Journal of Computing and Applied Informatics (JoCAI) Vol. 02, No. 1, 2018 | 1 - 11

talenta PUBLISHER

DATA SCIENCE

Journal of Computing and Applied Informatics

A Framework to Ensure Data Integrity and Safety

Mochammad Azmi Fauzan[1], Eric Paulus[2]

[1,2]Universitas Pandjajaran, Sumedang, Indonesia

Abstract. The technology development allows people to more easily communicate and convey information. The current communication media can facilitate its users to send and receive digital data, such as text, sound or digital image. But in terms of security, communications media not always ensure the confidentiality and authentication of data traffic. Most people rely solely on the security provided by the communications media providers in securing their data, which is essentially still inadequate. This paper presents the development of a data security framework by applying the principles of cryptography and digital signatures, such as authenticity, integrity, and data confidentiality. The application is designed using the SHA-256 algorithm as digital signature, AES algorithm as file encryption, and RSA algorithm as asymmetric key in digital file distribution and signature. Then, several simulation testing was performed to ensure the robustness of the framework. Furthermore, we also evaluated the speed of framework based on CPU and memory capacity. Based on the experiment, our proposed framework can be a reliable solution for securing data in data transaction.

Keyword: cryptography, digital signature, rsa, aes, sha-256.

Abstrak. *Perkembangan teknologi memungkinkan manusia semakin mudah berkomunikasi dan menyampaikan informasi. Media komunikasi saat ini memfasilitasi penggunanya untuk mengirim dan menerima data berupa teks, suara atau gambar. Namun dari sisi keamanan, tidak semua media komunikasi memastikan kerahasiaan dan integritas lalu lintas data. Kebanyakan orang hanya mengandalkan keamanan yang disediakan oleh penyedia media komunikasi dalam mengamankan data, yang pada dasarnya masih kurang. Makalah ini memaparkan pengembangan skema keamanan data dengan menerapkan prinsip-prinsip kriptografi dan tanda tangan digital, seperti keaslian, integritas, dan kerahasiaan data. Aplikasi dirancang menggunakan algoritma SHA-256 sebagai tanda tangan digital, algoritma AES sebagai enkripsi file, dan algoritma RSA sebagai kunci asimetris dalam pendistribusian file dan tanda tangan digital. Pengujian ketahanan dilakukan terhadap skema keamanan dengan menggunakan beberapa simulasi gangguan. Selanjutnya, pengujian skema keamanan berdasarkan waktu juga dilakukan dengan mengevaluasi faktor kapasitas CPU dan memori. Skema keamanan yang diperoleh dapat menjadi solusi yang andal untuk mengamankan data ketika terjadi proses transaksi data.*

Kata Kunci: *kriptografi, tanda tangan digital, rsa, aes, sha-256*

Received 27 December 2017 | Revised 3 January 2018 | Accepted 28 January 2018

*Corresponding author at: Department of Computer Science, Universitas Padjadjaran, Jalan Raya Bandung-Sumedang KM21 Jatinangor, Kampus UNPAD Jatinangor, Sumedang 45363, Indonesia

E-mail address: afauzan92@yahoo.co.id, erick.paulus@unpad.ac.id

ISSN: 2580-6769 | e-ISSN: 2580-829X | DOI: 10.32734/jocai.v2.i1-90
Journal Homepage: https://talenta.usu.ac.id/JoCAI

1. Introduction

The Internet is one medium to exchange information or communicate. Almost everyone uses the internet because it has become a major requirement in their daily life, whether for education, business, entertainment, and others. But along with the rapid development of the Internet, security issues are also increasingly complex. One of the most important elements in security is cryptography[1].

Cryptography, known as secret writing, has classical task to offer confidentiality by encryption algorithm[2]. Cryptography can be grouped into symmetric key methods[3][4], asymmetric key methods[1][5], and message digest methods[6][7]. The difference of the symmetric key and asymmetric key is number of keys[8]. The method of symmetric key uses Single key. But, the method of asymmetric key uses two keys (public and private). Message digest method is used to produce message digest based on a input message. Message digest is the result of a hash computation algorithm, some examples are SHA and MD-5, which are the best known algorithms for message digest. The integrity message is obtained by running the hash function iteratively. If there is any modification on the message, the different message digest is resulted[1][9]. Digital signature scheme is designed using two algorithms, one is using public key cryptographic and the other is hash algorithm[8]. Asymmetric key and symmetric key cannot produce any authentication data but they can make the data more secure. However, public key cryptographic is used to ensure secrecy the distribution of secret keys[2].

Based on the information above, this paper proposed a digital signature scheme. In this scheme, symmetric algorithm AES, a standard of symmetric algorithm provided by NIST, used for the encryption of the main file. We implement AES because it can work fast for minimum hardware and software specification. Therefore, it is well-suited to encrypt the large amounts of data[2]. For the authentication, we implement SHA-256 for the hash algorithm and RSA for the key distribution.

2. Digital Signature and Encryption Algorithm

2.1. Digital Signature

Digital signature is one of the concepts of modern cryptography. The usefulness of digital signatures is similar to signatures in real versions, namely to provide certainty of authenticity and approval of documents by signatories. However, the definition of digital signatures is not the analog signature of the document owner that was inserted by means of a scanned example, but rather to a unique code that indicates the document was late created and sent by the owner of the legitimate document[2].

The principle used in digital signature is that the document sent must be signed by the sender and the signature can be checked by the recipient to verify the authenticity of the submitted

document. Its function is to validate the data sent. Digital signature method uses the hashing algorithm to produce a unique combination of characters called a message digest. In this way, the sender is responsible for the contents of the document and can be checked the authenticity of documents by the recipient.

The uniqueness is that if in the middle of the trip the data is modified, deleted or secretly taped by an irresponsible person even if only 1 character only, then message digest residing on the recipient will be different from that sent at first. Another uniqueness is that the message digest can not be returned to its original form as it was before signature, so it is called a one-way hash.

The way digital signature works as shown in Figure 1 is as follows:

1. The sender performs a hashing process to provide a message digest
2. After the hashing done, Sender signs the message digest by using the public key used to form the digital signature.
3. Then Sender sends the digital signature along with the document to the Receiver.
4. The receiver then verifies the message sent by the sender. In the process of verifying the message in hashing first generate message digest and digital signature will be unsign by using private key. If the message digests the same, then this message is original and the message is from the actual sender and vice versa.

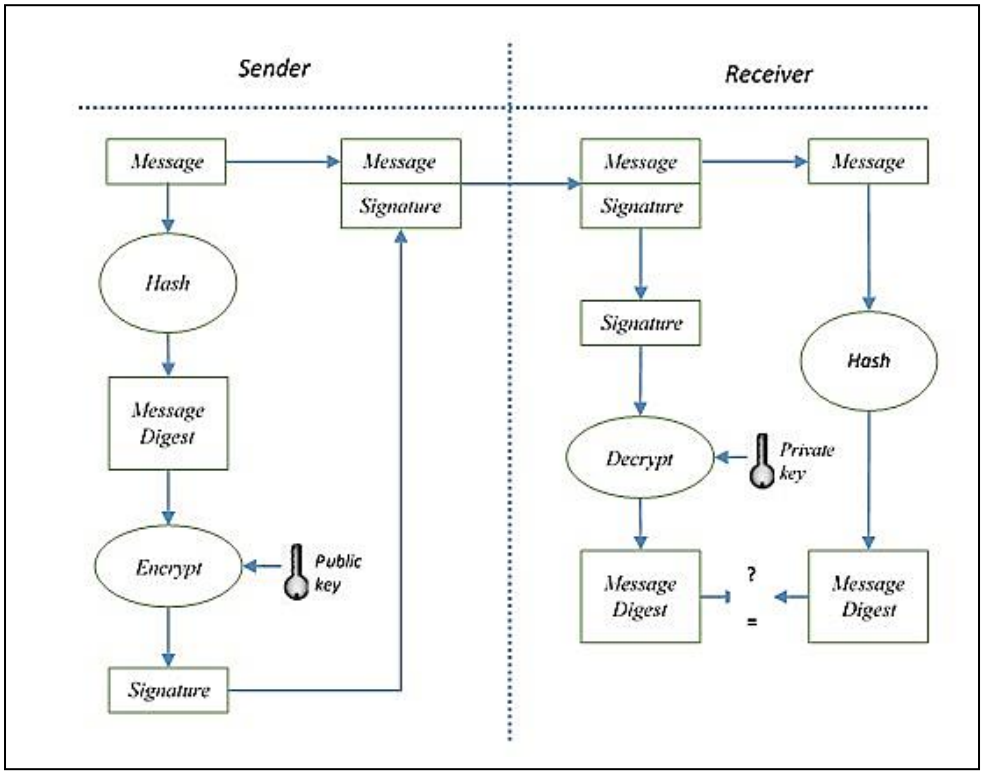


Figure 1. General Concept of Digital Architecture

2.2. Hash Function

The hash function is a function that accepts a string of input whose length is arbitrary and converts the input into a string that has a fixed length and generally becomes smaller than its original length. The output of a hash function is called a hash value or a message digest. The hash function is a one-way function that can generate signatures of data. A single bit change will dramatically change the hash output. The hash function is usually used to ensure integrity and digital signature[2].

The hash function basically works in one direction. It means that the original message will be converted to a message digest. But, the resulting message digest can not be reversed into original message. The sender and receiver have a way so that the integrity of the data can be investigated.

2.3. Secure Hash Algorithm 256 (SHA-256)

SHA-256 was developed by NIST (National Institute of Standards and Technology). SHA-256 can be applied in the use of digital signatures. SHA-256 will produce 256 bits message digest. The message digest length can range between 256 to 512 bits. Secure Hash Algorithm-256 is one of the most common hash types used. This function is a variant of SHA-1, SHA-256 is made because it has been discovered the problem of SHA-1. Until now no one can solve the algorithm for SHA-256. SHA-256 is commonly used as an intermediary function for other functions, including the MAC hash function, HMAC, and some digital signature generating functions[2].

Message digest is a value (value) derived from a data or message that has a unique nature that indicates that the message has a certain quantity. Calculation of message digest using SHA-256 hash on message or data file given as input. SHA-256 converts the input message into a 256 bit digest message [1]. Based on Secure Hash Standard, an input message that is shorter than 264 bits long, must be operated by 512 bits in a group and becomes a 256- bit diggest message.

2.4. RSA Algorithm (Rivest-Shamir-Adleman)

RSA cryptographic algorithm is a public key cryptography method (asymmetric). It was first discovered in 1977 from MIT (Massachusetts Institute of Technology) which is an international standard algorithm and can be applied to digital signature, key exchange, and encryption. The RSA letter itself comes from the initials of their name (Rivest-Shamir-Adleman). RSA was developed in 1978 at MIT which provides key authentication and encryption.

As a public key algorithm, RSA has two keys, namely public key and private key. Public keys may be shared and known to anyone, and used for encryption. While private keys are confidential only certain parties are allowed to know, and used for the decryption process[10]. First, the document is hashed and generates a message digest. Then, the message digest is

encrypted by a public key into a digital signature. RSA algorithm security lies in the difficulty of factoring large numbers into prime factors [11]. Factoring is done to obtain private key. By combining Private Key with Public Key from other people then both parties can share secret message which only known by both parties [8]. Until now RSA is still trusted and widely used on the internet. RSA consists of three processes: key generation algorithm, encryption process, and decryption process.

Three main formulas are used to generate RSA key pair. Equation (1) and (2) are used to calculate parameter e and d, where m = p * q and ø (m) =(p-1)*(q-1). Parameter p and q are large prime number that can be chosen randomly. Based on the calculation result of equation (1) and (2), we can obtain (n,d) as the private key and (e,m) as public key.

$$\gcd(\emptyset(m), e) = 1 \qquad (1)$$

$$d = e^{-1}(mod\ \emptyset\ (n)) \qquad (2)$$

Once the public key is generated, anyone can use the public key. The RSA encryption algorithm uses the exponential function in modular m, as shown in equation (3).

$$c = p^e\ mod\ m \qquad (3)$$

Similarly as encryption, the RSA decryption algorithm is a modular exponential function m using the private key, as shown in equation (4).

$$p = c^d\ mod\ m \qquad (4)$$

A. AES Algorithm (Advanced Encryption Standard)

A contest of encryption system is arranged by NIST (1997) to replace the Data Encryption Standard (DES) with the Advanced Encryption Standard (AES). After some selections, NIST selected the Rijndael encoding system developed by Joan Daemen and Vincent Rijment as the AES encryption system in 2000.

The key expansion process has an initial key input with 4 word length. In the AES algorithm with 128 bit length, the expansion process produces 44 key words by performing several operations for 10 iterations[12]. The operations performed in each iteration are as follows:

1. Rot-Word, is a circular left shift operation on the last word of a state.
2. Sub-Word, substituting each word element with S-Box.
3. Word results from sub-word operations performed XOR operations with round-constant.

In this scheme, AES is used in main file encryption. AES encryption algorithm with 10 rounds using 4 transformations, Sub-Bytes, Shift-Row, Mix-Column and AddRound-Key and Add-Round-Key transformation done before the 10th encryption process. In round 1 - 9, the AES

encryption algorithm performs all the transformations in sequence while in round 10 the transformations are SubBytes, Shift-Row and Add-Round- Key.

AES decryption algorithm with 10 rounds using 4 transformations, Inverse Shift-Row, Inverse Sub-Byte, Inverse Mix-Column and Add-Round-Key and Add-Round-Key transformation done before decryption process. In round 1 - 9, the AES decryption algorithm performs all the transformations while in round 10 the transformations are Inverse Shift-Row, Inverse Sub-Byte and Add-Round-Key.

3. Methodology

We proposed a data security framework using combination digital signature method and encryption method. There are two roles in the framework that is as sender and receiver. Figure 2 shows the process of this framework.

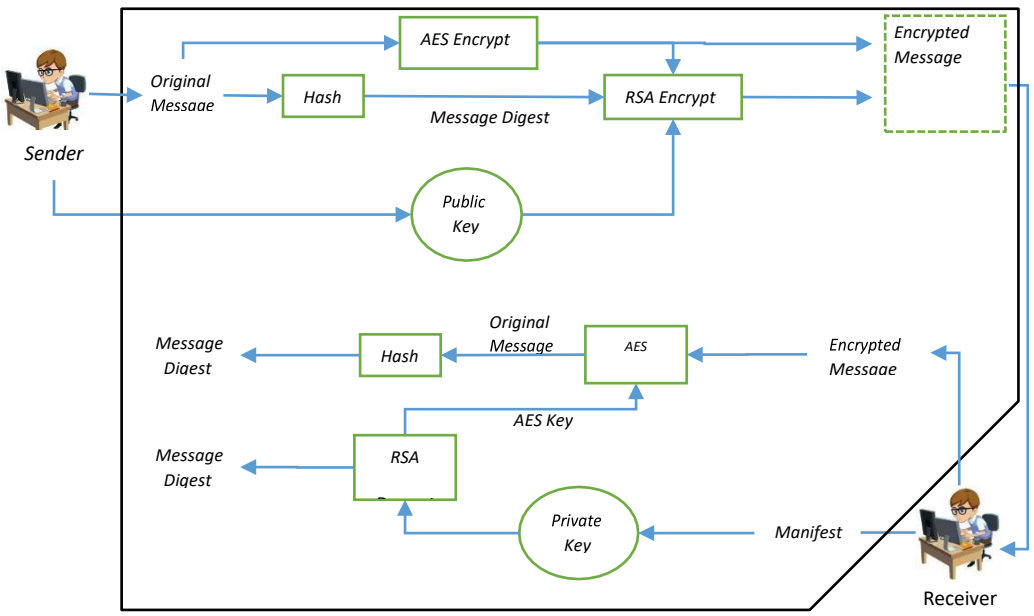


Figure 2. Digital Signature and Cryptographer Framework

Digital signature and encryption process by the sender:

1. The sender must have the pairs of keys.
2. The message is hashed to generate a message digest.
3. The result of message digest or digital signature will be encrypted using private key using RSA algorithm, while the main file is encrypted using AES. Information from the whole encryption result will be summarized in the manifest file, where there are also AES keys for the main file decryption process. Manifest files are encrypted using RSA.
4. Then the result of the digital signature and the manifest file is placed with an encrypted message, then both can be sent.

Digital signature and encryption process by the recipient:

1. Messages and signatures are decrypted first.
2. The signature is decrypted using the sender's private key, generating the original message digest. While the main message is decrypted using information from the manifest file.
3. Rehashed the message to get message digest.
4. Next, the message digest will be compared. If the message digest is the same with the signature then the authentication valid and the received signature are from the correct sender.

4. Experiments and Results

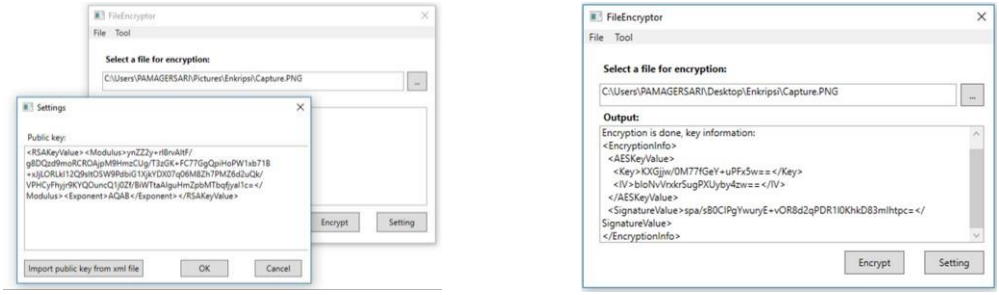
This research used some image samples with different size and type, shown in Table 1. The image samples are obtained from philologist in Sundanese Department, Universitas Padjadjaran that contain Sundanese manuscript [13]. There were two experiments to evaluate the performance of digital signature and cryptography framework. First, we changed the main file, primary key file, signature or manifest file. Then we used those files to evaluate the robustness of the framework. Second, we investigated the computation speed of the framework based on various hardware specifications.

Table 1. The size of image samples

File	Name	Type	Size (B)
F1	3306_GT1.bmp	image	123,942
F2	dokumen1.rtf	Text	123,975
F3	3309.tif	Image	3,199,368
F4	dokumen2.docx	Text	3,199,488
F5	3-19-30-2.nef	Image	18,267,794
F6	3-19-90-12.tif	Image	48,690,172

4.1. Encryption and Decryption Simulation

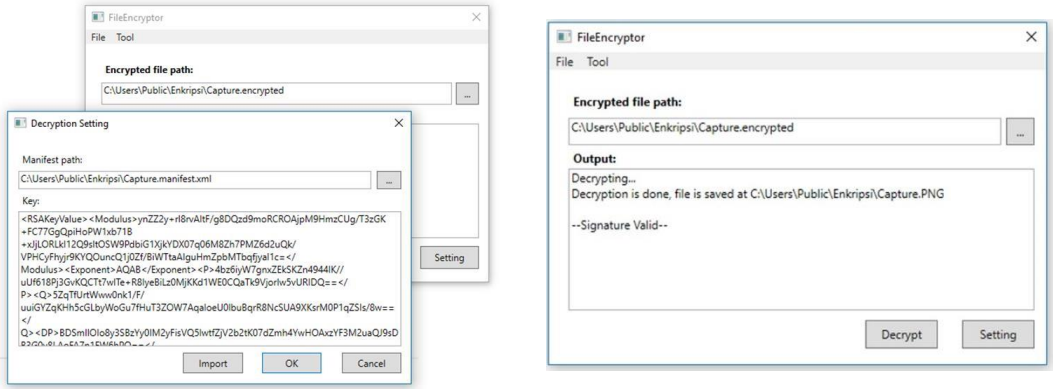
The first step is to generate public key and private key. Then, we input the public key and the original file. The key information and the signature value can be viewed after the encryption process succeeds. This process can be seen in Figure 3. There are two outputs of the encryption process, i.e. filename.encrypted and filename.manifest. The encrypted file cannot be opened by the regular way. Those two files can be sent separately to the receiver.



(a) (b)

Figure 3. Sample of Encryption Interface: (a) Input File and Public Key Setting, (b) Encryption Info Results

After the file is received, the recipient decrypts the file using the available data. In the process, the application system decrypts the first manifest file using the private key to get the AES key values, the signature file, and the actual file extension. By using AES key information, then the system can reverse the file back to its original form, shown in Figure 4. Then we can check the authentication of the original file.



(a) (b)

Figure 4. Sample of Decryption Interface: (a) Input Encrypted File and manifest File and Private Key setting, (b) Decryption Info Results

4.2. Attacks Simulation

We assumed that an attacker sends an unauthorized attachment file with an unauthorized encryption file, replacement of a public key, or a replacement in the signature. It is also assumed that the attacker does not damage the private keys, but only replaces the manifest file with a fake one.

Table 2. Attacks Simulation

Simulation Number	Parameters				Output Result	
	Original File Authenticity	Valid Signature	Valid Private Key	Valid AES Key (Manifest)	Decryption	Authentication
1	√	√	x	√	x	x
2	√	√	√	x	x	x
3	√	x	√	√	√	x
4	x	√	√	√	x	x

Table 2 shows which parameter is being attacked or modified. Symbol √ can be meant as authentic, valid, or succeed. Symbol x can be meant as inauthentic, invalid or failure. From those results, any change of parameters makes the authentication fails and the decryption cannot be done. In the simulation 3, the decryption process can succeed because both the key and the file are not changed, but failed in through the authentication process.

4.3. Speed and Memory Usage Analysis

In this section, the decryption and encryption process perform on different devices. The purpose of this experiment is to analyze the effect of memory and CPU capacity on the performance of our proposed framework. The specifications of each device are described in the Table 3. The results of the speed comparison of each device are shown in the Table 4. The symbolic meaning of F1-E and F1-D are F1 (file 1), E (encryption time), and D (decryption time). The speed comparison is measured in seconds.

Table 3. Devices Specification for Experiment

Device	Memory Size (GB)	Processor Size
1	4	AMD APU(TM) A6-5200 @ 2.00GHz (4CPUs)
2	4	AMD Phenom(TM) II X2 550 @ 3.10GHz (2CPUs)
3	4	Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz (2 CPUs)
4	8	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz (2 CPUs)
5	8	Intel(R) Core(TM) i7-3537U CPU @ 2.00GHz (4 CPUs)

Table 4. The Speed Comparison on Encryption and Decryption Process

Device	F1-E	F1-D	F2-E	F2-D	F3-E	F3-D	F4-E	F4-D	F5-E	F5-D	F6-E	F6-D
1	0.025	0.009	0.027	0.026	0.112	0.087	0.098	0.091	0.466	0.462	1.332	1.223
2	0.021	0.016	0.023	0.170	0.275	0.266	0.276	0.262	1.405	1.349	3.753	3.641
3	0,040	0,015	0,137	0,147	0,176	0,757	0,138	0,289	0,622	0,601	1,488	2,644
4	0,034	0,024	0,017	0,003	0,180	0,056	0,058	0,052	0,314	0,284	0.885	0.743
5	0.016	0.008	0.015	0.020	0.067	0.055	0.066	0.056	0.313	0.282	0.819	0.739

Based on the analysis of Table 4, there is a time gap between decryption and encryption. It can be occurred due to the RSA key decryption process to get AES keys in the manifest file. The differences capacity of memory and processor are also very influential in the process of

decryption and encryption. There is a considerable time interval between each device. The file size also greatly affects the length of processing time. Device 5 shows that though the size of F6 (48,690,172 Byte) is about 392 times from F1 (123,942 Byte). But the encryption time ratio of F6-E (0.819 Sec) and F1-E (0.016 Sec) is about 51 times. We can conclude that the time consumption is not linear with the file size. The file type does not significantly affect algorithm speed. F1-D (image type) and F2-D (text type) present that the working time is almost the same. Device 4 and 5 show that the amount of CPU core makes the speed of computation a little bit faster, though the frequency of device 5 (2.00GHz) is lower than device 4 (2.30GHz). Device 3 and 4 present that bigger memory capacity can result in faster computing time.

5. Conclusion

We have built the digital signature and cryptography framework using SHA-256 as a hashing process, AES as symmetric key, and RSA as the distribution of encryption keys. Based on the experiment, encrypted file cannot be read or opened except by the corresponding way and the right keys. It means the message confidentiality and the security of the file in the delivery process can be guaranteed. The application system is very sensitive to the slightest changes, either on the main file or the parameter file. It causes the process of decryption cannot be done. Encryption and decryption can still be done if only the signature is changed, though the main file and the keys are unchanged. But in the authentication process it shows that the signature provided does not match to the original. Furthermore, the speed of the process is influenced by several aspects, such as the file size and the device specifications.

REFERENCES

[1] P. Patil, P. Narayankar, N. D.G., and M. S.M., "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," *Procedia Comput. Sci.*, vol. 78, no. Supplement C, pp. 617–624, 2016.

[2] H. Delfs and H. Knebl, *Information Security and Cryptography: Principles and Applications*, Third. Verlag: Springer, 2015.

[3] S. Heron, "Advanced Encryption Standard (AES)," *Netw. Secur.*, vol. 2009, no. 12, pp. 8–12, 2009.

[4] U. Farooq and M. F. Aslam, "Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 29, no. 3, pp. 295–302, 2017.

[5] P. Patel, R. Patel, and N. Patel, "Integrated ECC and Blowfish for Smartphone Security," *Procedia Comput. Sci.*, vol. 78, no. Supplement C, pp. 210–216, 2016.

[6] J. Ó. Ruanaidh, H. Petersen, A. Herrigel, S. Pereira, and T. Pun, "Cryptographic copyright protection for digital images based on watermarking techniques," *Theor. Comput. Sci.*, vol. 226, no. 1, pp. 117–142, 1999.

[7] N. R. Chandran and E. M. Manuel, "Performance Analysis of Modified SHA-3," *Procedia Technol.*, vol. 24, no. Supplement C, pp. 904–910, 2016.

[8] K. V Pradeep and V. Vijayakumar, "Survey on the Key Management for Securing the Cloud," *Procedia Comput. Sci.*, vol. 50, no. Supplement C, pp. 115–121, 2015.

[9] Q. H. Dang, "Secure Hash Standard (SHS)," Federal Inf. Process. Stds. (NIST FIPS) 180-4, 2015.

[10] Sankalp Prakash and Mridula Purohit, "An Efficient implementation of PKI architecture based Digital Signature using RSA and various hash functions (MD5 and SHA variants)," *IOSR J. Comput. Eng.* , vol. 15, no. 6, pp. 27–33, 2013.

[11] V. R. Pallipamu, T. R. K, and S. V. P, "Design of RSA Digital Signature Scheme Using A Novel Cryptographic Hash Algorithm," vol. 4, no. 6, pp. 609–613, 2014.

[12] J. Nechvatal *et al.*, "Report on the development of the Advanced Encryption Standard (AES)," *J. Res. Natl. Inst. Stand. Technol.*, vol. 106, no. 3, p. 511, 2001.

[13] E. Paulus, M. Suryani, and S. Hadi, "Improved Line Segmentation Framework for Sundanese Old Manuscripts," in *the 2nd International Conference on Computing and Applied Informatics*, 2017.