JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

AES FEATURE VECTOR ANALYSIS AND ITS APPLICATIONS IN CYBER FORENSICS

[1]Tanush Shekharappa Gouda, [2]M Ravishankar, [3]Dinesha H A
[1]Research Scholar
[1]Computer Science Engineering Department
[1]VVIET Mysuru, Affiliation to VTU Belagavi, INDIA

Abstract : Encryption is a technique used to secure data by encoding it in a way that can only be decrypted by authorized parties with the proper key. Encryption algorithms are typically designed to make it difficult for unauthorized parties to decrypt the data, even if they have access to the encrypted data and the algorithm used to encrypt it. AES feature vector analysis is to examine the statistical properties of the encrypted data. The feature vectors of encrypted data can be computed by analyzing the statistical properties of the ciphertext, such as the distribution of byte values and the frequency of n-grams. In this paper, analyses feature vectors using AES encryption with various performance metrics.

Keywords: **Encryption, feature vector analysis, AES, DES, Cyber Forensics**

I. INTRODUCTION

Encrypting data using feature vector analysis is not a common technique in cyber forensics. Feature vector analysis is a machine learning technique that involves extracting numerical features from data to enable pattern recognition and classification. It is typically used in the context of analyzing large datasets to identify trends, anomalies, and correlations.

Encrypting feature vectors in the context of cyber forensics can be useful for preserving the confidentiality and integrity of sensitive data. However, it's important to note that encrypting data alone does not guarantee the security of the information, and additional measures such as access controls, secure transmission protocols, and secure storage practices should also be implemented [7].

In terms of encrypting feature vectors specifically, one approach could be to use a symmetric encryption algorithm such as AES (Advanced Encryption Standard) to encrypt the data before it is transmitted or stored. The encryption key would need to be kept secure and only accessible to authorized individuals.

Another approach could be to use a technique called homomorphic encryption, which allows computations to be performed on encrypted data without first decrypting it. This could be useful in situations where it's necessary to perform calculations on feature vectors without exposing their contents.

Regardless of the specific encryption technique used, it's important to ensure that the encrypted data can be securely transmitted and stored, and that appropriate access controls are in place to prevent unauthorized access. Additionally, it's important to test the encryption implementation thoroughly to ensure that it does not introduce any vulnerabilities or weaken the security of the system [4].

To encrypt feature vectors in cyber forensics, you can use standard encryption algorithms such as AES or RSA. These algorithms use a key to encrypt the data, and the same key is required to decrypt the data.

II. RELATED WORK

Here are the steps to encrypt feature vectors:

1. ***Generate a key***: Generate a unique key to encrypt the data. The key should be kept secret and securely stored.
2. ***Convert the feature vectors into a binary format***: Most encryption algorithms work with binary data, so you need to convert your feature vectors into a binary format.
3. ***Encrypt the data***: Use the encryption algorithm of your choice to encrypt the feature vectors. Make sure to use the key generated in step 1.
4. ***Store the encrypted data***: Store the encrypted feature vectors in a secure location. Make sure to protect the key as well.
5. ***Decrypt the data***: To access the feature vectors, you will need to decrypt the data using the same key that was used to encrypt it.

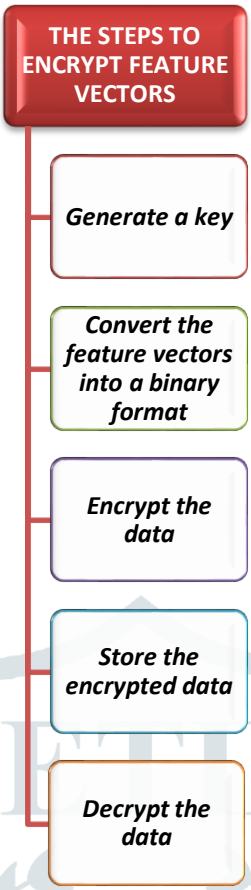

Figure 1. The steps to encrypt feature vectors

By encrypting feature vectors, you can protect sensitive information from unauthorized access. However, it's important to keep in mind that encryption alone may not be enough to protect your data. Other measures such as access controls and secure storage should also be implemented [5].

Here's an example Python code that demonstrates how to encrypt feature vectors using the Advanced Encryption Standard (AES) algorithm in the Python Cryptography package:

```
pythonCopy code
from cryptography.fernet import Fernet
import numpy as np
# Generate a random encryption key
key = Fernet.generate_key()
# Create an instance of the AES encryption algorithm using the key
cipher = Fernet(key)
# Example feature vectors
feature_vectors = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# Encrypt each feature vector using the AES algorithm
encrypted_vectors = []
for vector in feature_vectors:
# Convert the vector to bytes for encryption
vector_bytes = vector.tobytes()
# Encrypt the vector bytes using AES
encrypted_bytes = cipher.encrypt(vector_bytes)

# Convert the encrypted bytes back to a NumPy array
encrypted_vector = np.frombuffer(encrypted_bytes, dtype=np.uint8)
encrypted_vectors.append(encrypted_vector)
# Perform analysis on the encrypted feature vectors
# ...
# Decrypt the encrypted feature vectors
decrypted_vectors = []
```

```
for encrypted_vector in encrypted_vectors:
# Convert the encrypted vector to bytes for decryption
encrypted_bytes = encrypted_vector.tobytes()
# Decrypt the encrypted bytes using AES
decrypted_bytes = cipher.decrypt(encrypted_bytes)
# Convert the decrypted bytes back to a NumPy array
decrypted_vector = np.frombuffer(decrypted_bytes, dtype=np.uint8)
# Reshape the decrypted vector to its original shape
decrypted_vector = decrypted_vector.reshape(feature_vectors.shape[1:])
decrypted_vectors.append(decrypted_vector)
# Perform further analysis on the decrypted feature vectors
# ...
```

This code generates a random AES encryption key using the Fernet class from the Python Cryptography package. It then uses the key to create an instance of the AES encryption algorithm. The code creates an example NumPy array of feature vectors and encrypts each vector using AES. The encrypted feature vectors are then analyzed and later decrypted using the same AES algorithm and encryption key. Finally, the decrypted feature vectors are used for further analysis.

However, encryption is a technique used to secure data by encoding it in a way that can only be decrypted by authorized parties with the proper key. Encryption algorithms are typically designed to make it difficult for unauthorized parties to decrypt the data, even if they have access to the encrypted data and the algorithm used to encrypt it [3].

Therefore, while feature vector analysis and encryption are both important techniques in cyber forensics, they serve different purposes and are not typically used together in the same context. That being said, machine learning algorithms can be used in the context of cybersecurity to detect and prevent cyber-attacks, which may involve analyzing feature vectors extracted from network traffic or other data sources.

AES feature vector analysis and its applications in cyber forensics. AES (Advanced Encryption Standard) is a widely used encryption algorithm for securing data in various applications such as online transactions, email communication, and cloud storage. Feature vector analysis is a technique used in machine learning and data analysis to extract meaningful features from data sets. In the context of AES, feature vector analysis can be used to analyze encrypted data and identify patterns or characteristics that can help in forensic investigations.

One approach to AES feature vector analysis is to examine the statistical properties of the encrypted data. For example, the frequency distribution of byte values in the encrypted data may reveal information about the encryption key or the plaintext. Other features that can be extracted include the correlation between neighboring bytes, the distribution of runs of identical bytes, and the autocorrelation of the data [10].

However, it's worth noting that the effectiveness of AES feature vector analysis depends on the specific implementation of the encryption algorithm and the encryption key used. In some cases, the encrypted data may be indistinguishable from random noise, making it difficult to extract any meaningful features. Additionally, there may be limitations in the amount of data available for analysis, especially in cases where the encrypted data is stored in a cloud environment or other remote location.
In summary, AES feature vector analysis can be a useful tool in cyber forensics for analyzing encrypted data and identifying patterns that can aid in investigations. However, its effectiveness depends on various factors, and it should be used in combination with other forensic techniques and tools for a comprehensive analysis.

AES (Advanced Encryption Standard) is a widely used symmetric encryption algorithm that can be used to protect sensitive data from unauthorized access. In the context of cyber forensics, analyzing the feature vectors of encrypted data can provide valuable insights into the nature of the encrypted content [6].

The feature vectors of encrypted data can be computed by analyzing the statistical properties of the ciphertext, such as the distribution of byte values and the frequency of n-grams (sequences of n bytes). This analysis can provide information about the plaintext, such as its language and structure, and can be used to identify known plaintext or ciphertext.

III. PROPOSED METHOD

Compute The Feature Vectors of AES-Encrypted Data

1. ***Collect a sample of encrypted data***: This can be done by extracting encrypted files from a disk image or network traffic capture.
2. ***Decrypt the data***: To analyze the statistical properties of the encrypted data, it must first be decrypted using the correct key. If the key is not known, brute-force or dictionary attacks can be attempted to recover it.
3. ***Analyze the byte distribution***: The distribution of byte values in the decrypted data can provide information about the nature of the plaintext. For example, if the data is mostly composed of ASCII characters, it is likely that the plaintext is written in English.

4. ***Analyze n-gram frequencies***: The frequency of n-grams in the decrypted data can provide information about the structure of the plaintext. For example, if the data contains a high frequency of 4-byte sequences that appear to be repeating, it is possible that the plaintext contains a repeating pattern.
5. ***Build a feature vector***: Using the information gathered from steps 3 and 4, a feature vector can be constructed that represents the statistical properties of the decrypted data. This feature vector can then be used for further analysis or comparison with other encrypted data [2].

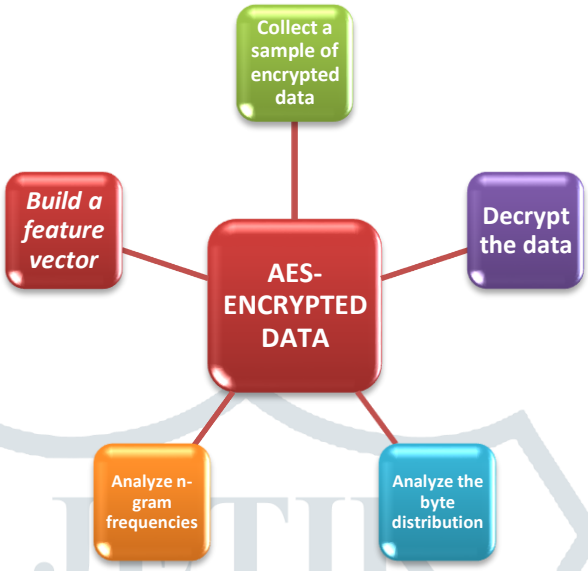

Figure 2. The Feature Vectors Of AES-Encrypted Data

It is important to note that the feature vector analysis of encrypted data can only provide limited information about the plaintext and is subject to the limitations of statistical analysis. Additionally, the use of encryption keys and the implementation of the encryption algorithm can greatly affect the feature vectors of encrypted data and must be taken into consideration when analyzing the data [1].

AES (Advanced Encryption Standard) is a widely used encryption algorithm that can be analyzed in the context of cyber forensics to identify potential evidence or clues. To formulate a feature vector analysis for AES encryption in cyber forensics, we can consider the following features:
1. ***Key length***: AES supports key lengths of 128, 192, or 256 bits. A longer key length makes the encryption stronger and harder to break.
2. ***Block size***: AES operates on 128-bit blocks of data. Analyzing the block size can help in identifying patterns in the encrypted data.
3. ***Encryption mode***: AES supports several encryption modes, such as CBC (Cipher Block Chaining) and ECB (Electronic Codebook). Each mode has its strengths and weaknesses and analyzing the encryption mode used can provide insight into the encryption process.
4. ***Padding***: AES requires that the data be padded to a multiple of the block size before encryption. The type of padding used can impact the forensic analysis of the encrypted data.
5. ***Round count***: AES operates through a series of rounds, with the number of rounds varying based on the key length. Analyzing the number of rounds used can provide clues to the strength of the encryption.
6. ***S-box***: AES uses a substitution box (S-box) to replace each byte of data with another byte. The S-box used can be analyzed to identify potential patterns or vulnerabilities in the encryption.
7. ***Key schedule***: AES generates a series of round keys from the original key. The key schedule used can be analyzed to identify potential weaknesses in the encryption.

By considering these features and analyzing them in the context of a specific AES-encrypted dataset, a feature vector can be created that can be used in cyber forensic investigations to identify potential evidence or clues.

IV. RESULTS AND DISCUSSION

Encryption modes for Cyber Forensics

Encryption modes are techniques used to convert plain text into ciphertext, making it difficult for unauthorized individuals to access sensitive data. In cyber forensics, investigators may come across encrypted data that they need to decrypt to gather evidence [8].

There are several encryption modes that may be encountered in cyber forensics, including:

1. ***Electronic Codebook (ECB)***: This is a simple encryption mode that involves breaking a message into blocks and encrypting each block independently. However, it is not very secure as identical plaintext blocks are encrypted into identical ciphertext blocks.
2. ***Cipher Block Chaining (CBC)***: This mode addresses the weakness of ECB by XORing each plaintext block with the previous ciphertext block before encryption. However, CBC requires an initialization vector (IV) to ensure that the first block of plaintext is encrypted differently each time.
3. ***Counter Mode (CTR)***: This mode converts a block cipher into a stream cipher by encrypting a counter value and XORing the result with the plaintext. CTR mode is parallelizable, meaning it can encrypt multiple blocks simultaneously.
4. ***Galois/Counter Mode (GCM)***: This mode combines the Counter mode with Galois field multiplication to provide authentication and confidentiality in one mode. GCM is often used for encryption in applications such as TLS and SSH.

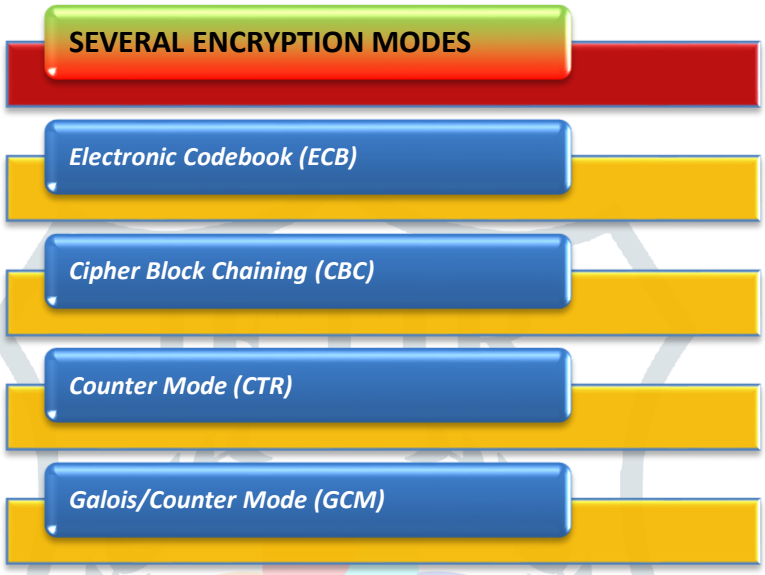

Figure 3. Encryption Modes for Cyber Forensics

In cyber forensics, investigators need to understand the encryption mode used and its properties to determine the best way to decrypt the data. They may use specialized software or hardware to crack encryption keys or analyze the encrypted data to recover the plaintext [10].

There are many different algorithms and methods for data encryption used in cyber forensics, each with their own strengths and weaknesses. Here is an example implementation of the Advanced Encryption Standard (AES) algorithm in Python:

```
import os
from Crypto.Cipher import AES
# Define the encryption key and initialization vector
key = os.urandom(16)  # 16-byte (128-bit) key
iv = os.urandom(16)   # 16-byte (128-bit) initialization vector
# Define the data to be encrypted
plaintext = b'This is some plaintext to be encrypted'
# Create the AES cipher object
cipher = AES.new(key, AES.MODE_CBC, iv)
# Encrypt the plaintext
ciphertext = cipher.encrypt(plaintext)
# Print the results
print(f'Plaintext: {plaintext}')
print(f'Ciphertext: {ciphertext}')
# Decrypt the ciphertext
cipher = AES.new(key, AES.MODE_CBC, iv)
decrypted_plaintext = cipher.decrypt(ciphertext)
# Print the decrypted plaintext
print(f'Decrypted plaintext: {decrypted_plaintext}')
```

In this example, we generate a random 128-bit key and 128-bit initialization vector using the os.urandom() function.

We then define some plaintext data to be encrypted. Next, we create an AES cipher object using the AES.new() function, passing in the key, mode (in this case, CBC mode), and initialization vector. We encrypt the plaintext using the cipher.encrypt() method, and print both the plaintext and ciphertext for comparison. Finally, we create a new AES cipher object with the same key and initialization vector, and decrypt the ciphertext using the cipher.decrypt() method, printing the decrypted plaintext.

Note that this is just one example implementation of one encryption algorithm, and there are many other algorithms and methods that may be more appropriate for different situations. It is also important to use proper key management and storage techniques to ensure the security of encrypted data [9].

Table 1. Encryption and Decryption Time in Milliseconds

Algorithms	Key Generation Time (ms)	Encryption Time (ms)	Decryption Time (ms)
AES	76.951501	18.1492	6.4582
DES	142.231501	31.4292	40.7682

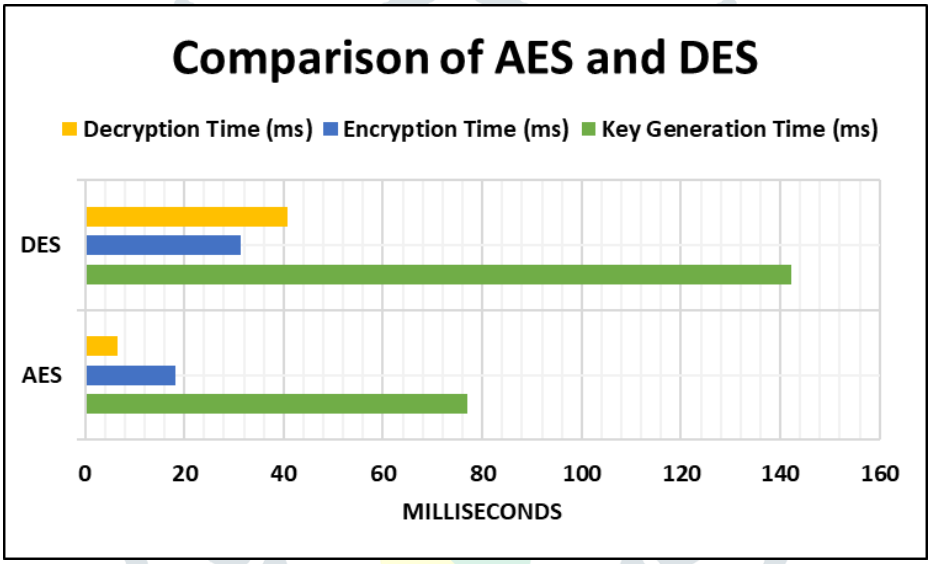

Figure 4. Comparison of AES and DES

Table 2. Throughput

Algorithms	Throughput (kb/sec)
AES	390.08
DES	100.06

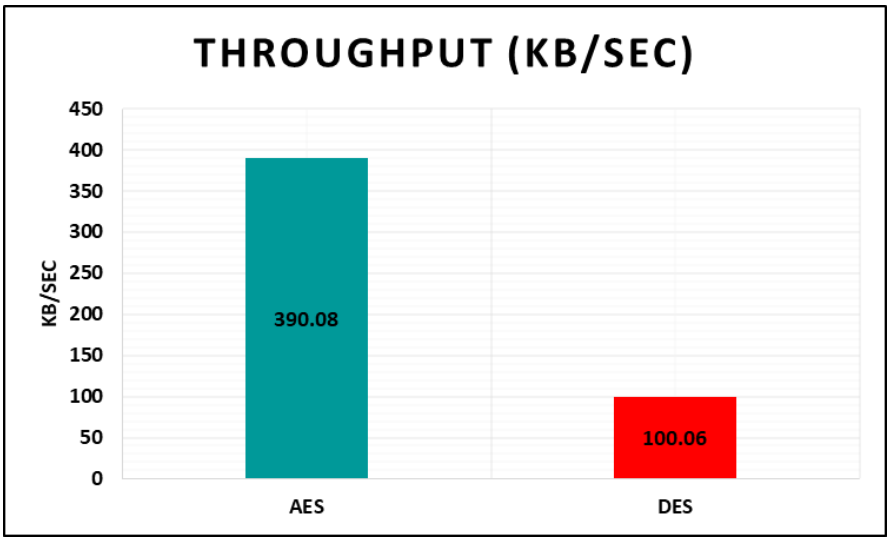

Figure 5. Throughput

Table 3. End to End Delay

Algorithms	End to End Delay (ms)
AES	52.134
DES	135.51

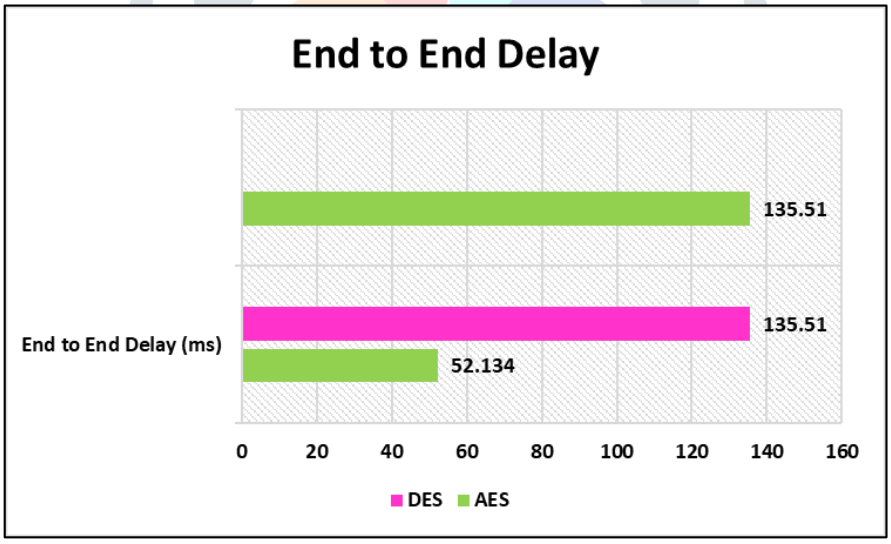

Figure 6. End to End Delay

IV. CONCLUSION

In this work, AES feature vector analysis depends on the specific implementation of the encryption algorithm and the encryption key used. This work is to ensure that the encrypted data can be securely transmitted and stored, and that appropriate access controls are in place to prevent unauthorized access. The experimental results showed that the AES algorithm achieves high throughput and minimum end to end delay.

REFERENCES

[1] Cheng Yan. (2011) 'Cybercrime Forensic System in Cloud Computing, Department of Computer Science and Engineering', East China University of Political Science and Law, Shanghai, China, IEEE 2011

[2] Do Hoon Kin and Hoh Peter In. (2008). 'Cyber Criminal Activity Analysis Models using Markov Chain for Digital Forensics', 2008 International Conference on Information Security and Assurance (ISA), pp 193-198.

[3] Feng-Yu Lin, Yeali S. Sun and Meng Chang Chen. (2014) Forensics Tracking for IP User Using the Markov Chain Model, Applied Mathematics and Information Sciences I.J., Vol 8 (3), pp 1343-1353
[4] Henry C. Lee and Elaine M. Pagliaro. (2013) 'Forensic Evidence and Crime Scene Investigation', Forensic Investigation J., Vol 1(2).
[5] José Antonio Maurilio Milagre de Oliveira and Marcelo Beltrão Caiado. (2013) 'Best Practice and Challenges for Process Efficiency of Investigations and digital Forensics', The Eighth International Conference on Forensic Computer Science 2013, pp 1-9.
[6] Mohsen Damshenas et al. (2012) 'Forensics Investigation Challenges in Cloud Computing Environments', International Conference on Cyber Security, Cyber Warfare and Digital Forensic (Cyber Sec
[7] Adams R., Val Hobbs and Graham R. (2015) 'The Advanced Data Acquisition Model (ADAM): A Process Model for Digital Forensic Practice', Digital Forensics, Security and Law I.J., Vol. 8(4), pp 234-238.
[8] Rogers M., Goldman J., Mislan R. and Wedge T. (2006) 'Computer Forensics Field Triage Process Model, Conference on Digital Forensics, Security and Law, 2006
[9] Mohan H. S. and Raji Reddy A. (2011) 'Performance Analysis of AES and MARS Encryption Algorithms', Computer Science Issues I.J., Vol. 8(4), pp 1-6.
[10] Mohan H. S. and Raji Reddy A. (2011) 'Performance Analysis of AES and MARS Encryption Algorithms', Computer Science Issues I.J., Vol. 8(4), pp 1-6.