INTERNAL SECURITY IN METROPOLITAN AREA NETWORK USING KERBEROS

[1]Sushmitha M.S, [2] Dr.Mahesh Kaluti

[1]P.G Research Scholars, Department of CSE, P.E.S College of Engineering
[2]Associate Professor, Dept of CSE, P.E.S College of Engineering

---

Abstract— *The main aim of current computer network security systems is to secure the network from external attackers; however, securing the network from its own users is still an unattended problem. In metropolitan area networks, the risk of having internal attacks is high because of their network topologies, due to high traffic and the amount of network users. This work proposes a new approach to identify whether a network user is having a normal behavior or not, by using Kerberos. Kerberos has a strong encrypted security protocol that deals with establishing a secure connection using more encryptions steps between the client(s) and the server. Kerberos issues ticket to the client which is used to access the services of a particular server along with a lifetime. There are number of encryption processes that take place before issuing ticket. After obtaining mutual authentication, client can communicate with the server by sending messages. These messages are encrypted using AES algorithm. The document presents an experiment how secure communication happens between two organization; that is in metropolitan area network.*

Keywords—Metropolitan area network, Kerberos, AES algorithm

I. INTRODUCTION

On current computer networks, traditional security methods like firewalls, access control systems and simple Intrusion Detection Systems (IDS) are no longer enough to protect computer systems; day after day, intruders find new ways to attack computers and systems. Nowadays, attackers use advanced techniques to go undetected by IDSs, including the following: IP address spoof, encrypted payload, or even social engineering techniques. A common symptom of an attack using these techniques is that the host under attack is experiencing unexpected network behavior. This is why the use of profiles to determine whether the user is having the expected behavior or not has become necessary as a new way to detect intrusions.

A major goal of current computer network security systems is to protect the network from outside attackers; however, protecting the network from its own users is still an unattended problem. This paper focus on providing internal network security to the systems using Kerberos. Kerberos is a protocol for authenticating service requests between trusted hosts across an untrusted network, such as the internet. The Kerberos protocol defines how clients interact with a network authentication service. Clients obtain tickets from the Kerberos Key Distribution Centre (KDC), and they present these tickets to servers when connections are established. Kerberos tickets represent the client's network credentials.

II. METHODOLOGY

Kerberos is a computer network authentication protocol that works on the basis of tickets. The Kerberos authentication protocol provides a mechanism for mutual authentication between a client and a server before a network connection is opened between them. The protocol assumes that initial transactions between clients and servers take place on an open network — an environment where most clients and many servers are not physically secure and packets travelling along the network can be monitored and modified at will. In other words, the protocol is designed for an environment that is much like today's Internet, where an attacker can easily pose as either a client or a server and can readily eavesdrop on or tamper with communications between legitimate clients and servers.

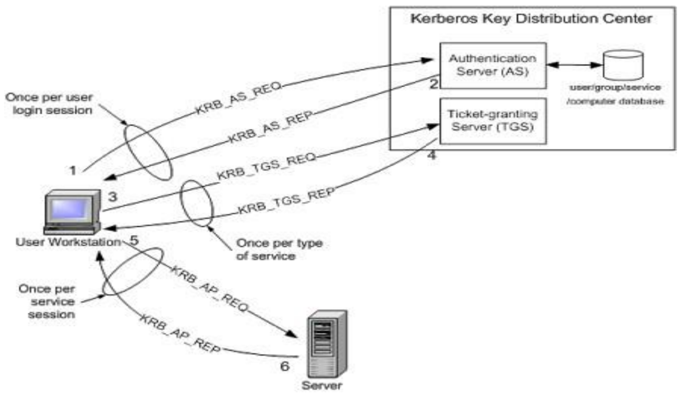

Kerberos Key Distribution Center
Authentication Server (AS)
user/group/service /computer database
Ticket-granting Server (TGS)
Once per user login session
KRB_AS_REQ
KRB_AS_REP
KRB_TGS_REQ
KRB_TGS_REP
1
2
3
4
5
6
User Workstation
Once per type of service
Once per service session
KRB_AP_REQ
KRB_AP_REP
Server

Fig : Working of Kerberos

A. User Client-based Logon

1. A user enters a username and password on the client machine(s).
2. The client transforms the password into the key of a symmetric cipher. This either uses the built-in key scheduling, or a one-way hash, depending on the cipher-suite used.

B. Client Authentication

1. The client sends a clear text message of the user ID to the AS (Authentication Server) requesting services on behalf of the user. (Note: Neither the secret key nor the password is sent to the AS.)
2. The AS checks to see if the client is in its database. If it is, the AS generates the secret key by hashing the password of the user found at the database (e.g., Active Directory in Windows Server) and sends back the following two messages to the client:
   - Message A:Client/TGS Session Key encrypted using the secret key of the client/user.
   - Message B: Ticket-Granting-Ticket (TGT, which includes the client ID, client network address, ticket validity period, and the client/TGS session key) encrypted using the secret key of the TGS.

3. Once the client receives messages A and B, it attempts to decrypt message A with the secret key generated from the password entered by the user. If the user entered password does not match the password in the AS database, the client's secret key will be different and thus unable to decrypt message A. With a valid password and secret key the client decrypts message A to obtain the Client/TGS Session Key. This session key is used for further communications with the TGS. (Note: The client cannot decrypt Message B, as it is encrypted using TGS's secret key.) At this point, the client has enough information to authenticate itself to the TGS.

C. Client Service Authorization

1. When requesting services, the client sends the following messages to the TGS:
   - Message C: Composed of the TGT from message B and the ID of the requested service.
   - Message D: Authenticator (which is composed of the client ID and the timestamp), encrypted using the Client/TGS Session Key.
2. Upon receiving messages C and D, the TGS retrieves message B out of message C. It decrypts message B using the TGS secret key. This gives it the "client/TGS session key". Using this key, the TGS decrypts message D (Authenticator) and compare client ID from message C and D, if they match server sends the following two messages to the client:
   - Message E: Client-to-server ticket (which includes the client ID, client network address, validity period and Client/Server Session *Key*) encrypted using the service's secret key.
   - Message F: Client/Server Session Key encrypted with the Client/TGS Session Key.

D. Client Service Request

1. Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the Service Server (SS). The client connects to the SS and sends the following two messages:
   - Message E from the previous step (the client-to-server ticket, encrypted using service's secret key).
   - Message G: a new Authenticator, which includes the client ID, timestamp and is encrypted using Client/Server Session *Key*.
2. The SS decrypts the ticket (message E) using its own secret key to retrieve the Client/Server Session *Key*. Using the sessions key, SS decrypts the Authenticator and compare client ID from message E and G, if they match server sends the following message to the client to confirm its true identity and willingness to serve the client:
   - Message H: the timestamp found in client's Authenticator (plus 1 in version 4, but not necessary in version 5[3][4]), encrypted using the Client/Server Session Key.
3. The client decrypts the confirmation (message H) using the Client/Server Session Key and checks whether the timestamp is correct. If so, then the client can trust the server and can start issuing service requests to the server.
4. The server provides the requested services to the client.

E. AES Algorithm

Once the user becomes authenticated and authorized, secure connection is established with the server. Then the user can send the messages (data) to the server in encrypted form and decrypted at receiver side. Advanced Encryption Standard (AES) algorithm is used for encrypting and decrypting the messages. The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption

Standard (AES). It is found at least six time faster than triple DES.

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

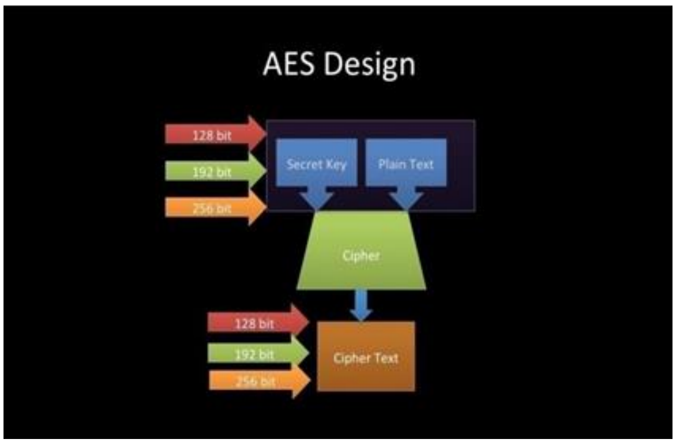
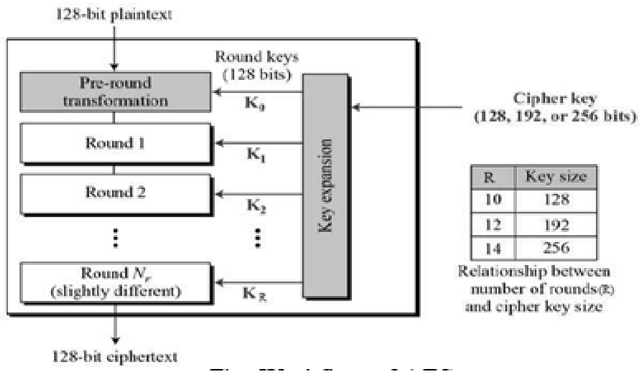


Fig: AES Design

Working of AES algorithm

The schematic of AES structure is given in the following illustration



R	Key size
10	128
12	192
14	256

Fig: Workflow of AES

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

F. EncryptionProcess

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below.

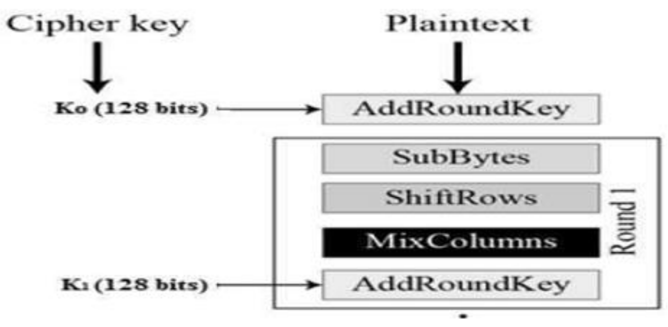


Fig : Round 1

1. High-level description of the algorithm

1. Key Expansions—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. Initial Round
    1. AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.
3. Rounds
    1. SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
    2. ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
    3. MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
    4. AddRoundKey

4. Final Round (no MixColumns)
    1. SubBytes
    2. ShiftRows
    3. AddRoundKey.

G. The SubBytes Step

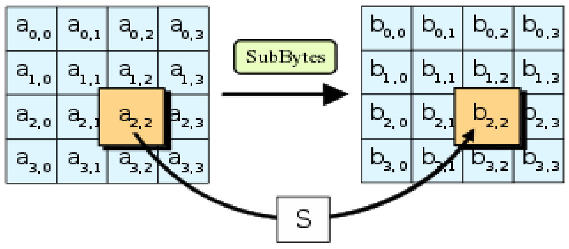


Fig: In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, *S*; $b_{ij} = S(a_{ij})$.

In the SubBytes step, each byte in ai,j the *state* matrix is replaced with a SubByte $S(a_{i,j})$ using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF(2^8), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), i.e.,, and also any opposite fixed points, i.e. While performing the decryption, the InvSubBytes step (the inverse of SubBytes) is used, which requires first taking the inverse of the affine transformation and then finding the multiplicative inverse.

H. The ShiftRows Step

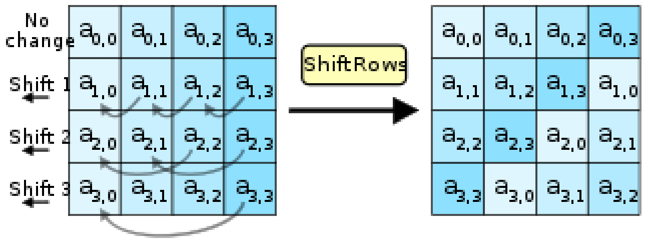


Fig : In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

The ShiftRows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For blocks of sizes 128 bits and 192 bits, the shifting pattern is the same. Row "n" is shifted left circular by "n-1" bytes. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. (Rijndael variants with a larger block size have slightly different offsets). For a 256-bit block, the first row is unchanged and the shifting for the second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectively—this change only applies for the Rijndael cipher when used with a 256-bit block, as AES does not use 256-bit blocks. The importance of this step is to avoid the columns being encrypted independently, in which case AES degenerates into four independent block ciphers.

I. The MixColumns Step

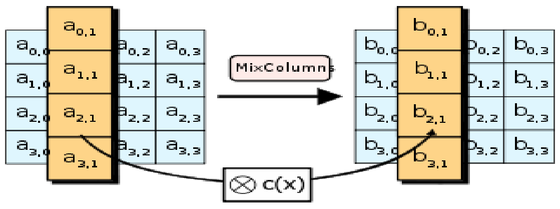


Fig : In the MixColumns step, each column of the state is multiplied with a fixed polynomial

In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes.

Together with ShiftRows, MixColumns provides diffusion in the cipher.

During this operation, each column is transformed using a fixed matrix (matrix left-multiplied by column gives new value of column in the state):

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Matrix multiplication is composed of multiplication and addition of the entries. Entries are 8 bit bytes treated as coefficients of polynomial of order x^7. Addition is simply XOR. Multiplication is modulo irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. If processed bit by bit then after shifting a conditional XOR with $1B_{16}$ should be performed if the shifted value is larger than FF_{16} (overflow must be corrected by subtraction of generating polynomial). These are special cases of the usual multiplication in $\mathrm{GF}(2^8)$.

In more general sense, each column is treated as a polynomial over $\mathrm{GF}(2^8)$. and is then multiplied modulo with a fixed polynomial $z^4 + 1$

$$c(z) = 03_{16} \cdot z^3 + z^2 + z + 02_{16}.$$

The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from $\mathrm{GF}(2)[x]$. The MixColumns step can also be viewed as a multiplication by the shown particular MDS matrix in the finite field $\mathrm{GF}(2^8)$. This process is described further in the article Rijndael MixColumns.

J. The AddRoundKey Step

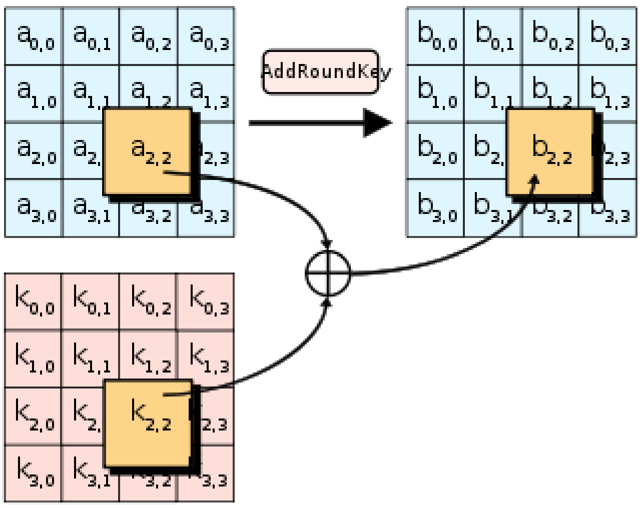


Fig : In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation (⊕).

In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

III. CONCLUSION

This paper presents the concept of using the Kerberos to improve the security in metropolitan area network. Kerberos provides secure authentication and authorization functionality for the client. After the client is authenticated and authorized, client can communicate and transfer the messages to the server using AES algorithm. Thus AES algorithm provides secure means of information transformation to the server.

REFERENCES:

[1] Alvaro Parres-Peredo, Ivan Piza-Davila, Francisco Cervantes "Towards a User Network Profiling for Internal Security using Top-K Rankings Similarity Measures", Oct 23,2017.

[2] Jiangzhuo Chen, V. S. Anil Kumar, Madhav V. Marathe, Ravi Sundaram, Mayur Thakur, Sunil Thulasidasan "A study of the structure and vulnerabilities of metropolitan area networks", 2016 8th International Conference on Communication Systems and Networks (COMSNETS), Mar 24, 2016.

[3] Fadi Al-Ayed, Hang Liu "Synopsis of Security: Using Kerberos Method to Secure File Transfer Sessions", 2016 International Conference on Computational Science and Computational Intelligence,Mar 20,2017.

[4] Nan Zhangl, Xiaoyu Wul, Cheng Yangl, Yinghua Shenl, Yingye Chengl "A lightweight authentication and authorization solution based on Kerberos",Advanced Information Management, Communicates, Electronic and Automation Control Conference(IMCEC), 2016 IEEE, Mar 02, 2017.

[5] Khaled Bakour, Gulesin Sena Das, H.Murat Unver "An Intrusion Detection System Based on a Hybrid Tabu-Genetic Algorithm", (UBMK'17) 2nd International Conference on Computer Science and Engineering, Nov 02, 2017.

[6] https://en.m.wikipedia.org/wiki/kerberos_(protocol).

[7] https://en.m.wikipedia.org/wiki/Advanced_Encryption_Standard.