CHAPTER 1: INTRODUCTION

This chapter discusses the background of this research. This chapter is divided into eight sections, rationale, theoretical framework, conceptual framework/paradigm, statement of the problem, objective and hypotheses, assumption, scope and delimitation, and significance of the study.

1.1 Rationale

Recently, many technologies are being developed to anticipate future Quantum attacks[1]. Quantum computing or quantum computers are supercomputers that have many times the capabilities of today's advanced computers. By using Quantum Computing, attackers can solve complex mathematical problems very quickly. So, it is necessary to develop methods to deal with these attacks[2]. Several data security methods have been proposed to overcome Quantum Computing attacks, one of which is using a hash function[3], [4]. Additionally, hashes are considered a good candidate for security due to their ease of implementation and customization.

Several hash methods that are resistant to quantum attacks are Hash to Obtain Random Subset (HORS) [3], Hash to Obtain Random Subset-Tree (HORST) [3], , PRNG to Obtain Random Subset(PORS)[3], Forest to Obtain Random Subset(FORS) [3]. HORS is a scheme that utilizes hashes to create digital signatures. HORS is weak in the signature creation process because only one round of hashing is carried out, which results in security vulnerabilities in this method[3], [5]. This deficiency was then overcome by developing the Hash to Obtain Random Subset-Tree (HORST) [6]and Hash to Obtain Random Subset and Integer Composition (HORSIC) methods[7]. Hash to Obtain Random Subset-Tree is a development of the HORS method, which combines the Merkel Tree[8], [9] in creating public keys and signatures. However, HORST has the disadvantage that the public key is long because it is the result of combining all the seeds, and the security level is low because of the large number of signatures used. Since the signature is distributed, an attacker can guess the pattern of the signature using the public key[6], [10].

Hash to Obtain Random Subset and Integer Composition combines the HORS and Integer Composition methods[11], where the signature creation process uses the Integer Composition Algorithm. The Integer Composition Algorithm is the process of combining several positive integers to form a specific integer. However, the time required for creating compositions from the Integer Composition algorithm is quite large, which is a weakness of this method[12]. Therefore, a method is needed to improve the level of security and the relatively lower time consumption of the two methods.

1.2 Theoretical Framework

HORS [5], HORST [6], and HORSIC [13] use hashes for key generation. The method is that the seed (random value created by the signature maker) is used as a private key and will be hashed using SHA-256 to create a public key. Messages are hashed using SHA-256, and the results of the message hash are then grouped into eight hexadecimal bits. Each group is converted into a decimal (integer), and the resulting decimal number is used as an index. The index selects part of the public key for signature use.

The HORST process is similar to HORS, but the difference is in creating the public key and signature. After the index creation process uses messages, the seed that has been hashed is the private key. The private key will then be selected using the index to form a Merkle tree, where each leaf will be a signature, and the top of the tree will be the public key.

HORSIC is similar to HORS and HORST, which use a seed generation process to create the private key, and then the public key is obtained using the private key hash process for n rounds. The message is used for index creation, similar to HORS and HORST. The message will also be used to create Integer Composition. The seed will be hashed by n rounds used in the public key creation process minus the Integer Composition value obtained from the message, thus producing (n - integer Composition) to create a signature. The result of this subtraction will be used to calculate the number of hash rounds for the signature creation process.

1.3 Conceptual Framework

There are several processes involved in this research. For the key generation process in creating the seed, use the Random Number Generator function and the hash, namely SHA-256[14]. The result of the key creation process is a private key. The Merkel tree is used at the signature creation stage. The signature creation process is almost similar to HORST, where the message is used as an index, and the seed or private key is selected using the index as data combined in the Merkle tree creation process. In the leaf creation process, leaves will be hashed using SHA 256 after combining the leaves. The output of the signature creation process is the signature at the tree's root and the public key on each leaf of the Merkle tree. For the signature validation process, the public key and the signature that has been obtained will be matched.

1.4 Statement of the Problem

The problems in this research were found in several previous methods, such as HORST, which utilizes a Merkle tree. In the process, the private key is used as a node in the Merkle tree. Each leaf resulting from the concatenation of the private key is a signature that connects the leaf to the root of the Merkle tree. The root of the Merkle tree serves as the public key. This method addresses the security issues of the HORS method [15], which only uses one hash round in creating signatures, thus having a low level of security [16], [17]. However, HORST has a weakness, because the public key is the result of combining all the signatures, so it is easier to forge the signature using the public key[6], [10].

HORSIC is a method that combines HORS and integer composition. However, adding the integer composition method increases the processing of the signature creation. Additionally, HORSIC key generation uses repeated hashing, which increases the time consumption required [13]. Therefore, it remains a challenge to design a digital signature method that addresses the shortcomings of HORST in terms of the security level and HORSIC in terms of time consumption.

1.5 Hypothesis

This method aims to increase security, decrease time consumption, and reduce key length in previous methods. In the proposed method, the key generation uses a Merkel tree, where each leaf that has been concatenated will be hashed to the top of the Merkle tree. The hash uses SHA-256, which is considered safe and resistant to many types of cryptographic attacks. Public keys are taken from some leaf of each level of the Merkel tree. In the message section, the message will be hashed and converted into an index value without going through the Integer composition process.

This proposal hypothesizes that using SHA-256 as a hash will increase key security and overcome the key length problem due to concat in the Merkle tree. In this case, the private key is used as the bottom leaf of the Merkle tree. Then, a pair of the bottom leaves is concatenated. Furthermore, the concatenation result is hashed using SHA-256. The time consumption is also reduced because it eliminates the process of determining composition integer values, which requires quite a lot of time.

1.6 Assumption

This research assumes that the seed is a random value from 0 - 255, generated with a Pseudo Random Number Generator (PRNG). The hash process uses the SHA-256 or 256-bit Secure Hash Algorithm. The sender and Receiver have the same PRNG.

1.7 Scope and Delimitation

This research focuses on the development, implementation, and analysis of algorithms in the design of the modified HORST method. The scope of the study includes algorithm design, security analysis, and practical applications in the context of authentication protocols and digital signatures. The research findings cover key length, processing time consumption, and security levels using supporting methods such as SHA-256, Merkle tree, and integer composition.

1.8 Importance of the Study

Related to HORST and HORSIC, which play a role in overcoming the growing problem of quantum attacks by utilizing hash functions resistant to cryptanalysis attacks, the proposed method aims to overcome the cryptanalysis attack using an improved hash function[18], [19]. As a component of creating digital signatures, hash functions influence the level of signature security, so they are resistant to pre-image and collision attacks [12]. Therefore, developing HORST modifications is essential.