World Research Journal of Computer Architecture
ISSN: 2278-8514 & E-ISSN: 2278-8522, Volume 1, Issue 1, 2012, pp.-11-15.
Available online at http://www.bioinfo.in/contents.php?id=97

CLOUD COMPUTING SECURITY WITH STEGANOGRAPHY AND CRYPTOGHRAPY AES ALGORTHM TECHNOLOGY

WAWGE P.U. AND RATHOD A.R.

Department Information Technology, J.D.I.E.T. College, Yavatmal, MS, India.
*Corresponding Author: Email- pwawge@gmail.com

Received: March 15, 2012; Accepted: May 11, 2012

Abstract- The growth of high speed computer networks and that of the Internet, in particular, has increased the ease of Information Communication. Ironically, the cause for the development is also of the apprehension - use of digital formatted data. In comparison with Analog media, Digital media offers several distinct advantages such as high quality, easy editing, high fidelity copying, compression etc. But this type of advancement in the field of data communication in other sense has hiked the fear of getting the data snooped at the time of sending it from the sender to the receiver. So, Information Security is becoming an inseparable part of Data Communication. In order to address this Information Security, Steganography plays an important role This paper includes review of existing methods and techniques for image based steganography and a new steganographic technique based on the file hybridization. In contrast to other methods of steganography where data embedded in image work on the principle of only one image file, the proposed method works on more than one image and focus on increased multilevel security. Although the cloud computing model is considered to be a very promising internet-based computing platform, it results in a loss of security control over the cloud-hosted assets.
This is due to the outsourcing of enterprise IT assets hosted on third-party cloud computing platforms. Moreover, the lack of security constraints in the Service Level Agreements between the cloud providers and consumers results in a loss of trust as well. Obtaining a security certificate such as ISO 27000 or NIST-FISMA would help cloud providers improve consumers trust in their cloud platforms' security. A brief explanation of the terms 128-AES and 256-AES: AES[1] is asymmetric key algorithm. AES encrypts and decrypts data in 128-bit blocks, using 128-, 192- or 256-bit keys. AES nomenclature for the different key sizes is AES-x, where x is the key size.) The Advanced Encryption Standard (AES)[2] specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher[3] that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encryptand decrypt data in blocks of 128 bits.
Keywords- privacy, security, cloud computing.

Citation: Wawge P.U. and Rathod A.R. (2012) Cloud computing security with Steganography and Cryptoghrapy AES algorthm Technology. World Research Journal of Computer Architecture, ISSN: 2278-8514 & E-ISSN: 2278-8522, Volume 1, Issue 1, pp.-11-15.

Introduction

The term "cloud" is analogical to "Internet". The term "Cloud Computing" is based on cloud drawings used in the past to represent telephone networks and later to depict Internet in .Cloud computing is Internet based computing where virtual shared servers provide software, infrastructure, platform, devices and other resources and hosting to customers on a pay-as-you-use basis. All information that a digitized system has to offer is provided as a service in the cloud computing model. Users can access these services available on the "Internet cloud" without having any previous know-how on managing the resources involved. Thus, users can concentrate more on their core business processes rather than spending time and gaining knowledge on resources needed to manage their business processes.
Cloud computing customers do not own the physical infrastructure; rather they rent the usage from a third-party provider. This helps them to avoid huge. They consume resources as a service and pay only for resources that they use. Most cloud computing

infrastructures consist of services delivered through common centers and built on servers. Sharing resources amongst can improve, as servers are not unnecessarily left idle, which can reduce costs significantly while increasing the speed of application development.

Cloud computing[4] is clearly one of today's most enticing technology areas due, at least in part, to its cost-efficiency and flexibility. However, despite the surge in activity and interest, there are significant, persistent concerns about cloud computing that are impeding momentum and will eventually compromise the vision of cloud computing as a new IT procurement model. In this paper, we characterize the problems and their impact on adoption. In addition, and equally importantly, we describe how the combination of existing research thrusts has the potential to alleviate many of the concerns impeding adoption. In particular, we argue that with continued research advances in trusted computing and computation-supporting encryption, life in the cloud can be advantageous from a business intelligence standpoint over the isolated alternative that is more common today.

Working of Cloud Computer

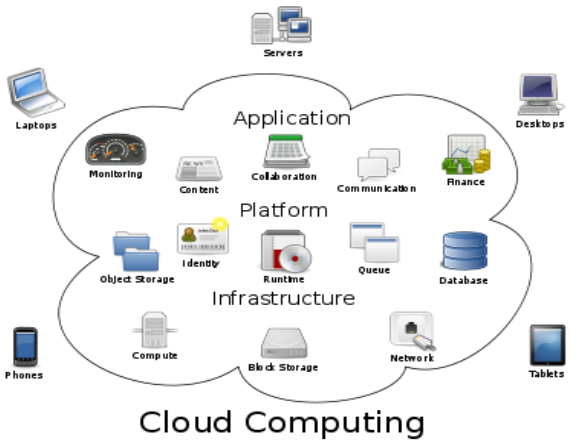

Fig. 1- cloud computing structure

Cloud computing is a marketing term for technologies that provide computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. A parallel to this concept can be drawn with the electricity grid wherein end-users consume power without needing to understand the component devices or infrastructure required to provide the service. Cloud computing describes a new supplement, consumption, and delivery model for IT services based on Internet protocols, and it typically involves provisioning of dynamically scalable and often virtualized resources. It is a byproduct and consequence of the ease-of -access to remote computing sites provided by the Internet. This may take the form of web-based tools or applications that users can access and use through a web browser as if the programs were installed locally on their own computers.

Fig 1[4] shows that Cloud computing providers deliver applications via the internet, which are accessed from web browsers and desktop and mobile apps, while the business software and data are stored on servers at a remote location. In some cases, legacy applications (line of business applications that until now have been prevalent in thin client Windows computing) are delivered via a screen-sharing technology, while the computing resources are consolidated at a remote data center location; in other cases, entire business applications have been coded using web-based technologies such as AJAX.

Steganography security

Steganography [5] comes from the Greek words Steganos (Covered) and Graptos (Writing). The term Steganography came into use in 1500's after the appearance of Trithemius book on the subject Steganographia The word Steganography technically means covered or hidden writing. Its ancient origins can be traced back to 440 BC. In ancient times, messages were hidden on the back of wax writing tables, written on the stomachs of rabbits, or tattooed on the scalp of slaves. Invisible ink has been in use for centuries for fun by children and students and for serious espionage by spies and terrorists The majority of today's steganographic systems uses multimedia objects like image, audio, video etc as cover media because people often transmit digital pictures over email and other Internet communication In modern approach, depending on the nature of cover object, steganography can be divided into five types: Text Steganography, Image Steganography, Audio Steganography, Video Steganography and Protocol Steganography. So, in the modern age so many steganographic techniques have been designed which works with the above concerned objects. With respect to Steganography there is a problem of unauthorized data access.

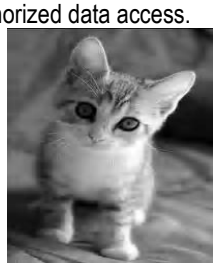

Fig. 2- The cover image

Fig. 3- The stego-image (after A is inserted)

Image steganography

To hide information, straight message insertion may encode every bit of information in the image or selectively embed the message in noisyll areas that draw less attention—those areas where there is a great deal of natural color variation. The message may also be scattered randomly throughout the image. A number of ways exist to hide information in digital media.

Common approaches include

- Least significant bit insertion
- Masking and filtering
- Redundant Pattern Encoding
- Encrypt and Scatter
- Algorithms and transformations

Each of these techniques can be applied, with varying Degrees of success.

Least Significant bit inserting

Least significant bit (LSB) insertion is a common and simple approach to embed information in an image file. In this method the LSB of a byte is replaced with an M's bit. This technique works

well for image, audio and video steganography. To the human eye, the resulting image will look identical to the cover object For example, if we consider image steganography then the letter A can be hidden in three pixels (assuming no compression). The original raster data for 3 pixels (9 bytes) may be

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

The binary value for A is 10000001. Inserting the binary value for A in the three pixels would result in

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)

On average, LSB requires that only half the bits in an image be changed.

Cryptography Security

Goals of AES algorithm

This standard specifies the Rijndael algorithm ([3] and [4]), a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. Rijndael was designed to handle additional block sizes and key lengths, however they are not adopted in this standard. Throughout the remainder of this standard, the algorithm specified herein will be referred to as "the AES algorithm." The algorithm may be used with the three different key lengths indicated above, and therefore these different "flavors" may be referred to as "AES-128", "AES-192", and "AES-256". This specification includes the following sections:

1. Definitions of terms, acronyms, and algorithm parameters, symbols, and functions;
2. Notation and conventions used in the algorithm specification, including the ordering and numbering of bits, bytes, and words;
3. Mathematical properties that are useful in understanding the algorithm;
4. Algorithm specification, covering the key expansion, encryption, and decryption routines;
5. Implementation issues, such as key length support, keying restrictions, and additional block/key/round sizes.

The standard concludes with several appendices that include step -by-step examples for Key Expansion and the Cipher, example vectors for the Cipher and Inverse Cipher, and a list of references. These are the honest entity when its operations abide by the system's specification. An honest entity can be curious: it attempts to infer knowledge from its own information (e.g., its secrets, state, and protocol communications). An honest entity becomes corrupt when it is compromised by an attacker, and hence, reveals its information at the time of compromise, and operates under the attacker's full control, possibly deviating from the specification.

Input and Output Conversion

The input and output for the AES algorithm each consist of sequences of 128 bits (digits with values of 0 or 1). These sequences will sometimes be referred to as blocks and the number of bits they contain will be referred to as their length. The Cipher Key for the AES algorithm is a sequence of 128, 192 or 256 bits. Other input, output and Cipher Key lengths are not permitted by this standard.

The bits within such sequences will be numbered starting at zero and ending at one less than the sequence length (block length or key length). The number *i* attached to a bit is known as its index and will be in one of the ranges 0 £ *i* < 128, 0 £ *i* < 192 or 0 £ *i* < 256 depending on the block length and key length (specified above).

Byte Comversion

The basic unit for processing in the AES algorithm is a byte, a sequence of eight bits treated as a single entity. The input, output and Cipher Key bit sequences described are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes For an input, output or Cipher Key denoted by *a*, the bytes in the resulting array will be referenced using one of the two forms, *an* or *a*[*n*], where *n* will be in one of the following ranges:

Key length = 128 bits, 0 £ *n* < 16;
Block length = 128 bits, 0 £ *n* < 16;
Key length = 192 bits, 0 £ *n* < 24;
Key length = 256 bits, 0 £ *n* < 32.

All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between braces in the order {*b*7, *b*6, *b*5, *b*4, *b*3, *b*2, *b*1, *b*0}. These bytes are interpreted as finite field elements using a polynomial representation:

$$,b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^{7} b_i x^i .$$

For example, {01100011} identifies the specific finite field element. It is also convenient to denote byte values using hexadecimal notation with each of two groups of four bits being denoted by a single character Bit Pattern Character Bit Pattern Character Bit Pattern Character Bit Pattern Character

0000 0 0100 4 1000 8 1100 c
0001 1 0101 5 1001 9 1101 d
0010 2 0110 6 1010 a 1110 e
0011 3 0111 7 1011 b 1111 f

Hence the element {01100011} can be represented as {63}, where the character denoting the four-bit group containing the higher numbered bits is again to the left. Some finite field operations involve one additional bit (*b*8) to the left of an 8-bit byte. Where this extra bit is present, it will appear as '{01}' immediately preceding the 8-bit byte; for example, a 9-bit sequence will be presented as {01}{1b}.

it is possible for a user to "frame" an honest user who later obtains the same IP address. Nonframeability holds true only against attackers with different identities (IP addresses).

Implementation

Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the State. The State consists of four rows of bytes, each containing *Nb* bytes, where *Nb* is the block length divided by 32. In the State array denoted by the symbol *s*, each individual byte has two indices, with its row number *r* in the range 0 £ *r* < 4 and its column number *c* in the range 0 £ *c* < *Nb*. This allows an individual byte of the State to be referred to as either *sr*,*c* or *s*[*r*,*c*]. For this standard, *Nb*=4, i.e., 0 £ *c* < 4 . At the start of the Cipher and Inverse Cipher described in Sec. 5, the

input – the array of bytes
*in*0, *in*1, … *in*15 – is copied into the State array as illustrated The Cipher or Inverse Cipher operations are then conducted on this State array, after which its final value is copied to
the output – the array of bytes *out*0, *out*1, … *out*15.

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])              // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                          // See Sec. 5.1.1
      ShiftRows(state)                         // See Sec. 5.1.2
      MixColumns(state)                        // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Key expansion

The AES algorithm takes the Cipher Key, *K*, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of *Nb* (*Nr* + 1) words: the algorithm requires an initial set of *Nb* words, and each of the *Nr* rounds requires *Nb* words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [*wi*], with *i* in the range 0 £ *i* < *Nb* (*Nr* + 1).

The expansion of the input key into the key schedule proceeds according to the pseudo code in SubWord() is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function RotWord() takes a word [*a*0,*a*1,*a*2,*a*3] as input, performs a cyclic permutation, and returns the word [*a*1,*a*2,*a*3,*a*0].

The round constant word array, Rcon[i], contains the values given by [*xi-1*,{00},{00},{00}], with *x i*-1 being powers of *x* (*x* is denoted as {02}) in the field GF(28), as discussed in Sec. 4.2 (note that *i* starts at 1, not 0). it can be seen that the first *Nk* words of the expanded key are filled with the Cipher Key. Every following word, w[i], is equal to the XOR of the previous word, w[i-1], and the word *Nk* positions earlier, w[i-*Nk*]. For words in positions that are a multiple of *Nk*, a transformation is applied to w[i-1] prior to the XOR, followed by an XOR with a round constant, Rcon[i]. This transformation consists of a cyclic shift of the bytes in a word (RotWord()), followed by the application of a table lookup to all four bytes of the word (SubWord()). It is important to note that the Key Expansion routine for 256-bit Cipher Keys (*Nk* = 8) is slightly different than for 128- and 192-bit Cipher Keys. If *Nk* = 8 and i-4 is a multiple of *Nk*, then SubWord() is applied to w[i-1] prior to the XOR.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Note that *Nk*=4, 6, and 8 do not all have to be implemented; they are all included in the conditional statement above for conciseness. Specific implementation requirements for the Cipher Key are presented in Sec. 6.1.

Preliminaries

Cryptographic Primitives

Nymble uses the following building blocks

- Secure cryptographic hash functions. These are Onaway and collision-resistant functions that resemble random oracles [6]. Denote the range of the hash functions by H.
- Secure message authentication (MA) [7]. These consist of the key generation (MA.KeyGen), and the message authentication code (MAC) computation (MA.Mac) algorithms. Denote the domain of MACs by M.
- Secure symmetric-key encryption (Enc) [8]. These consist of the key generation (Enc.KeyGen), encryption (Enc.Encrypt), and decryption (Enc.Decrypt) algorithms. Denote the domain of ciphertexts .
- Secure digital signatures (Sig) [9]. These consist of the key generation (Sig.KeyGen), signing (Sig.Sign), and verification (Sig.Verify) algorithms.

Advantages

Side-channel attacks

In this current implementation does not fully protect against side-channel attacks, we mitigate the risks. This paper has implemented various algorithms in a way that their execution time leaks little information that cannot already be inferred from the algorithm's output. Also, since a confidential channel does not hide the size of the communication, we have constructed the protocols so that each kind of protocol message is of the same size regardless of the identity or current legitimacy of the user.

References

[1] AESpageavailablevia, *http://www.nist.gov/CryptoToolkit*.

[2] Daemen J. and Rijmen V. (1999) *AES Proposal, Rijndael*, AES Algorithm Submission.
[3] Daemen J. and Rijmen V. *The block cipher Rijndael*, *Smart Card research and Applications*, LNCS 1820, 288-296.
[4] *http://www.luitinfotech.com/kc/what-is-cloud-computing.pdf*.
[5] Chandra Sekhara Reddy T., Prasad D. and Venkateswara Reddy B. *IJCTA journal*.
[6] Bellare M. and Rogaway P. (1993) *First ACM Conf. Computer and Comm. Security*, 62-73.
[7] Bellare M., Canetti R. and Krawczyk H. (1996) *Proc. Ann. Int'l Cryptology Conf*. (*CRYPTO*), 1-15.
[8] Bellare M. Desai A., Jokipii E. and Rogaway P. *Proc. Ann. Symp. Foundations in Computer Science* (*FOCS*), 394-403.
[9] Goldwasser S., Micali S. and Rivest R.L. (1988) *SIAM J. Computing*, 17(2), 281-308, 1988.