Very Secure Hash Algorithm

Edwin Lunando/13509024
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
edwinlunando@gmail.com

***Abstract*—the secure hash algorithm-1(SHA-1) is one of the most widely used in the world. Unfortunately, in the meantime, there were a lot of people found a way to get collision value of SHA-1 that makes the SHA-1 is not secure enough. We need a better hash function. In this paper, the writer will present an implementation and analysis of a new hash function named very secure hash algorithm (VSHA). This algorithm is an improvement of the SHA-1.**

***Index Terms*—SHA-1, hash function, cryptography**

I. INTRODUCTION

In these days, security is an important aspect that cannot be neglected in our life. There are a lot of private data that were communicated every day and there are many people that would like to break it for their own matter. Every kind of applications needs a secure way to communicate or store data. In the meantime, most application use cryptography to send or store the data securely.

One of the most widely used cryptography technique is hash function. Hash function is used to check the integrity of one data. It returns a value that theoretically will be different if the data is not the same. The hash function that mostly used is the secure hash algorithm (SHA-1). In order to keep the confidentiality of a password, most application only stores the hash value of the password. So that, the owner of the account knows the password, not even the admin or developer knows about the password.

Unfortunately, many researchers have found ways to break the hash function. In 2005 they have found a way to find collision with a computational effort fewer than 2^{69} operations. On the same year, an improvement on the SHA-1 attack was announced and lowering the complexity required for finding the collision in SHA-1 to 2^{63} operations. In 2008, an attack methodology may produce hash collisions with an estimated theoretical complexity of 2^{51} to 2^{57} operations. If we estimate the increasing of number of operations per second by moore's law, we can find a collision on SHA-1 in less than a minute in the few next year. Clearly, we need a better hash function.

With the new design of hash function, very secure hash algorithm (VSHA), we are a step closer to a more secure way to send and store data. The main thing to note is that we need to sacrifice speed in order to gain security. We need extra operations to make the algorithm more complex.

The VSHA has a lot of improvement from SHA-1 even though the structure is still the same. SHA-1 has a good and efficient structure. We only need to make it more complex and add some new operation to make it more secure.

The analysis and implementation of the VSHA will be presented at this paper. This paper will also show that the VHSA is theoretically unfeasible to be broken so that, the VSHA could be used as the new hash function, replacing the SHA-1.

II. THEORY

II.I Cryptographic Hash Function

A cryptographic hash function is a hash function, that is, an algorithm that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an (accidental or intentional) change to the data will (with very high probability) change the hash value. The data to be encoded is often called the "message," and the hash value is sometimes called the message digest or simply digests.

The ideal cryptographic hash function has four main or significant properties:

- It is easy to compute the hash value for any given message.
- It is infeasible to generate a message that has a given hash.
- It is infeasible to modify a message without changing the hash.
- It is infeasible to find two different messages with the same hash.

Cryptographic hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information security contexts, cryptographic hash values are sometimes called (digital) fingerprints, checksums, or just hash values, even though all these terms stand for functions with rather different properties and purposes.

Most cryptographic hash functions are designed to take a string of any length as input and produce a fixed-length hash value.

A cryptographic hash function must be able to withstand all known types of cryptanalytic attack. As a minimum, it must have the following properties:

- Pre-image resistance: given a hash h, it should be infeasible to find any message m such that h = hash (m). This concept is related to that one-way function.
- Given an input m1, it should be difficult to find another input m2 where m1 != m2 such that hash (m1) = hash (m2). This property is sometimes referred to as weak collision resistance.
- Collision resistance: it should be hard to find two different messages m1 and m2 such that hash (m1) and hash (m2). Such a pair is called a hash collision.

II.II Secure Hash Algorithm-1

In cryptography, SHA-1 is a cryptographic hash function designed by the United States National Security Agency and published by the United States NIST as a U.S. Federal Information Processing Standard. SHA stands for "secure hash algorithm". The three SHA algorithms are structured differently and are distinguished as SHA-0, SHA-1, and SHA-2. SHA-1 is very similar to SHA-0, but corrects an error in the original SHA hash specification that led to significant weaknesses. The SHA-0 algorithm was not adopted by many applications. SHA-2 on the other hand significantly differs from the SHA-1 hash function.

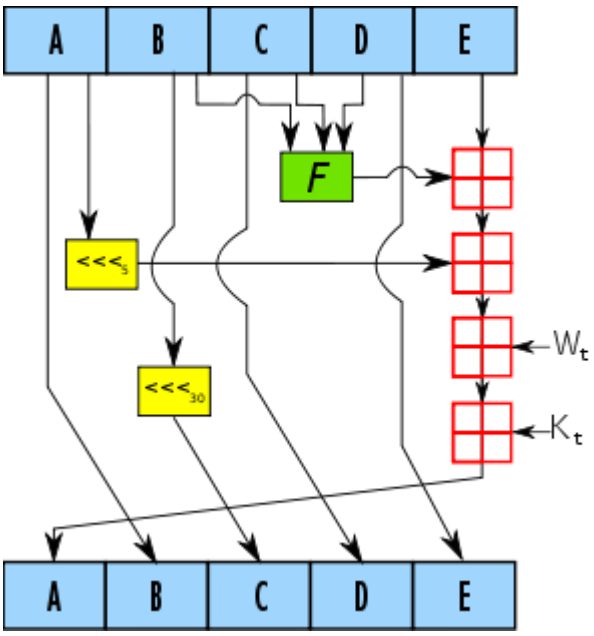

Figure 1 SHA-1 operation

SHA-1 is the most widely used of the existing SHA hash functions, and is employed in several widely used applications and protocols. In 2005, security flaws were identified in SHA-1, namely that a mathematical weakness might exist, indicating that a stronger hash function would be desirable. Although no successful attacks have yet been reported on the SHA-2 variants, they are algorithmically similar to SHA-1 and so efforts are underway to develop improved alternatives. A new hash standard, SHA-3, is currently under development — an ongoing NIST hash function competition is scheduled to end with the selection of a winning function in 2012.

SHA-1 produces a 160-bit message digest based on principles similar to those used by Ronald L. Rivest of MIT in the design of the MD4 and MD5 message digest algorithms, but has a more conservative design.

The original specification of the algorithm was published in 1993 as the Secure Hash Standard, FIPS PUB 180, by US government standards agency NIST (National Institute of Standards and Technology). This version is now often referred to as SHA-0. It was withdrawn by NSA shortly after publication and was superseded by the revised version, published in 1995 in FIPS PUB 180-1 and commonly referred to as SHA-1. SHA-1 differs from SHA-0 only by a single bitwise rotation in the message schedule of its compression function; this was done, according to NSA, to correct a flaw in the original algorithm which reduced its cryptographic security. However, NSA did not provide any further explanation or identify the flaw that was corrected. Weaknesses have subsequently been reported in both SHA-0 and SHA-1. SHA-1 appears to provide greater resistance to attacks, supporting the NSA's assertion that the change increased the security.

In cryptographic practice, "difficult" generally means "almost certainly beyond the reach of any adversary who must be prevented from breaking the system for as long as the security of the system is deemed important." The meaning of the term is therefore somewhat dependent on the application, since the effort that a malicious agent may put into the task is usually proportional to his expected gain. However, since the needed effort usually grows very quickly with the digest length, even a thousand-fold advantage in processing power can be neutralized by adding a few dozen bits to the latter.

In some theoretical analyses "difficult" has a specific mathematical meaning, such as not solvable in asymptotic polynomial time. Such interpretations of difficulty are important in the study of provably secure cryptographic hash functions but do not usually have a strong connection to practical security. For example, an exponential time algorithm can sometimes still be fast enough to make a feasible attack. Conversely, a polynomial time algorithm (e.g., one that requires n20 steps for n-digit keys) may be too slow for any practical use.

These are examples of SHA-1 digests. ASCII encoding is used for all messages.

SHA1("The quick brown fox jumps over the lazy dog")
= 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12

SHA1("The quick brown fox jumps over the lazy cog")
= de9f2c7f d25e1b3a fad3e85a 0bd17d9b 100db4b3

III. VERY SECURE HASH FUNCTION (VSHA) STRUCTURE

The pseudo-code of the VSHA will be given first. Every single line will be analyzed to proof the strength of the algorithm. Here's the pseudo-code:

```
h0 = 0x24839348
h1 = 0xDBED3423
h2 = 0x92304EBE
h3 = 0xDF89Ab93
h4 = 0xA0494834

Pre-processing:
append the bit '1' to the message
append 0 = k < 512 bits '0', so that the resulting message length (in bits) is congruent to 448 (mod 512)
append length of message (before pre-processing), in bits, as 64-bit big-endian integer

Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 = i = 15

    Extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 99
        w[i] = (w[i-3] xor w[i-5] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

    Initialize hash value for this chunk:
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    Main loop:[31]
    for i from 0 to 99
        if 0 = i = 19 then
            f = (b and not c) or not  ((not b) and d)
            k = 0x5A827999
        else if 20 = i = 39
            f = (b and c) xor not c xor (d or not b)
            k = 0x6ED9EBA1
        else if 40 = i = 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 = i = 79
            f = b xor c xor d xor not c
            k = 0xCA62C1D6
        else
          f = (not b and c) xor (b and not d)
           k = 0xBA92DE21

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d leftrotate 20 xor e
        d = c rightrotate 15 xor not d xor (c leftrotate 10 and d rightrotate 20)
        c = (b leftrotate 30) xor (b rightrotate 15) xor (b leftrotate 20)
        b = (not a rightrotate 20 and a leftrotate 20) xor a leftrotate 30
        a = temp

    Add this chunk's hash to result so far:
    h0 = h0 + a
    h1 = h1 + b
    h2 = h2 + c
    h3 = h3 + d
    h4 = h4 + e

Produce the final hash value (big-endian):
digest = hash = h2 append h0 append h3 append h4 append h1
```

This function relies on the Merkle-Damgard construction to achieve great complexity and fixed length output. At the first time, the first value of the function (h0, h1, h2, h3, h4) is changed from SHA-1. The value needs to be changed because it is the most basic way to change something. We want to create a different value from SHA-1. The default output length is still 512 bit, although we could modify the algorithm to produce different length output. For the chunk iteration, it was extended into 100 iterations. The cryptanalysis could break the SHA-1 algorithm easily if the numbers of the iterations are below 60. So, we need to increase the iterations.

The operations that are meant to fill the w array is also improved. The VSHA add an additional xor operation with w[i-5] to achieve more complexity.

In the main loop, the number of iteration is increased as the first iteration number needs to be same as the main loop. The bitwise operations located in the main loop have been improved. At those iterations, mainly the VHSA adds xor and not operations. So, all operations have minimum bitwise operations of 4. The VHSA add a lot of bitwise function because it was not as costly as other complex function. The k variable is still the same because changing by changing it, it do not give better result.

After the bitwise function, the algorithm comes to the next function, The Merkle-Damgard construction. A hash function must be able to process an arbitrary-length message into a fixed-length output. This can be achieved by breaking the input up into a series of equal-sized blocks, and operating on them in sequence using a one-way compression function. The compression function can either be specially designed for hashing or be built from a block cipher. A hash function built with the Merkle–Damgård construction is as resistant to collisions as is its compression function; any collision for the full hash function can be traced back to a collision in the

compression function.

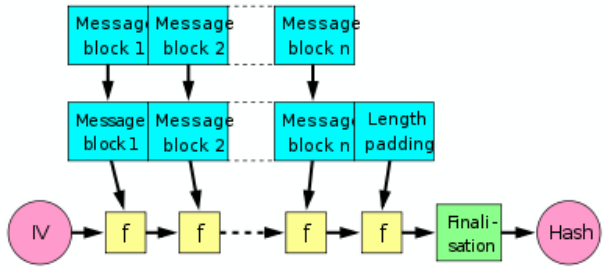

Figure 2 Merkle-Damgard construction

The Merkle-Damgard construction is made different for VHSA because this is the heart of a hash function. The VSHA were added a lot of operations. It also use a right rotate and left rotate. In SHA-1 it only used left rotate. The VSHA also added xor operation and some bitwise operation in order to be more complex.

IV. IMPLEMENTATION AND ANALYSIS OF VSHA

This is the result of an implementation of VHSA tested on a text file. The value of the text file is this.

The quick brown fox jumps over the lazy dog

The result of the VSHA function is:

9fcc1b7b22a8605034fc5f934bb6292917a06041

If the input file is changed by a single character, for example.

ahe quick brown fox jumps over the lazy dog

The result of the VSHA function will be changed dramatically.

2ac973f65e502e0bdcfb28c9ba9e48cbef2af6

If the input file is added one character, for example.

The quick brown fox jumps over the lazy dog1

The result of the VSHA function will be changed dramatically again.

b7a987ddafb1017e677a072562dc6f689986188e

It is true that the performance of VSHA is slightly lower that SHA-1 because of the added operations, but as the computational power will increase by the moore's law, it would be an endless race. While the algorithm will be made more complex and it needs high computational power to break it.

It is very hard for now to define a secure hash function as according to pigeonhole principle, as long as we try all the combination of a hash function, we'll be able to find a collision. There are a lot of attacks to hash function.

For a hash function for which L is the number of bits in the message digest, finding a message that corresponds to a given message digest can always be done using a brute force search in 2L evaluations. This is called a preimage attack and may or may not be practical depending on L and the particular computing environment. The second criterion, finding two different messages that produce the same message digest, known as a collision, requires on average only 2L/2 evaluations using a birthday attack.

In terms of practical security, a major concern about these new attacks is that they might pave the way to more efficient ones. Whether this is the case has yet to be seen, but a migration to stronger hashes is believed to be prudent. Some of the applications that use cryptographic hashes, such as password storage, are only minimally affected by a collision attack. Constructing a password that works for a given account requires a preimage attack, as well as access to the hash of the original password (typically in the shadow file) which may or may not be trivial. Reversing password encryption (e.g., to obtain a password to try against a user's account elsewhere) is not made possible by the attacks. (However, even a secure password hash cannot prevent brute-force attacks on weak passwords.).

The VHSA has the same operation as SHA-2, but the structure is using the SHA-1 style. So, the VSHA is more efficient than SHA-2 and has stronger security power that SHA-1.

It is very difficult to try to break a hash function because it needs great computational power which cannot be bought by any people. However, since there are similar operations like SHA-2, we could believe that the VHSA is eligible to become a secure hash function that could replace the SHA-1.

V. CONCLUSION

In short, the VSHA is an improvement from SHA-1 structure but using the SHA-2 operations to make it complex. The VSHA could replace the broken SHA-1 algorithm since, it was powerful enough to withstand from attacks.

To improve the VSHA, you can use some of the world hardest problem like discrete logarithms, integer factorization, or subset sums to make the hash function is more powerful. The are many cryptographic algorithm that relies on the problem and most of the are widely used.

REFERENCES

[1] http://www.cut-the-knot.org/do_you_know/pigeon.shtml. Online. [Accessed 13 May 2012]
[2] http://crackstation.net/hashing-security.Online.[Accessed 13 May 2012]
[3] http://www.itl.nist.gov/fipspubs/fip180-1.htm. Online [Acessed 13 May 2012]

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010

ttd

Edwin Lunando/13509024