Thesis for the Degree of Master of Engineering

Providing Data Integrity for Container Dwelling Time in the Seaport

by

Sandi Rahmadika

Department of Information Systems (Interdiciplinary Program)
The Graduate School
Pukyong National University

February 2016

Thesis for the Degree of Master of Engineering

Providing Data Integrity for Container Dwelling Time in the Seaport

항구에서의 컨테이너 체류 시간에 대한 데이터 무결성 제공

by

Sandi Rahmadika

Department of Information Systems (Interdiciplinary Program)

The Graduate School

Pukyong National University

February 2016

Thesis for the Degree of Master of Engineering

Providing Data Integrity for Container Dwelling Time in the Seaport

항구에서의 컨테이너 체류 시간에 대한 데이터 무결성 제공

Advisor: Prof. Kyung-Hyune Rhee

by

Sandi Rahmadika

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Engineering

in the Department of Information Systems (Interdisplinary Program)
The Graduate School,
Pukyong National University

February 2016

Providing Data Integrity for Container Dwelling Time in the Seaport

A thesis

By

Sandi Rahmadika

Approved by:

(Chairman) ***Man-Gon Park***

(Member) ***Hilwadi Hindersah***

(Member) ***Kyung-Hyune Rhee***

26 February 2016

Contents

Contents i
List of Figures ii
List of Tables iii
Chapter 1. Introduction 1
1.1 Background 1
1.2 Related Work 2
1.3 Thesis Objective 3
1.4 Scope 3
1.5 Thesis Outline 4
Chapter 2. Preliminaries 4
2.1 Digital Signature 5
2.1.1 Principles of Digital Signatures 5
2.1.2 Basic Digital Signature Protocol 6
2.2 Security Services 7
2.3 The RSA Cryptosystem 8
2.3.1 Encryption and Decryption 9
2.3.2 Key Generation of RSA Algorithm 9
2.4 The RSA Signature Scheme 10
2.5 Hash Function 12
2.5.1 Basic Protocol for Digital Signatures with a Hash Function 13
2.5.2 Principal Input-Output of Hash Function 14
2.6 The Secure Hash Algorithm SHA-2 15
2.6.1 Preprocessing 17
2.6.2 Hash Computation 17
2.7 Cloud Computing and Storage 19
2.7.1 The SPI Framework for Cloud Computing 19
2.7.2 The Cloud Service Delivery Model 19
2.8 Container Dwelling Time 21

Chapter 3. System Requirements and Design 22
3.1 Data Access Architecture in the Cloud 22
3.2 The Impact Falsification of Dwelling Time 23
3.3 The Signed and Verification Process 24
3.3.1 Signed Process 24
3.3.2 Verification Process 25
3.3.3 Concept Digital Signature to Provide Data Integrity 26
3.4 Authorized and Unauthorized Person 27
3.4.1 The Authorized Person 27
3.4.2 Symbol and Notation between Authorized Person 28
3.4.3 The Unauthorized Person 30
3.5 Overall Flowchart of the Proposed System 30
3.6 Android Application for Dwelling Time Prediction 32
Chapter 4. System Implementation and Analysis 33
4.1 Overall System and Application 33
4.2 Container Import Dwelling Time Prediction 33
4.3 Digital Signature Implementation 34
4.3.1 User Interface of RSA Key 34
4.3.2 Signed the Dwelling Time Data 37
4.3.3 Verifying the Dwelling Time Data 38
4.4 System Database 39
4.5 Android Apps Amount of Equipment 40
4.6 Experimental Result 40
Chapter 5. Conclusion and Future Work 49
5.1 Conclusion 49
5.2 Future Work 49
References 50

List of Figure

Figure 2.1 Principle of Digital Signatures 6
Figure 2.2 Generic Digital Signature Protocol 7
Figure 2.3 RSA Key Generation 10
Figure 2.4 The RSA Signature Scheme 11
Figure 2.5 RSA Signature Algorithm 12
Figure 2.6 Signing of Long Messages with a Hash Function 13
Figure 2.7 Basic Protocol for Digital Signatures with a Hash Function 13
Figure 2.8 Principal Input-Output Behavior of Hash Function 14
Figure 2.9 The Three Security Properties of Hash Function 15
Figure 2.10 One iteration in a SHA-2 family compression function 16
Figure 2.11 Padding of a Message in SHA-1 17
Figure 2.12 Eighty-round compression function of SHA-1 18
Figure 2.13 SPI Services Model 19
Figure 2.14 Cloud Services Delivery Model 20
Figure 3.1 Data Access Architecture in the Cloud 22
Figure 3.2 Signed Dwelling Time Data Process 24
Figure 3.3 Verification for Dwelling Time Data Process 25
Figure 3.4 Digital Signature in Cloud 26
Figure 3.5 Comparing the Result of Hash Value 27
Figure 3.6 Structure of the Officer in the System 28
Figure 3.7 Communication between Authorized Person 29
Figure 3.8 Flowchart of the Overall System 31
Figure 4.1 Main Menu of Digital Signature Application 33
Figure 4.2 Menu of Container Dwelling Time Prediction 34
Figure 4.3 User Interface of RSA Algorithm 35
Figure 4.4 User Interface of Signed Process 37
Figure 4.5 Verifying Dwelling Time Data 39
Figure 4.6 System Database Using MySQL 39
Figure 4.7 Android Apps Amount of Equipment 40

Figure 4.8 Amount of Equipment 41
Figure 4.9 Dwelling Time Prediction 41
Figure 4.10 RSA Keys 43
Figure 4.11 Signing the Dwelling Time Data 44
Figure 4.12 Information in Database 44
Figure 4.13 Dwelling Time Data in Cloud 45
Figure 4.14 Verify the Dwelling Time Data 45
Figure 4.15 Indicator If Data Is Valid 46
Figure 4.16 The Original of Dwelling Time Data 46
Figure 4.17 The Attacker Change the Data 46
Figure 4.18 Message Digest Value With the Data Changed 47
Figure 4.19 Indicatior When the Data Has Been Changed 47

List of Tables

Table 2.1 The MD4 Family of Hash Function.. 16

Table 3.1 The Impact Falsification of Dwelling Time 23

Table 4.1 Different Key Size 35

항구에서의 컨테이너 체류 시간에 대한 데이터 무결성 제공

산디 라흐마디카

부경대학교 대학원 정보시스템협동과정

요약

데이터 무결성은 정보보호에서 기본적인 요소로서 데이터 생성부터 수용까지의 과정에서 데이터가 변경되지 않았다는 것을 보장해준다. 본 연구에서 다뤄진 항구에서의 컨테이너 체류 시간은 항구에서 컨테이너가 머무른 시간에 대한 정보와 관련된 것이다. 컨테이너 체류 시간에 대한 데이터가 데이터 생성자로부터 클라우드에 저장될 때까지 고의적인 공격이나 어떤 실수에 의해서 변경된 것을 검증할 수 있는 기술로 데이터 무결성을 고려할 수 있다. 체류 시간 데이터에 대한 데이터 무결성을 제공하기 위해 SHA-256 해시 알고리즘을 사용한 디지털 서명 방식을 사용하였다. 디지털 서명 기법으로는 공개키 알고리즘인 RSA 알고리즘을 사용하였고 사용된 키는 1,024 비트, 2,048 비트, 3,072 비트 중 선택하여 사용할 수 있다. 시뮬레이션을 통하여 구현한 방법이 체류 시간에 대한 완전한 무결성을 제공해줌을 보였다.

Chapter 1.   Introduction

1.1   Background

Data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle [1] and a critical aspect to the design, implementation and usage of any system which stores, processes or retrieves data [2]. Data integrity is a fundamental component of information security and as a process data integrity verifies the data has remained unaltered in transit from creation to reception. Data integrity is the opposite of data corruption, which is a form of data loss [3]. The overall intent of any data integrity technique is the same, ensure the data is recorded exactly as intended (such as a database correctly rejecting mutually exclusive possibilities) and upon later retrieval, ensure the data is the same as it was when it was originally recorded. In short, data integrity aims to prevent unintentional changes to information. Any unintended changes to data as the result of a storage, retrieval or processing operation, including malicious intent, unexpected hardware failure and human error are failure of data integrity [1]- [2].

The sharing of dwelling time prediction and equipment optimization data in cloud computing provides many advantages for user like time efficiency, ease in communicating, cost effective and many others but the security issues to keep the data from being stolen or changed by the attacker is one of the considerations in the cloud [4]. As cloud computing is being widely adopted, data security is becoming one of the major concerns of data owners. Data integrity is an important factor in almost any data and computation related context. It is not only one of the qualities of service but also an important part of data security and privacy [5]. In this research using the seaport data from our previous research about dwelling time prediction and equipment optimization in JICT seaport, Indonesia [6].

Data exchange in the cloud must be completely safe and secure to avoid the unauthorized user (attacker) to change the data. The data received by officer is very important and vital therefore the data from an authorized sender should be original without any data changes made by the attacker. The data needs to be maintained its integrity because it will affect the performance of the seaport. In this research, digital signature with the hashing algorithm (SHA-1) which will be encrypted and decrypted using the keys from RSA algorithm are proposed in this research to verify data received and to ensure the data is still original from the sender without any modification from attacker during stored in the cloud.

1.2 Related Work

In this section, we briefly explain about several studies and works which related to the thesis. Dwelling time prediction and equipment optimization data can be created, gathered and managed by authorized officer and by using digital signature to sign and verify data. A VLSI implementation of the digital signature scheme is proposed by P. Kitsos, N. Sklavos and O. Koufopavlou for efficient usage in any cryptographic protocol. The architecture is based on Secure Hash Algorithm. The whole desing was captured by using VHDL language and a FPGA device was used for the hardware implementation of the architecture. The proposed VLSI implementation of digital signature scheme achieves a data throughput up to 32 Kbit/sec [7]. Don Jomar, Corazon Gracia, Chaves Enrico and friends using digital signature based on data encryption standard integrating HMAC and implemented in multi cast messenger application to provide data integrity verification and messages non-repudiation of the symmetric cryptosystem [8].

In this thesis we use a pairing key from RSA algorithm to encrypt and decrypt messages digest from the hashing result. Thangavel M., Varalakshmi P., Muralli Mukund and friends in 2014 proposed enhanced and secure RSA key generation scheme (ESRKGS) and the public component n is the product of two large prime numbers but the values of encryption *(E)* and decryption *(D)* keys are based on the product of four large prime numbers *(N)* making the system highly secured. With the existing factorization techniques, it is possible only to find the primes p and q. A comparison was done between the traditional RSA scheme a recent RSA modified scheme and the author scheme to show that the proposed technique is efficient [9]. Digital signature using RSA algorithm is also proposed by Chang Lin and Chen Chang in 2006 which is digital signature schemes with fault tolerance make possible for error detections and corrections during the processes of data computations and transmissions, the author improved Zhang's and Lee and Tsai's schemes to overcome the security flaw, the improved scheme also meets all requirements for digital signature with fault tolerance [10].

In this thesis, our contribution is implementation the digital signature using Secure Hash Algorithm-1 (SHA-1) to ensure the data integrity of dwelling time prediction and equipment optimization in the seaport. The secret and public key are obtained from RSA algorithm.

1.3 Thesis Objective

This research aims to develop a user interface using digital signature to ensure the data integrity in the seaport (that is nobody changes the data) and to avoid the fake dwelling time and equipment optimization data. Dwelling time prediction and equipment optimization data are obtained from the author's other research [6]. The data should be maintained its integrity in order not to be changed by the attacker with a different motivation. There are several main objectives in this research by using digital signature to protect the integrity of data in the seaport which are:

1. Signing process. Make a signed and attached to the data before send to the cloud. The signing is used by authorized recipient to comparing the hash value with the data decryption.
2. Verify process. This process can be used to guarantee the integrity of the data by checking and comparing the hash value with the data decryption. If the decryption result is equal to the value of signing it means there are no changes in the data (still intact). Otherwise, if the hash value is not equal it means the data has been changed.
3. The result of digital signature can be considered for officers to apply it directly in the seaport. If the data integrity is still intact then the officer can use that data to be applied at the seaport. Otherwise, if it is proven that the data has been changed, then the officer must be able to take precaution steps.

1.4 Scope

Digital signature has been widely used for secure and protect the data over the Internet such as cloud. In this thesis, we use Secure Hash Algorithm-256 (SHA-256) to compress a message of the data in the seaport. RSA algorithm is used in this thesis to produce a pair of key, namely public and secret key which are used to encrypt and decrypt a message digest. Secret key is used by the sender to encrypt the message digest of a data before sending to the cloud storage. Public key is used by the recipient for decrypting the message digest and the recipient gets the value and checking the integrity of that data by comparing signing value with the decrypted value.

Container dwelling time (CDT) data in the seaport is used in this thesis and CDT gives detailed information about the length of time which the container stayed in the seaport. RSA scheme with the hashing algorithm (SHA-256) are used in this thesis to verify the received data is the original one from the sender without any modification from attacker during stored in the cloud storage.

1.5 Thesis Outline

The thesis has been divided into five chapters which are:

Chapter I, Introduction: introduction consists of thesis background, related work, thesis objective, scope and thesis outlines.

Chapter II, Preliminary: literature review explains the theoretical supports and methods. It includes explanation about digital signature with hash value, and RSA algorithm. In this session also explained about basic cloud and android application which are used in the research.

Chapter III, System Requirements and Design: In this chapter contains model system of digital signatures and explained process using flowchart systems. Symbol and notation explained briefly and diagram whole system also explained in this chapter.

Chapter IV, System Implementation and Analysis: Explained detail implemented digital signature, user interface all of program like user interface for RSA Key. In this chapter also give example original data and data that has been changed by the attacker with different value of message digest.

Chapter V, Conclusion and Future Work: this chapter contains conclusions and additional features that is required.

Chapter 2

Chapter 2. Preliminaries

2.1 Digital Signature

Digital signature are one of the most important cryptographic tools and widely used today. Applications for digital signatures range from digital certificates for secure e-commerce to legal signing of contracts and secure software updates. Together with key establishment over insecure channels, they form the most important instances for public key cryptography [11]. Digital signature shares some functionality with handwritten signatures. In particular, they provide a method to assure that a message is authentic to one user, i.e., it in fact originates from the person who claims to have generated the message. However, they actually provide much more functionality [12] [11].

2.1.1 Principles of Digital Signatures

The property of proving that a certain person generated a messages is obviously also very important outside the digital domain. In the real, "analog" world, this is achieved by handwritten signatures on paper. For instance, if we sign a contract or sign a check, the receiver can prove to a judge that we actually signed the message [1]. Digital signature can also be used to testify (or certify) that a public key belongs to a particular person. This is done by signing the combination of the key and the information about its owner by a trusted key. The digital signature by a third party (owner of the trusted key), the public key and information about the owner of the public key are often called certifies [7]. Digital signatures are based on public key cryptography, also known as asymmetric cryptography. Using a public key algorithm such as RSA, one can generate two keys that are mathematically linked: one private and one public. To create a digital signature, signing software (such as an email program) creates a one-way hash of the electronic data to be signed. The private key is then used to encrypt the hash.

As with conventional hand-written signatures, only the person who creates a digital message must be capable of generating a valid signature. In order to achieve this with cryptographic primitives, we have to apply public-key cryptography. The basic idea is that the person who signs the message uses a private key, and the receiving party uses the matching public key [13].

The principle of digital signature scheme is shown in Fig. 2.1.

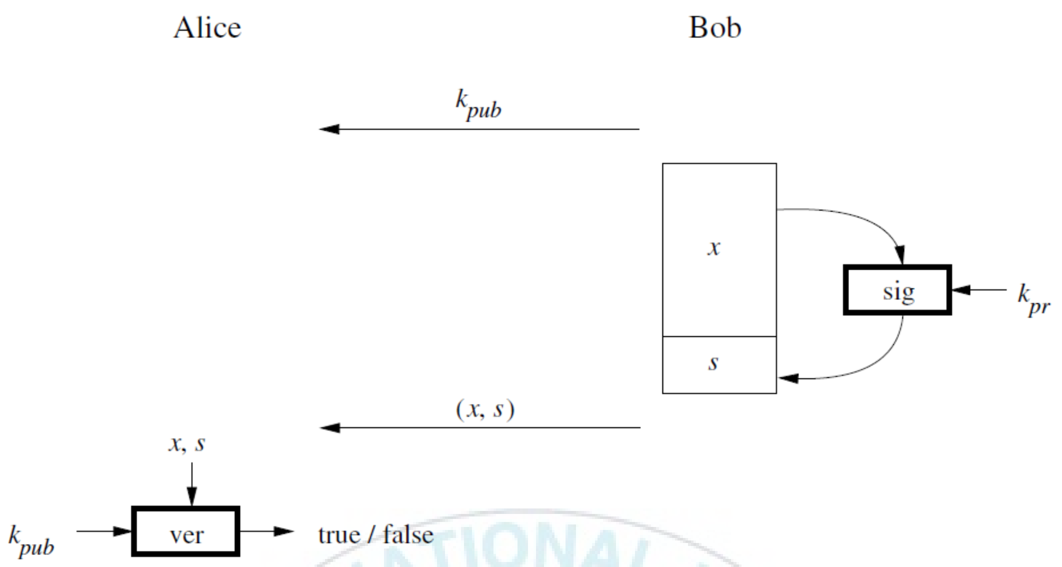


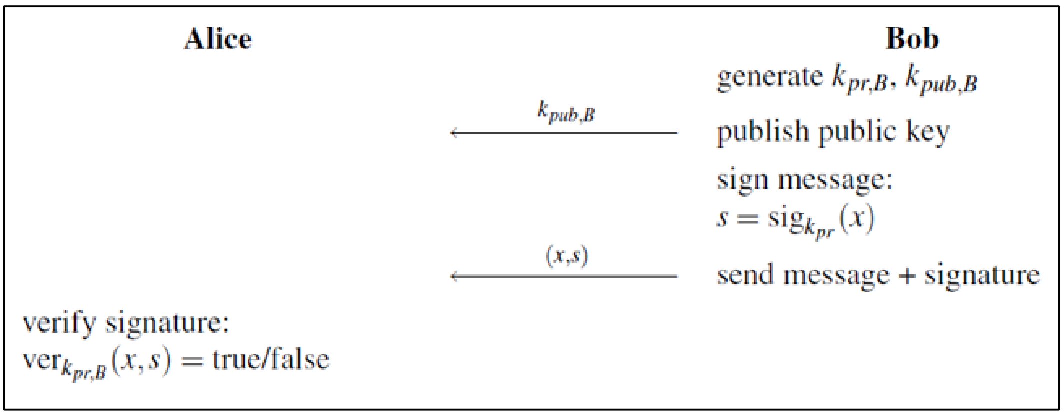
Figure 2.1 Principle of digital signatures [1]

The process starts with Bob signing the message *x*. The signature algorithm is a function of Bob's private key, K_{pr}. Hence, assuming he in fact keeps his private key, only Bob can sign a message *x* on his behalf. In order to relate a signature to the message, *x* is also an input to the signature algorithm. After signing the message, the signature *s* is appended to the message *x* and the pair (*x*, *s*) is sent to Alice. It is important to note that a digital signature by itself is of no use unless it is accompanied by the message. A digital signature without the message is the equivalent of a handwritten signature on a strip of paper without the contract or a check that is supposed to be signed.

2.1.2 Basic Digital Signature Protocol

The digital signature itself is merely a (large) integer value, for instance, a string of 2048 bits. The signature is only useful to Alice if she has means to *verify* whether the signature is valid or not [13]. For this, a verification function is needed which takes both *x* and the signature *s* as inputs. In order to link the signature to Bob, the function also requires his public key. Even though the verification function has long inputs, its only output is the binary statement "true" or "false". If *x* was actually signed with the private key that belongs to the public verification key, the output is true, otherwise it is false [1] [13].

From these general observations we can easily develop a generic digital signature protocol as shown in Figure 2.2.



Alice
Bob
generate $k_{pr,B}$, $k_{pub,B}$
$k_{pub,B}$
publish public key
sign message:
$s = \mathrm{sig}_{k_{pr}}(x)$
(x,s)
send message + signature
verify signature:
$\mathrm{ver}_{k_{pr,B}}(x,s) = \text{true/false}$

Figure 2.2. Generic digital signature protocol

From this set-up, the core property of digital signatures follows: A signed message can unambiguously be traced back to its originator since a valid signature can only be computed with the unique signer's private key. Only the signer has the ability to generate a signature on his behalf. Hence, we can *prove* that the signing party has actually generated the message [1].

2.2 Security Services

The act of ensuring data is not lost when critical issues arise. Since most information is stored on computers in our modern era, information assurance is typically dealt with by IT security specialists. One of the most common methods of providing information assurance is to have an off-site backup of the data in case one of the mentioned issues arise [2]. There are existing many security services, but the most important ones which are desirable in many applications are as follows:

1. **Confidentiality:** Information is kept secret from all but authorized parties.
2. **Integrity:** Messages have not been modified in transit.
3. **Message Authentication:** The sender of a messages is authentic. An alternative term is *data origin authentication.*
4. **Nonrepudiation:** The sender of a message cannot deny the creation of the message. Different application call for different sets of security services. For instance, for private e-mail the first three functions are desirable, whereas a corporate e-mail

system might also require nonrepudiation. As another example, if we want to secure software updates for a cell phone, the chief objectives might be integrity and message authentication because the manufacturer primarily wants to assure that only original updates are loaded into the handheld device.

5. **Identification/entity authentication:** Establish and verify the identity of an entity, e.g., a person, a computer or a credit card.
6. **Access control:** Restrict access to the resources to privileged entities.
7. **Auditing:** Provide evidence about security relevant activities [1].

Which security services are desired in a given system is heavily applicationspecific. For instance, anonymity might make no sense for an e-mail system since e-mails are supposed to have a clearly identifiable sender. On the other hand, carto-car communication systems for collision have a strong need to keep cars and drivers anonymous in order to avoid tracking. As a further example, in order to secure an operating system, access control to certain parts of a computer system is often of paramount importance. Most but not all of these advanced services can be achieved with the crypto algorithms from this book. However, in some cases noncryptographic approaches need to be taken. For instance, availability is often achieved by using redundancy, e.g., running redundant computing or storage systems in parallel. Such solutions are only indirectly, if at all, related to cryptography [1].

2.3 The RSA Cryptosystem

The RSA crypto scheme, sometimes referred to as the Rivest–Shamir–Adleman algorithm, is currently the most widely used asymmetric cryptographic scheme, even though elliptic curves and discrete logarithm schemes are gaining ground. RSA was patented in the USA (but not in the rest of the world) until 2000 [10]. However, it should be noted that RSA encryption is not meant to replace symmetric ciphers because it is several times slower than ciphers such as AES. This is because of the many computations involved in performing RSA [9]. RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this symmetry is based on practical difficulty of factoring the product of two large prime numbers, the factoring problem.

2.3.1 Encryption and Decryption

The RSA encryption and decryption is done in the integer ring Z_n and modular computations play a central role. RSA encrypts plaintexts *x*, then consider the bit string representing *x* to be an element in $Z_n = \{0,1, . . . ,n-1\}$. As a consequence the binary value of the plaintext *x* must be less than *n*. Encryption with the public key and decryption with the private key are as shown below:

- **RSA Encryption** given the public key $(n, e) = k_{pub}$ and the plaintext *x*, the encryption function is:

$$y = e_{k_{pub}}(x) \equiv x^e \bmod n \tag{2.1}$$

  where $x, y \in \boldsymbol{Z_n}$

- **RSA Decryption** given the private key $d = k_{pr}$ and the ciphertext *y*, then:

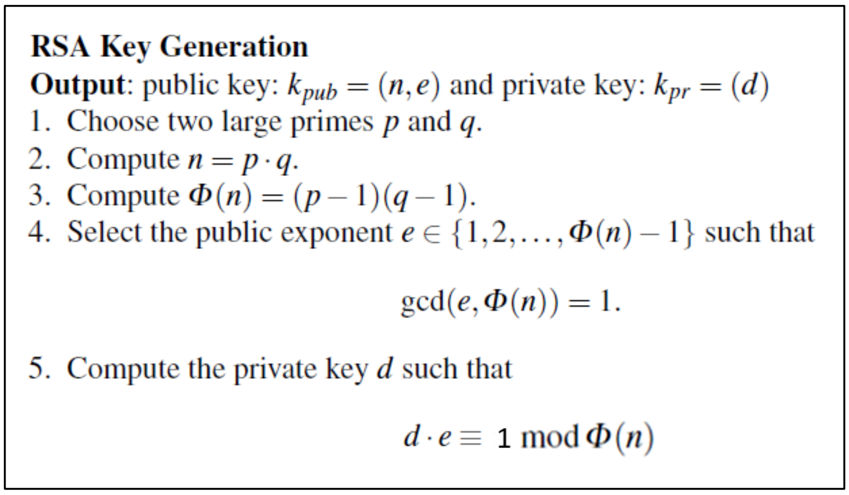
$$x = d_{k_{pr}}(y) \equiv y^d \bmod n \tag{2.2}$$

  where $x, y \in \boldsymbol{Z_n}$

In practice, *x*, *y*, *n* and *d* are very long numbers, usually 1024 bit long or more. The value *e* is sometimes referred to as *encryption exponent* or *public exponent*, and the private key *d* is sometimes called *decryption exponent* or *private exponent*. If Alice wants to send an encrypted message to Bob, Alice needs to have his public key (*n*,*e*), and Bob decrypts with his private key *d* [1].

2.3.2 Key Generation of RSA Algorithm

A distinctive feature of all asymmetric schemes is that there is a set-up phase during which the public and private key are computed. Depending on the public-key scheme, key generation can be quite complex. As a remark, we note that key generation is usually not an issue for block or stream ciphers. Here are the steps involved in computing the public and private-key for an RSA cryptosystem is shown in Figure 2.3. A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime numbers can feasibly decode the message.


RSA Key Generation

Output: public key: $k_{pub} = (n, e)$ and private key: $k_{pr} = (d)$

1. Choose two large primes p and q.
2. Compute $n = p \cdot q$.
3. Compute $\Phi(n) = (p-1)(q-1)$.
4. Select the public exponent $e \in \{1, 2, \ldots, \Phi(n)-1\}$ such that

$$\gcd(e, \Phi(n)) = 1.$$

5. Compute the private key d such that

$$d \cdot e \equiv 1 \bmod \Phi(n)$$

Figure 2.3. RSA key generation [13]

State a few requirements for the RSA cryptosystem:

1. Since an attacker has access to the public key, it must be computationally infeasible to determine the private key d given the public key values e and n.
2. Since x is only unique up to the size the modulus n, we cannot encrypt more than l bits with one RSA encryption, where l is the bit length of n.
3. It should be relatively easy to calculate $x^e \bmod n$, i.e., to encrypt, and $y^d \bmod n$, i.e., to decrypt. This means we need a method for fast exponentiation with very long numbers.
4. For a given n, there should be many private key/public key pairs, otherwise an attacker might be able to perform a brute-force attack. (It turns out that this requirements is easy to satisfy) [1].

2.4 The RSA Signature Scheme

The RSA signature scheme is based on RSA encryption. Its security relies on the difficulty of factoring a product of two large primes (the integer factorization problem). Since its first description in 1978, the RSA signature scheme has emerged as the most widely used digital signatures scheme in practice [14]. Suppose Bob wants to send a signed message x to Alice. He generates the same RSA keys that were used for RSA encryption. At the end of the setup he has the following parameters:

- Bob's private key: $k_{pr} = (d)$
- Bob's public key: $k_{pub} = (n, e)$

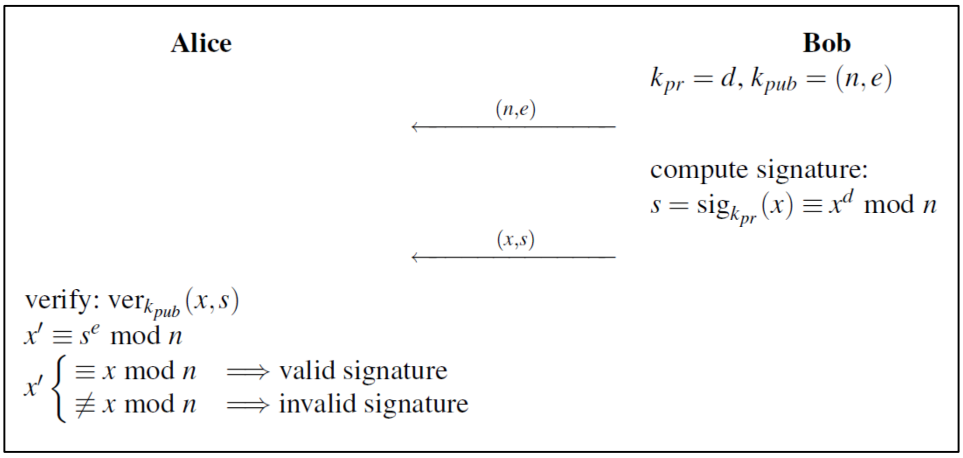

Alice
Bob
$k_{pr} = d,\ k_{pub} = (n,e)$
(n,e)
compute signature:
$s = \mathrm{sig}_{k_{pr}}(x) \equiv x^d \bmod n$
(x,s)
verify: $\mathrm{ver}_{k_{pub}}(x,s)$
$x' \equiv s^e \bmod n$
$x' \begin{cases} \equiv x \bmod n & \Longrightarrow \text{valid signature} \\ \not\equiv x \bmod n & \Longrightarrow \text{invalid signature} \end{cases}$

Figure 2.4. The RSA signature scheme

As can be seen from the protocol, Bob computes the signature *s* for a message *x* by RSA-encrypting *x* with his private key k_{pr}. Bob is the only party who can apply k_{pr}, and hence the ownership of k_{pr} authenticates him as the author of the signed message. Bob appends the signature *s* to the message *x* and sends both to Alice. Alice receives the signed message and RSA-decrypts *s* using Bob's public key k_{pub}, yielding *x*. If *x* and x' match, Alice knows two important things: First, the author of the message was in possession of Bob's secret key, and if only Bob has had access to the key, it was in fact Bob who signed the message. This is called message authentication. Second, the message has not been changed in transit, so that message integrity is given. We recall from the previous section that these are two of the fundamental security services which are often needed in practice [14].

For example, suppose Bob wants to send a signed message ($x = 4$) to Alice. The first steps are exactly the same as it is done for an RSA encryption: Bob computes his RSA parameters and sends the public key to Alice. In contrast to the encryption scheme, now the private key is used for signing while the public key is needed to verify the signature [1]. RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and ϕ(n) must also be kept secret because they can be used to calculate d.

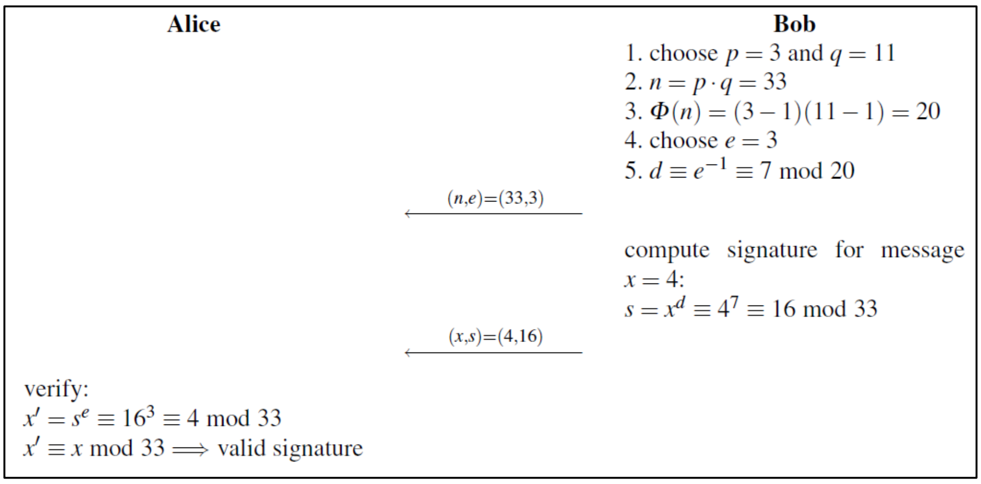

Alice
Bob
1. choose $p = 3$ and $q = 11$
2. $n = p \cdot q = 33$
3. $\Phi(n) = (3-1)(11-1) = 20$
4. choose $e = 3$
5. $d \equiv e^{-1} \equiv 7 \bmod 20$
$\xleftarrow{(n,e)=(33,3)}$
compute signature for message $x = 4$:
$s = x^d \equiv 4^7 \equiv 16 \bmod 33$
$\xleftarrow{(x,s)=(4,16)}$
verify:
$x' = s^e \equiv 16^3 \equiv 4 \bmod 33$
$x' \equiv x \bmod 33 \Longrightarrow$ valid signature

Figure 2.5. RSA signature algorithm

Alice can conclude from the valid signature that Bob generated the message and that it was not altered in transit, i.e., message authentication and message integrity are given. On the other side of the communication channel, RSA encryption requires the use of the private key by the receiver, while the digital signature scheme applies the public key for verification [1] [13].

2.5 Hash Function

Hash functions are an important cryptographic primitives and are widely used in protocols. They compute a *digest* of a message which is a short, fixed-length bitstring. For a particular message, the message digest, or *hash value*, can be seen as the fingerprint of a message, i.e., a unique representation of a message. Hash functions are also widely used for other cryptographic applications, e.g., for storing of password hashes or key derivation [14]. A hash function is any function that can be used to map digital data of arbitrary size to digital data of fixed size. The values returned by a hash function are called hash values, hash codes,hash sums, or simply hashes. One use is a data structure called a hash table, widely used in computer software for rapid data lookup. Hash functions accelerate table or database lookup by detecting duplicated records in a large file [15].

Hence, for performance as well as for security reasons we would like to have one *short signature* for a message arbitrary length. The solution to this problem is hash function. If we had a hash function that somehow computes a fingerprint of the message *x*, we could perform the signature operation as shown in Figure 2.6 [14].

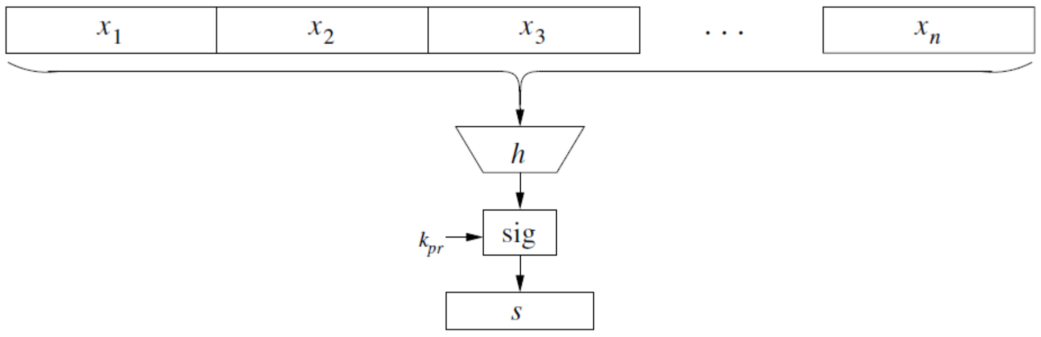


Figure 2.6. Signing of long messages with a hash function

Assuming we possess such a hash function and to describe how a basic protocol for a digital signature scheme with a hash function. Bob wants to send a digitally signed messaged to Alice.

2.5.1 Basic Protocol for Digital Signatures with a Hash Function

Assuming we possess such a hash function, and describe a basic protocol for a digital signature scheme with a hash function. Bob wants to send a digitally signed message to Alice as we can see in Figure 2.7.

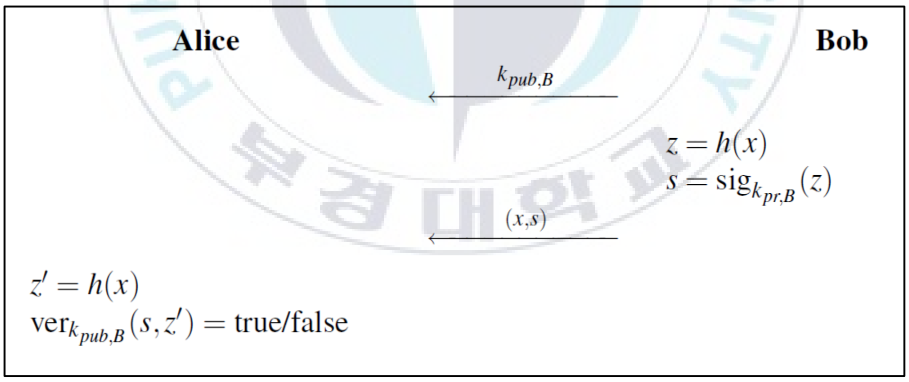

Alice
Bob
$k_{pub,B}$
$z = h(x)$
$s = \mathrm{sig}_{k_{pr,B}}(z)$
(x,s)
$z' = h(x)$
$\mathrm{ver}_{k_{pub,B}}(s,z') = \mathrm{true/false}$

Figure 2.7 Basic Protocol for Digital Signatures with a Hash Function [1]

Bob computes the hash of the message *x* and signs the hash value *z* with his private key k_{pr},*B*. On the receiving side, Alice computes the hash value *z*_ of the received message *x*. She verifies the signature *s* with Bob's public key k_{pub},*B*. We note that both the signature generation and the verification operate on the hash value *z* rather than on the message itself. Hence, the hash value represents the message. The hash is sometimes referred to as the *message digest* or the *fingerprint* of the message [1] [14].

2.5.2 Principal Input-Output of Hash Function

Every data have a different message digest and if we want to be able to apply a hash function to messages *x* of any size, and it is thus desirable that the function *h* is computationally efficient. Even if we hash large messages in the range of, say, hundreds of megabytes, it should be relatively fast to compute. Another desirable property is that the output of a hash function is of fixed length and independent of the input length. Practical hash functions have output lengths between 128–512 bits. Finally, the computed fingerprint should be highly sensitive to all input bits. That means even if we make minor modifications to the input *x*, the fingerprint should look very different. This behavior is similar to that of block ciphers. The properties which we just described are symbolized in Figure 2.8.

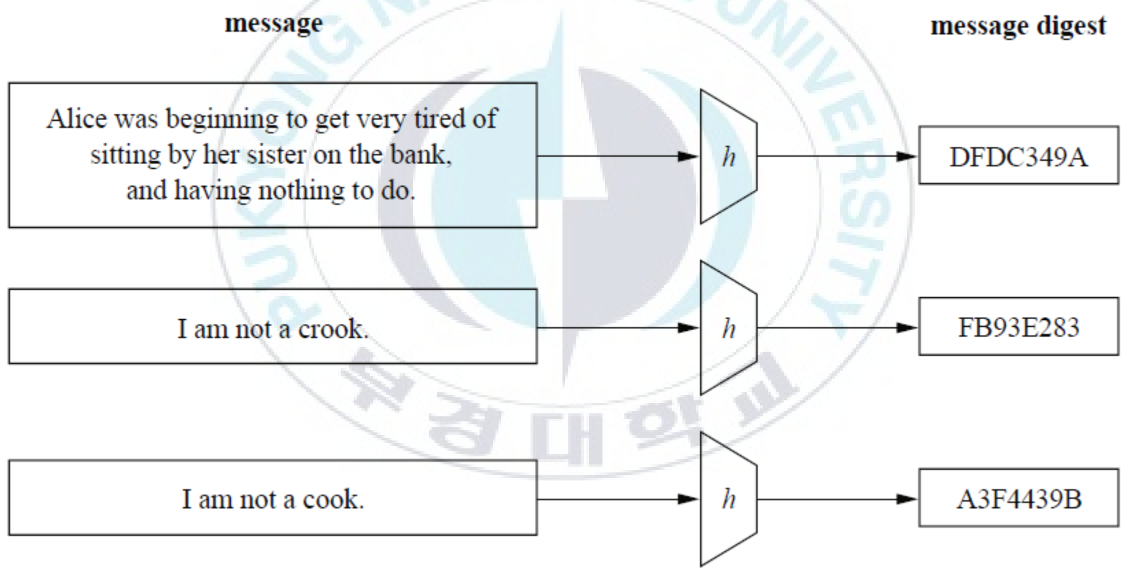

message
message digest
Alice was beginning to get very tired of sitting by her sister on the bank, and having nothing to do.
h
DFDC349A
I am not a crook.
h
FB93E283
I am not a cook.
h
A3F4439B

Figure 2.8 Principal input-output behavior of hash function [1]

As is often the case in cryptography, things can be tricky and there are attacks which use weaknesses of hash function. It turns out that there are three central properties which hash function need to possess in order to be secure:

1. Pre-image resistance (or one-wayness)
2. Second pre-image resistance (or weak collision resistance)
3. Collision resistance (or strong collision resistance)

These three properties are visualized in Figure 2.9. They are divided in the following.

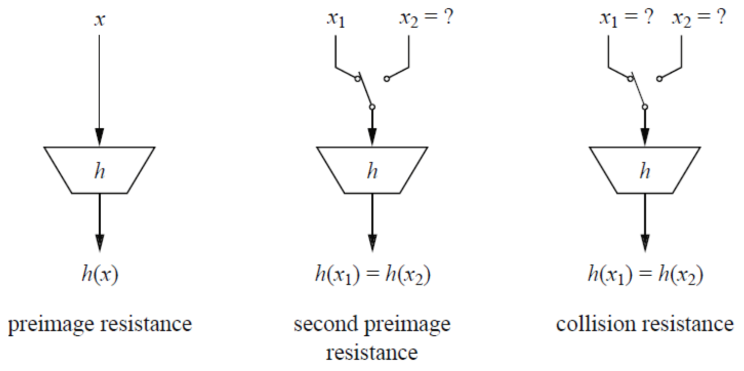


Figure 2.9 The three security properties of hash function

Properties of hash function:

1. **Arbitrary message size** *h(x)* can be applied to message *x* of any size.
2. **Fixed output length** *h(x)* produces a hash value *z* of fixed length.
3. **Efficiency** *h(x)* is relatively easy to compute.
4. **Pre-image resistance** for a given output *z*, it is impossible to find any input *x* such that $h(x) = z$, *h(x)* is one way.
5. **Second pre-image resistance** given x_1 and thus $h(x_1)$, it is computationally infeasible to find x_2 such that $h(x_1) = h(x_2)$
6. **Collision resistance** It is computationally infeasible to find any pairs $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$.

2.6 The Secure Hash Algorithm SHA-2

The Secure Hash Algorithm (SHA-2) is a set of cryptographic hash functions designed by the NSA. SHA stands for Secure Hash Algorithms. Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" (the output from execution of the algorithm) to a known and expected hash value, a person can determine the data's integrity. For example, computing the hash of a downloaded file and comparing the result to a previously published hash result can show whether the download has been modified or tampered with. A key aspect of cryptographic hash functions is their collision resistance: nobody should be able to find two different input values that result in the same hash output. SHA-2 includes significant changes from its predecessor, SHA-1. The SHA-2 family consists of six hash functions with digests (hash values) that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256.

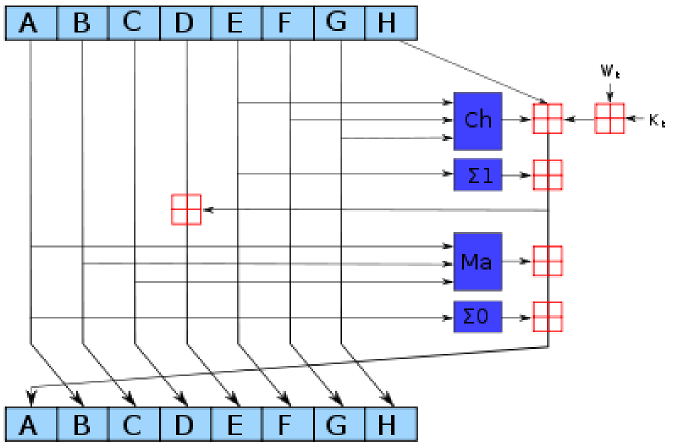


Figure 2.10 One iteration in a SHA-2 family compression function

Dedicated hash functions are algorithms that have been custom designed. A large number of such constructions have been proposed over the last two decades. In practice, by far the most popular ones have been the hash functions of what is called the MD4 family. MD5, the SHA family and RIPEMD are all based on the principles of MD4. MD4 is a message digest algorithm developed by Ronald Rivest. MD4 was an innovative idea because it was especially designed to allow very efficient software implementation.

Table 2.1 The MD4 family of hash function

Algorithm		Output [bit]	Input [bit]	No. of rounds	Collisions found
MD5		128	512	64	yes
SHA-1		160	512	80	not yet
SHA-2	**SHA-224**	224	512	64	no
	SHA-256	256	512	64	no
	SHA-384	384	1024	80	no
	SHA-512	512	1024	80	no

In 2004, collision-finding attacks against MD5 and SHA-0 where announced by Xiaoyun Wang. One year later it was claimed that the attack could be extended to SHA-1 and it was claimed that a collision search would take 2^{63} steps, which is considerably less than the 2^{80} achieved by the birthday attack. Table 11.2 gives an overview of the main parameters of the MD4 family. A further modification, SHA-224 was introduced in 2004 in order to fit the security level of 3DES. These four hash functions are often referred to as SHA-2.

2.6.1 Preprocessing

Before the actual hash computation, the message x has to be padded to fit a size of a multiple of 512 bit. For the internal processing, the padded message must then be divided into blocks. Also, the initial value H_0 is set to a predefined constant. Padding means that we assume have a message x with a length of l bit. To obtain an overall message size and the binary 64-bit representation of l. in Figure 2.11 illustrates the padding of a message x and consequently, the number of required zeros k is given by:

$$\begin{aligned} k &\equiv 512-64-1-l \\ &= 448-(l+1) \bmod 512. \end{aligned}$$

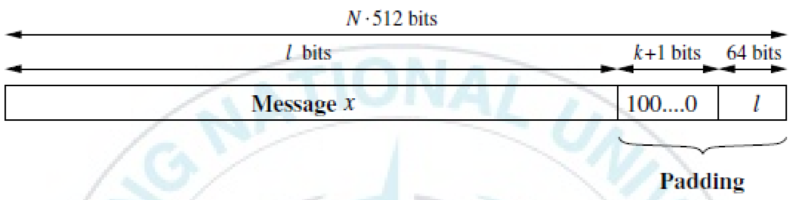


Figure 2.11 Padding of a message in SHA-1

2.6.2 Hash Computation

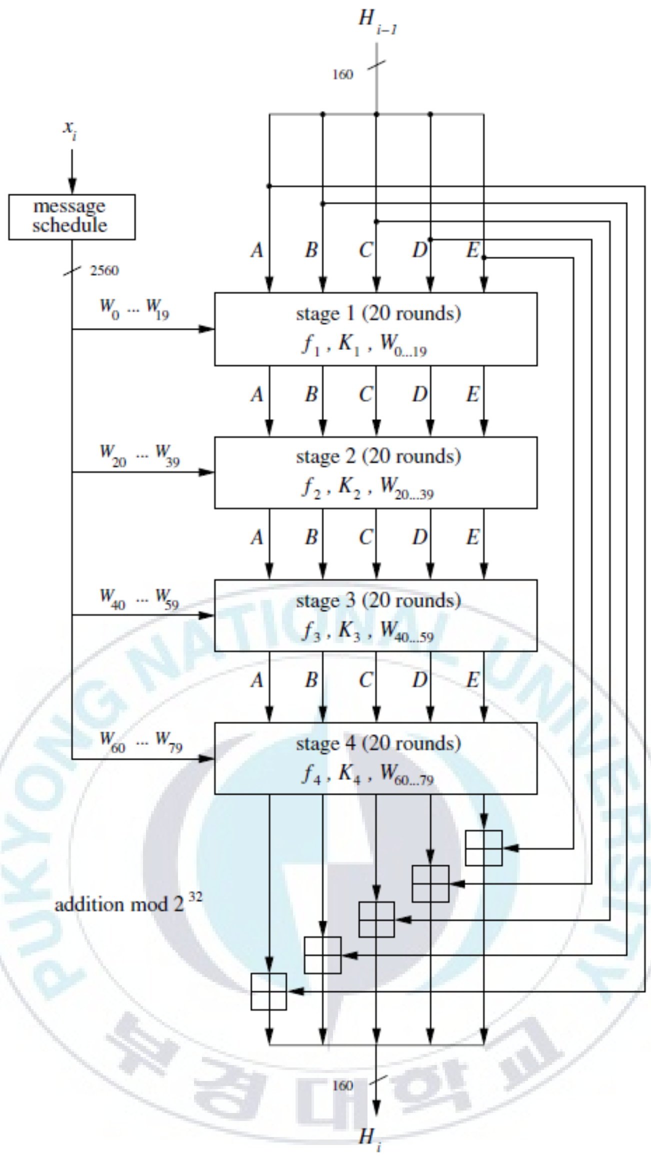
Each message block x_i is proposed in four stages with 20 rounds each as shown in Figure 2.12. The algorithm uses:

- A message schedule which computes a 32-bit word W_0... W_{79} for each of the 80 rounds. The words Wj are derived from the 512-bit message block as follows:

$$W_j = \begin{cases} x_i^{(j)} & 0 \leq j \leq 15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3}) \lll 1 & 16 \leq j \leq 79, \end{cases}$$

- Five working register of size of 32 bits A, B, C, D, E
- A hash value H_i consisting of five 32-bit words $H_i^{(0)}, H_i^{(1)}, H_i^{(2)}, H_i^{(3)}, H_i^{(4)}$. In the beginning, the hash value holds the initial value H_0, which is replaced by a new hash value after the processing of each single message block.

Hash functions are related to (and often confused with) checksums, check digits, fingerprints, randomization functions, error-correcting codes, and ciphers. Although these concepts overlap to some extent, each has its own uses and requirements and is designed and optimized differently.



H_{i-1}
160
x_i
message schedule
2560
$W_0 \cdots W_{19}$
$W_{20} \cdots W_{39}$
$W_{40} \cdots W_{59}$
$W_{60} \cdots W_{79}$
A B C D E
stage 1 (20 rounds) $f_1, K_1, W_{0...19}$
stage 2 (20 rounds) $f_2, K_2, W_{20...39}$
stage 3 (20 rounds) $f_3, K_3, W_{40...59}$
stage 4 (20 rounds) $f_4, K_4, W_{60...79}$
addition mod 2^{32}
160
H_i

Figure 2.12 Eighty-round compression function of Hash Function

The four SHA-1 stages have a similar structure but use different internal functions *ft* and constants *Kt* , where $1 \le t \le 4$. Each stage is composed of 20 rounds, where parts of the message block are processed by the function *ft* together with some stage-dependent constant *Kt* . The output after 80 rounds is added to the input value *Hi*−1 modulo 232 in word-wise fashion. Typically, the domain of a hash function (the set of possible keys) is larger than its range (the number of different table indexes), and so it will map several different keys to the same index. Therefore, each slot of a hash table is associated with (implicitly or explicitly) a set of records, rather than a single record. For this reason, each slot of a hash table is often called a bucket, and hash values are also called bucket indices.

2.7 Cloud Computing and Storage

Cloud computing has recently emerged as one of the buzzwords in the ICT industry. Numerous IT vendors are promising to offer computation, storage and application hosting services and to provide coverage in several continents, offering service-level agreements (SLA)-backed performance and uptime promises for their services. While these "clouds" are the natural evolution of traditional data centers, they are distinguished by exposing resources (computation, data/storage and applications) as standards-based web services and following "utility" pricing model where customers are charged based on their utilization of computational resources, storage and transfer of data [16].

2.7.1 The SPI Framework for Cloud Computing

A commonly agreed upon framework for describing cloud computing services goes by the acronym "SPI." This acronym stands for the three major services provided through the cloud: software-as-a-service (*S*aaS), platform-as-a-service (*P*aaS), and infrastructure-as-a-service (*I*aaS).

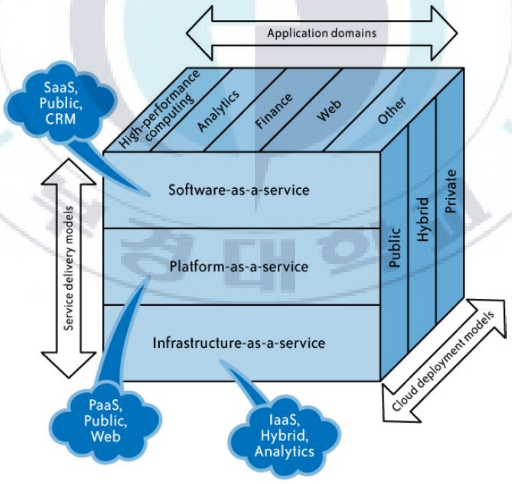
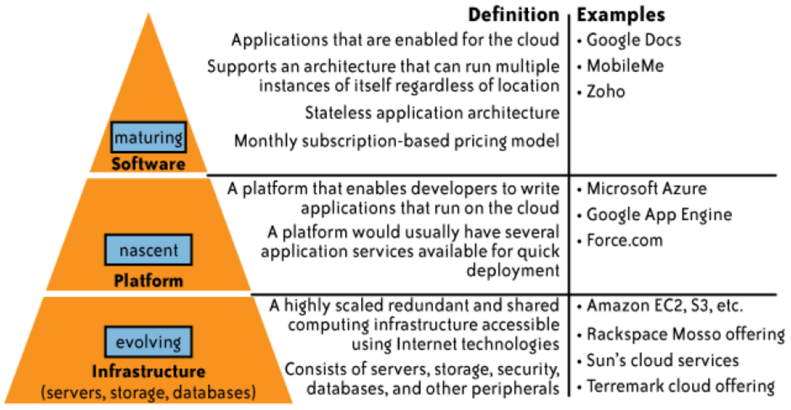


Figure 2.13 SPI service model [17]

2.7.2 The Cloud Service Delivery Model

A cloud service delivery model is commonly referred to as an SPI and falls into three generally accepted services as shown in Figure 2.14. The cloud as a utility can be defined as "on demand delivery of infrastructure, applications, and business process in a security-rich, shared, scalable and based computer environment over the internet for a fee" [16].



Layer	Definition	Examples
maturing Software	Applications that are enabled for the cloud Supports an architecture that can run multiple instances of itself regardless of location Stateless application architecture Monthly subscription-based pricing model	• Google Docs • MobileMe • Zoho
nascent Platform	A platform that enables developers to write applications that run on the cloud A platform would usually have several application services available for quick deployment	• Microsoft Azure • Google App Engine • Force.com
evolving Infrastructure (servers, storage, databases)	A highly scaled redundant and shared computing infrastructure accessible using Internet technologies Consists of servers, storage, security, databases, and other peripherals	• Amazon EC2, S3, etc. • Rackspace Mosso offering • Sun's cloud services • Terremark cloud offering

While cloud-based software services are maturing, cloud platform and infrastructure offerings are still in their early stages

Figure 2.14 Cloud services delivery model [17]

Definition of cloud computing is based on five attributes: shared resources, massive scalability, elasticity, pay as you go, and self-provisioning of resources [17].

- *Shared resources*: unlike previous models, which assumed dedicated resources (i.e., computing facilities dedicated to a single user or owner), cloud computing is based on a business model in which resources are shared (i.e., multiple users use the same resource).
- *Massive scalability*: Although organizations might have hundreds or thousands of systems, cloud computing provides the ability to scale to tens of thousands of systems, as well as the ability to massively scale bandwidth and storage.
- *Elasticity:* User can rapidly increase and decrease their computing resources as needed, as well as release resources for other user when they are no longer required.
- *Pay as you go:* User pay for only the resources they actually use and for only the time they require them.
- *Self-provisioning of resources:* Users self-provisions resources, such as additional systems (processing capability, software, and storage).

Cloud storage is a model of data storage where the digital data is stored in logical pools, the physical storage spans multiple servers (and often locations), and the physical environment is typically owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible, and the physical environment protected and running. People and organizations buy or lease storage capacity from the providers to store user, organization, or application data.

2.8 Container Dwelling Time

Seaport is part of the development of a region or country as a main gateway for exports and imports between countries and islands, therefore the port system should be structured properly including management of container dwell time. Prediction of container dwell time is required because it deals directly with consumer satisfaction, the longer a container remains in the port, the more expensive costs are charged to the costumers. CDT values are obtained from three processes, namely stevedoring, cargodoring and receiving [6]. Stevedoring process is characterized by the cargo container ship docked at the pier, then the container is removed using quay crane one by one to be placed on the truck and after that the truck is taken to warehouse. Corgodoring is a second process to obtain value of CDT and it is a process that requires the longest time duration than the others because the container is checked by officers for purposes such as country of origin, the contents of container, administration etc. Receiving is a port activity when the containers are transported out of the port and payment is done for several administrative to the port operator. Duration for this process is relatively short compared to the other processes [6].

Jakarta International Container Terminal (JICT) is the largest container terminal service in Indonesia (covers 100 hectares) and handles more than 2.2 million TEUs (twenty equivalent units) per year. Data from January 2011 to November 2012 showed that the average dwell time is 5.3 days whereas maximum dwell time for all of container is three days. Predicted results are expected to help officer to take several steps to prevent container from exceeding the maximum time (3 days). The output from prediction is “days” which means how long a container takes when stacked in terminal [18].

The sharing of dwelling time prediction and equipment optimization data in cloud computing provides many advantages for user like time efficiency, ease in communicating, cost effective and many others but the security issues to keep the data from being stolen or changed by the attacker is one of the considerations in the cloud [4]. As cloud computing is being widely adopted, data security is becoming one of the major concerns of data owners. Data integrity is an important factor in almost any data and computation related context. It is not only one of the qualities of service but also an important part of data security and privacy [5].

Chapter 3. System Requirements and Design

3.1 Data Access Architecture in the Cloud

In this approach, we assumed that the seaport has a storage in the cloud. For every dwelling time and equipment optimization data are stored in the cloud. Dwelling time and equipment optimization data can be accessed, edit, download and upload via the cloud by authorized person. The unauthorized person could have come from anywhere like from outsider and insider who may have different motivations to change the data. If data already changed by the attacker, it will make some serious problems in the seaport system. The goal of digital signature concept in the seaport is to ensure the data integrity of dwelling time prediction and equipment optimization result and to make sure nobody change the data, and to avoid the fake dwelling time.

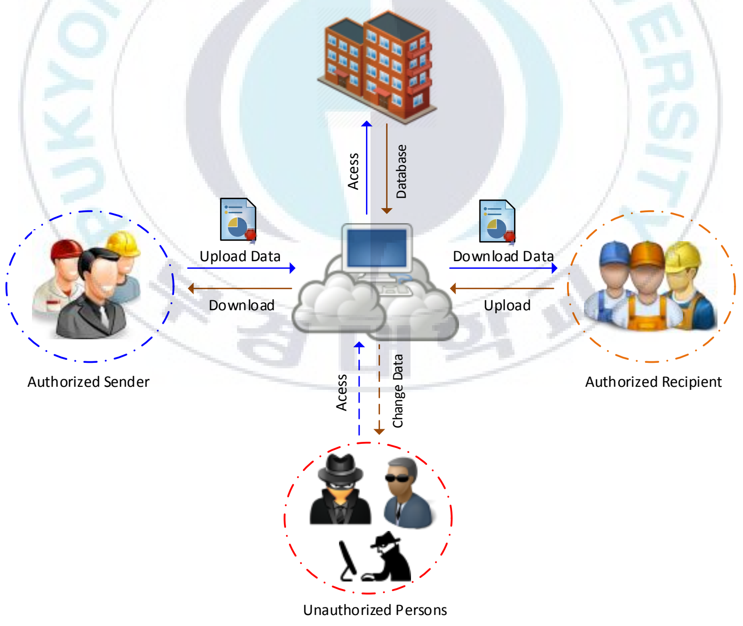


Figure 3.1 Data access architecture in the cloud

In figure 3.1 there are online communication and data exchange in the cloud storage between authorized sender and authorized recipient. In this thesis, authorized sender is capable to sign and upload the dwelling time data in google drive as a cloud storage. Dwelling time prediction provides detailed information about the length of time where the container stayed in the seaport.

3.2 The Impact Falsification of Dwelling Time

There are many motivations from unauthorized person to changes the dwelling time data. For the example the actual prediction time of dwelling time is 6 days, then the attacker changes the dwelling time data in the cloud storage becomes 8 days (longer) or 4 days (shorter). The dwelling time data changes by the unauthorized person gives several impact in the seaport such as the more expensive cost, administrative complexity and time efficiency. The longer dwelling time the more expensive cost will be paid by consumers. Disproportionate number of equipment and containers will affect to the seaport performance. In general, impact of falsification dwelling time is system in the seaport would not be efficient because the data changes by the attacker as shown in table 3.1.

Table 3.1 the impact falsification of dwelling time

8 days (Longer than actual time)	**4 days (Less than actual time)**
a. The Seaport will spend more costs for equipment operation to handle a number of container.	**a.** The process will be longer than it should be because of inadequate equipment.
b. The longer dwelling time the more expensive cost to be paid by consumers.	**b.** Disproportionate number of equipment and containers will affect to seaport performance.
c. There will be a lot of equipment that is not used in the field, etc.	**c.** System in the field would not be efficient, etc.

The goal of digital signature concept in the seaport is to provide data integrity of dwelling time in the seaport, to ensure nobody change the data and to prevent the falsification of dwelling time data. The integrity of data is so important to keep the original data from the authorized sender and recipient. If the attacker change the dwelling time data to longer or shorter than normal time, it causes some harm to the company or the customer. Therefore, the dwelling time data needs to be maintained originality that really comes from an authorized sender without any changes during the processing from outsider and insider who may have different motivations to change the dwelling time data [6].

3.3 The Signed and Verification Process

A digital signature is the term used for marking or signing an electronic document, by a process meant to be analogous to paper signatures, but which makes use of a technology known as public-key cryptography. Additional security properties are required of signatures in the electronic world. This is because the probability of disputes rises dramatically for electronic transactions without face-to-face meetings, and in the presence of potentially undetectable modifications to electronic documents.

3.3.1 Signed Process

One of the main differences between a digital signature and a written signature is that the user does not "see" what he signs. The user application presents a hash code to be signed by the digital signing algorithm using the private key. An attacker who gains control of the user's PC can possibly replace the user application with a foreign substitute, in effect replacing the user's own communications with those of the attacker. This could allow a malicious application to trick a user into signing any document by displaying the user's original on-screen, but presenting the attacker's own documents to the signing application. The general idea is to provide some means for both the user application and signing application to verify each other's integrity. For example, the signing application may require all requests to come from digitally signed binaries.

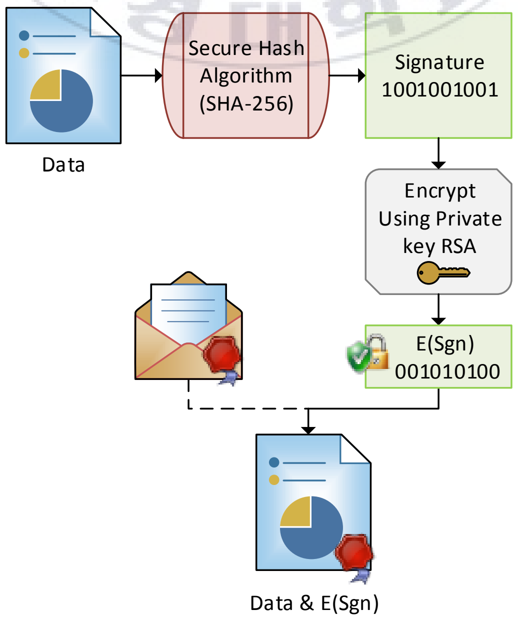

Data
Secure Hash Algorithm (SHA-256)
Signature 1001001001
Encrypt Using Private key RSA
E(Sgn) 001010100
Data & E(Sgn)

Figure 3.2 Signed dwelling time data process

Signing process based on Figure 3.2 is begins by taking the hash value of the data using secure hash algorithm-1 to produce a message digest. The value of message digest will be totally different although the dwelling time data are almost identical for each other. A pair of key (secret and public key) is derive from RSA algorithm and used to encrypt the message digest. A secret key from RSA is used to encrypt the message digest which is every message digest has a fixed length and different for each data. In the signing process a digital signature will be attached in the dwelling time data then the signature and data will be upload to the cloud.

3.3.2 Verification Process

The verification process carried out by the authorized recipient in a way to decrypt the message using the public key from RSA algorithm and then compare with the value of signing that has been attached by the sender. Based on Figure 3.3 if decryption result equal to the value of signing, it means there are no changes in data (dwelling time data still intact), the otherwise if the value is not equal means the data has been changed by the attacker.

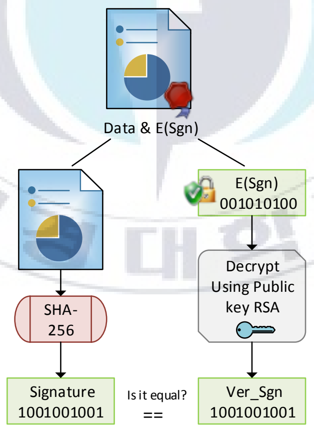


Figure 3.3 Verification for dwelling time data process

Creating and verifying signatures uses the public/private key pair in an operation different from encryption and decryption. A signature is created using the private key of the signer. The signature is verified using the corresponding public key. A common use of digital signatures is to sign use net postings or email messages. In such situations it is undesirable to compress the document while signing it. When digital signatures is invalid then the officer can able to make several step like, contact the sender and let them know that there is a problem with the signature, and inform the system administrator in charge of seaport's security infrastructure. A trustworthy signature is valid, on the user account, on the computer that states it as valid. If the signature were opened on another computer, or another account, the signature may appear as invalid because that account may not trust the certificate issuer. Also, for a signature to be valid, the cryptographic integrity of the signature must be intact. This means that the signed content was not tampered with, and the signing certificate is not expired or revoked.

3.3.3 Concept Digital Signature to Provide Data Integrity in Cloud Storage

A digital signature is created by a series of mathematical processes that transform data (e.g., a Word document, PDF, or text file) into a uniquely coded "message digest." The sender encrypts the message digest then attaches it to or embeds it in a file, and sends the package to the intended recipient. Once the package is received and the message digest is decrypted, a determination of integrity can be made.

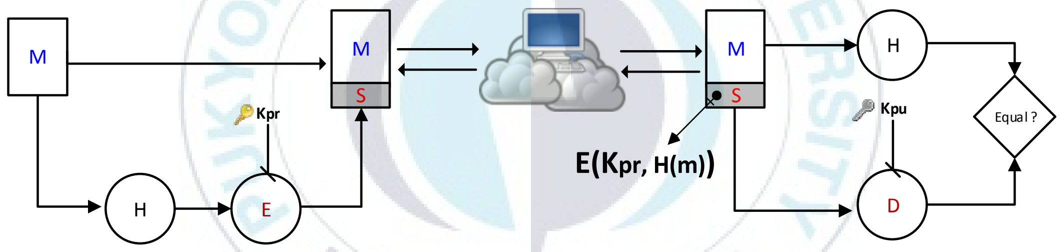


Figure 3.4 Digital signature in cloud

$$S = E\ (K_{pr}, H(M)) \qquad (3.1)$$

Then for verification,

$$h = D\ (K_{pub}, S) \qquad (3.2)$$

where:

S	:	Signature
E	:	Encryption
D	:	Decryption
H	:	Hash value
h	:	Message digest
K_{pr}	:	Secret key the sender
K_{pub}	:	Public key the sender

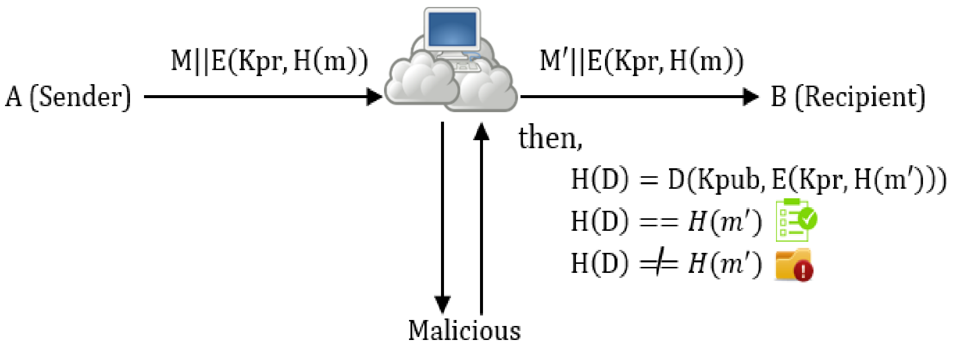


Figure 3.5 Comparing the result of hash value

Based on Figure 3.4 and Figure 3.5 for the signing process, user signed a dwelling time message by encrypting value of message digest with own private key and after that the signature will be attached in the dwelling time data. It means there are 2 types data that will be upload to the cloud storage, dwelling time data and signature. If the digital signature matches the identity of the sender one can be reasonably assured that it was sent by the individual associated with the digital signature (at worst the actual sender had access to the digital identity of the supposed sender). If the message digest received matches the message digest as calculated by the recipient the document has not been altered after it was digitally signed.

3.4 Authorized and Unauthorized Person

3.4.1 The Authorized Person

The people who are categorized as authorized officers in the seaport is the people who assigned to control and supervise the performance of the seaport. The authorized officers communicate with each other via phone or online to give information on the amount of equipment that is ready for operation. Then the other officer will be made an analysis for dwelling time prediction is based on the amount of equipment available in the field. In general there are three groups of officer, namely Officer 1, Officer 2 and Field Officer. Each officer communicates via different way, for example using Android apps for communication between field officer and officer 1. After the officer obtain the result of dwelling time prediction, then the data should be signed before uploaded to the cloud and sent to an authorized recipient. Structure of the officer in this system more details can be seen in Figure 3.6.

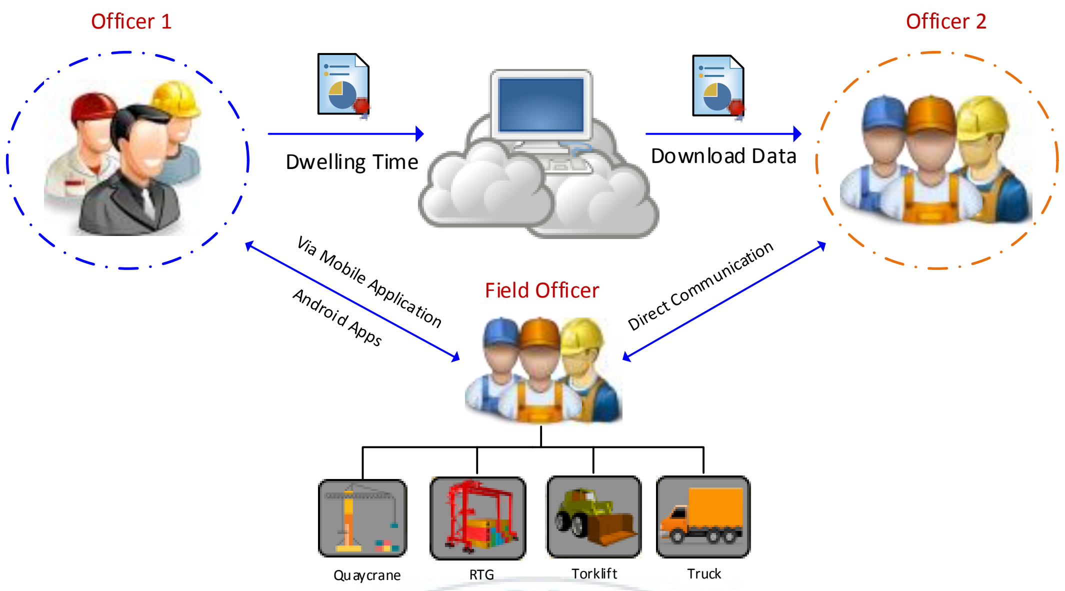


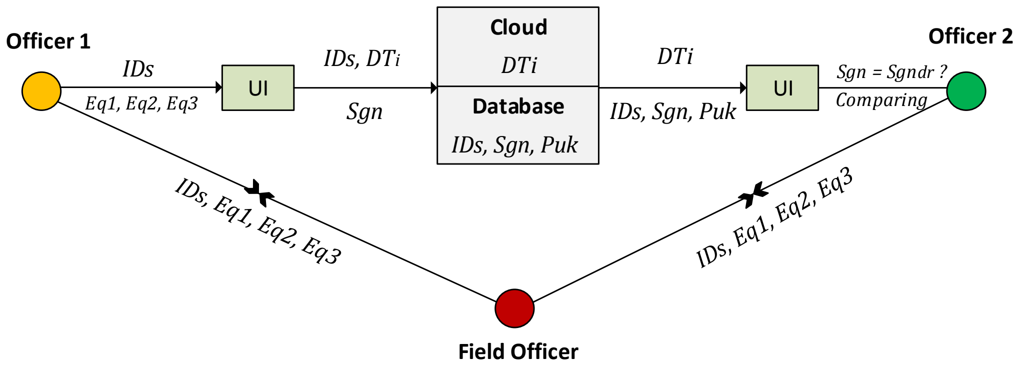
Figure 3.6 Structure of the officer in the system

Based on Figure 3.6 there are three group of officer that assumed in the system, namely field officer, officer 1, officer 2. Each group consisting of two or more people and have a different task, the following details:

a. **Field officer:** This group is a field officer in charge to control and report the amount of equipment which are available at the time. Officer 1 can directly communication via android application.
b. **Officer 1:** This officer works in the office and work with using a computer to predict the dwelling time is based on the amount of equipment that has been reported by the Officer 1.
c. **Officer 2:** Tasked to analyze and provide precautions dwelling time is based on reports from Officer 1. Officer 2 communicates directly with the Officer 1 instance ordered additional equipment so that dwelling time can be avoided.

3.4.2 Symbol and Notation between Authorized Person

There are several communication and data exchange between authorized officers in the seaport which are related to dwelling time prediction. Based on Figure 3.7 has been described in detail every process that will be happened between the officers. In the process also using database to keep some of data like information about public key, ship ID, the value of message digest etc.



Picture 3.7 Communication between Authorized Persons

Where,

IDs	:	Ship ID
Eq1	:	∑ Equipment in stevedoring
Eq2	:	∑ Equipment in corgodoring
Eq3	:	∑ Equipment in receiving
DTi	:	Dwelling time prediction data
DTdr	:	Decrypted the signature
Sgn	:	The signatures
Puk	:	Public key
UI	:	User interface application

After the field officer report a number of equipment which available in the field then officer 1 will predict dwelling time using user interface application based on report by officer field officer. Officer 1 will input some data like *ID*, number of crane, quaycrane, truck, top loader and etc. Based on Figure 3.7, after using *UI* application there are some output like *IDs*, *DTi*, *MD*, and *Puk* then the output for *IDs* and *DTi* will be send to the cloud then for message digest and public key will send to database. Dwelling time prediction data can be accessed by authorized persons and can be downloaded directly, while the message digest of data will be stored in a database and used to complete the verification process that compares the data encrypted by the data contained in the database. User interface application in this research is a computer based desktop application that are various menus and options such as dwelling time prediction and digital signature applications. Dwelling time prediction is the other research conducted by the authors aimed to determine the length of the existence of containers at the seaport and the case study in JICT, Tanjung Priok, Indonesia.

3.4.3 The Unauthorized Person

The unauthorized person is also called as the attacker is an unwanted party within the system that can lead to problems such as changing the dwelling time data so that the system will be chaotic and causes many losses to seaport and consumers. The attacker in this system are categorized into two groups, insider and outsider with different motivation. The following details:

a. **Insider attacker:** This attacker comes from internal officers in the seaport that is not part of the authorized officer (officer 1, 2 and 3). In the seaports there are many kind of officers with different tasks, for the all officer except officer 1, 2 and 3 who are trying to change the dwelling time data are categorized as the attacker. An insider attack is one of the biggest threats faced by modern enterprises, where even a good working culture might not be sufficient to prevent it. Companies implement sophisticated technology to monitor their employees but it’s not always easy for them to distinguish between an insider and an outside attack.

b. **Outsider attacker:** The outsider attacker can be anyone other than the seaport officer who want to disrupt the seaport system with some different motivation like revenge, resentful and etc, but the all motivations will give serious impact for the seaport. In this system should understand that attackers are people too, who differ in resources, motivation, ability and risk propensity. There are a number of insider attackers who are merely pawns for another inside or outside mastermind. He or she is usually persuaded or trained to perpetrate or facilitate the attack, alone or in collusion with other agents, motivated by the expectation of personal gain.

3.5 Overall Flowchart of the Proposed System

The data exchange process is initiated by the sender who uploads the data into the user interface in a program that will be built. The data will be signed and upload to database, then the authorized recipient will be downloaded and verify the dwelling time data by comparing message digest encrypted and message digest in database. The overview of the system sequentially shown in Figure 3.8.

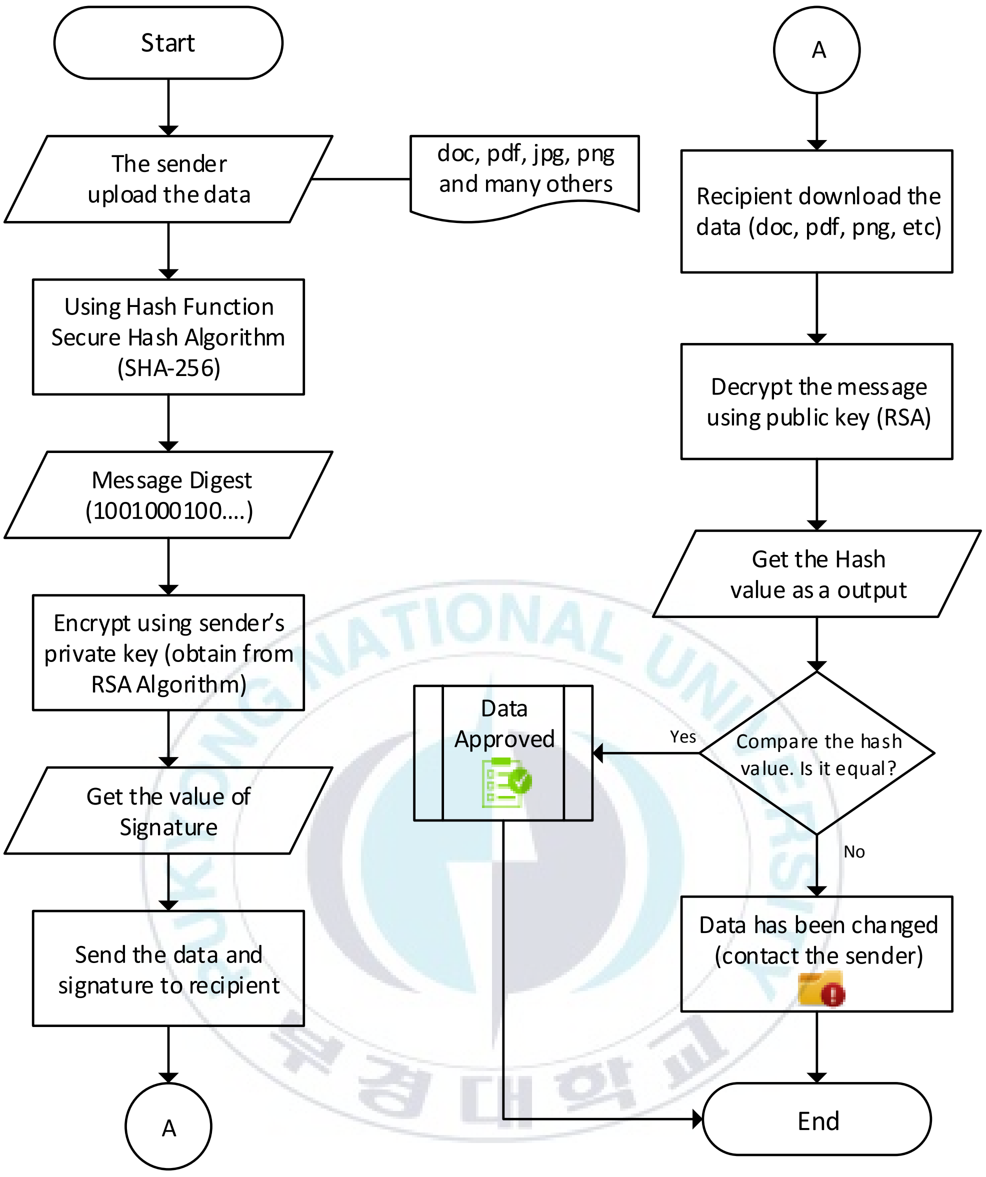

Start
The sender upload the data
doc, pdf, jpg, png and many others
Using Hash Function Secure Hash Algorithm (SHA-256)
Message Digest (1001000100....)
Encrypt using sender's private key (obtain from RSA Algorithm)
Get the value of Signature
Send the data and signature to recipient
A
A
Recipient download the data (doc, pdf, png, etc)
Decrypt the message using public key (RSA)
Get the Hash value as a output
Compare the hash value. Is it equal?
Yes
No
Data Approved
Data has been changed (contact the sender)
End

Figure 3.8 Flowchart of the Overall System

To verify the data, the authorized recipient must download the data in the cloud, then using a RSA public key that data will be decrypted and the recipient will obtain a hash value as an output and the value will be compared with the signed value. Dwelling time data received by the recipient is original if the decryption result is equal to the signed value. The original message digest can be obtained from database. If the authorized recipient finds a match message digest value between decrypted and value from database then it can be said the data has not been changed by anyone it means the data still original from the sender. The otherwise is the value is not match, it means the data has been changed by attacker.

3.6 Android Application for Dwelling Time Prediction

Communication between field officers and officer 1 is used an android application that provides information of dwelling time which is based on the amount of equipment in the seaports at one time. Android is mobile operating system (OS) based on the Linux kernel and currently developed by Google. Android is popular with technology companies which require a ready-made, low-cost and customizable operating system for high-tech devices. Android's open nature has encouraged a large community of developers and enthusiast to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices which were officially released running other operating system. Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software.

The Field officer using the dwelling time application (Android) as an input calculate or estimate amount of equipment available at the port such as the number of trucks, rubber tyred gantry, torklift and toploader for each process like stevedoring, corgodoring and receiving which is the operational standards in every seaport in the world. In the application there are several important information like *ship ID*, number of crane, truck, top loader which are should be secret information, it means only authorized person knows that information. Via the applications, field officer also can send the data into database that means every authorized user can accesses all of the related information in database. Some of data is important to keep in database, it will help the officer to avoid manually to input the data and it reduces the mistakes made by the attacker.

The predicted dwelling time data can be accessed by authorized persons and can be downloaded directly, while the message digest of data will be stored in a database and used to complete the verification process that compares the data encrypted by the data contained in the database. All of that information will be stored and keep in the cloud storage by field officer and can be loaded anytime by the authorized officer. The information will be used by officer 1 from the cloud storage and used as basic information for signing and verifying data and that data also will be uploaded to the cloud which is same storage with android apps.

Chapter 4. System Implementation and Analysis

4.1 Overall System and Application

Concept of digital signature for the data integrity of the dwelling time data is implemented into a java-based application that will be used by the authorized officer to process, edit, create, upload and download the dwelling time data. Authorized officer uses the same application, which includes a process of digital signature. In the application there are four menus with functions and information, namely ‘About”, “Dwelling Time”, “Digital Signature” and “Exit” as shown in Figure 4.1, but the main menu in this thesis is “Digital Signature” menu.

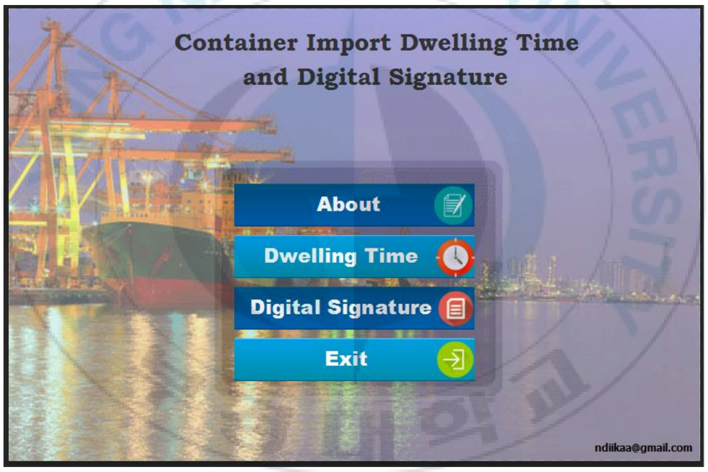
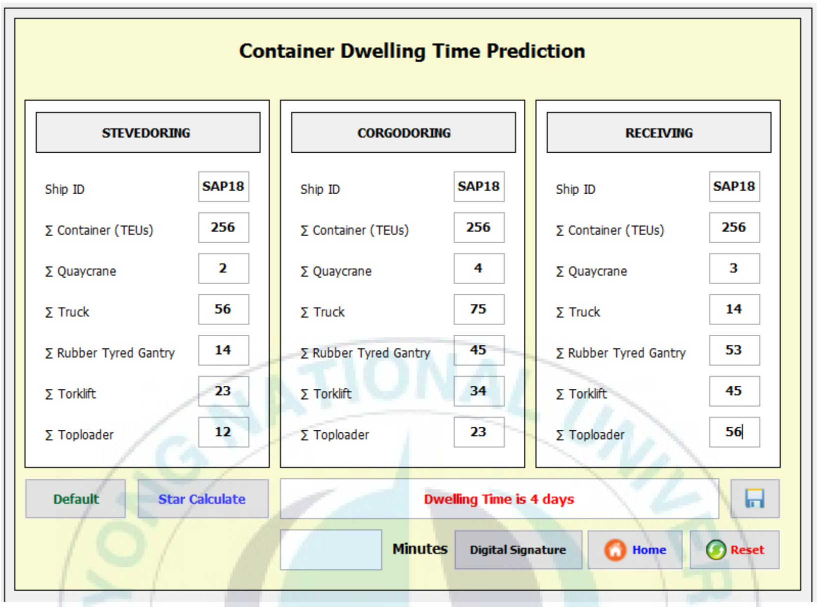


Figure 4.1 Main menu of digital signature application

The following is an explanation for each menu as shown in Figure 4.1. “About” describes a general overview of the application where the dwelling time and digital signature are interconnected in the seaport system. This menu also contains the information that is considered as important ones.

4.2 Container Import Dwelling Time Prediction

Container import dwelling time application is not the main focus in this study because the dwelling time prediction is done from other work [6], it means there is no detailed explanation about how to get the predicted results based on the amount of equipment available in the seaport. This menu application is used by the authorized officer (officer 1) based on reports from the field officer and the appearance of the menu shown in Figure 4.2. There are some different amount of equipment for each process such as stevedoring, corgodoring and receiving and also there is a *Ship ID* which is the identity of a ship carrying a number of containers.



Container Dwelling Time Prediction

	STEVEDORING	CORGODORING	RECEIVING
Ship ID	SAP18	SAP18	SAP18
Σ Container (TEUs)	256	256	256
Σ Quaycrane	2	4	3
Σ Truck	56	75	14
Σ Rubber Tyred Gantry	14	45	53
Σ Torklift	23	34	45
Σ Toploader	12	23	56

Dwelling Time is 4 days

Figure 4.2 Menu of container dwelling time prediction

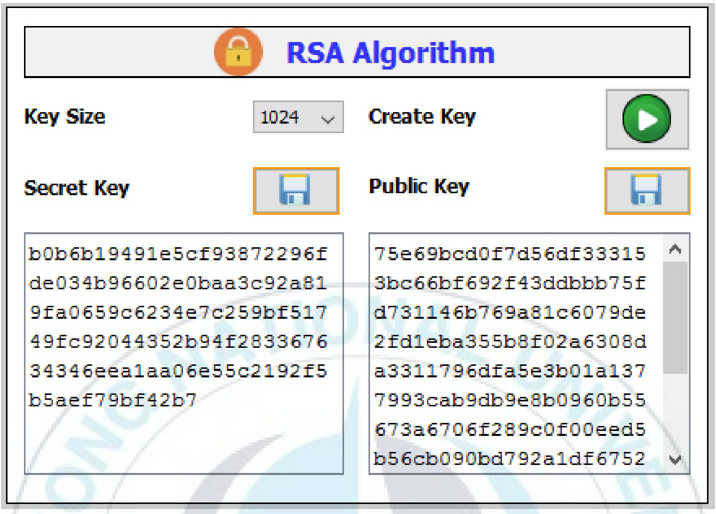
In this research, the data output of container dwelling time prediction is subsequently used for signing using digital signature concept and then uploaded to the cloud by officer 1.

4.3 Digital Signature Implementation

Digital signature application is the main focus in this research. Data of dwelling time prediction is used by the authorized officer, and it is uploaded and downloaded in the cloud but the data should be signed to keep the originality of the data. In the digital signature process. We need a pair of key, public key and secret key which is derived from the RSA algorithm.

4.3.1 User Interface of RSA Key

RSA algorithm is used to derive a pair of key, secret key and public key. Secret key is used by officer 1 to encrypt a message digest of dwelling time data. The encrypted file is not the value of original message but the value of message digest of dwelling time data. A value of message digest is derived from using hash value which has a fixed length for every message. RSA algorithm in this thesis uses three kinds of key-length, 1024 bits, 2048 bits and 3072 bits. Based on Figure 4.3 there are two windows to provide the key, the first one is for a secret key and the other one is for a public key. Secret key is used by officer 1 to encrypt the message digest of dwelling time data.



Key Size: 1024

Secret Key:

```
b0b6b19491e5cf93872296f
de034b96602e0baa3c92a81
9fa0659c6234e7c259bf517
49fc92044352b94f2833676
34346eea1aa06e55c2192f5
b5aef79bf42b7
```

Public Key:

```
75e69bcd0f7d56df33315
3bc66bf692f43ddbbb75f
d731146b769a81c6079de
2fd1eba355b8f02a6308d
a3311796dfa5e3b01a137
7993cab9db9e8b0960b55
673a6706f289c0f00eed5
b56cb090bd792a1df6752
```

Figure 4.3 User interface of RSA algorithm

The key size used in particular application of cryptography depends on the necessary security level of the applications and the actual key size also depends on what cryptographic algorithm is being used. In most cryptographic functions, the key length is an important security parameter. The used RSA key (public and secret) should be random, it means the key will not be the same as the key used previously or afterwards. The following is a key length of 1024 bits (random key) in hexadecimal notation.

Table 4.1 Different Key Size

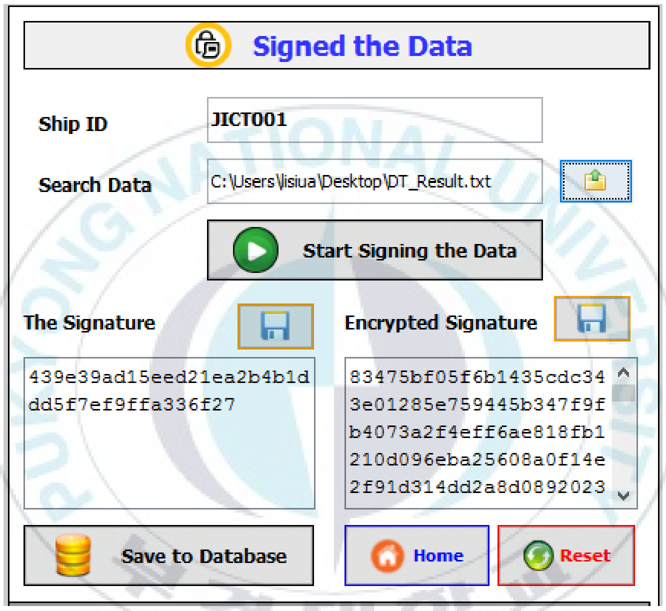
Key Size	Secret Key	Public Key
1024 bits	b0b6b19491e5cf93872296fde 034b96602e0baa3c92a819fa0 659c6234e7c259bf51749fc92 044352b94f283367634346eea 1aa06e55c2192f5b5aef79bf4 2b7	75e69bcd0f7d56df333153bc66bf 692f43ddbbb75fd731146b769a81 c6079de2fd1eba355b8f02a6308d a3311796dfa5e3b01a1377993cab 9db9e8b0960b55673a6706f289c0 f00eed5b56cb090bd792a1df6752 70a8e7bbacbd968c1436282a589b 42138f56b7eedae878c8cd6b2a65 be1d7db8b2ff7e61ade77f23b924 e6f7

Table 4. 1 (Continued)

Key Size	Secret Key	Public Key
2048 bits	d4dc0eaa36a0cc355eaa2229e 9a28a789cd7a4995b560dba8e c072bc48d02013ccc8574f137 22303f230001bbfc2ebcb256f 0388270e4a46f42485f45ec9d ed45f7e9bdb3665955f933f86 932683594c5cf9b46a30660d0 0860d5ee171fbad165c0265e3 1505c8d5f8c1b40cc9cc58348 db314dd4ea9e7ac7e08efc19f a695ef	106904fd2e2de3b00ca227be3e2e4e 2aceb919464db23d404bd069f2e464 7fc49c3477254320e31d7c11cfb7f7 3369006186f625bf52068b9369518 ef7fd0056d125831c500117d5f17dc 86fb16538d08f54d4b5eff76882b12 d6e1bc37687249346c5493b47e9966 11629b3a2b23d5831a30f4b02db341 3160961a6fc58658683674 6fefce93 287f637bb783a86c5c39ad95fd58df a0972a024d7a04ef88ac27a366eb36 04a63a003d80c5716c8d511b867c79 4655208c3231d4d7e65b0fb7da9e11 5b17057f3c9cfd1e2c8afff85806b6 ce47a19d0f9e57cc4c6e28cbd16c26 f2b6a3370526eeae4149b898390de6 6f368356b4069183fb7dc187193edb 4b
3072 bits	eb4cdace055ccaa621c5bb897 6a83a7f838233d470779d346a 076f8086f30044e740f46fb2a 92bdb6065e3499206dffe0252 fcac535f42799473cc1fbff59 815012f628e558285c654eee0 2889ba08cd13611c280334769 458f8bc2da4485c4b6f488a0e 60921371eb4f51b0c2ea4f229 37430493b4e5dbbac123ff21c c6ee03e252a3d8889145fb8b2 a552c91841b7c8cc3205d5a96 39bcac6d3d6f143c97940f576 4e707a6642777f6e95200f9a9 8a2e7eb75615e0173fd8be626 6b665f5af	1ea373b046f8c5fdb8479d7f2afc1f 3ec16a64e97806621df438410cad7b e41d9ba6aa934912a61286e2fbf8c4 2033f202373c122337410d1d180f55 5b7d8858e79abdd8cb725dd10e8179 dac3ede20ea6362ad625eb79af0b59 2adc1a74c895a1c945f1edfab617b1 8e0e6c29b0c99be66017194c7d21b3 f8591636e6842f3875f4a1ace46cb7 35b253c35ce60f4328343c50fbccfb 6f683c5c88b1a86ca0ec63e6498a16 1d687f72406e3b42cb5015484dcc8d 83e3d859de9dd1f64b172989e7a46f 845ad08e2a38a72a4be9bd1c7cb83b a2d0ff8dd9d5d60938cdb5a55418b2 0cc652affed68ec44ccab3be98320d 279caf60cab3c29ca7a4e85b167266 287de6ac57493cd34c84598fcc2790 59a152cb7cedafbea1f6219ffa55d3 281451af96f69ab3a9977775bebcae 4c0ed16e56c5abb5cffcb1a5aad002 d614546391408c52e7c7039c86dddc e1de565508d905103b068f301f8fec 8d8b460a363cb83e432b45c9c5fc92 32e440b9ff8c2fd29e4737b0bc79e7 39c987bb4e247a3dbf

4.3.2 Signed the Dwelling Time Data

Dwelling time prediction process provides an output in the form of long existence of containers in the seaport as shown in the Figure 4.2 and the format for this data is "*txt" (notepad). Signing and verifying process is always related to RSA keys because the process uses the public and the secret key to encrypt and decrypt the dwelling time data. In the Signed the data form, there is a "Ship ID" which gives the information of a ship carrying a number of containers. Data of Ship ID will be stored in the cloud that will provide information on the number of containers and can be used anytime.



Ship ID: JICT001
Search Data: C:\Users\lisiua\Desktop\DT_Result.txt

The Signature:
```
439e39ad15eed21ea2b4b1d
dd5f7ef9ffa336f27
```

Encrypted Signature:
```
83475bf05f6b1435cdc34
3e01285e759445b347f9f
b4073a2f4eff6ae818fb1
210d096eba25608a0f14e
2f91d314dd2a8d0892023
```

Figure 4.4 User interface of signed process

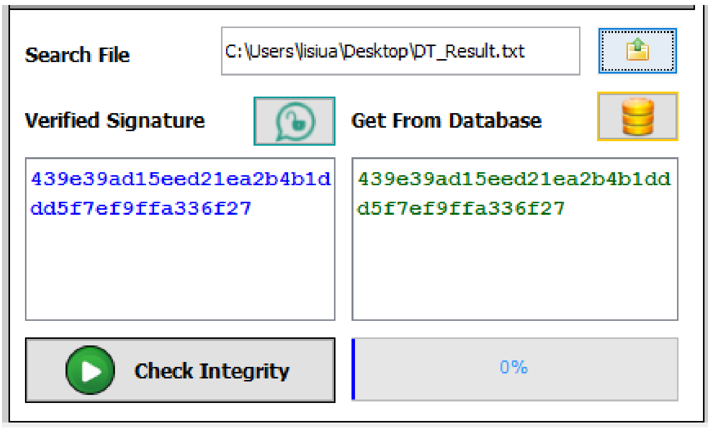
From the Figure 4.4, there are two windows namely "Message Digest" and "The Signatures". Message digest is a cryptographic hash function containing a string of digits created by a one-way hashing formula. Message digest is designed to protect the integrity of data to detect changes and alterations to any part of messages. Message digest values represent specific files containing the protected work. One message digest is assigned to particular data content. If there are any changes deliberately or accidentally, it prompts the owner to identify the modifications by checking the hash value of the dwelling time data.

Message digest is encrypted with private/secret keys to create a digital signature. This result is a type of validation assurance that the appropriate user is accesses to the protected information. Message digest protects one-way hash algorithms taking random data and transmitting a set length hash value. To begin the process a message digest is initialized. Then the data is processed through the message digest by using updates. Final operations include padding, during which the message digest completes the hash computation and resets itself. However, the digest can be reset at any time during the process.

Message digest in this research represent the value of dwelling time data after using the hash algorithm whereas the signatures is encrypted value of message digest using secret key officer 1. Message digest and the signature is not same even comes from the same data source (dwelling time data). After the officer 1 input some data like Ship ID and get message digest value and the signatures value, then the officer will be able to save that data in database. All of information about ship id, amount of container, value of message digest can be stored in database so the officer can use to verifying process by access in database. Value of message digest and signatures can be save in the local hard drive by click the icon button as shown in Figure 4.4.

4.3.3 Verifying the Dwelling Time Data

The verification process carried out by the authorized recipient in a way to decrypt the message using the public key from RSA algorithm and then compare with the value of signing that has been attached by the sender, if decryption result equal to the value of signing, it means there are no changes in data (dwelling time data still intact), the otherwise if the value is not equal means the data has been changed by the attacker. Based on Figure 4.5 there are two kinds of value, "Decrypt Signature" and "Get from Database". Those value should be same to indicate the dwelling time data still original it means there is no changes in the data.



C:\Users\lisiua\Desktop\DT_Result.txt

```
439e39ad15eed21ea2b4b1d
dd5f7ef9ffa336f27
```

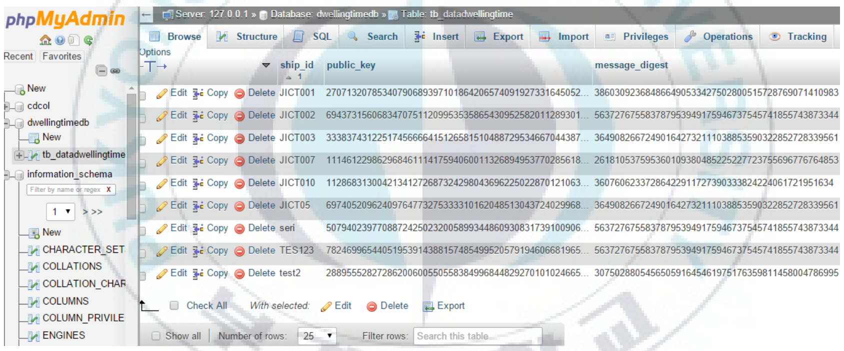
```
439e39ad15eed21ea2b4b1dd
d5f7ef9ffa336f27
```

Figure 4.5 Verifying dwelling time data

Decrypt signature value obtained from the decryption by using the public key by authorized officer (officer 2). Once the data is downloaded by the officer 2 then decrypt the data by using public key to obtain the hash value of the data decrypt result. After get the decrypt signature value then the officer comparing with the hash value from database which is the original hash value from the sender (officer 1). If the value is same then there is no change in the data, otherwise if its value is not same then the data has been modified by the attacker.

4.4 System Database

Database in this research containing some information like ship id, public key, message digest, date created and other as shown by Figure 4.6. MySQL is the most popular open source SQL database management system and applied in this research.



Server: 127.0.0.1 » Database: dwellingtimedb » Table: tb_datadwellingtime

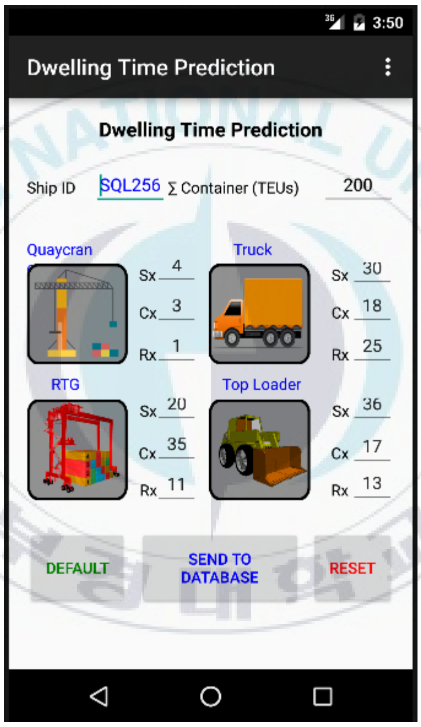
ship_id	public_key	message_digest
JICT001	27071320785340790689397101864206574091927331645052...	386030923684866490533427502800515728769071410983
JICT002	69437315606834707511209953535865430952582011289301...	563727675583787953949175946737545741855743873344
JICT003	33383743122517456666415126581510488729534667044387...	364908266724901642732111038853590322852728339561
JICT007	11146122986296846111417594060011326894953770285618...	261810537595360109380485225227723755696776764853
JICT010	11286831300421341272687324298043696225022870121063...	36076062337286422911727390333824224061721951634
JICT05	69740520962409764773275333310162048513043724029968...	364908266724901642732111038853590322852728339561
seri	50794023977088724250232005899344860930831739100906...	563727675583787953949175946737545741855743873344
TES123	78246996544051953914388157485499520579194606681965...	563727675583787953949175946737545741855743873344
test2	28895552827286200600550558384996844829270101024665...	307502880545650591645461975176359811458004786995

Figure 4.6 System database using MySQL

Based on Figure 4.6 there are some tables in database like ship id, public key, message digest and other. Ship id gives information about amount of container transported by ship. Every ship has different id and also different number of container. Public key also important stored to database because with this public key officer 2 can decrypt the message digest of dwelling time data. Message digest in database is the original value of message digest from dwelling time data, this value will be compare with decrypt value by officer 2. If the value of message digest from database is same with value of message digest from decrypt, it means the data still original (there is no changes in the data). A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows and columns, offers a flexible programming environment.

4.5 Android Apps Amount of Equipment

Communication between field officer and officer 1 is using Android apps as shown in Figure 4.7. This application is using by field officer to give information how many equipment in the seaport that is ready to use for operation. Field officer always check manually amount of equipment in the seaport and report to officer 1 and send the result to database by using this application.



3:50

Dwelling Time Prediction

Dwelling Time Prediction

Ship ID SQL256 ∑ Container (TEUs) 200

Quaycran		Truck	
Sx	4	Sx	30
Cx	3	Cx	18
Rx	1	Rx	25
RTG		**Top Loader**	
Sx	20	Sx	36
Cx	35	Cx	17
Rx	11	Rx	13

Figure 4.7 Android apps amount of Equipment

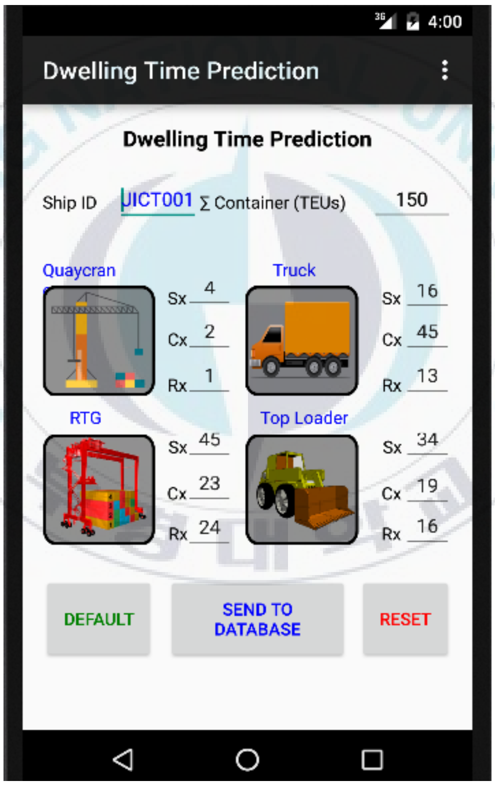
4.6 Experimental Result

In this session briefly explained the whole of process and result with changes and original data to easily understand the different message digest value if data changed by the attacker. In this experiment using the default data like amount of equipment to predict the dwelling time data. For the result of dwelling time data also using a default data because the main point in this research is to detect the data has been changed or still original from the authorized sender.

For example case, the information is following:

- *Ship ID* : JICT001
- *Amount of Container* : 150 TEUs

1. The first step is field officer reports to officer 1 about the amount of equipment in the field by using android apps as shown in Figure 4.8. Field officer input one by one equipment for each process in the seaport (stevedoring, corgodoring and receiving). The amount of equipment in the field is not always same at the time, it means the amount of equipment always changes caused some factors. Then, the information will be send to database by field officer.



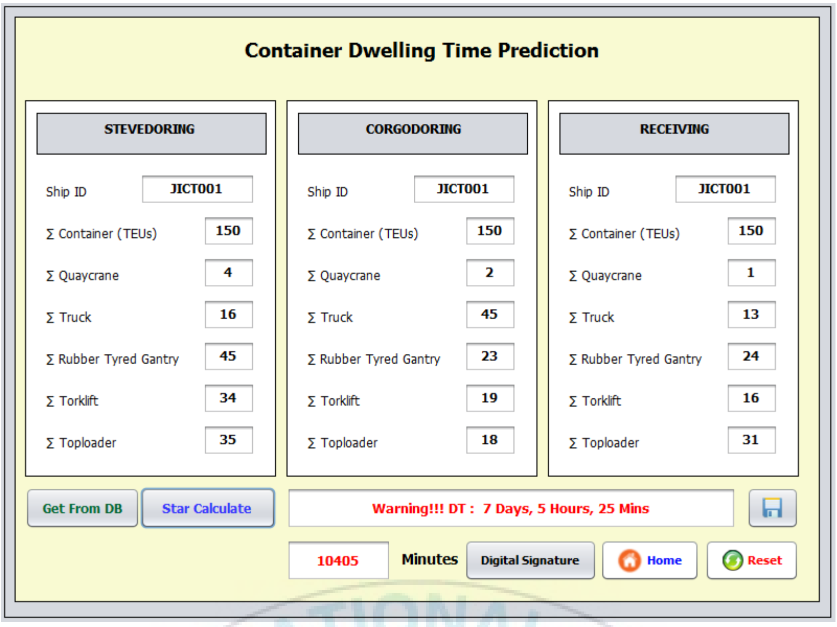
Dwelling Time Prediction

Ship ID JICT001 Σ Container (TEUs) 150

Quaycran	Sx	4	Truck	Sx	16
	Cx	2		Cx	45
	Rx	1		Rx	13
RTG	Sx	45	Top Loader	Sx	34
	Cx	23		Cx	19
	Rx	24		Rx	16

Figure 4.8 Amount of equipment

2. Based on report result by field officer, then officer 1 predicts the dwelling time data by simply input "the ship id" in the applications then officer 1 gets all of kind amount of equipment information from database as shown in Figure 4.9.



Container Dwelling Time Prediction

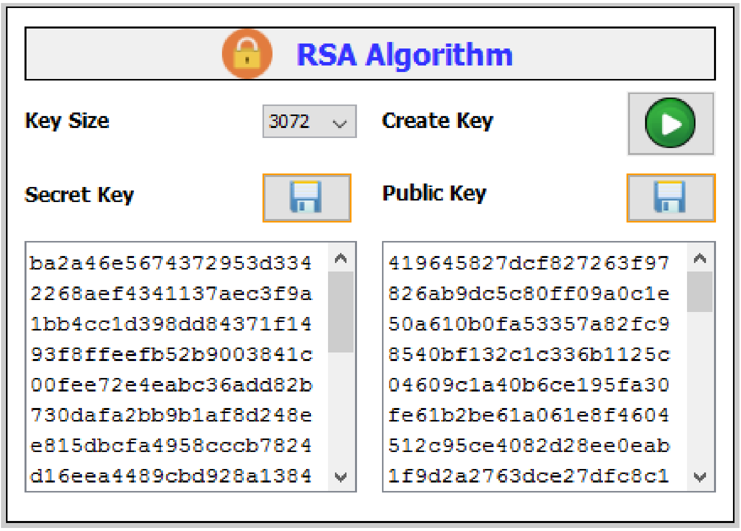
	STEVEDORING	CORGODORING	RECEIVING
Ship ID	JICT001	JICT001	JICT001
Σ Container (TEUs)	150	150	150
Σ Quaycrane	4	2	1
Σ Truck	16	45	13
Σ Rubber Tyred Gantry	45	23	24
Σ Torklift	34	19	16
Σ Toploader	35	18	31

Warning!!! DT : 7 Days, 5 Hours, 25 Mins

10405 Minutes

Figure 4.9 Dwelling time prediction

3. The dwelling time data in notepad (.txt) format. After the officer save in hard drive computers, then the officer 1 should signed the data first then save the information to database before send to cloud. To signing the dwelling time data, officer 1 should make a pair of key from RSA algorithm. In this case, officer 1 choose 3072 bits key size as shown in Figure 4.10. There are many variations for key size in this research like 1024 bits, 2048 bits and 3072 bits. The longer key size the harder attacker to decrypt and attack the message. After make a pair of key, then the officer can also save the key to computer hard drive.



RSA Algorithm

Key Size 3072

```
ba2a46e5674372953d334
2268aef4341137aec3f9a
1bb4cc1d398dd84371f14
93f8ffeefb52b9003841c
00fee72e4eabc36add82b
730dafa2bb9b1af8d248e
e815dbcfa4958cccb7824
d16eea4489cbd928a1384
```

```
419645827dcf827263f97
826ab9dc5c80ff09a0c1e
50a610b0fa53357a82fc9
8540bf132c1c336b1125c
04609c1a40b6ce195fa30
fe61b2be61a061e8f4604
512c95ce4082d28ee0eab
1f9d2a2763dce27dfc8c1
```

Figure 4.10 RSA Keys

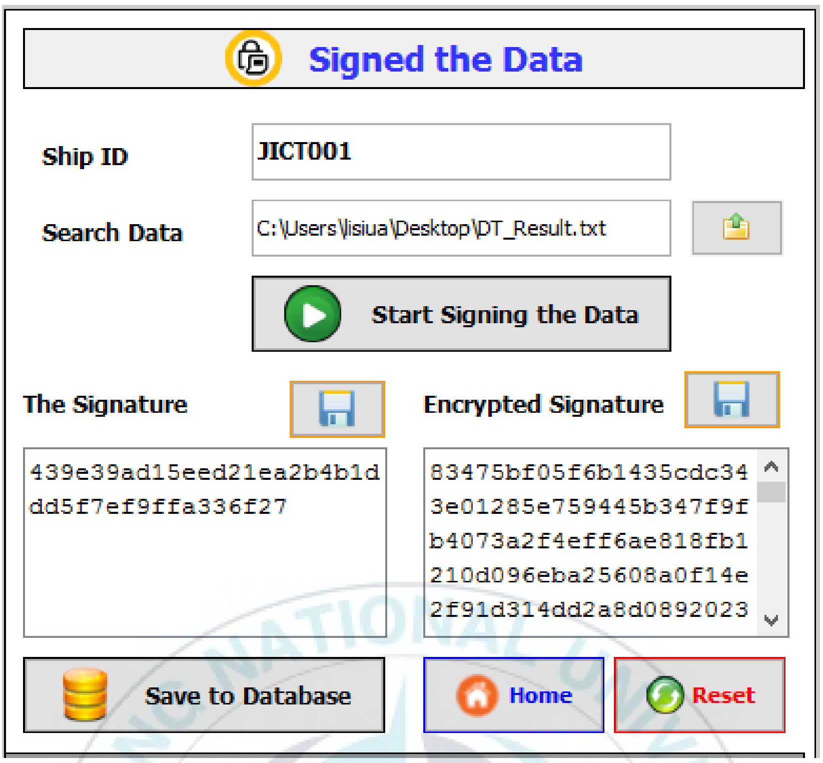
Secret key:

```
ba2a46e5674372953d3342268aef4341137aec3f9a1bb4cc1d398dd84371f1493f8ffe
efb52b9003841c00fee72e4eabc36add82b730dafa2bb9b1af8d248ee815dbcfa4958c
ccb7824d16eea4489cbd928a1384b637a6cf45daeed14c0e45409eddab129dd752c230
d6e8d9725e9559b61f612b2f19f06b74dd816d70ef8fc5e298869057e19c6ee340d7bf
6f5cc12c1c672f2f118185d2821526d52a2630a0f48fd058c608d6477668e55e9c0e86
7d3bf324b7f93f9b6a190da7d08bfd2387
```

Public Key:

```
419645827dcf827263f97826ab9dc5c80ff09a0c1e50a610b0fa53357a82fc98540bf1
32c1c336b1125c04609c1a40b6ce195fa30fe61b2be61a061e8f4604512c95ce4082d2
8ee0eab1f9d2a2763dce27dfc8c13257444fc1f4d81605bc8551f66b7ad6fc7de45dff
c349082c4f4a7d1640c7f36e84ad261adf5a6cea73b3254d34c8b84614fb5db9e23fe1
0e83fc266ea7cededb22d515a9829a7a7415b5b1ec4ad18e1139140a8284424662421 9
8fd49fcd95470909f331dee318f279a9b03fb71eae8e8a5ed04d4d8d0914385bded155
b85fc6523643e5b672cd2d47630effa7305248ff6b0a6e17b2035492c20bc4f2fdb45d
44c3125cb57b4bc2773d325356ffb49da1e8cbd9d449fbf29544e8df4cbb7e0c1603d5
f2874ce391a991dcbf383f6bda65b27152c79291e4315ada14afda4f72a0f0b44c1483
073c22855a925d064caeb0b1dbd5c8123435688ef7c9c70614791310416 1ecd12565bc
94d79092d70c6cd0d85169c6ef98ff8092464ac0d2ded9f666190fa7b13857c4d4ef
```

4. Signing process by officer 1 by input ship id and upload the dwelling time data and then send the public key and message digest to database as shown in Figure 4.10 and Figure 4.11. The data is important to save in database because by input the ship ID the officer can get the all kind of information such as public key, value of message digest, the signatures, date created etc. The value of message digest will be totally different although the dwelling time data are almost identical for each other. A pair of key (secret and public key) is derive from RSA algorithm and used to encrypt the message digest. A secret key from RSA is used to encrypt the message digest which is every message digest has a fixed length and different for each data. In the signing process a digital signature will be attached in the dwelling time data then the signature and data will be upload to the cloud.



Ship ID: JICT001

Search Data: C:\Users\lisiua\Desktop\DT_Result.txt

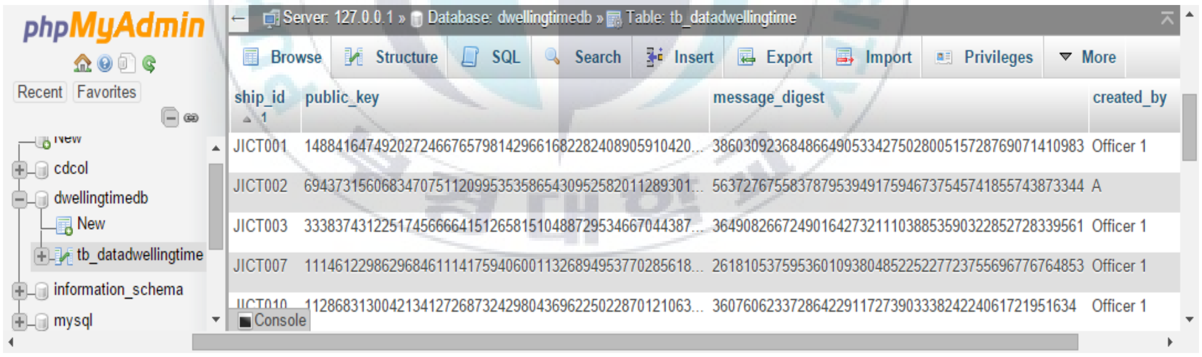
The Signature:

```
439e39ad15eed21ea2b4b1d
dd5f7ef9ffa336f27
```

Encrypted Signature:

```
83475bf05f6b1435cdc34
3e01285e759445b347f9f
b4073a2f4eff6ae818fb1
210d096eba25608a0f14e
2f91d314dd2a8d0892023
```

Figure 4.10 Signed the dwelling time data



Server: 127.0.0.1 » Database: dwellingtimedb » Table: tb_datadwellingtime

ship_id	public_key	message_digest	created_by
JICT001	14884164749202724667657981429661682282408905910420...	386030923684866490533427502800515728769071410983	Officer 1
JICT002	69437315606834707511209953535865430952582011289301...	563727675583787953949175946737545741855743873344	A
JICT003	33383743122517456666415126581510488729534667044387...	364908266724901642732111038853590322852728339561	Officer 1
JICT007	11146122986296846111417594060011326894953770285618...	261810537595360109380485225227723755696776764853	Officer 1
JICT010	11286831300421341272687324298043696225022870121063...	360760623372864229117273903338242224061721951634	Officer 1

Figure 4.11 Information in database

Value of Signature:

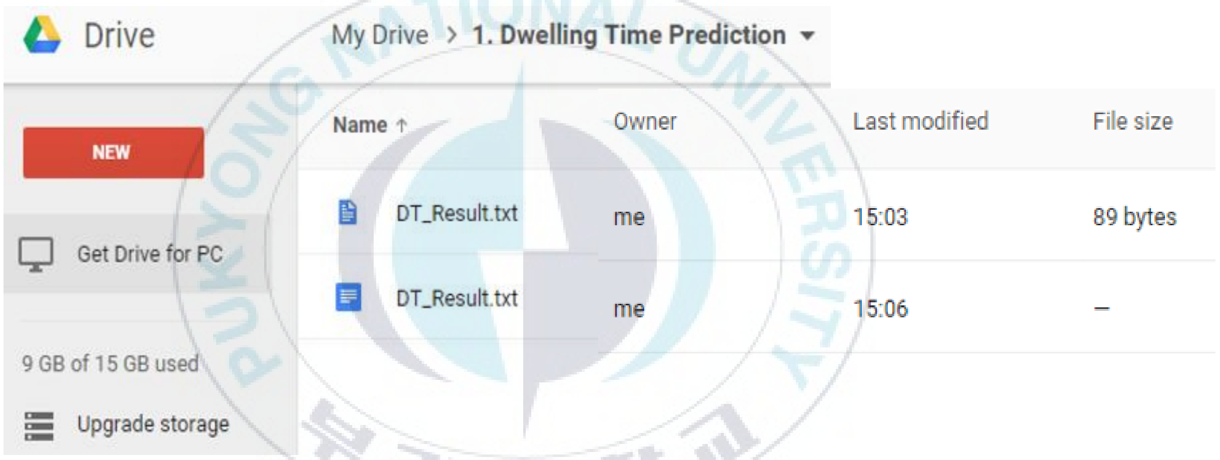
```
439e39ad15eed21ea2b4b1ddd5f7ef9ffa336f27
```

Value of Encrypted Signature:

```
83475bf05f6b1435cdc343e01285e759445b347f9fb4073a2f4eff6ae818fb1210d096
eba25608a0f14e2f91d314dd2a8d089202307c46f0a8136623a76fff1e8dd731a4e3a6
fc602b8f79b6d9b4fd2bacc7f0acb560b548a021bebc53bcb7fe8b075843be899edf6e
666cb2545292b2563f237c394241621a60aca176aa405d25735486d5b2a88352d82f18
```

```
e307c3858cda8752fec34c5fff3fd1d2030dbd8adf3a477d277eef66cd2658b6572742
f9276d3d193a5aa5c92b3c0858d999e546d45ac7501c7355ba2c63de335d8f880ba180
265a943a82e39c503e1002d308398711647c6466ed0508ef5aa6c60633c5c858cd3d68
2bf9ef24d97dcfe88b2adf6190ac295a0583ea13d32f8a07e6b082d2b7d3e79a6afbbb
b7393dfec1d98496682ffb4f9412c642fd0580765d16e37f7a99afbb375f457d93f9b0
62afe0069e6a93b8ef6c65cd1fbcad193ced22a2182cb238b6103ac81a84760c57fbf9
2e937cff495817ffb1af2fcd6d1426f0add11b260c9a1c06199e2ae29808f478667e
```

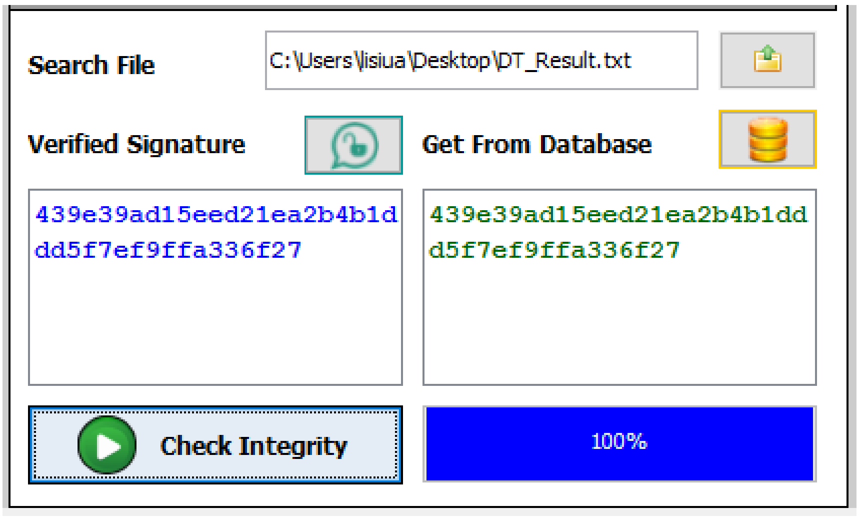
5. Dwelling time data will be upload to the cloud. In this research, using google drive as cloud as shown in Figure 4.12. In this cloud, the attacker possible to change the dwelling time data, and in this case digital signature will be necessary to protect originality of the data.



Name ↑	Owner	Last modified	File size
DT_Result.txt	me	15:03	89 bytes
DT_Result.txt	me	15:06	–

Figure 4.12 Dwelling time data in Cloud

6. After officer 1 upload the dwelling time data in the cloud, then officer 2 can download anytime to verify that data still original from the authorized sender. Officer 2 can verify the data by input ship id and upload the dwelling time data that already downloaded, then when decrypted is done the officer will get the value of message digest. The value will be compare with value from database, if the value is same then the data is still original, but the otherwise if the value is not same, it means the data has been changed by attacker.


Search File

C:\Users\lisiua\Desktop\DT_Result.txt

Verified Signature

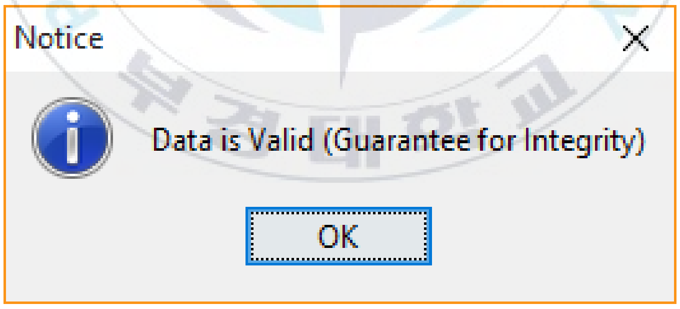
Get From Database

439e39ad15eed21ea2b4b1d
dd5f7ef9ffa336f27

439e39ad15eed21ea2b4b1dd
d5f7ef9ffa336f27

Figure 4.13 Verify the dwelling time data

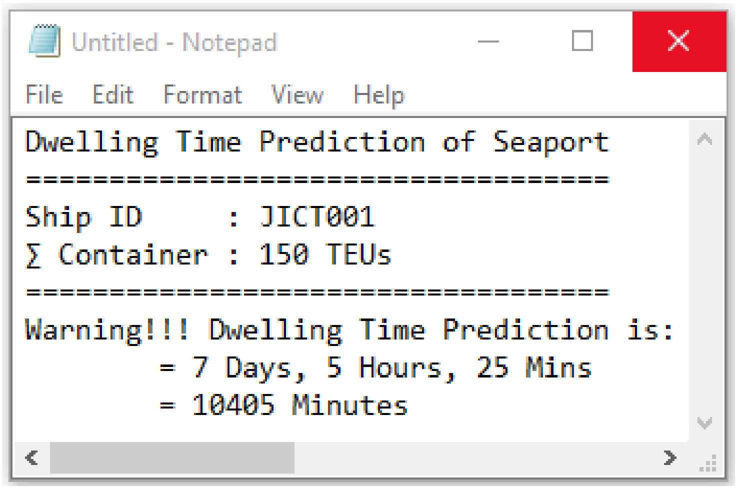
After the officer 2 get the decrypt signature result of decrypt from dwelling time data, then the officer compare with value from database. The message digest value from database represented the original value (data original) of dwelling time data before the attacker change the data. There is an indicator to show the data is valid or not by click in "check integrity" button as shown in Figure 4.14. By using this indicator, the officer 2 can easily decide the data is still valid or not.


Notice

Data is Valid (Guarantee for Integrity)

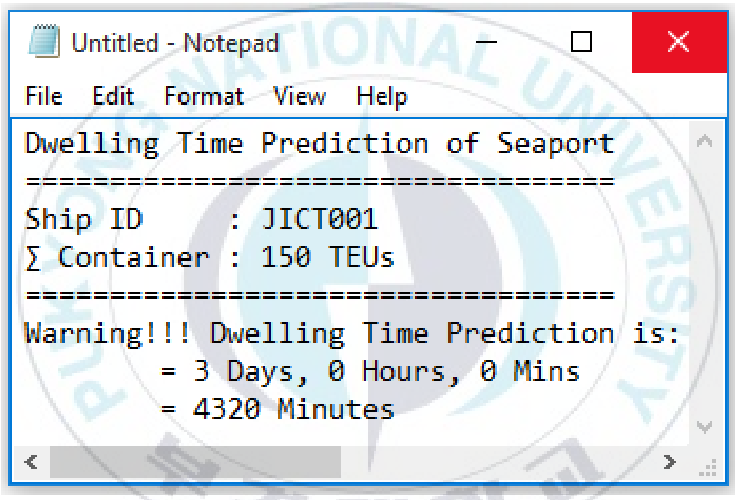
Figure 4.14 Indicator if data is valid

7. In the case if the attacker change the dwelling time data. For the example the original dwelling time prediction is 7 days, 5 hours and 25 minutes, then the attacker changes the data in the cloud storage becomes 3 days, 0 hours, 0 minutes as shown in Figure 4.16. If the officer 2 verify the data changed, the officer 2 will find out that "the verified signature" is not same with "value from database", it means the data has been changed by the attacker as shown in Figure 4.17.



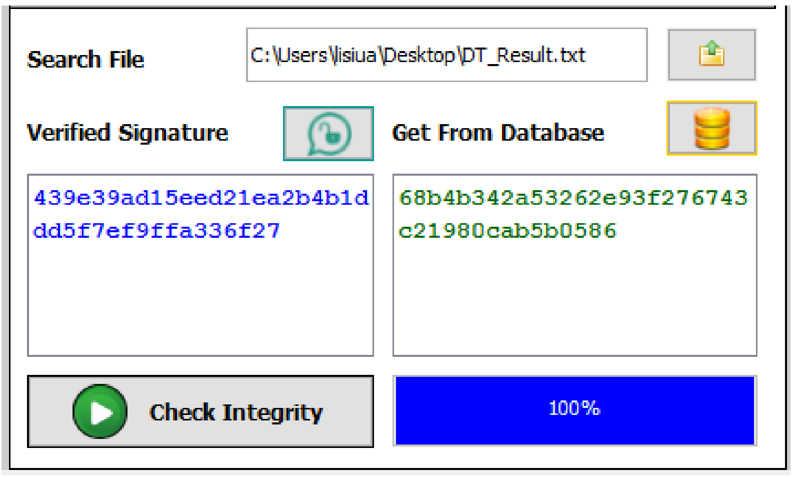
```
Dwelling Time Prediction of Seaport
===================================
Ship ID     : JICT001
∑ Container : 150 TEUs
===================================
Warning!!! Dwelling Time Prediction is:
        = 7 Days, 5 Hours, 25 Mins
        = 10405 Minutes
```

Figure 4.15 The original of dwelling time data



```
Dwelling Time Prediction of Seaport
===================================
Ship ID     : JICT001
∑ Container : 150 TEUs
===================================
Warning!!! Dwelling Time Prediction is:
        = 3 Days, 0 Hours, 0 Mins
        = 4320 Minutes
```

Figure 4.16 The attacker change the data



C:\Users\lisiua\Desktop\DT_Result.txt

```
439e39ad15eed21ea2b4b1d
dd5f7ef9ffa336f27
```

```
68b4b342a53262e93f276743
c21980cab5b0586
```

Figure 4.17 Message digest value with the data changed

Figures above explained if the attacker change the data. The original data is show in Figure 4.15 then the attacker change the data as shown in Figure 4.16. The attacker can change anything and give the wrong information to officer 2.

Comparing message digest of dwelling time data:

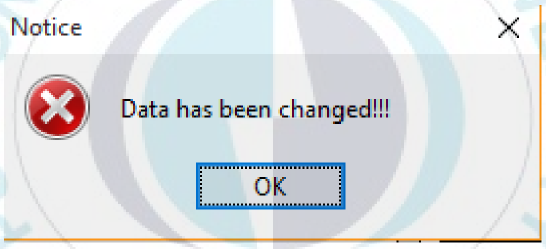
Decrypt Signature (the changed data):

```
439e39ad15eed21ea2b4b1ddd5f7ef9ffa336f27
```

Get from database:

```
68b4b342a53262e93f276743c21980cab5b0586
```

The officer 2 can see the value of message digest from decrypt signature and value from database. When the value of message digest is not same then there is an indicator as shown in Figure 4.18.



Data has been changed!!!

Figure 4.18 Indicator when the data has been changed

If the value from decrypt the signature and value from database is not same, it means the data has been changed by the attacker. Even the attacker changed only one bit, the value of message digest will totally changes. The value of message digest is not same to every kind of data, no matter how similar for each data, the value of message digest will totally different to each other. Message digest has a fixed length, even when the officer 1 using different key size like 1024 bits, 2048 bits and 3072 bits, the length of message digest will not change. At this point, if the officer 2 find that data already changed by attacker, then the officer can make some step to prevent apply that data in real situation. The officer can make a contact to authorized sender (original sender) to confirm manually the data. In this research is assumed that every officer using same desktop application because there are directly communication between officer.

Chapter 5. Conclusion and Future Work

5.1 Conclusion

Digital signature with using hashing algorithm SHA-256 is implemented to provide data integrity of dwelling time data by signed and verified the dwelling time data. The key is obtained from RSA algorithm and used to encrypt and decrypt the message digest from dwelling time data are 1.024 bits, 2.048 bits and 3.072 bits. Based on experimental results, digital signatures with SHA-256 can be used to detect the dwelling time data still original from the sender or not. Even the data only changed one character, the message digest totally changed. The officer can see easily comparing message digest decrypted with message digest from database by input ship id then the result will show up via indicator which is give information the data has been changed by attacker or still original from the authorized sender. The key size from RSA algorithm is not affect with the length of message digest because characteristic of message digest is has a fixed length. This apps also works for any format data like doc, pdf, *jpg, *png, mp3, mp4 and etc. We have successfully developed digital signature applied to provide data integrity of dwelling time data in the cloud storage.

5.2 Future Work

Digital signatures to provide data integrity is a fundamental component of information security which is verified the data has remained unaltered in transit from creation to reception. Related in this research, hopefully future research can provide many types of data like video, mp3 and other format. The future design for user interface also hopefully more secure and privacy via authorized officer like gives username and password to authenticity before use the program.

References

[1] C. Paar dan J. Perlz, Understanding Cryptography, Berlin: Springer, 2010.

[2] N. Ferguson, Cryptography Engineering: Design, Principles and Practical Applications, Indianapolis: Wiley Publishing, Inc, 2010.

[3] T. B. Idalino, L. Moura, F. R. Custodio dan D. Panario, "Locating Modifications in Signed Data for Partial Data Integrity," *Information Processign Letters*, vol. 115, no. 10, pp. 731-737, 2015.

[4] C. Liu , C. Yang , X. Zhang dan J. Chen , "External Integrity Verification for Outsourced Big Data in Cloud and IoT: A Big Picture," *Future Generation Computer Systems*, vol. 49, pp. 58-67, 2015.

[5] K. Liang , M. H. Au, J. K. Liu, W. Susilo, D. S. Wong, G. Yang dan A. Yang, "A secure and efficient Ciphertext-Policy Attribute-Based Proxy Re-Encryption for cloud data sharing," *Future Generation Computer Systems*, vol. 52, pp. 95-108, 2015.

[6] S. Rahmadika, "Using Naive Bayes Algorithms to Predict Container Dwell Time: Case Study JICT, Indonesia," dalam *KMMS Spring, Conference*, Andong, South Korea, 2015.

[7] P. Kitsos, N. Sklavos dan O. Koufopavlou, "An Efficient Implementation of the Digital Signature Algorithm," Greece, 2002.

[8] D. J. Hombrebueno, G. C. Sicat, J. Niguidula, E. Chavez dan A. Hernandez, "Symmetric Cryptosystem Based on Data Encryption Standard Integrating HMAC and Digital Signature Scheme Implemented in Multi-Cast Messenger Application," dalam *International Conference on Computer and Electrical Engineering*, Manila, 2009.

[9] M. Thangavel, P. Varalakshmi, M. Murali dan K. Nithya, "An Enhanced and Secured RSA Key Generation Scheme (ESRKGS)," *Journal of Information Security and Applications*, vol. 20, pp. 3-10, 2015.

[10] I. C. Lin dan C. C. Chang, "Security Enchancement for Digital Signature Schemes with Fault Tolerance in RSA," *Information Sciences*, vol. 177, no. 19, pp. 4031-4039, 2007.

[11] A. J. Menezes, P. C. V. Oorschot dan S. A. Vanstone, Handbook of Applied Cryptography, CRC-Press 1996: USA, 1996.

[12] R. Burton, Handbook of Financial Cryptography and Security, Australia: Boca Raton: Chapman & Hall/CRC, 2010.

[13] B. A. Forouzan dan D. A. College, Cryptography and Network Security, India: McGraw-Hill Higher Education, 2008.

[14] H. Delfs dan H. Knebl, Introduction to Cryptography, Berlin: Springer-Verlag Berlin Heidelberg, 2007.

[15] G. Pereira, C. Puodzious dan P. Barreto, “Shorter hash-based signatures,” *The Journal of Systems and Software*, pp. 1-6, 2015.

[16] R. Buyya, J. Broberg dan A. Goscinski, Cloud Computing Principles and Paradigms, New Jersey: John Wiley & Sons, Inc, 2011.

[17] T. Mather, S. Kumaraswamy dan S. Latif, Cloud Security and Privacy, United States of America: O'Reilly Media, Inc., 2009.

[18] M. Filip, “The Issue of Dwell Time Charges to Optimize Container Terminal Capacity,” dalam *IAME Annual Conference*, 2005.

[19] S. Hashimi, S. Komatineni dan D. Maclean, Pro Android 2, India: Apress; 2009 edition (June 23, 2009), 2009.

[20] R. Meier, Professional Android Application Development, Indianapolis, Indiana: Wiley Publishing, Inc, 2009.

Acknowledgements

Immeasurable appreciation and deepest gratitude for the help and support are extended to the following who in one way or another have contributed in making this thesis possible. The final outcome of this thesis required a lot of guidance and assistance from many people and I am extremely fortunate to have got this all along the completion of my thesis. Whatever I have done is only due to such guidance and I would not forget to thank them.

First, I am deeply indebted to my supervisor **Prof. Kyung-Hyune Rhee** for his guidance, motivation, patience and encouragement to look my research and my work in different ways and for opening my mind. I learned a lot about this topic and also many other things while staying in LISIA Lab and your support was essential to my success here.

I heartily to thank **Prof. Man-Gon Park**, Chairman, for assistance and words of encouragement, also for his time and effort in checking this thesis. I am extremely grateful to Professor for providing such a nice support and guidance though he had busy schedule managing the academic affairs. I also would like to thank **Prof. Bong-Kee Sin, Prof. Chang-So Kim and Dr. Hilwadi Hindersah** for useful knowledge and information during lectures, also providing indispensable advice and support on different aspects.

Many of my experimental work will not be completed without the help of **Dr. Park, Bayu, Lewis** and all of LISIA members. Thank you for all kinds of your support for me.

I devoted all of my works to my father, mother, older brother and younger sister who always love me and pray for my well-being. They always give me spirit and motivation when I’m tired by only hearing their voice on the phone. Things that are not to be forgotten are they never ceases to remind me to pray and thanks to Allah SWT. And to all my families and relatives who care for me wherever you are, thank you. I love you.

For all 14 members of double degree programs ITB-PKNU, I love you all, we are like brothers and sisters. Sometimes we did a stupid things, laughed, sad, and happy together.

Last but not least, thanks to all Indonesian students in PKNU who always give me a helping hand. I’ve got awesome experiences from you guys. Also to any person or institution whose name I could not mention here, thank you for our kind of support.

Appendix

A. Access to Database

```
/*
 @author Sandi Rahmadika
 @LISIA
 */
package Sandi_Thesis;
/**
 * @author Sandi Rahmadika
 */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Date;

public class AFDatabase_access {

private Connection connect = null;
private PreparedStatement preparedStatement = null;
private ResultSet resultSet = null;

public   void   insert_database(String   ship_id,   String   public_key,   String
message_digest, String user) throws Exception {
    try {
      // This will load the MySQL driver, each DB has its own driver
      Class.forName("com.mysql.jdbc.Driver");
      // Setup the connection with the DB
      connect = DriverManager
          .getConnection("jdbc:mysql://127.0.0.1/dwellingtimedb?"
              + "user=root&password=");
      preparedStatement = connect
          .prepareStatement("insert  into   dwellingtimedb.tb_datadwellingtime
values (?, ?, ?, ? , ?)");
      // "myuser, webpage, datum, summery, COMMENTS from feedback.comments");
      // Parameters start with 1
      preparedStatement.setString(1, ship_id);
      preparedStatement.setString(2, public_key);
      preparedStatement.setString(3, message_digest);
      preparedStatement.setString(4, user);
      preparedStatement.setDate(5, new java.sql.Date(2015, 12, 11));
      preparedStatement.executeUpdate();
    }
    catch (Exception e) {
      throw e;
    } finally {
       if (connect != null) {
        connect.close();
      }}}

public String[] retrieve_database(String ship_id) throws Exception {
    String[] result = new String[2];
    try {
      // This will load the MySQL driver, each DB has its own driver
      Class.forName("com.mysql.jdbc.Driver");
      // Setup the connection with the DB
      connect = DriverManager
          .getConnection("jdbc:mysql://127.0.0.1/dwellingtimedb?"
              + "user=root&password=");
      preparedStatement = connect
          .prepareStatement("select    public_key,       message_digest    from
dwellingtimedb.tb_datadwellingtime where ship_id=?");
      // "myuser, webpage, datum, summery, COMMENTS from feedback.comments");
      // Parameters start with 1
     preparedStatement.setString(1, ship_id);
     resultSet = preparedStatement.executeQuery();
     while (resultSet.next()) {

         result[0] =  resultSet.getString("public_key");
         result[1] =  resultSet.getString("message_digest");

         System.out.println(result[0]);      }
     return result;
    }
    catch (Exception e) {
      throw e;
    } finally {
       if (connect != null) {
        connect.close();
      }
    }}}
```

B. Main Menu Operation

```
package Sandi_Thesis;

import java.awt.Color;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintWriter;
import java.math.BigInteger;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JProgressBar;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;
import javax.swing.filechooser.FileNameExtensionFilter;
public class ACDigital_Signature extends javax.swing.JFrame {

    static int bitleg;
    static BigInteger ciphertext;

    static ADRSA_KeyAlgorithm rsa;

    public ACDigital_Signature() {

        initComponents();

        barComplete.setStringPainted(true);
        barComplete.setForeground(Color.blue);
        bitleg = Integer.parseInt((String) jComboBox1.getSelectedItem());
        rsa = new ADRSA_KeyAlgorithm();
        rsa.KeyRSA(bitleg);

        this.setLocation(300, 50);
        this.setResizable(false);
    }

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        barComplete.setValue(0);
        String filename = signed_searchData.getText();
        filename = filename.replace('\\', '/');
        if ("".equals(filename)) {
            barComplete.setIndeterminate(true);
            JOptionPane.showMessageDialog(null,   "Please   Input   the   Data",
"Announcement", JOptionPane.ERROR_MESSAGE);
            barComplete.setMaximum(200);
            barComplete.setValue(0);
            barComplete.setIndeterminate(false);
        } else {
            AEHashFunction_SHA2 sha1 = new AEHashFunction_SHA2();
            try {
                barComplete.setIndeterminate(true);
                BigInteger              sh1              =              new
BigInteger(sha1.md(filename).abs().toString());
                signed_messageDigest.setText(sh1.toString());
                BigInteger             sha1t              =              new
BigInteger(signed_messageDigest.getText());
                signed_signatures.setText(rsa.encrypt(sha1t).toString());
                jButton8.setEnabled(true);
                barComplete.setMaximum(200);
                barComplete.setValue(200);
                barComplete.setIndeterminate(false);
                JOptionPane.showMessageDialog(null, "The Signature successfully
created!");

            } catch (Exception ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
            }

        }
    }

    private   void   searchData_SignedActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        barComplete.setValue(0);
        JFileChooser chooser = new JFileChooser();
        FileNameExtensionFilter  filter  =  new  FileNameExtensionFilter("txt",
"doc", "docx", "pdf", "jpg", "png");
```

```
        chooser.setFileFilter(filter);
        int returnVal;
        returnVal = chooser.showOpenDialog(null);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = chooser.getSelectedFile();
            String attach = file.toString();
            signed_searchData.setText(attach);
        }
    }

    private  void  signed_searchDataActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        BigInteger bigInteger = new BigInteger(RSA_secretkey.getText());
        try {
            OutputWrite(getSaveLocation(), bigInteger, "Keyprivate.txt");
        } catch (FileNotFoundException ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        barComplete.setIndeterminate(true);
        barComplete.setIndeterminate(false);
    }
    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        bitleg = Integer.parseInt((String) jComboBox1.getSelectedItem());
        barComplete.setValue(0);
    }

    private  void  btnCheckIntegrityActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        barComplete.setValue(0);
        String filename = verified_searchData.getText();
        filename = filename.replace('\\', '/');
        if ("".equals(filename)) {
            barComplete.setIndeterminate(true);
            JOptionPane.showMessageDialog(null,  "Please  Input  the  Data  ",
"Notice", JOptionPane.ERROR_MESSAGE);
            barComplete.setMaximum(200);
            barComplete.setValue(0);
            barComplete.setIndeterminate(false);
        } else {
            AEHashFunction_SHA2 sha1 = new AEHashFunction_SHA2();
            try {
                barComplete.setIndeterminate(true);
              //  jTextArea5.setText(sha1.md(filename).abs() + "");

              //  BigInteger dsrsa = new BigInteger(jTextArea4.getText());
               // jTextArea6.setText(rsa.decrypt(dsrsa).toString());
                barComplete.setMaximum(200);
                barComplete.setValue(200);
                barComplete.setIndeterminate(false);
                if
(verified_database.getText().equals(verified_decrypt.getText())) {
                    JOptionPane.showMessageDialog(null,    "Data    is    Valid
(Guarantee for Integrity)", "Notice", JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(null,    "Data    has    been
changed!!!", "Notice", JOptionPane.ERROR_MESSAGE);
                }

            } catch (Exception ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
            }
        }
    }

    private  void  searchData_verifiedActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        barComplete.setValue(0);
        JFileChooser chooser = new JFileChooser();
        FileNameExtensionFilter  filter  =  new  FileNameExtensionFilter("pdf",
"doc", "docx", "jpg", "png", "txt");
        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(null);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = chooser.getSelectedFile();
            String attach = file.toString();
            verified_searchData.setText(attach);
        }
    }

    private  void  verified_searchDataActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

         JOptionPane.showMessageDialog(null,  "Successfully  Sent  to  Database",
"Notice", JOptionPane.INFORMATION_MESSAGE);

        AFDatabase_access d = new AFDatabase_access();
        try {
            String ship_id = shipID.getText();
            String public_key = RSA_publickey.getText();
            String message_digest = signed_messageDigest.getText();
            String user = "Officer 1";
            d.insert_database(ship_id, public_key, message_digest, user);
            //String[] data = d.retrieve_database(ship_id);
        } catch (Exception ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }

    }

    private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
        AFDatabase_access d = new AFDatabase_access();
        try {
            String ship_id = shipID.getText();
            String[] data = d.retrieve_database(ship_id);
            verified_database.setText(data[1]);
        } catch (Exception ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }
    }

    private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {
         BigInteger bigInteger = new BigInteger(signed_messageDigest.getText());
        try {
            OutputWrite(getSaveLocation(), bigInteger, "DigitalSignature.txt");
        } catch (FileNotFoundException ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }

    private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
        AAMain_Menu movequickly = new AAMain_Menu();
        movequickly.setVisible(true);
        this.dispose();
    }

    private void btnDecrypt_signatureActionPerformed(java.awt.event.ActionEvent
evt) {
        BigInteger dsrsa = new BigInteger(signed_signatures.getText());
                verified_decrypt.setText(rsa.decrypt(dsrsa).toString());
    }

    public       static       void       main(String       args[])       throws
UnsupportedLookAndFeelException {
        try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }    catch    (ClassNotFoundException    |    InstantiationException    |
IllegalAccessException | UnsupportedLookAndFeelException e) {
            e.printStackTrace();
        }

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ACDigital_Signature().setVisible(true);
            }
        });
   bitleg = Integer.parseInt((String) jComboBox1.getSelectedItem());
```

```
        rsa = new ADRSA_KeyAlgorithm();
        rsa.KeyRSA(bitleg);
        this.setLocation(300, 50);
        this.setResizable(false);
    }

    public void OutputWrite(File saveLocation, BigInteger EncryptCodes, String
name)
            throws FileNotFoundException {

        PrintWriter file
                = new PrintWriter(new FileOutputStream(new File(saveLocation,
name)));
        file.println(EncryptCodes);
        file.close();
    }

    private File getSaveLocation() {
        JFileChooser chooser = new JFileChooser();
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        int result = chooser.showSaveDialog(this);

        if (result == chooser.APPROVE_OPTION) {
            return chooser.getSelectedFile();
        } else {
            return null;
        } }
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        BigInteger bigInteger = new BigInteger(signed_signatures.getText());
        try {
            OutputWrite(getSaveLocation(), bigInteger, "DigitalSignature.txt");
        } catch (FileNotFoundException ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,  null,
ex);
        }
    }

private void searchData_SignedActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        barComplete.setValue(0);
        JFileChooser chooser = new JFileChooser();
        FileNameExtensionFilter  filter  =  new  FileNameExtensionFilter("txt",
"doc", "docx", "pdf", "jpg", "png");
        chooser.setFileFilter(filter);
        int returnVal;
        returnVal = chooser.showOpenDialog(null);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = chooser.getSelectedFile();
            String attach = file.toString();
            signed_searchData.setText(attach);
        }
    }

    private  void  signed_searchDataActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }
    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        BigInteger bigInteger = new BigInteger(RSA_secretkey.getText());
        try {
            OutputWrite(getSaveLocation(), bigInteger, "Keyprivate.txt");
        } catch (FileNotFoundException ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }
    }
        barComplete.setIndeterminate(false);
    }
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        bitleg = Integer.parseInt((String) jComboBox1.getSelectedItem());
        barComplete.setValue(0);
    }
private void btnCheckIntegrityActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        barComplete.setValue(0);
        String filename = verified_searchData.getText();
        filename = filename.replace('\\', '/');
        if ("".equals(filename)) {
            barComplete.setIndeterminate(true);
            JOptionPane.showMessageDialog(null,  "Please  Input  the  Data  ",
"Notice", JOptionPane.ERROR_MESSAGE);
            barComplete.setMaximum(200);
            barComplete.setValue(0);
            barComplete.setIndeterminate(false);
        } else {
            AEHashFunction_SHA2 sha1 = new AEHashFunction_SHA2();
            try {
                barComplete.setIndeterminate(true);
              //  jTextArea5.setText(sha1.md(filename).abs() + "");

              //  BigInteger dsrsa = new BigInteger(jTextArea4.getText());
               // jTextArea6.setText(rsa.decrypt(dsrsa).toString());
                barComplete.setMaximum(200);
                barComplete.setValue(200);
                barComplete.setIndeterminate(false);
                if
(verified_database.getText().equals(verified_decrypt.getText())) {
                    JOptionPane.showMessageDialog(null,    "Data    is    Valid
(Guarantee for Integrity)", "Notice", JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(null,    "Data    has    been
changed!!!", "Notice", JOptionPane.ERROR_MESSAGE);
                }
} catch (Exception ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
            }
private void searchData_verifiedActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        barComplete.setValue(0);
        JFileChooser chooser = new JFileChooser();
        FileNameExtensionFilter  filter  =  new  FileNameExtensionFilter("pdf",
"doc", "docx", "jpg", "png", "txt");
        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(null);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = chooser.getSelectedFile();
            String attach = file.toString();
            verified_searchData.setText(attach);
        }
private void verified_searchDataActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

         JOptionPane.showMessageDialog(null,  "Successfully  Sent  to  Database",
"Notice", JOptionPane.INFORMATION_MESSAGE);

        AFDatabase_access d = new AFDatabase_access();
        try {
            String ship_id = shipID.getText();
            String public_key = RSA_publickey.getText();
            String message_digest = signed_messageDigest.getText();
            String user = "Officer 1";
            d.insert_database(ship_id, public_key, message_digest, user);
            //String[] data = d.retrieve_database(ship_id);
        } catch (Exception ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }
private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
        AFDatabase_access d = new AFDatabase_access();
        try {
            String ship_id = shipID.getText();
            String[] data = d.retrieve_database(ship_id);
            verified_database.setText(data[1]);
        } catch (Exception ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }
private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {
         BigInteger bigInteger = new BigInteger(signed_messageDigest.getText());
        try {
            OutputWrite(getSaveLocation(), bigInteger, "DigitalSignature.txt");
        } catch (FileNotFoundException ex) {

Logger.getLogger(ACDigital_Signature.class.getName()).log(Level.SEVERE,   null,
ex);
        }
private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
        AAMain_Menu movequickly = new AAMain_Menu();
        movequickly.setVisible(true);
        this.dispose();
    }

    private void btnDecrypt_signatureActionPerformed(java.awt.event.ActionEvent
evt) {
        BigInteger dsrsa = new BigInteger(signed_signatures.getText());
                verified_decrypt.setText(rsa.decrypt(dsrsa).toString());
    }

    public      static      void      main(String      args[])      throws
UnsupportedLookAndFeelException {
        try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }   catch   (ClassNotFoundException   |   InstantiationException   |
IllegalAccessException | UnsupportedLookAndFeelException e) {
            e.printStackTrace();
        }

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ACDigital_Signature().setVisible(true);
            }
        });
```

C. Hash Function SHA-256 and RSA Keys

```
package Sandi_Thesis;

import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.math.BigInteger;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import static sun.security.krb5.Confounder.bytes;

public class AEHashFunction_SHA2 {
    private static byte[] bytes;
  private static int BUFFER_SIZE = 32 * 1024;

  public static void main(String[] args) throws Exception {
  }

  public BigInteger md(String f) throws Exception {
    BufferedInputStream file = new BufferedInputStream(new FileInputStream(f));
    MessageDigest md = MessageDigest.getInstance("SHA-256");
  //  md.update(f.getBytes());
 //   return bytesToHex(md.digest());
   DigestInputStream in = new DigestInputStream(file, md);
    int i;
    byte[] buffer = new byte[BUFFER_SIZE];
    do {
      i = in.read(buffer, 0, BUFFER_SIZE);
    } while (i == BUFFER_SIZE);
    md = in.getMessageDigest();
    in.close();

    return new BigInteger(md.digest());
}};
```
-
```
package Sandi_Thesis;

//package atnf.atoms.mon.util;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class ADRSA_KeyAlgorithm {
    private BigInteger n, d, e;
    public BigInteger getN() {
        return n;
    }
    public void setN(BigInteger n) {
        this.n = n;
    }
    public BigInteger getD() {
        return d;
    }
    public void setD(BigInteger d) {
        this.d = d;
    }
    public BigInteger getE() {
        return e;
    }
    public void setE(BigInteger e) {
        this.e = e;
    }
    /**
     * Create an instance that can encrypt using someone elses public key.
     */
    public ADRSA_KeyAlgorithm(BigInteger newn, BigInteger newe) {
        n = newn;
        e = newe; }

    /**
     * Create an instance that can both encrypt and decrypt.
     */
    public ADRSA_KeyAlgorithm() {
    }

    public void KeyRSA(int bits){
         SecureRandom r = new SecureRandom();
        BigInteger p = new BigInteger(bits / 2, 100, r);
        BigInteger q = new BigInteger(bits / 2, 100, r);
        n = p.multiply(q);
        BigInteger m = (p.subtract(BigInteger.ONE)).multiply(q
                .subtract(BigInteger.ONE));
        boolean found = false;
        do {
            e = new BigInteger(bits / 2, 50, r);
            if (m.gcd(e).equals(BigInteger.ONE) && e.compareTo(m) < 0) {
                found = true;
            }
        } while (!found);
        d = e.modInverse(m);
    }
```

```
        /**
         * Encrypt the given plaintext message.
         */
        public synchronized String encrypt(String message) {
            return (new BigInteger(message.getBytes())).modPow(d, n).toString();
        }
        /**
         * Encrypt the given plaintext message.
         */
        public synchronized BigInteger encrypt(BigInteger message) {
            return message.modPow(d, n);
        }
        /**
         * Decrypt the given ciphertext message.
         */
        public synchronized String decrypt(String message) {
            return      new      String((new      BigInteger(message)).modPow(e,
n).toByteArray());
        }
        /**
         * Decrypt the given ciphertext message.
         */
        public synchronized BigInteger decrypt(BigInteger message) {
            return message.modPow(e, n);
        }
        /**
         * Trivial test program.
         */
        public static void main(String[] args) throws Exception {
        }
        void setN(int bitleg) {
            throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
        }
```