Available Online through www.ijptonline.com

A PROPOSED DESIGN FOR SECURE HASH ALGORITHM-2 (SHA-2) USING 640 BITS
Saurav k. Shaw¹, Mudit Jain², Jitesh Shaw³, P.M. Durai Raj Vincent⁴

¹,²,³ II MCA, VIT University.
⁴ Associate Professor, SITE, VIT University.

Email: sauravkumar.shaw2015@vit.ac.in.

Received on 25-10-2016

Accepted on 02-11-2016

Abstract:
In today’s world, it is common to first convert (encode) our message of varying length into some message form of constant length and then use this converted message for generating digital signatures or in other cryptographic applications. The fixed length message generated is generally referred to as message digest. The algorithm used for such conversion must be designed in such a manner that it does not generate same digest for two different messages otherwise we can’t guarantee the authenticity of the original message. What we do in this type of algorithms is to compress the message and keep on compressing it unless it generates such a fixed length message that is unique for each of the possible messages. Most of such algorithms which were designed earlier and were considered to be secure have now become completely obsolete as they have been cracked due to the increasing processing and computing capacities. In order to ensure that the algorithm designed in today’s time for generating a digest from a varying length message survives the attacks that might become possible due to the increasing processor capacities, it has become necessary that the functions used in the algorithm are safe and fast to calculate and the generated message length must also be of reasonable size. Bigger digest length with better encoding functions ensure more security from different types of attacks.

In this paper, we are going to discuss and describe the design of SHA-2 algorithm using 640 bits, the possible benefits that one might get through this approach and its applications.

Keywords: Message digest, Secure Hash Algorithm (SHA), digital signature, Secure Hash Standard (SHS).

Introduction:
SHA-2 (Secure Hash Algorithm 2) is an algorithm for generating a message digest (hash) corresponding to a given message by performing certain operations on the bits of the message. SHA 2 algorithm was first published in the FIPS
publication 180-2 by National Security Agency and the document consisted of three new algorithms: SHA-256, SHA-384 and SHA-512. SHA-2 algorithms are successor of SHA-1 algorithms which was used as a replacement for MD5 algorithm.[2] Nowadays, SHA-2 algorithms are commonly used for testing the integrity of any data and in digital signature applications.[3] Different hash functions belonging to the SHA-2 family generate hash of different length. For example, SHA-256 generates hash of length 256 bits, SHA-512 generates hash of 512 bits.[1]

SHA-2 using 640 bits in the hash, isa one-way encryption algorithm, i.e., we can encrypt a message using the hash functions and generate a message digest corresponding to that particular message but we can’t decrypt the generated message digest to get back the corresponding message. Hence, this type of algorithms are used for testing the integrity to the message, i.e., to check whether the message is intact or has it been altered in transit.[3] SHA-640 inherits many of the properties of SHA-512 but not all of them.

SHA-640 uses big endian notation like all the other hash algorithms belonging to the SHA family.

SHA-640 processes a block of 1024 bits at a time using the hash functions. If the length of the message is less than 1024 bits then it must be padded. Padding is done in the following manner: The message is padded with 1 and remaining 0’s up till 895th bits and the length of the original message is also added at the end of the block in the remaining 128 bits.

SHA-640 can generate message digest for messages of length l, where 0 ≤ l < 2128. This is because the length of the original message added at the end of the block is of 128 bits. If l1 be the 64-bit word containing the most significant bits of the 128-bit length l and l0 be the 64-bit word containing the least significant bits then l can be represented as l=l1 * 2^64 + l0. So, this implies that SHA-640 doesn’t work for message having length greater than 2128.

SHA-640 has a message schedule array (M) of size 80 with 64-bit each. SHA-640 has 80 round constants (keys K) for each of the rounds and the round constants are initialized with the same value as in SHA-512. SHA-640 has ten states (buffers) of 64-bit each. These ten states are initialized with the 64 bits extracted from the fractional parts of Pi serially. The states are initialized with the values given below, where a…j denotes the ten states.

\[
\begin{align*}
a &= 0x243F6A8885A308D3 \\
b &= 0x13198A2E03707344 \\
c &= 0xA4093822299F31D0 \\
d &= 0x082EFA98EC4E6C89 \\
e &= 0x452821E638D01377 \\
f &= 0xB5E5466CF34E90C6C \\
g &= 0xC0AC29B7C97C50DD \\
h &= 0x3F84D5B5B5470917 \\
i &= 0x9216D5D98979FB1B \\
j &= 0xD1310BA698DFB5AC
\end{align*}
\]
The initial value of the states are extracted from the fractional parts of Pi because Pi’s fractional part is non-repetitive and random in nature.[4]

The intermediate results of the functions applied on a word of message schedule and a round constant during the rounds are assigned to the temporary copy of states and after the end of 80 rounds these temporary copies are added to the states.

The round function of SHA-640 are same as that of SHA-512 but the result of these functions are added in a different way to the ten states. The functions used are:

\[ \text{CH}(p,q,r) = ((p) \& (q)) \oplus (~((p) \& (r))) \]

\[ \text{MAJ}(p,q,r) = ((p) \& (q)) \oplus ((p) \& (r)) \oplus ((q) \& (r)) \]

\[ \text{EP0}(p) = \text{ROTRIGHT}(p,28) \oplus \text{ROTRIGHT}(p,34) \oplus \text{ROTRIGHT}(p,39) \]

\[ \text{EP1}(p) = \text{ROTRIGHT}(p,14) \oplus \text{ROTRIGHT}(p,18) \oplus \text{ROTRIGHT}(p,41) \]

\[ K_i \text{ denotes } i^{th} \text{ round constant (key) and } M_i \text{ denotes the } i^{th} \text{ word of the message schedule in round } i. \, \] [1]

The functions performed in each round is depicted in the diagram given below.

Fig.1. Iteration diagram of SHA-640.
So from the diagram we can see that the results of MAJ(a,b,c) and EP0(a) are added to the old value of h and the results of CH(e,f,g), EP1(e), K_i, M_i and the old value of j is added to the old value of d and also with the results of MAJ(a,b,c) and EP0(a) which eventually becomes the new value of a, where a…j denotes the ten states. Also just like any other SHA algorithm the values of the states are rotated after each iteration. This is one of the most necessary and important step for this type of algorithm in order to ensure that each of the bits are encrypted properly. The algorithm finally outputs a message digest of 640 bits.

**Literature Survey:**

A number of research works has been done on the Secure Hash Standards (SHS). Some of the related works are:

Penny Pritzker (Secretary of US Dept. of Commerce) et al.’s [1] have specified the various Secure Hash Standards and the algorithms to generate the unique fixed length message digest. The algorithms are iterative, one-way hash functions which process the original message and produce an encoded message called a message digest which proves to be useful for generating and verifying digital signatures. Every algorithm is defined in two stages: first being *preprocessing* and second being *hash computation*. The difference among the algorithms are based on their security strengths which are provided over the data/message to be hashed. Moreover, the algorithms differ in terms of block-size, word-size or message-digest size. A summary to the various specified algorithms would be:

**Table.1 Comparison of the various algorithms.**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Message Size(bits)</th>
<th>Block Size(bits)</th>
<th>Word Size(bits)</th>
<th>Message Digest Size(bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>&lt;2^64</td>
<td>512</td>
<td>32</td>
<td>160</td>
</tr>
<tr>
<td>SHA-224</td>
<td>&lt;2^64</td>
<td>512</td>
<td>32</td>
<td>224</td>
</tr>
<tr>
<td>SHA-256</td>
<td>&lt;2^64</td>
<td>512</td>
<td>32</td>
<td>256</td>
</tr>
<tr>
<td>SHA-384</td>
<td>&lt;2^128</td>
<td>1024</td>
<td>64</td>
<td>384</td>
</tr>
<tr>
<td>SHA-512</td>
<td>&lt;2^128</td>
<td>1024</td>
<td>64</td>
<td>512</td>
</tr>
<tr>
<td>SHA-512/224</td>
<td>&lt;2^128</td>
<td>1024</td>
<td>64</td>
<td>224</td>
</tr>
<tr>
<td>SHA-512/256</td>
<td>&lt;2^128</td>
<td>1024</td>
<td>64</td>
<td>256</td>
</tr>
</tbody>
</table>

N. Sklavos (Electrical & Computer Engineering Dept.) et al.’s [7] have proposed a VLSI (Very Large Scale Integration) architecture for SHA-2 family. They have designed a shared architecture which is used separately for each of the 3 secure hash algorithms of the SHA-2 family i.e. SHA-2 (256,384,512). The PADDR (one of the components of the
architecture) pads the input message converting them to blocks each of n-bit size. The process of padding follows the process: a logic ‘1’, followed by m ‘0’s, and then a 64-bit integer, all along are appended at the end of the input message to result into a padded message with length equaling to a multiple of n.

Wanzhong Sun (Institute of Electronic Tech., Information Engineering Univ.) et al.’s [10]- have proposed a reconfigurable architecture that supports multi-mode operation in the fact that it can perform each of the three secure hash functions of the SHA-2 family. Moreover, its performance is higher as comparative to other implementations in terms of resources as well as throughput. This increased performance results from the use of critical data path optimization and the method used for reducing the word length.

**Algorithm:**

Begin:

Step 1:

Initialize the states of the SHA context:(extracted from the fractional part of Pi by taking 64 bits at a time[^4] ) and other fields like data length and bit length to 0.

Step 2:

Initialize the round constants (keys):

The round constants are initialized with the same 80 round constants of 64-bit each used in SHA-512, SHA-384, SHA-512/224 and SHA-512/256, i.e., hexadecimal value of the first 64 bits extracted from the fractional parts of the cube root of the first eighty prime numbers.[^1]

Step 3:

Read the data whose message digest is to be generated.

Step 4:

If the data length is greater than 1024 bits then club the data into blocks of 1024 bits and go to Step 9. Update the bit length field of the SHA context accordingly.

Step 5:

If the message length is greater than 896 then pad the message with 1 and rest 0’s until the length becomes 1024 and then go to Step 9. Set all the first 896 bits of data to 0.
Otherwise pad the message with 1 and rest 0’s until the length becomes 896 bits.

Step 6:
Add the length of the message in the block in the remaining 128 bits and then go to Step 9.

Step 7:
Extract the message digest from the final states of the SHA context.

Step 8:
Display the message digest.

End

Step 9:
-Step 9a) Create a message array of size 80 with each message word of 64 bits.
-Step 9b) Copy the data from the data block of 1024 bits into the first 16 words of the message array.
-Step 9c) Generate the rest of the message words from the previous message words using the following functions:

\[ m_i = \text{SIG1}(m_{i-2}) + m_{i-7} + \text{SIG0}(m_{i-15}) + m_{i-16}, \text{ where} \]

\[ \text{SIG1}(x) = \text{ROTRIGHT}(x,1) \oplus \text{ROTRIGHT}(x,8) \oplus ((x) \gg 7) \]

\[ \text{SIG0}(x) = \text{ROTRIGHT}(x,19) \oplus \text{ROTRIGHT}(x,61) \oplus ((x) \gg 6) \]

-Step 9d) Copy the states of the SHA context into the temporary variables a..j.

-Step 9e) for each of the round (80 rounds) constant \( (k_q) \) and message word \( (m_q) \) rotate the values of the temporary variables among themselves, i.e., the value of a is assigned to b, b to c and so on, along with the following operations:

\[ t1 = j + \text{EP1}(e) + CH(e,f,g) + k_q + m_q \]

\[ t2 = \text{EP0}(a) + \text{MAJ}(a,b,c) \]

\[ j = i \]

\[ i = h + t2 \]

\[ h = g \]

\[ g = f \]

\[ f = e \]

\[ e = d + t1 \]
\[ d = c \]
\[ c = b \]
\[ b = a \]

\[ a = t_1 + t_2, \text{ where } t_1 \text{ and } t_2 \text{ are temporary variables, } q \text{ denotes the } q^{th} \text{ round and functions are defined as:} \]

\[ \text{CH}(\alpha, \beta, \gamma) = ((\alpha) \& (\beta)) \land \neg ((\alpha) \land (\gamma)) \]

\[ \text{MAJ}(\alpha, \beta, \gamma) = ((\alpha) \& (\beta)) \lor ((\alpha) \& (\gamma)) \lor ((\beta) \& (\gamma)) \]

\[ \text{EP0}(\alpha) = \text{ROtright}(\alpha, 28) \lor \text{ROtright}(\alpha, 34) \lor \text{ROtright}(\alpha, 39) \]

\[ \text{EP1}(\alpha) = \text{ROtright}(\alpha, 14) \lor \text{ROtright}(\alpha, 18) \lor \text{ROtright}(\alpha, 41) \]

-Step 9f) After the end of loop add the values of the temporary variables to the respective states of the SHA context.

-Step 9g) Return to the calling point.

Result analysis:

**Fig. 2. Sample Output 1.**

**Fig. 3. Sample Output 2.**
Cryptanalysis:

Increasing the length of the generated message digest increases the security from brute force attack (preimage attack) and birthday attack. As for a message digest of length $m$ bits it takes $2^m$ evaluations to find a message that corresponds to a given message digest using brute force attack and $2^{m/2}$ evaluations to find two different messages that produce the same message digest, i.e., collision, using birthday attack.\[^{[15]}\] So this implies that for SHA-640 it requires $2^{640}$ evaluations to find the original message from a message digest and $2^{640}/2 = 2^{320}$ evaluations to find collision using birthday attack. Whereas for SHA-512 it requires $2^{512}$ evaluations using preimage attack and $2^{256}$ evaluations using birthday attack. Hence, SHA-640 is more secure than SHA-512 in terms of preimage attack and birthday attack. Also looking at the above equations one may imagine that by increasing the length of message digest one may have more security, however this should be kept in mind that the security doesn’t only depend on the length of the message digest but also on the functions that operate on the original message to generate the message digest, the number of rounds and the rotations among buffers (states). Also the length of the message digest shouldn’t be very large as transmitting the message digest and using the message digest in other applications requires extra time and space.\[^{[15]}\]

Applications:

SHA algorithms are used in many applications. These are used to test the integrity of the message, i.e., using SHA we can determine whether the message is intact or has it been altered, as the message digest of a message gets altered significantly if there is any change in the message unless and until it’s not the message that produces the same message digest as the original one. But the complexity of finding such message using birthday attack is high so we need not worry about that.\[^{[16]}\] The passwords are encrypted using SHA algorithms and the generated message digest is stored in the database or sent to the server side.\[^{[17]}\] Cracking such passwords require great effort as first the account name needs to be determined and then the password needs to be cracked. Also SHA algorithms are used in digital signature applications.\[^{[15]}\] At the sender side, the message whose digital signature is to be generated is first encrypted using SHA algorithm to generate a uniform length message digest. This uniform length message digest is then encrypted using the private key of any asymmetric key algorithm to generate the digital signature. The message along with its digital signature and public keys are transmitted to the intended receiver(s). At the receiver side the digital signature is decrypted using the public key to generate the message digest. At the receiver side a message digest is also generated.
from the message received using the same hash algorithm. Finally, the two message digest are compared. If the message digests matches then the message is authentic and is coming from the verified user else the message is discarded. [16]

SHA-2 algorithms are currently used by many federal and financial organizations. [17]

**Conclusion:**

SHA-640 is overall an implementation of SHA-2 with 640 bit message digest as an output. SHA-640 uses the same hash functions that belong to SHA-2 family hence it implements SHA-2 only. But the hash functions are not added in the same way to the states completely. SHA-640 uses eighty rounds and ten states (buffers) of 64 bits each to generate a message digest corresponding to a message whereas the other algorithms belonging to SHA-2 family uses eight states. Hence, SHA-640 does not completely share same properties with SHA-2 algorithms. This algorithm increases the security of the message from preimage attack and birthday attack and hence makes one way encryption more secure.

**References:**

1. Federal Information Processing Standards Publication 180-4, August 2015, Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8900.