A preliminary study of bit-integer hash function for bilinear pairing identity-based cryptosystem

D E Saputra*

Department of Informatics, STMIK AMIK BANDUNG, Jalan Jakarta No 28, Bandung, Indonesia

*dekastra@stmik-amikbandung.ac.id

Abstract. Hash function is a crucial part of identity-based cryptosystem, especially for identity-based cryptosystem that utilize bilinear pairing as its core mechanism. However, the hash function has specific requirement that most of commonly known function are not suitable for this purpose, especially the requirement that requires the output length of function is varies. A variable length hash function, such as HAVAL, only provide multiple option for its output length. While the identity-based cryptosystem require that the length of the output can be any number. This study aims to create a hash function to fulfill the identity-based cryptosystem requirement. This study modifies the MD5 hashing algorithm to produce a variable length output, which called the Bit-Integer hash function. The result shows a proof-of-concept that Merkle-Damgård construction can be used to create a variable length hash function. The hash value of Bit-Integer hash function can produce random-like result which is essential for any cryptographic hash function.

1. Introduction

Identity-based cryptosystem is an asymmetric cryptosystem that uses user identity as its public key [1]. The cryptosystem offers an encryption and signature scheme that does not require any third party to store user’s public key, thus reducing the complexity of an encryption scheme. The most common method to implement identity-based cryptosystem is by using bilinear pairing as its main mechanism [2].

This bilinear pairing-based identity-based cryptosystem requires 3 (three) types of hash function as part of its algorithm. Hash function is a one-way function, it is easy to compute the result for a certain input but extremely difficult to obtain that certain input given the result and the function. The first hash function takes a variable length bit string and map it to a member of a predetermined group, \( H_1: \{0,1\}^* \rightarrow G \). The second function convert an arbitrary bit string into an integer with certain order, \( H_2: \{0,1\}^* \rightarrow \mathbb{Z}_q \). The third hash function takes an input of arbitrary integer and maps it into a bit string with variable length, \( H_3: \mathbb{Z}_q \rightarrow \{0,1\}^n \).

The requirement of each hash function is so specific that most of known cryptographic hash function is not suitable for this purpose. The hash functions require a change of data types, for example from integer to bit string and vice versa. The commonly known hash function, for example Rivest, Carreira and Shen, can fulfill this requirement [3-5]. The MD5 can convert any message (usually in string) into a binary hash value (usually represented as hexadecimal). However, the problem lies within the variable length requirement.
The common hash function produces a fixed length output. For example, MD5 always produce a 128-bit long hash value. This condition is not suitable for the implementation of identity-based cryptosystem, especially for $H_3$. There are several hash functions that can produce a variable length hash value, such as Zheng et al. and Wang et al. However, the functions in Zheng and Wang are more suitable to be called as having multiple output rather than variable output [6,7]. HAVAL provides an output length option that can be chosen at the start of algorithm: 128, 160, 192, 224, or 256 bits [6]. The $H_3$ used in Malone-Lee needs to produce a hash value with length that matched the original message before encryption [8]. Since the length of message is arbitrary, HAVAL is not suitable to be implemented for this case.

This study is a preliminary study that aims to create a hash function that suitable for identity-based cryptosystem. Particularly, this study only focuses on creating $H_3$ function. The basic hypothesis is that such a function can be constructed by using Merkle-Damgård construction [9,10]. To achieve the proof-of-concept of this hypothesis, this study will modify MD5 function so that it can produce a variable length output.

2. Basic construction of bit-integer hash function

As one-way function, a hash function should have a certain characteristic. Given a certain input, there should be no entity that can predict the output. However, if the same input is used again in the hash function, then the output should still the same. An arithmetic circuit constructed using Merkle-Damgård construction method should have the characteristic [9].

A function in Merkle-Damgård construction is an iterative circuit. A plaintext (to be hashed) is usually divided into blocks with fixed length (figure 1). Using a constant initiation vector, each block is processed as part of each iteration. The hash value is produced at the end of iteration. The hash value usually has a fixed length (e.g. 128 bit). The result should act like a randomly generated, in sense that no one can predict the precise result from a particular input.

![Figure 1. Merkle-Damgård construction.](image)

The function $H_3$ in identity-based cryptosystem takes an input in form of an integer and produce an output in form of binary integer with a predetermined length. The length of the output is determined by the user before the calculation began. To formalize these requirements, the function $H_3$ can be defined as follow:

**Definition 1.** Let $H_3 = f(x, n)$ be a one-way function that takes input of $x \in \mathbb{Z}$ and a length of $n > 0, n \in \mathbb{Z}$. Function $H_3$ is suitable for identity-based cryptosystem if for any $x$, $H_3$ produce a random-like binary string with length $n$, $H_3 = \{0,1\}^n$. 
There are several popular Merkle-Damgård hash algorithm, including MD5 [3] and SHA-1. The MD5 and SHA-1 function is acting as random oracle [10], thus producing a random result. Both functions also operate on binary level, so it always produces a binary string as a result. However, both functions have fixed length (MD5 has 128-bit length and SHA-1 has 160-bit length), so both functions cannot satisfy Definition 1.

The proposed Bit-Integer hash function is based on MD5 algorithm. As preliminary study, this choice is sufficient, as this work aims to explore the possibility on how Definition 1 can be fulfilled. To overcome the fixed-length problem of MD5, similar approach from Zheng and Wang is used [6,7]. In Zheng and Wang, the iterative function produces a fixed-length result. The result is then post processed according to the chosen length option. By using AND operation on each bit element, a desired length of hash value is produced [6,7].

The complete steps of Bit-Integer hash function can be seen in figure 2. An input in form of integer is divided into blocks, using padding rule of MD5. To divide the integer, each number is treated as a string and parsed as a byte (an 8-bit string) and padded with padding bits so the total length is equal to modulo 512. The bits then divided into block with length 512-bits each. The MD5 algorithm will create a 128-bit long hash value. The value is then post-processed, so the length of the bit is equal to the desired length. The complete algorithm of Bit-Integer hash function can be found in Algorithm 1.

![Diagram of Bit-Integer hash function](image)

**Figure 2.** Operation of bit-integer hash function.

**Table 1.** Algorithm 1 Bit-integer hash algorithm (Input: \(x, n\)).

| Algorithm 1. Bit-Integer Hash Algorithm (Input: \(x, n\)) |
|---------------------------------------------------------|
| 1: Begin.                                               |
| 2: Parse each element in \(x\) as bit.                   |
| 3: Add padding bits into \(x\), so that \(l_x \mod 512 = 0\), where \(l_x\) is the length of \(x\). |
| 4: Divide \(x\) into blocks \(m_1, m_2, ..., m_i\), where each block has length of 512 bit. |
| 5: Run MD5 algorithm with blocks \(m_1, m_2, ..., m_i\) as input. Set \(h\) as result of this operation in byte array. |
Table 1. cont.

6: Set $h'$ as byte array with length $n$.
7: for each element in $h'$ do
8: if index of $h'$ $\geq$ length of $h$.
9: Set $h'[\text{index}] = h[\text{index} \mod l_h] \oplus h'[\text{index} - l_h]$
10: else if index of $h'$ > 0 and index $< l_h$ do
11: Set $h'[\text{index}] = h[\text{index}] \oplus h[\text{index} - 1]$
12: else do
13: Set $h'[\text{index}] = h[\text{index}] \ll n$
14: end if.
15: end for.
16: return $h'$ as result.
17: End.

In algorithm 1, the symbol “[index]” is used to represent the number of array element. While symbol “$\ll n$” means that the bit is left shifted by $n$. From the algorithm, the length of result is gained by adding each byte in $h'$. By doing so, the desired length can be controlled freely. The length $n$ in Boneh and Malone-Lee represent the length of original plaintext that will be encrypted using identity-based encryption algorithm [2,8]. Since the plaintext can be divided into each character and each character can be represented as a single byte (value of its bits representation), it can be concluded that $n$ is always a multiplication of 8. That is why, in algorithm 1 the length $n$ is controlled in bytes.

3. Verification of proposed hash function

The aim of this study is to achieve a proof-of-concept that Merkle-Damgård construction can be used to construct a suitable hash function for bilinear pairing identity-based cryptosystem, especially for a Bit-Integer type hash function. To prove that the constructed Bit-Integer hash function satisfies definition 1, the function is verified using several tests.

The first test is to see that the function can produce a random-like output for a certain input. Table 1 show the result of this test. There are 7 inputs for this test, each input slightly differs than the default value, with “1234567” as the default value. The result in table 1 shows that each input produces a different output. Hamming distance is used to show that the bits result has significant difference, thus illustrating that the output cannot be predicted even for two similar value [11]. The Hamming distance measure the difference of value per position between two string with equal length, which formulated as follow:

$$\text{Hamming Distance} = \sum_{i=1}^{n} |b_{1i} - b_{2i}|$$

where $i$ denotes the position of bit in the string, and $b_{1i}, b_{2i}$ denotes the compared bit-strings.

Table 2. The result of output testing with length 8 bytes.
The result of the first test show that each output greatly differs, with average Hamming distance of 19. This prove that the proposed Bit-Integer hash function can produce a random-like output. The closest distance is produced by input "1234567" and "1234467". Even on those input, the hash function still produces results with Hamming distance of 13. That means there are 13 bits difference from the entire 64 bits result. The result in table 1 also show that there is no certain pattern that can be seen from the output, thus create a random-like result.

The second test aims to see that the bit-integer hash function can produce an output with variable length and retain the random-like result. To prove this capability, a single input is feed to the function repeatedly, each with a different output length. The result of this test can be seen in table 2.

| Input       | Output Length (Byte) | Output Value (Bit) |
|-------------|----------------------|--------------------|
| 1234567     |                      |                    |
| 1           | 1                   | 11111000           |
| 2           | 2                   | 111000011101000    |
| 3           | 3                   | 1100000011010000   |
| 4           | 4                   | 10000000110100001  |
| 5           | 5                   | 00000000110100001  |
| 6           | 6                   | 00000000011010000  |
| 7           | 7                   | 00000000001101000  |
| 8           | 8                   | 00000000000110100  |

From the result, the Bit-Integer hash function can produce variable output length. The output length can be freely determined compared to the length options from function in Zheng and Wang [6,7]. However, as can be seen from the result, aside the first 8-bits all the bits are similar. This condition is generated from the post-processing phase in the Bit-Integer hash algorithm. From the algorithm, it can be seen that only the first byte has different input value that depends on the output length, while the rest of bytes is the same regardless the output length. However, this result is satisfying enough for the purpose of this preliminary study. Since, it shows the needed proof-of-concept that existing Merkle-Damgård construction can be used to construct a suitable hash function for identity-based cryptosystem.

4. Conclusion
The proposed Bit-Integer hash function to a certain extend can fulfil definition 1. Thus, making it suitable to be used in the implementation of identity-based cryptosystem, such as Boneh and Malone-Lee [2,8]. This study shows that any Merkle-Damgård based hash function can be modified to produce an output with variable length. The output length can be freely controlled, not like Zheng and Wang that provide an option of output length [6,7]. The hash value produced by Bit-Integer hash function also have the random-like characteristic, making it a proper cryptographic hash value.

However, the proposed function still has its limitation. For a certain input, the function still produces the same hash value despite the desired output length. This condition could enable a person to guess the hash value of an input of any length if the person has a value of a single length. To tackle this problem, the Merkle-Damgård construction can be used again. In this study, and similar study to produce variable output length, the output length is controlled in post processing phase. There is an opportunity to create an entirely new Merkle-Damgård-based hash function, where the number of iteration correspondence to the desired length of output. This approach, in theory, could produce a random-like result in various output, even for a single same output.

References
[1] Shamir A 1984 Identity-based cryptosystem and signature scheme. In Workshop on the Theory and Application of Cryptographic Techniques. Springer. p. 47-53.
[2] Boneh D and Franklin M 2001 Identity-based encryption from the Weil pairing. In Annual
International Crytoplogy Conference. Springer. p. 213-229.

[3] Rivest R 1992 The MD5 message-digest algorithm RFC 1321.
[4] Carreira-Perpinán M A and Raziperchikolaei R 2015 Hashing with binary autoencoders Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 557-566.
[5] Shen F, Shen C, Liu W and Shen H T 2015 Supervised discrete hashing Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition 37-45.
[6] Zheng Y, Pieprzyk J and Seberry J 1992 HAVAL-a one-way hashing algorithm with variable length output International Workshop on The Theory and Application of Cryptographic Techniques 81-104.
[7] Wang M J and Li Y Z 2015 Hash function with variable output length 2015 International Conference on Network and Information Systems for Computers (ICNISC) 190-193.
[8] Malone-Lee J 2002 Identity-based signcryption IACR Cryptology ePrint Archive 98.
[9] Merkle R 1979 Secrecy, auhtentiction, and public key systems (Thesis, California, Stanford University, Department of Electrical Engineering) Report No.: Technical Report No 1979-1.
[10] Coron J S, Dodis Y, Malinaud C and Puniya P 2005 Merkle-Damgård revisited: How to construct a hash function Annual International Cryptology Conference 430-448.
[11] Hamming R W 1950 Error detecting and error correcting codes Bell System Technical Journal 29(2) 147-160.