Original Research Paper

MDS Algorithm for Encryption

[1]Mani Arora, [2]Derick Engles and [3]Sandeep Sharma

[1]*Khalsa College, Amritsar, Punjab, India*
[2,3]*Guru Nanak Dev University, Amritsar, Punjab, India*

Article history
Received: 3-08-2013
Revised: 6-01-2015
Accepted: 13-03-2015

Corresponding Author:
Mani Arora
Khalsa College, Amritsar,
Punjab, India
Email: mani_mcain@yahoo.com

Abstract: Most of the encryption algorithms used today generates huge cipher messages as well as long encryption keys. These approaches require time and are computationally intensive .While sending data packets through the network a compression technique along with cryptography can be applied to reduce the data packet size for better bandwidth utilization and hence faster transmission of data. In this study we propose a new encryption technique which will encrypt data into reduced size cipher text while keeping the check on size of key. The algorithm was first introduced in an earlier paper. In this study we modified algorithm to preprocess text as well as numeric data.

Keywords: Encryption, Privacy, Cipher, Size, Text, Security

Introduction

In the digital world with the widespread use of information technologies and the rise of digital computer networks in many areas of the world, securing the exchange of information has become a crucial task (McConnell, 2002). Cryptography is the methods that allow information to be sent in a secure form in such a way that the only receiver able to retrieve this information. The era of modern cryptology is generally agreed to have started in 1949, when Shannon transformed cryptography from an art to a science with the publication of a paper entitled "Communication theory of secrecy systems" (Diffie and Hellman, 1976; 1979). However, while cryptology took a new fundamental direction from that point on, most of the major innovations in the field date from the last 30 years. Modern cryptography involves the use of keys for data signing, encoding and decoding. Some keys are distributed privately and some publicly. Level of protection is varied for every situation and depends on the encryption technique used for coding. One way to estimate techniques on this level is to estimate how much CPU time would be required on a machine of a given processing speed to iterate through all possible keys to the encoded data based upon the permutation and second is how much secure the data is while transmitting. The basic goal of our work is along with these two measures cryptography algorithm must also be efficient in reducing the size of encrypted text referred as cipher text. In our earlier paper (Mani and Derick, 2010) we have used the term reduced size cipher text to convert plain text to cipher text. In this study we are modifying the cryptographic technique to provide better results.

Proposed Technique

The proposed technique intends to encrypt the plain text with a prechosen mathematical function along with the objective of reduced size cipher text. During encryption and decryption process, two dictionaries are referred primary dictionary as shown in Table 1 and secondary dictionary as shown in Table 2. Dictionary technique is used as it provide compression efficiency as well as fast decompression mechanism. The basic idea is to take the advantage of commonly occurring instruction sequences by using a dictionary. The repeating occurrences are replaced by a codeword. Primary dictionary will be static in nature while secondary dictionary will be dynamic. It depends on the nature of text being encoded. The online building of the secondary dictionary in the primary memory ensures the single pass over the data and the dictionary need to be transmitted over the network. In this previous technique we not focused on frequent occurrence of alphanumeric and numeric data. Since any digit took 2 bytes i.e., 16 bits in memory but if we provide code of 12 bits to every digit, it will reduce the size of cipher text as well. So the modified primary dictionary will contain codes for words, numeric data and alphanumeric data. We have still focused on English language only while creating dictionaries.

Primary Dictionary

The dictionary contains words and numeric data which are probably most frequently used along with the codes* (which will be explained later). The dictionary will be fixed in size and the codes too. Even if someone cracks the dictionary and codes, still our technique is going to work because it is based on both primary and secondary dictionary and secondary dictionary is not fixed.

Table 1. Primary dictionary

Code	String
0000 0000 0001	0
0000 0000 0010	1
0000 0000 0011	2
0000 0000 0100	3
0000 0000 0101	4
0000 0000 0110	5
0000 0000 0111	6
0000 0000 1000	7
0000 0000 1001	8
0000 0000 1010	9
0000 0000 1011	after
0000 0000 1100	off
0000 0000 1101	I
0000 0000 1110	or
0000 0000 1111	an
0000 0001 0000	a
0000 0001 0001	as
0000 0001 0010	in
0000 0001 0011	ok
0000 0001 0100	It
0000 0001 0101	is
0000 0001 0110	on
0000 0001 0111	at
0000 0001 1000	of
0000 0001 1001	If
0000 0001 1010	we
0000 0001 1011	my
0000 0001 1100	do
0000 0001 1101	am
0000 0001 1110	pm
0000 0001 1111	be
0000 0001 0000	to
0000 0010 0001	by
0000 0001 0010	can
0000 0001 0011	the
0000 0001 0100	was
0000 0001 0101	sat
0000 0001 0110	for
0000 0001 0111	not
0000 0001 1000	has
0000 0001 1001	had
0000 0001 1010	him
0000 0001 1011	her
0000 0001 1100	other
0000 0001 1101	which
0000 0001 1110	where
0000 0001 1111	you
0000 0011 0000	your
0000 0011 0001	some
0000 0011 0010	too
0000 0011 0011	who
0000 0011 0100	Its
0000 0011 0101	and
0000 0011 0110	whatever
0000 0011 0111	herself
0000 0011 1000	bar
0000 0011 1001	can’t
0000 0011 1010	don’t
0000 0011 1011	there
0000 0011 1100	does
0000 0011 1101	into

Table 1. Continue

0000 0011 1110	this
0000 0011 1111	back
0000 0100 0000	were
0000 0100 0001	four
0000 0100 0010	that
0000 0100 0011	back
0000 0100 0100	from
0000 0100 0101	form
0000 0100 0110	most
0000 0100 0111	word
0000 0100 1000	whom
0000 0100 1001	able
0000 0100 1010	here
0000 0100 1011	must
0000 0100 1100	did
0000 0100 1101	didn’t
0000 0100 1110	like
0000 0100 1111	national
0000 0101 0000	.
0000 0101 0001	,
0000 0101 0010	‘
0000 0101 0011	“
0000 0101 0100	;
0000 0101 0101	:
0000 0101 0110	)
0000 0101 0111	(
0000 0101 1000	?
0000 0101 1001	/
0000 0101 1010	\
0000 0101 1011	|
0000 0101 1100	!
0000 0101 1101	@
0000 0101 1110	%
0000 0101 1111	&

Table 2. Secondary dictionary

0001 0000 0000	confidentiality
0001 0000 0001	Information
0001 0000 0010	cannot
0001 0000 0011	understood
0001 0000 0100	anyone
0001 0000 0101	unintended
0001 0000 0110	protection
0001 0000 0111	transmitted
0001 0000 1000	data
0001 0000 1001	passive
0001 0000 1010	attacks
0001 0000 1011	aspect
0001 0000 1100	traffic
0001 0000 1101	flow
0001 0000 1110	analysis
0001 0000 1111	requires
0001 0001 0000	attacker
0001 0001 0001	observe
0001 0001 0010	source
0001 0001 0011	destination
0001 0001 0100	frequency
0001 0001 0101	length
0001 0001 0110	characteristics
0001 0001 0111	communication
0001 0001 1000	facility

Table 3. Contents of key file

1111 0000 1111	Key

To maintain this dictionary plain text file is used and for processing we read the file into memory as associative arrays. Associative arrays maps arbitrarily typed objects to arbitrarily typed objects. Data structures used to represent associative array when initialized in memory will be linked list. For searching a word in the dictionary, simple linear search is used. We are using 12 bit codes so total 4096 words can be stored in this dictionary.

Secondary Dictionary

The dictionary is not fixed. This dictionary will be empty when initialized. Every time the algorithm come across the string in pass2, it will add it to dictionary and assign a code to it. Starting code for first string in secondary dictionary will be fixed. The next codes can be obtained by doing increment of one step. As this dictionary will be created during runtime so it's difficult to crack the encryption. This dictionary will also be stored in same file that for primary dictionary is used.

Encryption Algorithm

The algorithm will start with initializing a primary dictionary and variable S, which will be initially empty. We read the file containing dictionary into memory in linked list structure. The variable S will read plain text data file word by word and comparisons in dictionary will be made with help of this variable. In the algorithm each word of plain text will be first searched in primary dictionary, this is done using linear search. If the word is present in primary dictionary then it will be replaced by corresponding code assigned to it and stored in encoded output file. Codes are in binary as binary take less memory space than any other data type. Codes are fixed for each data word so it can't be changed later on. Size of code allocated to each string will be 12 bits that is any dictionary can contain maximum 4096 entries. In this modified version of the previous algorithm we are considering first 10 entries of the numeric data in primary dictionary. As we know otherwise numeric digit took 2 byte (16 bits) space in memory but here we are allocating 12 bits fixed code for every numeric digit which will also compress the data and finally the cipher text. Also some codes will be allocated to common alphanumeric data so that data can be compressed.

If the word is not found in primary dictionary then it is searched in secondary dictionary. If the word is present in secondary dictionary its corresponding code will be substituted to encoded output file, if not present then that word is substituted in secondary dictionary and the new code is generated to it by incrementing the code by 1 of last entry in secondary dictionary. The code for first entry in secondary dictionary will be fixed and rest codes will be obtained by incrementing each code by 1.Secondary dictionary will be of variable size.

After these pass we will get encoded output file. Further in the algorithm we use a predetermined mathematical function XOR to further encrypt the encoded output. As code assigned to each word is of 12 bits, so in pass3 each 12 bit block will be XOR with a secret key to get final output which will be a cipher text in reduced size then plain text.

The above procedure is also explained with the help of flowchart i.e., Fig. 1.

The Algorithm for Proposed Technique is:

```
1. Initialize S as an empty string
2. Read primary dictionary in memory
3. Do while NOT EOF
   Read the next word from the file
   If this word is in primary dictionary get corresponding
   code from primary dictionary
           Write the code in the output file
   else
           Read secondary dictionary in file
   If this word is in secondary dictionary see the
   corresponding code from secondary dictionary
           Write the code in the output file
   else
           add this word to secondary dictionary
           assign the next code obtained by
           incrementing previous code by 1 used to
           substitute the new code in the output file
4. Read the output file
5. Do while NOT EOF
   Read key from file in memory
   Read next 12 -bit block from file
   Perform XOR operation between key and 12-bit block
   Write the code in final output file
6. End
```

Primary Dictionary

Primary dictionary is fixed so sender and receiver both before transmission of cipher text will know it.

Algorithm to Create Primary Dictionary

1. Start with prechosen code and assign first word to it
2. Add new word to dictionary
3. Assign code to new word by incrementing previous code by 1

Secondary Dictionary

Secondary dictionary is dynamic in nature. It will be created at runtime.

Algorithm to Create Secondary Dictionary

```
1. Initialize A as an empty string
2. Do till EOF
   Read the file till any string is
   encountered
   Assign string to A
   Add this string to dictionary
3. Start with prechosen code and assign first string to it
4. Assign code to new word by incrementing previous
   code by 1
```

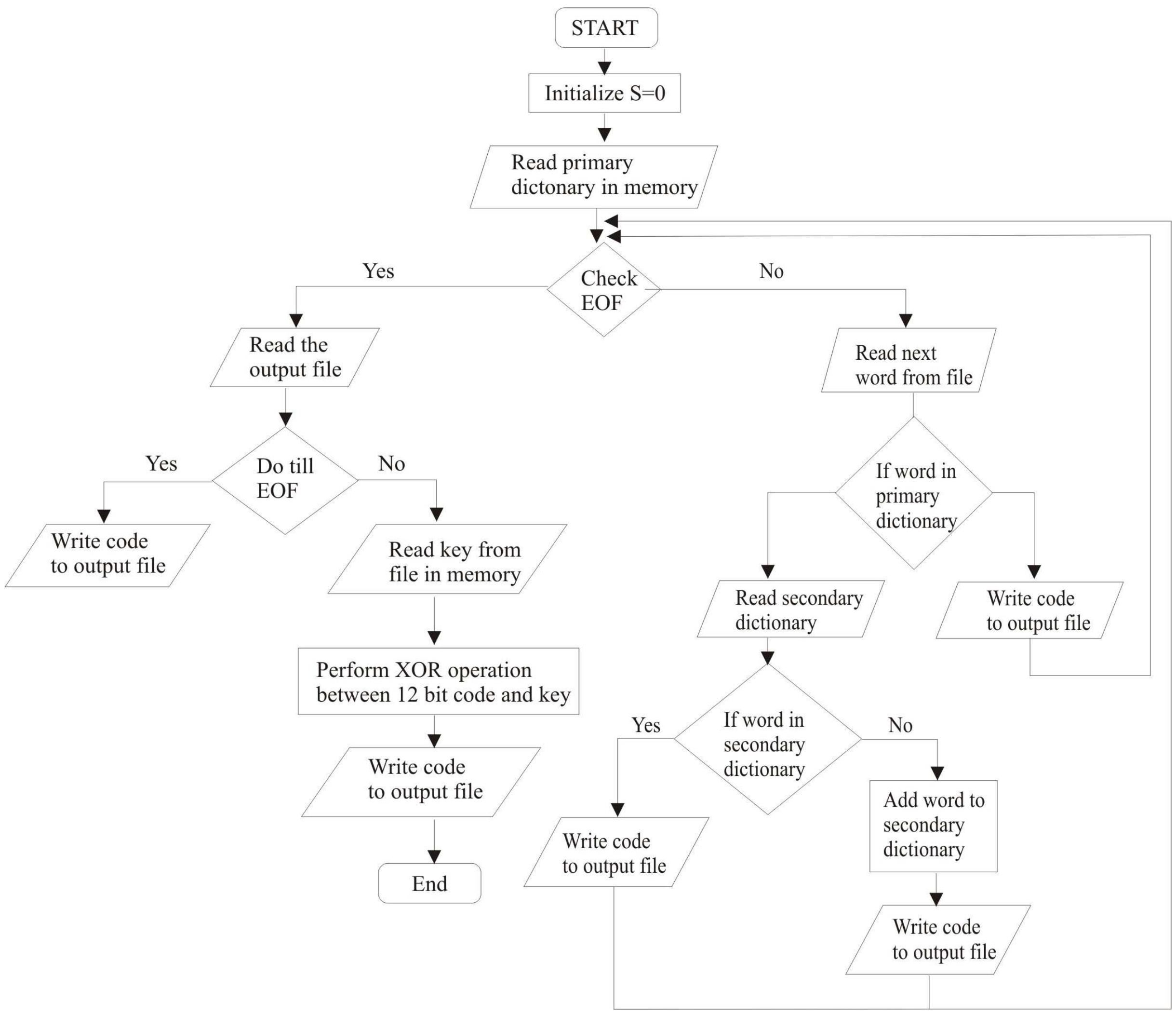
START
Initialize S=0
Read primary dictonary in memory
Check EOF
Yes
No
Read the output file
Do till EOF
Yes
No
Write code to output file
Read key from file in memory
Perform XOR operation between 12 bit code and key
Write code to output file
End
Read next word from file
If word in primary dictionary
Read secondary dictionary
Write code to output file
If word in secondary dictionary
Yes
No
Write code to output file
Add word to secondary dictionary
Write code to output file

Fig.1. Flowchart

Flowchart for the Algorithm is as Follows

Example

For Example the plain text given below is encrypted after pass1 and pass2 as shown below.

Plain Text that is to be Encrypted is:

Confidentiality

The information cannot be understood by anyone for whom it was unintended. Confidentially is the protection of transmitted data from passive attacks. The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination frequency length or other characteristics of the traffic on a communication facility.

Pass1: In this pass primary dictionary given as Table 1 and secondary dictionary given as Table 2 is referred for substituting codes in place of text:

0000 0000 0010 0000 0101 0110 0001 0000 0000 0000 0101 0101 0000 0001 0011 0001 0000 0001 0000 0001 0010 0000 0001 1111 0000 0001 0011 0000 0010 0001 0000 0100 0000 0001 0110 0000 0100 1000 0000 0001 0100 0000 0001 0100 0001 0000 0101 0000 0101 0000 0001 0000 0000 0000 0001 0101 0000 0001 0011 0001 0000 0110 0000 0001 1000 0001 0000 0111 0001 0000 1000 0000 0100 0100 0001 0000 1001 0001 0000 1010 0000 0101 0000 0000 0001 0011 0000 0001 1100 0001 0000 1011 0000 0001 1000 0001 0000 0000 0000 0001 0101 0000 0001 0011 0001 0000 0110 0000 0001 1000 0001 0000 1100 0001 0000 1101 0000 0100 0100 0001 0000 1110 0000 0101 0000 0000 0011 1110 0001 0000 1111 0000 0100 0010 0000 0000 1111 0001 0001 0000

0000 0001 0111 0000 0001 1111 0000 0100 1001 0000 0001 0000 0001 0001 0001 0000 0001 0011 0001 0001 0010 0000 0101 1111 0001 0001 0011 0001 0001 0100 0001 0001 0101 0000 0000 1110 0000 0001 1100 0001 0001 0110 0000 0001 1000 0000 0001 0011 0001 0000 1100 0000 0001 0110 0001 0001 0111 0001 0001 1000 0000 0101 0000

Pass2: In this pass each 12-bit block will be XOR with key, which is given in Table 3:

1111 0000 1101 1111 0101 1001 1110 0000 1111 1111 0101 1010 1111 0001 1100 1110 0000 1110 1111 0001 1101 1111 0001 0000 1111 0001 1100 1111 0010 1110 1111 0100 1111 1110 0110 1111 1011 1000 1111 1110 0100 1111 1110 0100 1110 1111 0101 1111 1010 0000 1110 1111 0000 1111 1110 0101 1111 1110 0011 1110 1111 0110 1111 1110 1000 1110 1111 0111 1110 1111 1000 1111 1011 0100 1110 1111 1001 1110 1111 1010 1111 1010 0000 1111 1110 0011 1111 1110 1100 1110 1111 1011 1111 1110 1000 1110 1111 0000 1111 1110 0101 1111 1110 0011 1110 1111 0110 1111 1110 1000 1110 1111 1100 1110 1111 1101 1111 1011 0100 1110 1111 1110 1111 1010 0000 1111 1100 1110 1110 1111 0000 1111 1011 0010 1111 1111 1111 1110 0001 1111 1111 0001 1000 1111 0001 0000 1111 1111 0001 1000 1111 0001 0000 1111 0100 0110 1111 0001 1111 1110 0001 1110 1111 0001 1100 1110 0001 1101 1111 0101 0000 1110 0001 1100 1110 0001 1011 1110 0001 1010 1111 0000 0001 1111 0001 0011 1110 0001 1001 1111 0001 0111 1111 0001 1100 1110 0000 0011 1111 0001 1001 1110 0001 1000 1110 0001 0111 1111 0101 1111

Conclusion

A new algorithm has been proposed that focused both on the security as well as length of cipher text. The proposed algorithm produced smaller cipher text in comparison to existing methods. In this algorithm dictionaries are acting as keys and both the encoding and decoding is not possible until key is not available. The code generation is simpler in comparison to existing algorithms which are mostly based on mixed operators or values and other mathematical operations. The transmission time of cipher text has been also reduced. Also it reduces the redundancy in data representation to decrease the storage required for that data. We extending this algorithm to our future work, we will try to provide a mechanism so that there should be no need to send secondary dictionary to receiver end.

Funding Information

The authors have no support or funding to report.

Author's Contributions

All authors equally contributed in this work.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Diffie, W. and M. Hellman, 1976. New directions in cryptography. IEEE Trans. Inform. Theory, 22: 644-654. DOI: 10.1109/TIT.1976.1055638

Diffie, W. and M.E. Hellman, 1979. Privacy and authentication: An introduction to cryptography. Proc. IEEE, 67: 397-427. DOI: 10.1109/PROC.1979.11256

Mani, A. and A. Derick, 2010. An algorithm to reduce the size of cipher text. Global J. Comput. Sci. Technol., 10: 50-54.

McConnell, M., 2002. Information Assurance in the twenty-first century. IEEE Comput., 35: 16-19. DOI: 10.1109/MC.2002.1012425