Indonesian Journal of Electrical Engineering and Computer Science
Vol. 21, No. 3, March 2021, pp. 1739~1750
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v21.i3.pp1739-1750

Enhancement in data security and integrity using minhash technique

Sa'ed Abed, Lamis Waleed, Ghadeer Aldamkhi, Khaled Hadi
Department of Computer Engineering, Kuwait University, Safat 13060, Kuwait

Article Info

Article history:

Received May 16, 2020
Revised Nov10, 2020
Accepted Dec 11, 2020

Keywords:

Advanced Encryption Standard
Cryptography
Data integrity
k-shingle technique
MinHash
Rivest-Shamir-Adleman

ABSTRACT

Data encryption process and key generation techniques protect sensitive data against any various attacks. This paper focuses on generating secured cipher keys to raise the level of security and the speed of the data integrity checking by using the MinHash function. The methodology is based on applying the cryptographic algorithms rivest-shamir-adleman (RSA) and advanced encryption standard (AES) to generate the cipher keys. These keys are used in the encryption/decryption process by utilizing the Pearson Hash and the MinHash techniques. The data is divided into shingles that are used in the Hash function to generate integers and in the MinHash function to generate the public and the private keys. MinHash technique is used to check the data integrity by comparing the sender's and the receiver's encrypted digest. The experimental results show that the RSA and AES algorithms based on the MinHash function have less encryption time compared to the normal hash functions by 17.35% and 43.93%, respectively. The data integrity between two large sets is improved by 100% against the original algorithm in terms of completion time, and 77% for small/medium data and 100% for large set data in terms of memory utilization.

Corresponding Author:

Sa'ed Abed
Department of Computer Engineering
Kuwait University, Safat 13060, Kuwait
Email: s.abed@ku.edu.kw

1. INTRODUCTION

Ciphers are used for performing encryption and decryption operations in various devices to ensure that any exchanged data and information between connected devices are well secured. There are two types of cipher algorithms: asymmetric and symmetric [1]. Asymmetric cipher algorithms use two keys (public and private keys) whereas symmetric algorithms use only one key which is a shared private key. Typically, asymmetric ciphers are more complex and secure than symmetric ones [2, 3]. Rivest-shamir-adleman (RSA) and elliptic-curve-cryptography (ECC) are examples of asymmetric ciphers [4] while data encryption standard (DES) and advanced encryption standard (AES) are common examples of symmetric ciphers [5]. Symmetric ciphers are used in resource-constrained devices (RCDs) because they provide privacy for the devices in addition to their high performance.

Cybersecurity is the practice of protecting networks and systems from digital attacks. These attacks can be either accessing, changing, or destroying the stored information. Using encryption algorithms, which are based on performing some mathematical processes to procedures to encrypt or hide data, a ciphertext is created, and a secured key is generated in order to get the original data. This brings the concept of Cryptography, which is the process of converting the plaintext, where the data needs to be transformed into an encrypted text and vice-versa. It constructs and analyses protocols to prevent any third parties, known as

adversaries, to interfere with the plaintext by eavesdropping. Cryptography [6] stores and transmits the data in a secured communication to allow the intended users only to read and process it. Cryptography is used in many daily applications that require a high level of security such as computer passwords, different types of transactions whether it is a banking transaction or e-commerce [7]. The process of hiding the information is called encryption, and when the information is unhidden for the end-user, it is called decryption. This process is accomplished by using specific algorithms called cryptographic algorithms. The plain text will be the input for these algorithms to encrypt and decrypt using another key input used to generate the ciphertext, which is the plain text after encrypting. The ciphertext is decrypted also by using a key at the receiver side.

The only difference in the three types of the cryptographic algorithms symmetric, asymmetric, and the hash function is the number of cipher keys that are used in the encryption/decryption process. This paper will consider the asymmetric algorithms only to generate the ciphertext and to decipher it. These algorithms need to satisfy some security goals like confidentiality, data integrity [8], authentication, and nonrepudiation to protect the data in a large organization.

The main issue that any organization might face is finding the best way to protect sensitive data and to ensure that no threats such as unauthorized access and use, misappropriation, destruction, and alteration happen within the organization. There are many efficient techniques to maintain the data in the transmission process and to ensure that the receiver receives it correctly. However, these techniques suffer from the delay due to the many stages that it goes through in order to have a successful transmission process. This delay was somehow enhanced by using the Hash functions [9] to generate the keys, but it did not have the ability to generate enough amount of keys to be used by the sender and the receiver, especially if the data size is large. Another issue was to inspect if any misbehavior has occurred by any security threat while sending the data from the sender to the receiver. This process of checking the correctness of the data, which is known as Data Integrity, costs additional delay to the transmission process. The existing data integrity checking techniques uses the hash functions, creates an encrypted value known as the digest at both the end-users, and then compares these values after the receiver gets the data [7].

In this work, different techniques are used in various phases to provide more data security and maintain data integrity for our proposed solution. AES and RSA algorithms are both used to encrypt and decrypt both the plaintext and digest after hashing the plaintext itself using the MinHash and Pearson Hashing. Encrypting the plain text and decrypting the ciphertext is to ensure security as usual. On the other hand, encrypting and decrypting the digest of hashing the plaintext is to ensure that the data has been not changed while receiving at the receiver side. Also, it will enhance the checking functionality of large data to be received as the original text. The MinHash function in this work is mainly used for encryption and comparison purposes. Our methodology consists of three phases as shown in Figure 1.

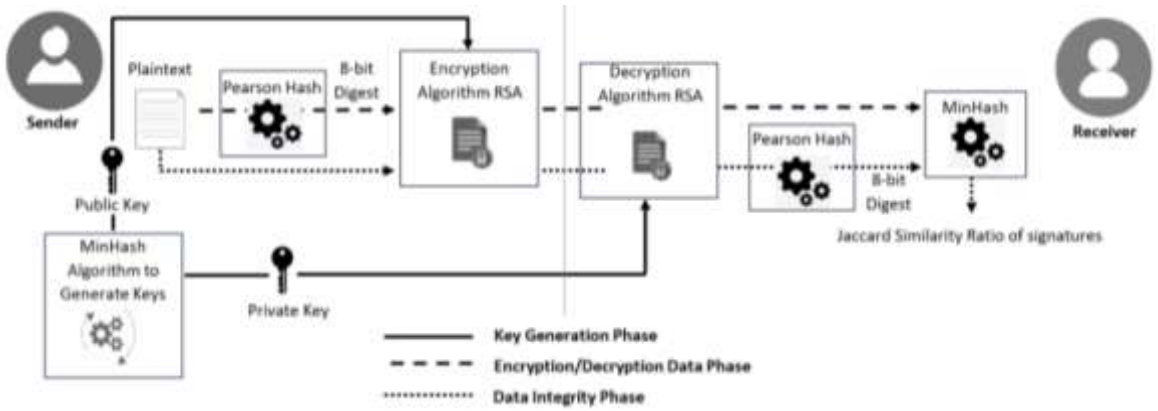

Figure 1. High level architecture for the proposed solution

Phase 1 generates different public-private pairs using the MinHash technique. Phase 2 encrypts/decrypts data and digests after hashing the plaintext using an encryption algorithm. The final phase checks the integrity of data using the MinHash similarity technique. This paper overcomes the delay in both the key generation phase and the data integrity phase by using a new form of the Hash function known as the MinHash Function. The MinHash function is considered to be a fast technique to estimate the similarity of two sets. It is used in the transmission process to check the data integrity without adding delay, which can slow the overall process, and providing accurate results to know if the data has been altered or not. The

MinHash technique is also used along with the Pearson Hash function to generate a small digest size of 8-bit. This will increase the level of security by having two-level hashing. The results after comparing the original RSA [10] and AES algorithms and the modified ones by using the MinHash technique showed that the MinHash technique improved the Encryption time whether the size of the plain text is small or large. It also improved the security level as well by using a two-level hashing algorithm which is the Hashing algorithm and the MinHash algorithm.

The results showed that the key generation process using the Hash along with the MinHash techniques is faster than using only the hash functions due to the two-level hashing. Moreover, it was proven that the Encryption time is faster using the MinHash function for AES by 17.35% and for RSA by 43.93% for both dynamic and static plain text size. The MinHash technique also generates more keys depending on the number of produced shingles. The data integrity checking process was accurate and fast by more than 100% for large sets compared with the original data integrity technique align with the speed, it saves capacity and enhances the memory usage by 100%.

The concept of maintaining privacy and security is addressed in various works using the MinHash technique. Related work can be analyzed into two major concepts: MinHash algorithm in different applications and Cryptographic algorithms from security aspects [1]. In [2], cryptographic algorithms such as DES, RSA, and AES were stated in terms of their performance and the cost evaluation. The evaluation attributes were the encryption time, the number of encoding bits, the Avalanche effect, and the memory space used. The key generation process was done by the "KeyGenerator" that is built-in in the packages of Java Security and Java Crypto. It was concluded that the DES algorithm is the best in terms of the bandwidth, AES for the cryptographic strength, and finally the BlowFish algorithm in terms of encryption time and memory.

Different evaluation parameters were discussed in [3] [4], that can be used to compare different encryption algorithms to choose the best data encryption algorithm to overcome the problem of security-related issues like authentication and integrity. The analysis shows the effectiveness of each algorithm in terms of the time it takes in the encryption/decryption process, speed, and throughput.

Krishna et al. [5] proposed a cipher method that encrypts two or multiple messages by using a pairing function at a time. The key is embedded by the transposition ciphers within the encrypted text. A de-pairing function is used to decrypt the encrypted message. The encryption/decryption process includes performing XOR operation, generating keys, and applying the algorithm to generate the cipher text.

A verification hash function (VERIHASH) was proposed in [11] where it ensures secure computing and identity authorization that affect the security and correctness. The main aspects of the VERIHASH are semantic modeling, analyzing all the operations that are done on huge sets of arrays and bits and supporting large-scale analysis via compositional verification. This technique works by splitting a cryptographic hash function implementation into multiple components are based on some variables. Finally, in order to get the final result, it merges all the obtained verification results of all the different components compositionally.

The partitioning of huge binary program collections that contain duplicates and near-duplicated using clustering algorithms based on MinHash function properties and the structure was discussed in [12]. It partitions the collections into smaller groups of similar elements. This is done when computing the disparity of two elements in the group by using the distance function. Boolean features from the binary program are extracted to compute the Jaccard distance between the sets. Using the MinHash functions helps to cognizance the probability of two sets having the same MinHash value similar to their Jaccard distance value. Then, applying the pairwise distances between the elements in the same partition.

A MinHash-based Jaccard similarity computation (MH-JSC) mechanism was proposed in [13] to overcome the privacy issues in the Jaccard similarity computation. The MinHash function satisfies the differential privacy definition regarding the set operations. Then, the strict differential privacy in MH-JSC is achieved by proposing the PrivMin algorithm. The algorithm consists of: Generating the Private MinHash value and MinHashing randomization. The results discovered that the proposed algorithm can assure privacy while computing similarity.

Orencik et al. [14] utilized the Minhash functions to propose an efficient privacy-preserving multi-keyword search method through encrypted cloud data. The uses of MinHash function in the proposed mechanism regarding the searching process was to ensure the efficient similarity between signatures of documents and queries to be compared. Also, the tf-idf approach has been used to prevent unnecessary communication and computation load on the user to maintain the security wise.

In [15], a MinHash algorithm used to search a fuzzy keyword was discussed to improve the efficiency of the ciphertext retrieval and lowering storage. In addition to minimizing the complexity, MinHash fingerprints have been used to avoid the construction of the fuzzy keyword search. The fuzzy keyword search process helps in reducing keyword index space storage, so it has been applied over several servers and users that have both properties of preserving the security and privacy of data. That led to retrieve

and prove the security of the keyword have when generated the keyword fingerprint improved efficiency and accuracy.

On the other hand, a one-way hash function algorithm was proposed in [16] to ensure data integrity, message authentication, and a digital signature of the security within a minimum delay of security called OSHA. The idea of the OSHA algorithm is to extend the number of bits to be used such as a variable of 16 bits of the message to be generated has been extended to eleven chaining which is more than SHA-1 and SHA-2 by using padding bits, dividing into blocks, and chaining variables process. As a result, the OSHA algorithm is more time efficient and faster than SHA-1 and SHA-2.

A different research related to implementing reconfigurable FPGA hardware components that enable deploying cryptographic algorithms was proposed in [17]. The purpose is to easily configure algorithms that use encryption and hashing solutions to allow data protection in various scenarios. This technique utilizes only a small part of an FPGA chip, which can be readily integrated with other processing needs.

In [18], the authors proposed an algorithm that used to get the near duplicate of documents in which these documents are similar to small differences called CentralMatch algorithm. Then, they compared between CentralMatch algorithm with the MinHash algorithm for finding the similarity in terms of short time execution and accuracy. The MinHashing is to find the similarity between two documents regardless of the location. Their proposed solution found that the CentralMatch is better performance and accuracy than MinHashing.

Chauhan et al. [19] proposed a fastest search technique of text documents by dividing the documents into shingles using k-shingle and finding the similarity between sets using Jaccard similarity. Then, the authors used Bloom Filtering to reduce the search time of the hashing shingles. As a result of their experiments, they used a new fast technique to find the similarity between signatures which is small size than the hashing values of shingles locality sensitive hashing (LSH). An implementation of signatures detection according to user-agent abnormality through malware HTTP traffic by using a systematic method was proposed in [20].

In [21], Ioffe proposed a new novel method that figures the drawing similarity between two inputs. He compared the proposed similarity method and the Jaccard similarity technique to find the probability of drawing the similarity of inputs. The other comparison came with MinHashing that divides the inputs into sets to find the identical drawing. The proposed solution reduces the running time of hashing which produces a smaller size of strings hashed using MinHash to give a more accurate estimation for identical drawing inputs.

In [22], recent robust encryption filed approach using the principles of MinHash technique and K-shingle was proposed to overcome the weaknesses in generating the cipher key for the cryptographic algorithms. This approach generates the block keys using the K-Shingle used in the MinHash technique to convert the text file into a sequence of consecutive words. The MinHash technique uses many hash functions to generate the cipher keys and then to encrypt the text files in many cryptographic algorithms like DES, Triple DES, AES, and Blowfish. The analysis of this approach shows that AES and Blowfish algorithms are the best in terms of throughput, CPU usage, and encryption time and memory space.

The work in [22] is the most related work to ours since it discusses the MinHash technique to encrypt files. The main aim of the research was to compare some types of cryptographic algorithms regarding the encryption time, throughput, memory used, and the avalanche effect. The common between our paper and [22] is that the keys are generated using a Hash function and the MinHash technique. However, our work uses another type of hash function called the Pearson function, which generates an 8-bit digest faster. The main drawback of [22] is not clearly stating the importance of the MinHash technique and its contribution in enhancing the performance of encrypting the files. Moreover, the steps of the encryption/decryption process are the same as the existing solutions and this work does not include any progress to check for the data integrity.

Our work, however, focuses on how to generate keys using the MinHash technique that creates an 8-bit digest with the help of Pearson Hash function and then encrypts it. This technique can create large amounts of keys that can be used in the transmission process later. MinHash function is also used in the proposed solution to check the data integrity since this digest will also be created at the receiver side, these two digests are then used to compare between them and check if any changes have occurred. The proposed procedure expedites the key generation and the data integrity process since it uses the Pearson Hash function and the MinHash function which are both a fast execution function.

As a conclusion, there are many applications where the MinHash technique has been used. The main scope of this paper is to focus on the MinHash technique in cryptographic algorithms to ensure transmitting the data without any changes and to speed up the comparison process to find the integrity of transmitting the data with the help of Pearson Hash function.

2. RESEARCH METHOD

In this paper, we propose an approach to enhance the security wise and speed up the checking criteria of the data transmission in a secured manner to ensure data integrity. The proposed solution consists of three main phases as shown in Figure 2 the key generation phase, encryption/decryption phase, and the data integrity checking phase

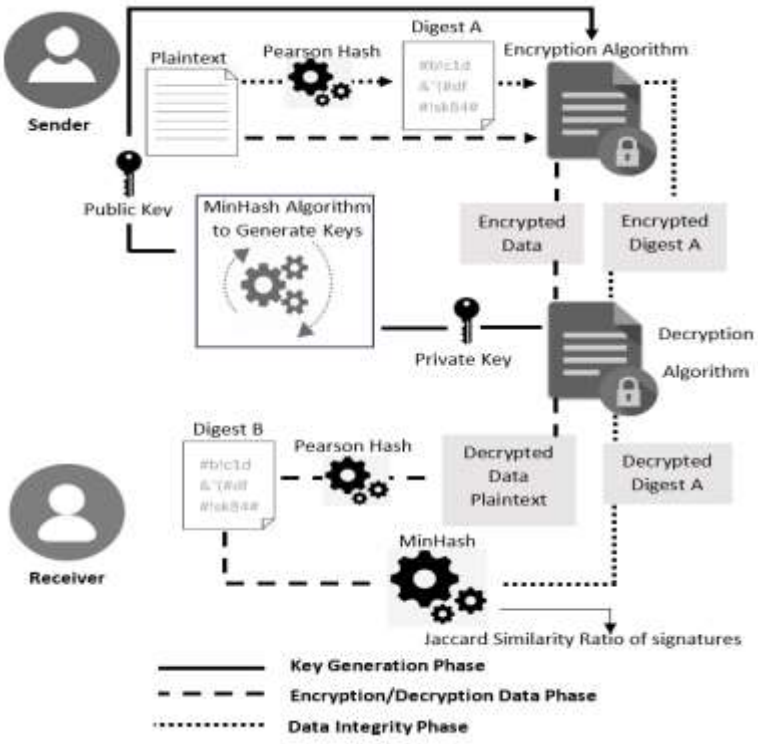

Figure 2. Detailed proposed methodology

Here are the steps for the proposed solution:

Phase 1: Generate different public-private keys for the sender and receiver using the k-shingle and MinHash technique as shown in Figure 3.

Phase 2: The Encryption/Decryption phase includes the following steps:

Step 1: Encrypt the plaintext using the RSA encryption algorithm with the public key as shown in Figure 4. At the same type, hash the plaintext using Pearson Hash function to generate 8-bit digest (Digest A) as shown in Figure 5. Then, encrypt the digest using the same RSA encryption algorithm with the public key.

Step 2: Decrypt both the digest and encrypted data using the private key from the receiver side as shown in Figure 4.

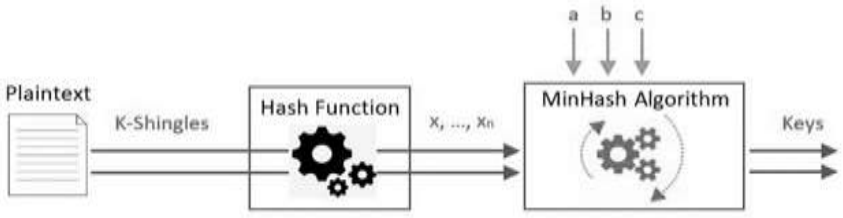

Figure 3. Key generation phase

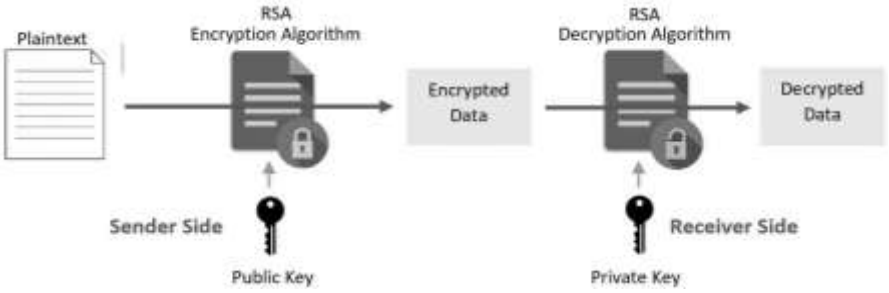

Figure 4. Encryption/Decryption data phase

Step 3: Hashing the decrypted data using the Pearson Hash function again to generate an 8-bit digest (Digest B) as shown in Figure 5.

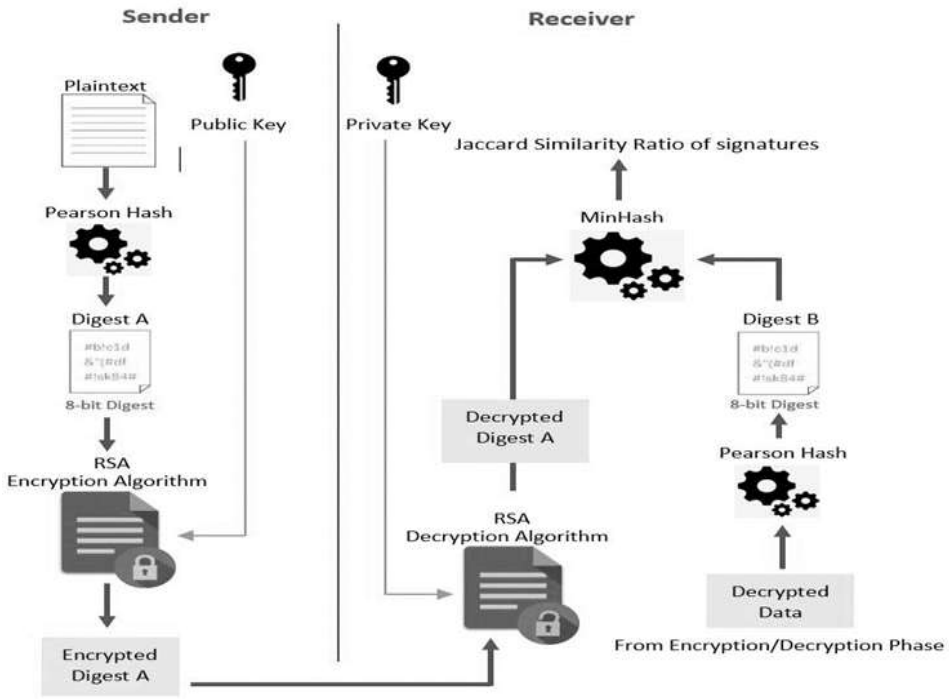

Figure 5. Data Integrity phase

Phase 3: To find the similarity between Digest B and Digest, both digests are divided using the k-shingle technique and then input to MinHash. The Signature matrix is next computed to reduce the execution time.

After that, the Jaccard similarity technique is used to obtain the similarity ratio of digest A signatures and digest B signatures as shown in Figure 5.

2.1. Key generation phase

This phase includes two different steps to generate a secure pair of keys for the sender and the receiver.

2.1.1. K- shingle

The first step uses the shingling process which splitting the input data into substrings depends on the length of K that is assumed to be a certain value from the beginning of the process, and then converting these substrings into integers. The number of shingles is equal to (n-K+1), where n is the number of words in the input data, K is the shingle length that set from the beginning.

For example; if the input data is "This is a plain text", and the value of K=4, then the number of generates shingles will be (5-4+1=2). This means that the generated shingles after removing the punctuation and the double spaces will be of length 2 and will be the following: "This is a plain", "is a plain text". Each of these shingles will be applied to a Hash function to generate different integers. The minimum result of the Hash function, which is known as the value "x" will be then applied to the MinHash to generate the keys.

2.1.2. MinHash function

MinHash function takes the value x as an input and applies it to (1) as shown in Figure 3:

$$\boldsymbol{h(x) = (ax + b)\% c} \quad (1)$$

The coefficients a and b are randomly chosen integers less than the value of x. c is a prime number that is slightly bigger than the value of x. The hash function will generate different values when the values of a & b are changed. For example, if we want to have 4 different keys, we have to generate 4 different hash functions with different values for a and b in order to use these keys in the cipher process later [23].

2.2. Encryption/decryption data phase

The transmission process of the data between the sender and the receiver needs to be secured. RSA encryption algorithm is used for the data encryption to ensure the data transit in a secure manner. Therefore, after generating the public and the private keys for the sender and the receiver from the previous phase, the generated keys will be used like the following to encrypt/decrypt data as shown in Figure 4:

a) The public key of the sender is used along with the plain text as an input to the RSA Algorithm, the result of the encryption algorithm is an encrypted data.
b) After receiving the encrypted data, the receiver decrypts it by using the sender's private key that is only known to the sender and to the receiver.

2.3. Data integrity

The transmission process of the data between the sender and the receiver needs to meet Data Integrity criteria, which is the process of guaranteeing that there is no change happened to the data from the sender and the receiver side. The RSA encryption algorithm used in this work does not guarantee Data Integrity and may take a long time to check the integrity of large data transmission. Hence, the data integrity process runs various steps as shown in Figure 5.

2.3.1. Hashing plaintext

At this stage, the Pearson Hash function is used to receive the digest from both the sender and receiver side; At the sender side, the original plaintext will be hashed to get Digest A with 8-bit. On the receiver side, the decrypted data will be hashed using the same type of hash function to get Digest B with 8-bit.

Person Hashing [24] is used to execute any input size of plaintext efficiently and generate an 8-bit digest even for a larger size. This will help to speed up the checking procedure when comparing two sets to find the similarity with 8-bit of digests using the MinHash function. The duplication of hashing the plain text with Pearson and MinHash function improves security.

2.3.2. Encryption/decryption digest

At this stage, we use the RSA algorithm to increase the security level of transferring the digest in parallel of securing the data transmission. To encrypt/decrypt the digest (Digest A), the same generated public and private keys that have been used for the data encryption process will be used as the following:

a) The public key of the sender is used along with the digest as an input to the Encryption Algorithm (RSA), the result of the encryption algorithm is an encrypted digest (*Encrypted Digest A*).
b) After the receiver receives the encrypted digest, it will decrypt by using the receiver's private key that is only known to the sender and to the receiver (*Decrypted Digest A*).

2.3.3. Digest comparison

At this stage, we explain the procedure used to determine the similarity between two digests using MinHash. The digests need to be represented as sets. First, the k-shingle of consecutive characters is used to divide the digest. In our work, we assume k = 2, so we obtain four shingles of 2-shingle for the digest. Second, obtain a big set containing all shingles without any duplication. In the third step, the flag matrix should be created that represents the digests as columns and the elements of the two digests combined as rows. A cell is given the value 1 when a shingle of the digest is part of the big set, 0 otherwise. At the same

time, the flags, also known as the signatures, are the values after using Pearson hash function and shingling. This what we called MinHash technique that allows us to replace a large set (digest) with a smaller probability value. Thus, the similarity of small lists which are the signatures predicts the similarity of the original sets (Digests) [23]. Therefore, from the first step, we speed up the process and eliminate the signature matrix by providing a set with 8-digit.

Jaccard Similarity technique is applied to determine the similarity of the signature vectors for the sets; Jaccard Similarity is the ratio of the size of the intersection of the two sets to the size of Union of the two sets.

For sets A and B $\rightarrow J(A, B) = |A \cap B| / |A \cup B|$

Data Integrity steps are as follows:

1. Break down the digests into shingles each shingle of size 2 after hashing the plaintexts using Pearson hash function.
2. Get a big set that includes all the shingles of both digest without any shingle duplication.
3. Find Flag for each shingle of each digest between 0 and 1 called signatures. The 1 flag means the shingle of digest appears at the big set, otherwise, it is 0 (MinHash Technique to replace shingle value with small value 0/1).
4. Find the similarity between the signatures using the Jaccard Similarity Technique.

3. RESULTS AND DISCUSSION

The three phases are implemented using Java language on Eclipse and NetBeans software. The AES and RSA algorithms are implemented using the SecretKeySpec [25] and PKCS8 [26] libraries, respectively. However, the similarity technique is originally developed according to the steps stated in the proposed method. In this work, the testing phase examines the original source code for AES, RSA, and data integrity. We, then, modify the source code according to our method and finally compare the results of the original and modified source code.

3.1. Key generation for encryption/decryption phase

3.1.1. AES

In this phase, the original AES algorithm along with the SecretKeySpec, which is a built-in class in Java. The SecretKeySpec constructs a secretKey from a byte array without consulting a provider as the SecretKeyFactor does. This algorithm generates one key for the sender and another key for the receiver. The modification incorporated in the AES algorithm is using the keys that were generated from the MinHash algorithm instead of the SecretKeySpec. The comparison between the two algorithms is accomplished according to the encryption time, which is the amount needed to convert plaintext to an encrypted text as a function of key and data block sizes.

Table 1 compares AES with the SecretKeySpec and AES with MinHash. It clearly shows that AES with MinHash outperforms AES with SecrtKeySpec by an average of 17.35%. Also, as the number of words in the plain text increases the encryption time increases. The results prove that the MinHash can improve the time of the encryption process for dynamic plain text sizes.

Table 1. AES Algorithms comparison

No. of words	AES with SecretKeySpec	AES with MinHash Algorithm	MinHash Algorithm's % compared to SecretKeySpec
200	19133.06	17647.38	8.4 %
400	21235.55	18093.41	17.36%
600	25057.06	22847.99	20.189%
800	29882.52	24827.49	20.36 %
1000	31327.71	27001.55	20.484%

3.1.2. RSA

The testing phase for the key generation of the RSA algorithm was done using the RSA with PKCS8, which is a standard syntax for storing private key information generated using the KeyGenerator library. The modification was done using the MinHash function to generate the private and the public key in the encryption and the decryption phase.

Table 2 shows the comparison between RSA with PKCS8 and RSA with the MinHash technique. It shows that our method achieves 43.93% in performance improvement against the original RSA algorithm.

This obvious enhancement is due to the poor performance of the original RSA algorithm, which is considered the slowest among other cryptographic algorithms. Our approach maintains outperforming the other one as the size of the plain text increases.

Table 2. RSA Algorithms comparison

No. of words	RSA with PKCS8	RSA with MinHash Algorithm	MinHash Algorithm's % compared to PKCS8
200	42849.1	29747.14	42.99%
400	50289.79	35169.27	44.173%
600	58047.62	42847.19	43.11%
800	65785.04	49274.93	45.12%
1000	74958.1	54815.4	44.28%

3.2. Data integrity

We perform data integrity by finding the similarity between the original set and the decrypted set as discussed earlier. Normal MinHash technique is usually used to find the similarity between sets, and it works as the following steps:

1. Break down the documents into a set of shingles.
2. Calculate the hash value for every shingle.
3. Store the minimum hash value found in step 2.
4. Repeat steps 2 and 3 with different hash algorithms with random times to get the min hash values.

Compared with our proposed methodology to find the similarities between two sets. Pearson hash function has been used to obtain 8-digit hash value to minimize the set size then comparing these hash values. Our proposed method again works as follows:

1. Hash two sets (original set and encrypted set) to get 8-digit hash values for each set.
2. Break down the digests into shingles, each of which has a size of 2.
3. Get the large set that includes all the unique shingles of both digests.
4. Find the flag for each shingle of each digest that takes a signature value between 0 and 1. The flag value of 1 means that the shingle of the digest appears in the large set. Otherwise, it appears in the small set when the value equals zero.
5. Find the similarity between the signatures utilizing Jaccard Similarity Technique.

Here is an example of a data integrity procedure:

Let's assume that we Hashed the A and B sets using the Pearson Hash function. Then get the shingles of size 2 for each digest.

Shingles of digest A = {ab, cd, ef, gh}

Shingles of digest B = {ab, cd, ot, ky}

The Large set let assume to be C = A U B = {ab, cd, ef, gh, ot, ky}, which is the large set that includes all the unique shingles of both digests.

After that need to compare each shingle of each digest with the large set to create a flag set for each with 0,1 values. The flag 1 means it includes in the C otherwise not, as the following:

Flag of A = {1, 1, 1, 1} Flag of B = {1, 1, 1, 1}

After that, using the Jaccard Technique to find the similarity ration between Flag A and Flag B as shows in Table 3.

Table 3. Data Integrity time results

Large Set C	Shingles of Digest A	Shingles of Digest B	Flag of A	Flag of B
ab	ab	ab	1	1
cd	cd	cd	1	1
ef	ef		1	0
gh	gh		1	0
ot		ot	0	1

For sets Flag A and Flag B = J (Flag A, Flag B)
= |Flag A ∩ Flag B| / | Flag A U Flag B| = 2/6 = 1/3

By comparing both techniques, we prove that our method has better performance than the existing method in finding the similarity between the two sets. And it has a better memory utilization for finding 8-digit hash value. Table 4 shows the sizes of the different scenarios used in the experiment. The original method to find the similarity is using either Jaccard or MinHash. But in our proposed method, the mixture between MinHash, Jaccard, and Pearson Hash has been used. The Pearson is used to minimize the data size input to MinHash to find the similarity instead of finding the similarity of the original data.

Table 4. Different scenarios with various sizes for Data integrity set

#	Scenario	Size of Original Set 1	Size of Original Set 2
1	Similar two small static* size sets	12	12
2	Similar two medium static size sets	68	68
3	Different two small static size sets	12	12
4	Different two medium static size sets	68	68
5	Different two sets with dynamic** size	68	12
6	Different two sets with dynamic large size	1337	2784
7	Different two sets with static large size	1337	1337
8	Similar two sets with static Large size	1337	1337
9	Similar two sets with static Large size	1337	1337

*two static size sets, which are two sets with similar number of words
**two dynamic size sets, which are two sets with different number of words

For a larger size set, our proposed method incurs a lower delay compared to the normal one. Moreover, it saves a memory resource by consuming a small amount of storage than the normal technique which utilizes a huge amount of storage. Hence, our proposed technique is more powerful especially for large data set compared with the existing technique to find the similarity. The results show a big gap in time between the two techniques, which leads to better performance in a term of speed using our approach. At the same time, the results show the increasing trend according to the memory utilization between the two techniques. From all these results as shown in Table 5, our approach has better performance in terms of fast time execution than the existing approach by at least 3% for small/medium size and exceed 100% between 288% to 295% for large size and has less memory utilization than existing approach by at least 15% for small/medium size increasing to exceed 100% for the large size.

Table 5. Data integrity enhancement ratio for different scenarios

Scenario	Proposed Result	Other Results (Existing)
1	4.84306	21.901423
2	8.285231	90.200064
3	4.876641	20.747053
4	5.920393	75.58318
5	5.818466	49.484228
6	6.241973	45.516624
7	75.48876	40771.17244
8	62.371531	16152.44004
9	62.745655	19465.74875

4. CONCLUSION

The key advantage of our proposed solution is to increase the level of security and integrity of the data transfer simultaneously. The power of this solution came from merging asymmetric encryption technique with the MinHashing technique. The asymmetric technique used at both encrypting/decrypting data and the digest of hashing the plaintext. Moreover, the MinHashing technique has been used for two various locations. The first use is for generating different pairs of public-private keys for the encryption phase. The other use is for determining the similarity for the two digests of the original plaintext with less delay to ensure the data integrity. This enhancement sacrifices the security of data transmission as the whole method is based on having a faster process by generating keys in a faster manner using the Hash function without changing anything in the algorithm that is used to handle the data transmission. But there is a small improvement at security aspects that the encryption/decryption algorithm based on the random keys generated from the MinHash and Shingling function which have a one-way output of digest and avoid

returning to the original data. Compared with the usual techniques for data security and integrity, using the MinHash technique for generating keys and finding the similarity with the help of Pearson hash function and different encryption algorithms demonstrates that the method can provide more data security and integrity. The experimental results show that the efficiency of engaging the MinHash to the proposed method can improve and enhance the data in terms of increasing the security wise and fast determination of data integrity in terms of time and accuracy. In addition, the results show better performance to transmit the data in a secure manner with more accuracy. For future work, more algorithms will be tested by using the MinHash technique in order to find the best algorithm in terms of security and accuracy. Also, more metrices can be added as the throughput in order to compare all the tested algorithms with more than one metric to have a deeper study on the behavior of the cryptographic algorithms.

REFERENCES

[1] Chandra, S., Paira, S., Alam, S.S. and Sanyal, G., "A comparative survey of symmetric and asymmetric key cryptography," In *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*, pp. 83-93. IEEE, 2014.

[2] Patil, P., Narayankar, P., Narayan, D.G. and Meena, S.M., "A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish," *Procedia Computer Science*, vol. 78, pp. 617-624, 2016.

[3] Rezai, A., and Keshavarzi, P., "High-performance scalable architecture for modular multiplication using a new digit-serial computation," *Microelectronics Journal*, vol. 55, pp. 169-178, 2016.

[4] Yadav, G., and A. Majare, "A comparative study of performance analysis of various encryption algorithms," In *International Conference on Emanations in Modern Technology and Engineering*, vol. 5, no, 3, pp. 70-73. 2017.

[5] Krishna, B.H., Reddy, I.R.S., Kiran, S. and Reddy, R.P.K., "Multiple text encryption, key entrenched, distributed cipher using pairing functions and transposition ciphers," In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 1059-1061. IEEE, 2016.

[6] Asis, K., Tapan, K. and Navaneethan, C., "Data cryptography based on musical notes on a fingerboard along with a dice," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 14, no. 3, pp.1286-1290, 2019.

[7] Kumari, Sarita, "A research paper on cryptography encryption and compression techniques," *International Journal of Engineering and Computer Science (IJEECS)*, vol. 6, no. 4, pp. 20915-20919, 2017.

[8] Sodhi, G., Gurjot, S., Lavish, K., Eduard, B., Mohammed, A., Sandeep, K. and Mehedi, M., "Preserving Authenticity and Integrity of Distributed Networks through Novel Message Authentication Code," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 12, no. 3, pp. 1297-1304, 2018.

[9] Quilala, Rogel, L., Ariel M. and Ruji, P., "QR Code Integrity Verification Based on Modified SHA-1 Algorithm," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 6, no. 4, pp. 385-392, 2018.

[10] Shahab, W., Raghad, Z. and Shadan, M., "An approach for enhancing data confidently in Hadoop," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 20, no. 3, pp. 1547-1555, 2020.

[11] Wang, D., Jiang, Y., Song, H., He, F., Gu, M. and Sun, J., "Verification of implementations of cryptographic hash functions," *IEEE Access*, vol. 5, pp.7816-7825, 2017.

[12] Oprisa, C., "A MinHash Approach for Clustering Large Collections of Binary Programs," *2015 20th International Conference on Control Systems and Computer Science*, Bucharest, pp. 157-163, 2015.

[13] Yan, Z., Liu, J., Li, G., Han, Z. and Qiu, S., "PrivMin: Differentially Private MinHash for Jaccard Similarity Computation," arXiv preprint arXiv: 1705.07258, 2017.

[14] Orencik, C., Kantarcioglu, M. and Savas, E., "A practical and secure multi-keyword search method over encrypted cloud data," In *2013 IEEE Sixth International Conference on Cloud Computing*, pp. 390-397, 2013.

[15] He, J., Wu, J., Zhu, N. and Pathan, M., "MinHash-Based Fuzzy Keyword Search of Encrypted Data across Multiple Cloud Servers," *Future Internet*, vol. 10, no. 5, p.38, 2018.

[16] Tiwari, N. and Sinhal, A., "An Implementation on Secure Hash Algorithm in Wireless Algorithms to Ensure the Integrity," *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 5, no. 3, pp. 4779-4781, 2014.

[17] Szefer, J., Chen, Y.Y. and Lee, R.B., "General-purpose FPGA platform for efficient encryption and hashing," In *ASAP 2010-21st IEEE International Conference on Application-specific Systems, Architectures and Processors*, pp. 309-312, July, IEEE, 2010.

[18] Park, H., Lee, S.C., Lee, S.H. and Kim, S.W., "Centralmatch: A fast and accurate method to identify blog-duplicates," In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp. 112-119. IEEE, 2010.

[19] Chauhan, S.S. and Batra, S., "Finding similar items using LSH and Bloom Filter," In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pp. 1662-1666. IEEE, 2014.

[20] Kheir, N., "Behavioral classification and detection of malware through http user agent anomalies," *Journal of Information Security and Applications*, vol. 18, no. 1, pp. 2-13, 2013.

[21] Ioffe, S., "Improved consistent sampling, weighted minhash and l1 sketching," In *2010 IEEE International Conference on Data Mining*, pp. 246-255. IEEE, 2010.

[22] Manaa, M.E. and Jwdha, R.H., "A Robust Encryption Files Approach using Minhash Technique," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 15, pp.169-183, 2018.

[23] Leskovec, J., Rajaraman, A. and Ullman, J.D., "Mining of massive datasets," *Cambridge university press*, 2014.

[24] "Pearson Hashing," https://programmingpraxis.com/2018/05/25/pearson-hashing/, created 24 May 2018.
[25] "Example of AES encryption and decryption in Java," https://gist.github.com/SimoneStefani/99052e8ce0550eb7725ca8681e4225c5, accessed 12 Feb. 2020.
[26] "RSA Encryption and Decryption in Java," https://www.devglan.com/java8/rsa-encryption-decryption-java, accessed 12 Sep. 2019.

BIOGRAPHIES OF AUTHORS

Sa'ed Abed received his B.Sc. and M.Sc. in Computer Engineering from Jordan University of Science and Technology, Jordan in 1994 and 1996, respectively. In 2008, he received his Ph.D. in Computer Engineering from Concordia University, Canada. He has previously worked at King Faisal University in Saudi Arabia from 1997-2003. He joined Hashemite University, Jordan, as an Assistant Professor from 2008-2014. Currently, Dr. Abed is an Associate professor in the Department of Computer Engineering at Kuwait University. His research interests include Formal Methods, VLSI Design and Image Processing. He also served as a reviewer for various international conferences and journals. Dr. Abed published over 90 papers in reputable journals and conferences.

Lamis Waleed received the B.S. degree in computer engineering from College of Petroleum and Engineering, Kuwait University, in 2018. She is completing her master's degree in computer engineering, Kuwait University. She is currently working as a computer instructor in Kuwait Technical college. Her research interests include wireless communications, network security, network management and artificial intelligence.

Ghadeer Aldamkhi grew up and graduated from Kuwait University, College of Petroleum and Engineering and received her Computer Engineering bachelor's degree in 2015. She is completing the master's degree in the Computer Engineering field. Currently, she is working at Telecom Company in Kuwait since 2017 and has a background about many aspects; in networking, development, system integration and hardware field.

Khaled Hadi received his Ph.D. from the University of Massachusetts, Amherst, in 2009, in Electrical and Computer Engineering and his MS Degree from the School of Information and Computer Sciences, University of California, Irvine, in 2004. He received the BS in Computer Engineering in 1999 from Kuwait University. He is currently an assistant professor of computer engineering at Kuwait University. His research interests include sensor networks, computer networks, distributed systems, and operating systems.