Providing Data Integrity and Authentication using RSA Digital Signature and MD5 Hash Algorithm

Khin Thet Nwe Swe
Computer University (Monywa) Myanmar
aha3.aha@gmail.com

Abstract

This paper presents how to get data integrity and authentication in receiver. Signing process is performed in sender and verifying process is performed in receiver. In signing process, sender computes hash value of message/file. The output is unique hash value. Sender encrypts this unique hash value with sender's private key. The output is digital signature. And then, sender sends original message/file and it's signature to receiver. In verifying process, receiver decrypts the signature with sender's public key. The output is unique hash value. And receiver computes hash value of original message/file. If two hash values are equal, the content of message/file is not modified after signed and was sent by valid sender. If two hash values are not equal, the content of message/file is modified after signed and was sent by invalid sender.

1. Introduction

Security is essential in data transmission and resource sharing via public network like internet. Security attacks take the form of eavesdropping, masquerading, tampering and denial of service. Integrity is one kind of security and it means that the information cannot be altered in storage or transit between sender and intended receiver without the alternation being detected. Authentication is that the sender and receiver can confirm each other's identity and the origin or destination of the information.

Cryptography is the science of information security. Two types of cryptography are symmetric key cryptography and public key cryptography. Symmetric key cryptography uses the same key for both encryption and decryption [2]. The modern study of symmetric key ciphers relates mainly to the study of block ciphers and stream ciphers [1]. Public key cryptography requires each user to have two keys: a public key, that is published and a private key that is kept secret [5].

Digital signature mechanism can be implemented by combining public key cryptography and hash function. A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value h that is, h = H(m) [3]. System uses two algorithms: one is signing in which a private key is used to process the message and another is verification in which the matching public key is used with the message to check the validity of the signature. With digital signature technology, any change in the signed message/file causes the verification process to fail.

2. Digital Signature

For message sent through an insecure channel, integrity is guaranteed in public key systems by using digital signature. A digital signature can be used to guarantee, beyond doubt, the validity of message integrity and that of non-repudiation. Public key cryptography and hash function provide a method for employing digital signature. These factors are important in digital signature:

Integrity: The sender and receiver of a message may have a confidence that the message has not been altered during transmission. If a message is digitally signed, any change in the message will invalidate the signature.

Authentication: Digital signature can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was send by that user.

Non-repudiation: A digital signature also provides non-repudiation which means that it prevents the sender from claming that did not actually send the information.

2.1. Hash Function

Hash function is used in the implementation of digital signature and denoted H(x). It produces a unique hash value from the arbitrary length input. This function is one-way hash function. So, it is hard to compute original input from the hash value. H(x) is collision-free. Hence, it is computationally infeasible to find any two messages x and y such that H(x) = H(y).

Hash functions are used to condense an arbitrary length message to a fixed size, usually for subsequence signature by a digital signature algorithm. There are several reasons to sign such a hash instead of the whole original input.

Efficiency: The signature will be much shorter and thus save time since hashing is generally much faster than signing in practice.

Compatibility: Messages are typically bit strings, but some signature schemes operate on other domains. Hash functions can arbitrary input into the proper format.

Integrity: Without the hash functions, the test "to be signed" may have to be split into blocks small enough for the signature scheme to act on them directly. However, the receiver of the signed blocks is not able to recognize if all appropriate order.

2.2. Signature with Public Key Cryptography

Public key cryptography is particularly well adopted for the generation of digital signatures because it is relatively simple and does not require any communication between the recipient of assigned document and the signer or any third party. Public key encryption uses key pairs called public key and private key. The public key is made public and is distributed widely and freely. The private key is never distributed and must be kept secret. Data encrypted with the public key can only be decrypted with it's private key; conversely, data encrypted with it's private key can only be decrypted with it's public key. This characteristic is used in implement encryption and digital signature. In digital signature, private key is used to sign the message/file and public key is used to verify that message/file.

3. Application of MD5 Hash Algorithm

MD5 is a widely used cryptographic hash function with a 128-bit hash value A MD5 hash is typically expressed as a 32 digital hexadecimal number. MD5 algorithm takes input message of arbitrary length and generates 128-bit long output hash. MD5 hash algorithm consists of 5 steps.

- Append Padding Bits
- Append Length
- Initialize MD Buffer
- Process Message in 16-Word Blocks
- Output

3.1. Append Padding Bits

The input message is broken up into chunks of 512-bit blocks. The message is "padded" so that its length is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512.

Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended [6].

3.2 Append Length

The remaining bits are filled up with a 64-bit representing the length of the original message or file, in bits. In the unlikely event that the original length is greater than 2 power 64, then only the low-order 64 bits of original length are used. At this point the resulting message has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words.

3.3 Initialize MD Buffer

A four-word buffer (A, B, C and D) is used to compute the message digest. Here each of A, B, C and D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first.

Word A: 0x01234567
Word B: 0x89ABCDEF
Word C: 0xFEDCBA98
Word D: 0x76543210

3.4 Process Message in 16-Word Blocks

In addition MD5 uses four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word. Functions apply the logical operators AND, OR, NOT and XOR to the input bits.

F (X, Y, Z) = (X and Y) or (not(X) and Z)
G (X, Y, Z) = (X and Z) or (Y and not(Z))
H (X, Y, Z) = X xor Y xor Z
I (X, Y, Z) = Y xor (X or not (Z))

MD5 further uses a table K that has 64 elements. Element number i is indicated as K_i. The table is computed beforehand to speed up the computations. The elements are computed using the mathematical sin function.

$$K_i = \text{abs}(\sin(i+1)) * 2^{32}$$

The contents of the four buffers (A, B, C and D) are now mixed with words of the input, using the four auxiliary functions (F, G, H and I). There are four rounds, each involves 16 basic operations.

3.5 Output

After all rounds have been performed, the buffers A, B, C and D contain the MD5 digest of the original input.

4. RSA Key Generation

Our system uses RSA key generation. RSA can be used for security (encryption), confidentiality (signature), and key exchange purposes. RSA select a private key uniformly at random from a set of possible private keys. The outputs are the private key and a corresponding public key. To find a key pair e, d:

Generate two large random primes, p and q, where p and q are distinct prime, and form

n=pq
phi= (p-1) (q-1)

Choose an integer e, 1<e<phi, such that

gcd (e, phi) = 1

Compute the secret exponent d, 1<d<phi, such that

ed = 1(mod phi)

The private key is (n, d) and the public key is (n, e).

5. RSA Signature Scheme

RSA signature is one of digital signature types created by using RSA algorithm. First, sender applies hash function to the message or file to produce a message with fixed length. Then, sender signs the message or file by using RSA algorithm to produce the signature. To verify the signature: first, receiver runs the received message or file through the hash function. Second, receiver decrypts the signature with sender's public key [4]. The process of signing uses the equation as follow:

Signing,

$$s = m^d \bmod n$$

The sender sends this signature s to the recipient. Receiver uses sender's public key to decrypt the signature as the following equation:

Verification,

$$v = s^e \bmod n$$

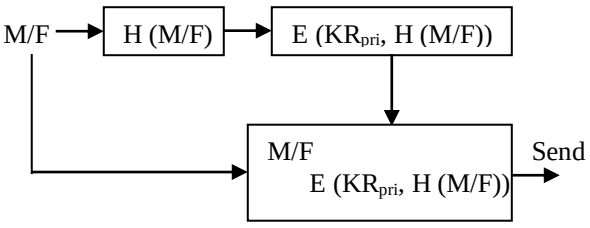
After decryption, the output is the 128 bit digest value and then again computes the digest value original message or file. If both message digests are identical, the signature is valid.

6. Overview of System Architecture

This system operates signing in sender site and verifying in receiver site.

6.1. Creating a Digital Signature (Signing)

In signing process, a hash algorithm is applied to the message or file to create a hash value for that message/file. The result is a hash value. This hash value represents a unique fingerprint for the document. If a cryptographic hash algorithm is used, then it should be impossible to compute another meaningful input message/file that will produce the same digest [7]. The hash value is encrypted using the sender's private key. The resulting encrypted hash value is the digital signature. The digital signature is attached to the message/file to create a digitally signed message or files and sends to the receiver.

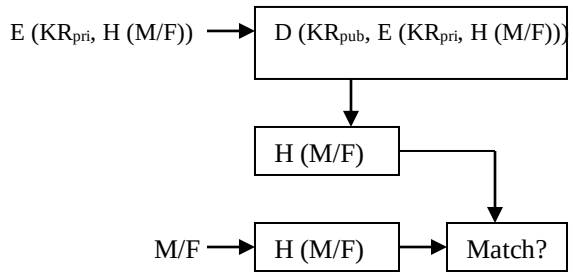


M/F = Message/File
H(M/F) =Hash value of Message/File
KR_{pri} = RSA private key
E = Encrypt

Figure 1. Creating a Digital Signature

6.2. Verifying a Digital Signature (Verifying)

In verification process, incoming input is divided into its components: the digital signature and the message or file itself. The sender's public key is applied to the digital signature. The result is the hash value from the original message/file. The same hash algorithm that was used in the signing process is then applied to the message/file to be verified. The result is the hash value for the signed message or file. The result is either the acceptance or denial of the digital signature, based on the following conclusions.



D=Decrypt
KR_{pub} = RSA public key

Figure 2. Verifying a Digital Signature

6.3. System Flow of Message/File

Figure 3 shows system flow for transmits message/file. First, the system computes hash value of message/file using MD5 hash algorithm. MD5 produces 128-bit fixed length hash value that is small and unique representation of the complete message/file. Second, system encrypts hash value with signer's private key using RSA algorithm. The output is digital signature of the original input. Finally, digital signature is attached to the message/file and the whole data is send to receiver.

Receiver gets the message/file and extracts the digital signature. Then, receiver decrypts the digital signature with signer's public key using RSA algorithm and computes hash value of message/file using the same algorithm using in signing. If the two hash values are matched, the system outputs that message/file was not modified during the data transmission. If the two hash values are not matched, system outputs that message/file was modified after signed and was send by invalid sender.

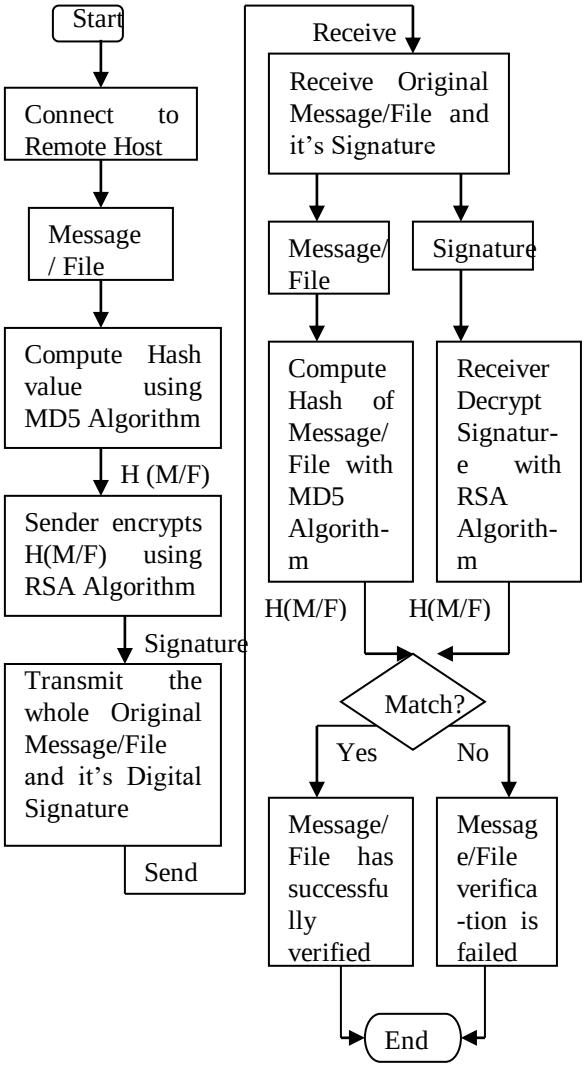


Figure 3. System flow diagram of Message-File

7. System Result

Figure 4 shoes that the message has successfully verified by MD5 algorithm. Receiver receives message and it's signature. Receiver decrypts the signature (encrypted MD5 hash). The output is unique hash value. Then, receiver computes hash value of message to verify data integrity and authentication. This message is not modified during transmission. So, receiver accepts this valid message.

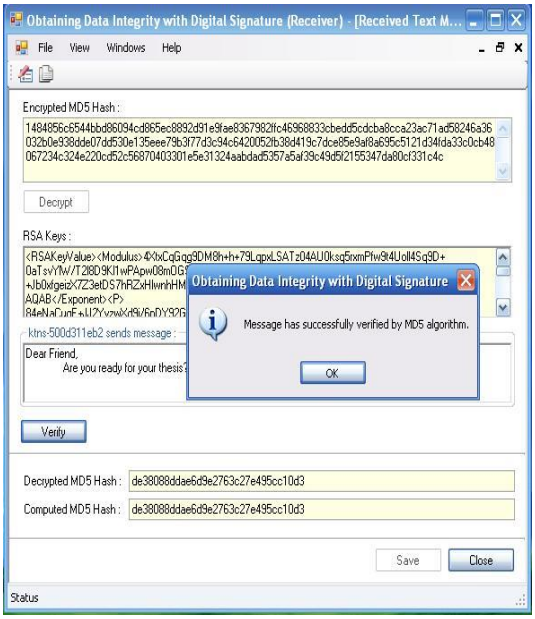


Figure 4. Receive Message with Success Verification

Figure 5 shows that message verification is failed by MD5 algorithm. This message is modified after signed. So, system outputs message verification is failed. Therefore, receiver rejects invalid message.

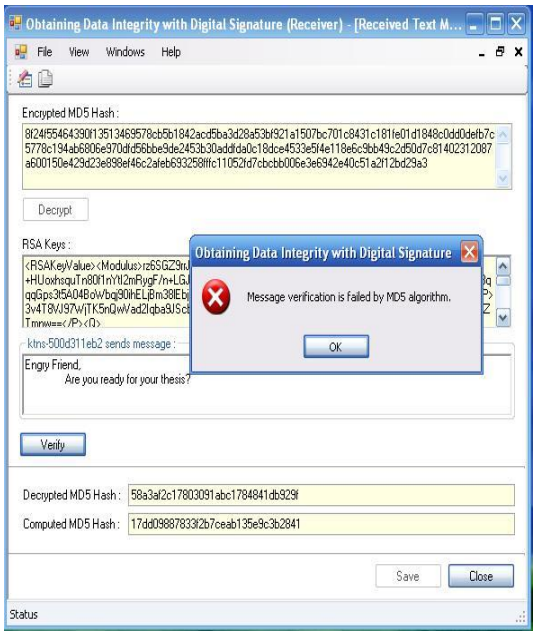


Figure 5. Receive Message with Fail Verification

Figure 6 shows that the file has successfully verified by MD5 hash algorithm. The content of file is not modified after signed and was send valid sender. So, receiver tests data integrity and authentication of file. Receiver accepts that file because system produces that file is valid.

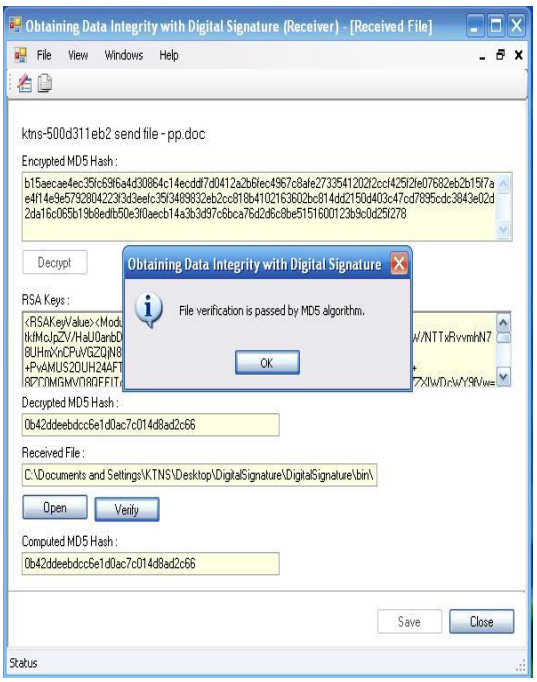


Figure 6. Receive File with Success Verification

Figure 7 shows that the file verification is failed by MD hash algorithm. Receiver decrypts the signature with RSA public key. The output is hash value of file. And, receiver computes hash value original file. Then, receiver compares the two hash values to find out data integrity and authentication of file. System outputs that file is failed in verification. So, receiver rejects that file.

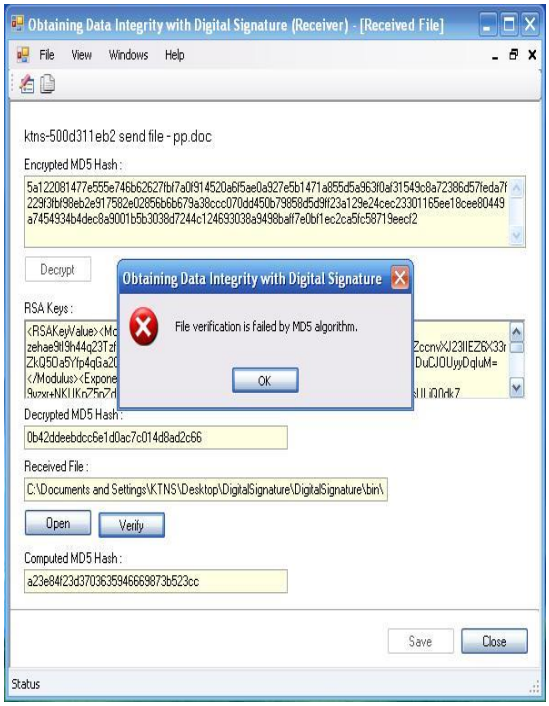


Figure7. Receive File with Fail Verification

8. Conclusion

This paper presents that by working MD5 hash algorithm and RSA algorithm together. MD5 hash algorithm is used to condense the message/file to 128-bits fixed length. This 128-bits hash value is one-way and collision-free. So, it is not able to produce input for given pre specified output. Any slightest change in the original message/file causes the verification process to fail. MD5 hash algorithm secure in digital signature application for guaranteeing integrity of data. RSA is one kind of public key cryptography and used for creating/verifying a digital signature. System produces received message/file is valid or not valid. By matching two hash values, receiver knows that message/file is modified after signed and was send by valid sender. So, system rejects file if two hash values are not matched.

8. REFERENCES

[1] David Aspinall, "Cryptography III: Symmetric Ciphers , Computer Security, Lecture 7" School of Information University of Edinburgh, 2nd February 2009.

[2] Gray C.Kessler, "An Overview of Cryptography", 17 August 2009.

[3] Mark Dermot Pyan, "One-way Secure Hash Functions", University of Brimingham, 2004.

[4] Pacharawit Topark-Ngarm, "RSA Signature with MD5 Hash Function", ECE575 Data Security and Cryptography, Prof. CetinK.Koc.

[5] Pradosh Kumar Mohapatra, "Public Key Cryptography", ACM Student Magazine.

[6] R.Rivest, "MD5 Message-Digest Algorithm", MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[7] Svetlin Nakov, "How Digital Signatures Work: Digitally Signing Messages", October 16,2003.