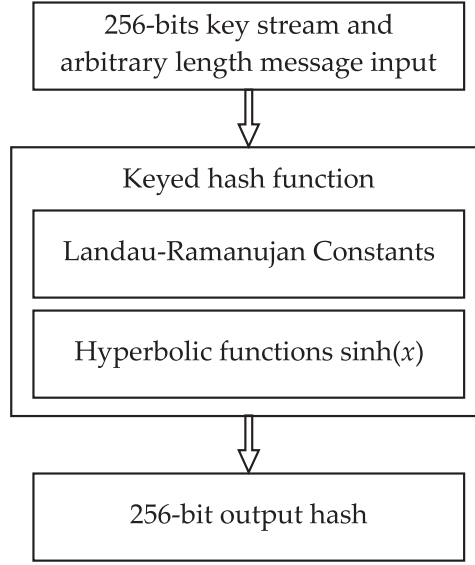


Figure 1
Keyed hash function

Security analysis

In Parameter table, the values of $PT[0\dots63]$ show randomness, which are derived from Landau-Ramanujan equation

$$K = \left[\frac{1}{2} \cdot \prod_{\substack{p \text{ prime} \\ p=4k+3}} \frac{1}{1-p^{-2}} \right]^{\frac{1}{2}} = 0.764223653.$$

This equation is efficient when compared to $p^{1/3}$, where p is the first 64 prime numbers. Since finding out $\text{function}_1 = ((1/1 - p^{-2}))^{1/2}$ for arbitrary prime p is difficult compared to $\text{function}_2 = (p)^{1/3}$ where p takes from first 64 primes. An attacker can guess whatever higher order of bits for p taken consecutively in function_2 .

In parameter table, the values of $PT[64\dots192]$ show randomness, which are derived from $\sinh(x)$ function. The constants are obtained by calculating the $\sinh(x) * 2^{32}$, where x varies from .064 to .192 (since

the $\sinh(x)$ gives the higher values than the $\sin(x)$ function). Due to this reason, $\sinh(x)$ is evaluated for $x = 0.064$ to 0.192 and $\sin(x)$ is for $x = 64$ to 192 . Though the $\sin(x)$ gives the randomness, the analysis on the graph obtained in $\sin(x)$ is bell shaped which may cause periodicity (period 2π) whereas in $\text{abs}(\sinh(x))$ exponentially increases.

Comparison of test vectors

In this paper, the proposed algorithm which is compared with the $\sin(x)$ keyed hash function, passes the validation tests. The following are the test vectors:

Test vectors for $\sin(x)$ Keyed hash function:

- (1) $\{\}$ = f06f7ded61a23cbad2852c72d13e5e8577812986714825092b07aa
699afb8c6a
- (2) $\{abc\}$ = 8a45b5150dcd4754f4daclf53185a8b892dbeb866a2ee33f3b
77eafe3dacdb
- (3) $\{abcdefghijklmnopqrstuvwxy\}$ = fa135974355f2c2c405d4f172636
9620617a260ddb2d25b557374
b6484f70aff

Test vectors for Landau-Ramanujan Keyed hash function:

- (1) $\{\}$ = 52d89f8c69274d067f35eObd1ac7b42573413475cedbe3512b9937
c4047bdal
- (2) $\{abc\}$ = 8690e6a9c7c268cfeed2091614cf226bfaecOfc42914f39f79af
c08694a2
- (3) $\{abcdefghijklmnopqrstuvwxy\}$ = c5bcde36a291879a82a238c6346
d73a5dc854f9096e751039900cd
d2ff571fd9

4. Conclusion

The proposed algorithm is tested well so that it can be used for message authentication and digital signatures. Since it is robust, it could be recommended for cryptographic applications.

References

- [1] M. Bellare, R. Canetti and H. Krawczyk, Keying hash functions for message authentication, *CRYPTO'96*, 1996, pp. 1–15.
- [2] M. Bellare, R. Canetti and H. Krawczyk, The HMAC construction, *Crypto Bytes*, 1996.
- [3] B. C. Berndt, *Ramanujan's Notebooks: Part IV*, SpringerVerlag, 1994.
- [4] *FIPS 180-1, Secure Hash Standard (SHS)*, U.S.Doc/NIST, 1995.
- [5] *Draft FIPS 180-2, Secure Hash Standard (SHS)*, U.S. Doc/NIST, 2001.
- [6] *FIPS 197, Announcing the Advanced Encryption Standard (AES)*, U.S. Doc/NIST, 2001.
- [7] T. M. Hsieh, Y. S. Yeh, C. H. Lin and S. H. Tuan, One-way hash functions with changeable parameters, *Information Sciences*, Vol. 118 (1999), pp. 223–239.
- [8] H. Krawczyk, M. Bellare and R. Canetti, *HMAC-MD5: Keyed-MD5 for Message Authentication*, submitted for RFC (Request for Comments) publication, 1996.
- [9] E. Landau, Über die einteilung der positiven ganzen zahlen in vier klassen nach der mindestzahl der zu ihrer additiven zusammensetzung erforderlichen quadrate, *Archiv der Math. und Phys.*, Vol. 3 (13) (1908), pp. 305–312; *Collected Works*, Vol 4, (L. Mirsky, I. J. Schoenberg, W. Schwarz and H. Wefelscheid, eds.), Thales Verlag, 1985, pp. 59–66.
- [10] W. J. LeVeque, *Topics in Number Theory, II*, Addison-Wesley, 1956.
- [11] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
- [12] The Web page gives the Landau-Ramanujan constants:
<http://www.worldwideschool.org/librarv/books/sci/math/MiscellaneousMathematicalConstants/chap51.html>
last accessed: August, 2004.

- [13] Y. S. Yeh and J. S. Choul, Keyed/Unkeyed RIPEMD-128, 192, 256, *Journal of Information & Optimization Sciences*, Vol. 22 (3), 2001, pp. 563–578.

Received August, 2004