Advanced SHA-1 Algorithm Ensuring Stronger Data Integrity

Siddhartha Rao
Vishwakarma Institute of
Information Technology, Pune
Konark Indrayu Enclave II
M-201, NIBM-Undri Road,
Pune-48

ABSTRACT
SHA-1 is a widely used cryptographic hashing algorithm for validating the integrity of data. Until recently, SHA-1 was considered to be the most secure hashing algorithm and also remains the most widely used hashing function till date. In this paper, we review various collision search attacks on the original full 80-step SHA-1 algorithm and present a new optimized version of the algorithm that reduces the chance of collision and increases the theoretical lower bound of the time complexity required to detect such a collision by an exponential factor of 2.

Keywords
SHA-1, SHA-0, Hash Functions, Collision Search Attacks, Improvements in SHA-1

1. INTRODUCTION
In 1990, Ron Rivest invented the hash function MD4. Two years later, he made various improvements to MD4 and developed and advanced version, called MD5. In 1993, the National Security Agency published a new hash function which was very similar to MD5, called SHA (Secure Hash Algorithm). Soon, in 1995, NSA cited a potential weakness in its developed SHA and made a minor change to it, which led to the creation of SHA-1. The weakness was never elaborated upon by NSA nor did it attempt to prove that their 'fix' removed or reduced the intensity of the weakness.

As stated above, SHA-1 is a hashing algorithm which was originally developed by The United States National Security Agency as SHA-0 and later published by The National Institute of Standards and Technology (NIST). It is the advancement to the original SHA-0 and it was first published in the year 1995. SHA-1 is presently the most widely used SHA hash function, even though it is going to be replaced by the latest and more secure group of hashing functions. SHA-1 is presently being used in a large range of applications which includes, TLS, SSL, SSH and PGP.

A hash function inputs a message of variable length size and gives out a fixed length message as its output. The output message of the hash function is called the hash or message digest of the original input message. The main plot behind creating a good, secured cryptographic hash function is to build a good compression function in which every input bit should modify as many output bits as possible.

In this paper, we have reviewed various collision search attacks conducted by various cryptanalysts on the SHA-1 function. These attacks have exposed various vulnerabilities in the full 80-step SHA-1 algorithm and many of these attacks have time complexity less than the theoretical bound of 280 cycles. We also propose a modified version of the algorithm, which requires the same number of cycles to execute as the original and improves upon the security of the original by doubling the message digest size and hash size, thereby exponentially increasing the probability of detecting a collision by a factor of 2 at the cost of increased storage size.

The original SHA-1 algorithm suffers from age. Computers have become more advanced than they were at the time SHA-1 was conceived. Over the years, cryptanalysts and security researchers have gained more insight when it comes to the analysis of hash functions. It was inevitable that SHA-1 would be broken soon. The current techniques will become more advanced in the future and hence measures must be taken to strengthen the current hashing algorithms to withstand such attacks or to make such attacks more costly to implement.

The rest of the paper is organized as follows. Section 2 outlines a brief description about the basics of a good hashing algorithm and various successful collision search attacks made on the SHA-1 algorithm. Section 3 proposes an advanced version of the same, detailing its working. Section 4 gives test results of the hash functions applies to various test cases and a compares Advanced SHA-1 with the original SHA-1 in terms of security and complexity.

2. PREVIOUS ATTACKS ON SHA-0 & SHA-1

2.1 Introduction
Hash functions are classified into a category called One-way hash functions. One-way hash functions and public-key algorithms are used in concurrence for both encryption and digital signatures. They are used in checking for integrity of the data and also in its authentication. There are a large number of applications for it in many different protocols. Modern cryptography is more relied and dependent on one-way hash functions rather than encryption algorithms.

One-way hash functions have two important properties that should be thoroughly satisfied to ensure high security:-

1. These functions are one way, which means that the hash value of the input message can be computed easily, but it is not possible to recreate the original message using the generated hash value within reasonable amount of cost and time.
2. These functions are collision free. By collision free, it means that, it is extremely difficult to find any two input messages produce same hash value to by the hash function.

To break a hash function exposes that either one of the defined property or both of the properties are not true.

2.2. Breaking of SHA

In 1997, Wang presented the first attack on SHA-0 based on an algebraic method and showed that collisions can be found with complexity 2^{58}. In 1998, Chabaud and Joux used the differential attack to find the same collision differential path independently.

In the more recent methods of attack, the algebraic method plays a crucial role as it is used to figure out message conditions on both SHA-0 and SHA-1 that should hold for a collision or a near-collision differential path and can be handled in advance.

The collision attacks mentioned above are of two types [4]:-

1. **Collision Attack**: Find two different messages m1 and m2 such that hash(m1) = hash(m2).
2. **Chosen Prefix Collision Attack**: Given two different prefixes p1, p2 find two appendages m1 and m2 such that hash(p1 ‖ m1) = hash(p2 ‖ m2) (where ‖ is the concatenation operation).

In 2005, three Chinese cryptographers broke the SHA-1 by proving that SHA-1 is not collision-free. They proved it by developing an algorithm for finding collisions faster than brute force.

The research team of Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu (mostly from Shandong University in China) have circulated a paper describing their results, which are as follows :-

1. Collisions in the full SHA-1 in 2^{69} hash operations, much less than the brute-force attack of 2^{80} operations based on the hash length.
2. Collisions in SHA-0 in 2^{39} operations.
3. Collisions in 58-round SHA-1 in 2^{33} operations.

This attack is built on previous attacks which were made on SHA-0 and SHA-1 algorithms, and this attack is a large cryptanalytic result. This result makes a major dent in reputation of SHA-1 as a hash function for digital signatures.

SHA-1 hashing function gives a 160-bit hash which means that it hashes down each message to a 160-bit hash value. There are an infinite number of possible messages that can hash into innumerable values. Hence the probability of finding one common hash value for two different messages is very small (1 in 280). This means if you hash 280 random messages, there is a possibility that two messages will hash to the same value. This is the brute force method of finding collisions i.e. we try every possible message and hash value combination till we detect the collision. But, "breaking" the hash function means being able to find collisions faster than that and that is what the 3 Chinese researches managed to do.

Their improved method can find collisions in SHA-1 in 269 calculations, which is about 2,000 times faster than brute force.

"Attacks always get better; they never get worse." This saying gives an indication about threat to electronic world after such attacks. As recent attack by Chinese cryptographers was built on other papers that described attacks against simplified versions of hashing algorithms such as SHA-1, SHA-0, MD4, and MD5, similarly further attacks have been built on this result.

There has been continuous improvement against attack on SHA-1, as can be seen from other attacks published as follows:-

1. Xiaoyun Wang, Andrew Yao and Frances Yao's attack (2005), lowering the complexity required for finding a collision in SHA-1 to 2^{63}.
2. Christophe De Cannière and Christian Rechberger's attack (2006) on a 64-round SHA-1 with a complexity of 2^{35}.
3. Marc Steven's near-collision attack against full SHA-1 (2012) working with an estimated complexity equivalent to $2^{57.5}$.

Hence, there is a need to migrate from SHA-1 towards a stronger and securer hashing mechanism.

Hash functions are the least-well-understood cryptographic primitive, and hashing techniques are much less developed than encryption techniques. Regularly there are surprising cryptographic results in hashing.

The National Institute of Standards and Technology already has standards for longer, and harder to break hash functions: SHA-224, SHA-256, SHA-384, and SHA-512. They're already government standards, and can already be used. Even though these produce longer hash messages, that are subsequently more difficult to break, the algorithms are more complex in terms of running time and storage space as compared to original SHA-1.

3. PROPOSAL

To improve the security and performance of the SHA1 algorithm, we propose a modified version of SHA1 algorithm. The proposal of an improved version of SHA-1 algorithm goes as follows:-

1. Change the hash constants A, B, C, D, E constant blocks to 64-bit size.
2. Change the message block size to multiple of 1024-bits
3. Provision to process a 128-bit message length, i.e. message with maximum size of 2^{128} bits.
4. Change the order of operations in the SHA-1 Hash function.

These changes lead to the following results being obtained:-

1. A 320-bit hash generated as a result.
2. The probability of collision is made negligible.

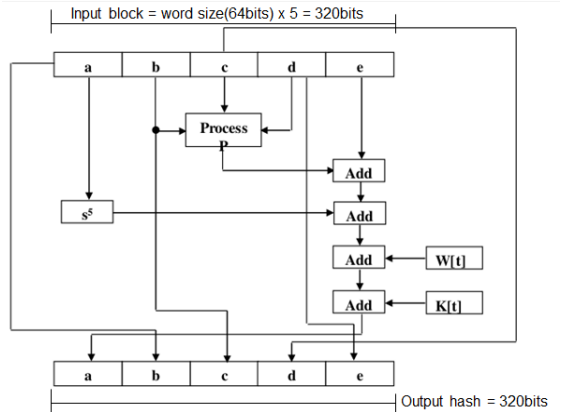

Fig 1: Modified SHA-1 Structure

Our changes do not extend towards the number of rounds required to evaluate a hash value via SHA-1, hence the time complexity of this "advanced" algorithm is same as SHA-1, theoretically. However, pragmatically, we find that our algorithm requires more time processing as the message size increases, since we use a bigger block size. This can be remedied by creating a parallel block processing algorithm which can be made to distribute the inter-block processing so as to reduce the time complexity.

The estimated approximate probability for finding a collision among k randomly generated values, given a space of N possible values is given by the equation $k^2/2N$. Considering that the space of N possible values is dependent on the bit-size of the hash, we can get an equation for a 160-bit hashing algorithm, such as the SHA-1 as $k^2/2*(2^{160})$.

The algorithm that we propose produces 320-bit hash sizes. Hence for our algorithm, the above probability equation is given as $k^2/2*(2^{320})$.

It is plainly observable that the probability of collisions is inversely dependent upon the size of the hash because the larger the size, greater is the number of possible values that can be generated, thereby decreasing the chance of a collision. It is safe to say that for a data set of size n << N, accidental collisions may not occur.

Regarding malicious collisions, even detecting a collision in the regular SHA-1 algorithm is a daunting task. If cryptanalysis research keeps on improving, attackers might be able to deliberately synthesize data that produces a hash collision with respect to other data. But, this is by no means cost-effective as the hash size increases because more computing or processing power is required as the number of bits increases.

4. TESTING & OBSERVATIONS

Based on our above propositions, we implemented the modified SHA-1 algorithm in Java by first developing a standard SHA-1 implementation for Java as defined in FIPS PUB 180-1 published April 17, 1995 and then making changes to it as stated above. The message blocks are stored in the form of short data type and the padding and circular shift functions are modified accordingly. No other changes are made to the original algorithm. The generated result is a 320-bit hash.

The algorithm was tested on a word data set containing 116219 elements (uppercase and lowercase words of the English language). No collisions were found on this test data. The execution time of the algorithm per individual word did not differ considerably from that of the original algorithm.

A 320 bit hash was generated in almost the same amount of time that it required to generate a 160 bit hash for small bits of data. However, as the size of the message increases, the difference in processing time becomes evident. For larger sizes of data, the processing time required by the modified algorithm is significantly larger than that required by the original SHA-1 algorithm.

```
Data            : abc

320bit #        : 5C05363459C0FB7AB7BFDA31FFBBA1FEF8B9BBA5
                  5DF0F31E867D5474067CD30B79B1E0A679A3982

160bit #        : A9993E364706816ABA3E25717850C26C9CD0D89D

320bitExecTime  : 0.097501
160bitExecTime  : 0.100922

--------------------------------------------------------------------------------

Data            : abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq

320bit #        : 5210A8229D9C02F44C1F31B64B275CB9F024A6B2
                  EFA3905B1029BE4B2E033522B689D1A664C8C772

160bit #        : 84983E441C3BD26EBAAE4AA1F95129E5E54670F1

320bitExecTime  : 0.163641
160bitExecTime  : 0.159080

--------------------------------------------------------------------------------

Data            : 1 million repititions of 'a'

320bit #        : 759739CF2F3389449528F00E0658E6AFE614E397
                  F7B5A917029B21BE9282424EB61440D5F2548C93

160bit #        : E98F0C78A84A8E2644219A613A3FBC67291EDA4B

320bitExecTime  : 29.567894
160bitExecTime  : 22.425802
```

Fig 2: Comparison between SHA-1 and Modified SHA-1

5. CONCLUSION

SHA-1 was created at a different point in time, a different age of security, and has become a fading algorithm, even though a successful cost-effective collision attack has yet to be found. SHA-2 & SHA-3 are the new security standard, being heavy and bulky algorithms that are currently said to be invincible. It is not entirely improbable that even these will be broken in the coming future. Instead of finding alternatives for existing algorithms, this paper aims to strengthen the existing one (SHA-1). Even as much as maintaining data integrity is a major challenge, so is trying to maintain existing algorithms for the same purpose and trying to strengthen them whilst also trying to reduce time and space complexity. After all, if the cost of maintaining data integrity is more than the cost of storing data, we are better off without maintain integrity. This paper presents a modification to SHA-1 that attempts to make it twice as much secure as the original one while maintaining the same computational complexity. The modifications do not involve changing the basic working of the algorithm. Rather, the focus is on making minor adjustments to specific values, tweaking them to achieve better performance.

6. REFERENCES

[1] Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu, Finding Collisions in the Full SHA-1, Crypto, 2005.

[2] SHA-1. Available online: http://en.wikipedia.org/wiki/SHA-1.

[3] Jeff Phreshing, http://preshing.com/20110504/hash-collision-probabilities/, 2011.

[4] Collision attack. Available online: http://en.wikipedia.org/wiki/Collision_attack.

[5] Mieliestronk's list of more than 58 000 English words: http://www.mieliestronk.com/wordlist.html.

[6] Dave Rockvam, SHA-2 Migration, Pt. 4, 2014.

[7] Bart Preneel, The First 30 Years of Cryptographic Hash Functions and the NIST SHA-3 Competition, 2007.