Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 01, January -2018

A FRAMEWORK USING SSL AND TLS PROTOCOLS IN BIG DATA ARCHITECTURE

[1]Ms. YOGA LAKSHMI.P, [2]Ms. MUTHU LAKSHMI. P, [3]Ms. AARTHI.E

** Department of Computer Science,*
FSH, SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,
Chennai, India.

ABSTRACT:- *Secure Socket Layer (SSL)protocol and Transport Layer Security(TLS) protocol are security protocols applied on the web, distributed systems, and private LANs to secure link between the server and client. In theory SSL and TLS uses public key encryption and thus public key encryption can provide flexible and strong way to authenticate the users in a distributed system such as distributed big data. For this reason, SSL and TSL are chosen as a protocol to argue the topic of this paper. SSL and TSL guarantee the integrity of the data served by various nodes without requiring the changes to the client. This paper serves as a resource to the user of the big data to know about the common protocols in the distributed data security. Through SSL and TLS are not only the secure protocols, but it was involved in sensitive data like password and financial transactions. Moreover, this protocols of choice will put into arguing why the most deployed user authentication mechanism like Kerberos will fail to provide protection against man in the middle attack in big data distributed system. Finally, this paper shall put into display how SSL and TLS can be used to provide secure transactions in the big data Distributed system and to give a complete framework in security management system that can be incorporated using SSL and TLS*

Keywords: *Big data, framework, MIMA, security*

Introduction

The big data has received considerable attention since it gives an excellent opportunity to mine knowledge from a massive amount of data. However the big data paradigm also brings some security challenges, as such data confidentiality, integrity and available are hindered. One of the major security challenge called the man in the middle which will also be part of the discussion in this paper is an attack that uses the security that undermines the flow of data from the client to a server in the big data environment.TLS/SSL can receive data from any application layer protocol, but usually, the protocol is https. That is why it is a suitable choice for the big data paradigm, especially when data was accessed over the internet. This paper will present a different approach to securing the big data environment by using a comparative study of secure socket layer protocol (SSL) and transport layer protocol (TLS) in other to safeguard the big data paradigm. It is intended to use G-Hadoop security model as a sample in this paper[7]. This paper will develop a new model that will mirror out our comparative study. This paper only presents a comparative model. Implementation of this model will not be reviewed in this paper.
The aim of the paper is to give a comparative study of SSL/TLS in a proposed security model and to guarantee that by using SSL/TLS to access data in a big data distributed environment, can help to achieve the four chief principle of security, principle of confidentiality, integrity, authentication, and non-reputation [10]. This paper aims to prove that SSL/TLS can be used in the G-Hadoop big data environment together to the traditional CA server to make a more stable secure big data environment. This paper shall also prove that man in the middle attack (MITM) can be better prevented when the CA server and SSL/TLS are used in Big data distributed environment. The model aimed at will have a better countermeasure against MITM.[7]

SSL is an Internet protocol for exchanging information between the web browser and a web server. Netscape created it in 1994; SSL comes in three versions which are version 2, 3, 3.1, the most famous of them is version 3 which was released in 1995. On the other hand, TLS is the IETF standard version of SSL with a slight difference between them but mind catching. The current version of TLS is TLSv1.0 [8].

SSL can be considered as an additional layer of the TCP/IP protocol suit, and the SSL layer is between the application layer and the transport layer.[8]

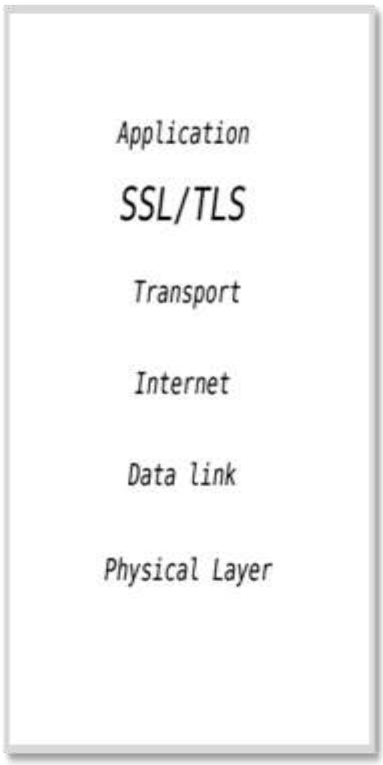


Figure1 Diagram showing the position of SSL and TLS in the TCP

As indicated in the above illustration, the application layer data is not passed directly to the transport layer. Instead, the application layer data is passed to the SSL or a TLS layer. The layer SSL or TLS layer performs encryption on the data received from the application layer and also adds its information header called SSL header to the encrypted data and the remaining layer also adds their corresponding header which is being passed over the network to the different computer.

On the recovery side, the process happens pretty similarly to how it happens in the case of a standard TCP connection until it reaches the SSL layer. The SSL layer decrypts the encryption data gives the plain text back to the application layer of the receiving computer.

A. Services provided by SSL/TLS

SSL provides several services to the data received by the application layer

Fragmentation: First SSL divided the data into block of 214 bytes or less

Compression: Each fragment of the data is compressed using one of the lossless compression methods negotiated between the client and server.

Message Integrity: to preserve the integrity of the data, SSL uses keyed hash function to create an MAC.

Confidentiality: To provide privacy, the data and the MAC are encrypted using symmetric key cryptography.

Framing: A header is added to the encrypted payload. The payload is then passed to a reliable transport layer protocol.

Digital Certificates

SSL/ TLS uses what is called *Digital certificate*. Digital certificates are electronic data which is used to authenticate the people and the resources in the web. It is used to enable the security between the client and the server which use encryption. While creating the certificate, the information is digitally signed by the digital authority. The signature on the certificate is like a detection seal, and packaging – any tampering with the contents is easily found.

The certificates are based on asymmetric cryptography which has a pair of keys, used for encrypting and decrypting with this type of cryptography, key work in the pairs of matched keys. In cryptographic systems, the term key means a value used by an algorithm to alter information, making that information secure and accessible only to individuals who have the related key to recover the information.[10]

A digital certificate can securely encapsulate your identity, as verified by a trusted third party, with your public key.

B. How SSL/TLS works

The SSL protocol uses digital certificates to create a secure, confidential communications “pipe” between two entities. Data transmitted over an SSL connection cannot be tampered with or forged without the two parties becoming immediately aware of the tampering.

Diffie HellMan protocol

Two parties can create a symmetric session key without the need of KDC. Before the create a symmetric key, the two parties need to choose two numbers p and g. The first number is a large prime number on the order of 300 decimal digits (1024 bits). The second number g is a generator of order p-1 in the group $<Z_{p*,}$ X> [8].
K is the symmetric key for the session.

$$K = (g^x \bmod p)^y \bmod p = (g^y \bmod p)^x \bmod p = g^{xy} \bmod p$$

Diffie HellMan is susceptible to man in the middle attack but by using the station to station key agreement that uses the digital signature with the public key certificate which is equivalent to the certificate verification when using an SSL and TLS.[10]
The Fixed Diffie HellMan uses parameter (g and p) each entity can create a fixed Diffie hellMan half key (g^x). Each half key is inserted into a certificate verified by the certificate authority.[10]

I. Proposed work

A. Big Data security model (G-Hadoop)

Apache Hadoop [1] is one of the well-known Map-reduce implementation in the big data paradigm [2, 3, 4] and has been used in many research areas around the globe by various company [6]. G-hadoop implementation is based on the master/salve architecture by a client submitting work to the master with one job tracker as the master and multiple Task Tracker acting in the role of the slave. G-Hadoop uses a special file system which is known as Hadoop Distributed File system, to manage the input /output data of the Map-Reduce application.

The G-Hadoop system enables massive data processing across multiple clusters and data centers. G-Hadoop targets distributed file system such as a grid infrastructure [8,7] in other to share data across multiple administrative domains. Not long ago grid society replaces G-Hadoop native distributed file system with the G-Farm Grid File system [19]. The G-Farm Grid file system was specifically designed to meet the requirements of providing a global virtual file system across multiple administrative domains. It is optimized for wide-area operation and offers the required location awareness to allow data-aware scheduling among clusters.[9]

The current G-Hadoop system reuses the Hadoop system mechanism for user authentication and job submission, which is designed for single cluster environment [7]. This mechanism applies the secure shell (SSH) protocol to establish a secure connection between the user and the target cluster. This procedure is not enough to have a perfect security for this kind of transaction and hence not suitable for a distributed environment. SSL/TLS are designed to be different handle authentication on a different cluster by assigning different certificate and private key for every cluster in a distributed environment. To make a strong case on SSL uses six cryptographic secrets. This paper proposes the use of Fixed Diffie-Hellman key algorithm; this is because two parties do not directly exchange the half-keys; the CA sends the half-keys in an RSA or DSS special certificate and the use it to calculate the pre-master plus the server half-key in the calculation. The server does the same but in the order direction. Besides SSL uses two variety of hash algorithm to provide data integrity such include MD5, SHA-1.[9]

B. The G-Hadoop security model integrated with SSL/TSL

The current G-Hadoop security model comprises of CA server 1, CA server 2, SSL/TLS, clusters and a master node. It will be seen in the diagram below; that the user and master node don't communicate directly, instead they communicate to the to a CA server that is installed on them in order to get their half keys since the proposed SSL and TLS key exchange algorithm for this model proposed only the use of Fixed Diffie-Hellman which does not allow direct communication between server and client. In this security model, the CA server 2 is an important component that issues proxy credentials to master node and slaves only. If there are many CA's in the environment each responsible for creating, sorting, issuing and receiving a limited number of certificates, to accomplish this kind of task, the trust model is used to define how a user can verify a certificate obtained from a CA. Such model to deploy can include the Hierarchical Model or the Mesh Model, which can be applied to CA server 1.

To achieve message integrity SSL need six cryptographic secrets, four keys, and two IVs, the client needs one key for the message authentication(HMAC), one key for encryption and one IV(initial vector) for block encryption. The master node needs the same. SSL requires that the key for one direction be different from the other direction. The parameters generated are as follows.

1) The client and master node exchange two random numbers, one is created by the client and the other by the master node.
2) The client and server exchange one pre-master secret using one of the key exchange algorithms
3) A 48-byte master secret is created from the premaster secret by applying two hash Function (SHA and MD5)
4) The master secret is used to create the variable-length key material by applying the same set of hash function and prepending with different constant.
5) Six different Keys are exchanged from the key material.

How TLS achieve message integrity in explained in section

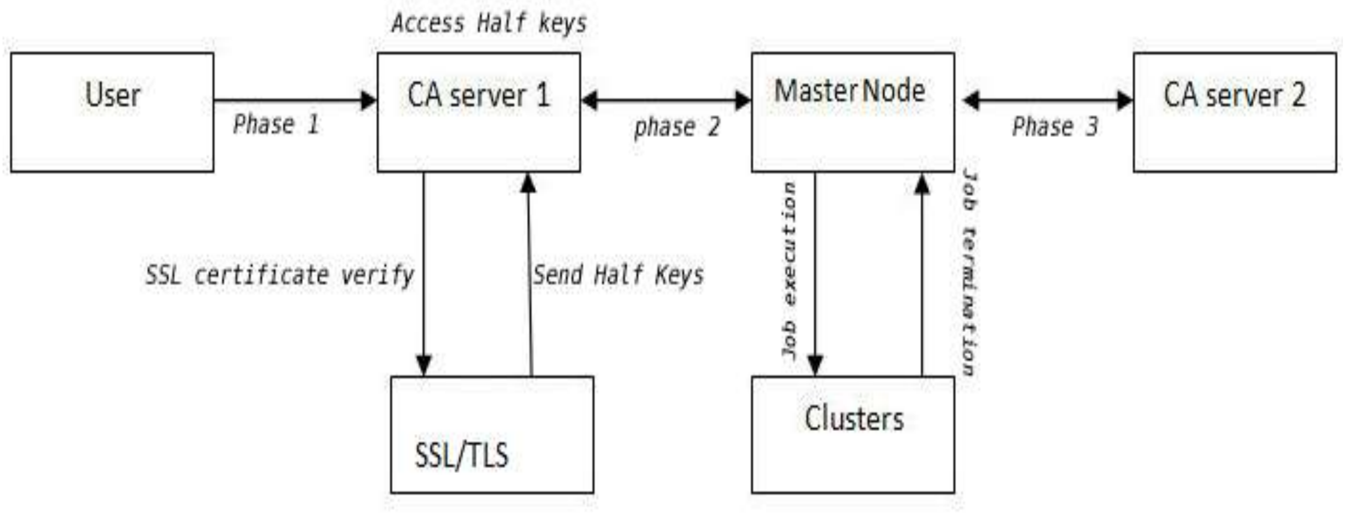


Figure 2: G-Hadoop security model with SSL/TSL

5.2 How does the server and the client exchange keys and authenticate them self in our G-Hadoop?
At first instance, the server announces their security capabilities and chooses those that are convenient for both (in our case the fixed differ hellman). In this phase, a session ID is established, and the cipher suite is chosen and also agreed on a particular compression method if any. in the end two random numbers a chosen by both participants for creating a master secret. Let's divide the exchange keys authenticate into two types

1) **Server key exchange and Authentication**

In this phase, the server authenticates itself if needed. The sender may send its certificate, public key and may also request certificates from the client. After a certificate message, the server sends a server Key-exchange message that includes its contribution to the pre-master secret. Since our security model requires the use of Fixed Diff-e-Hellman, the server sends an RSA or DSS digital signature that indicates it registered DH half-key. The certificate is signed by CA's private key and can be verified by the client using CA's public key. In other words, the CA is authenticated to the client, and the CA claims that the half-key belongs to the server.

2) **Client key exchange and Authentication**

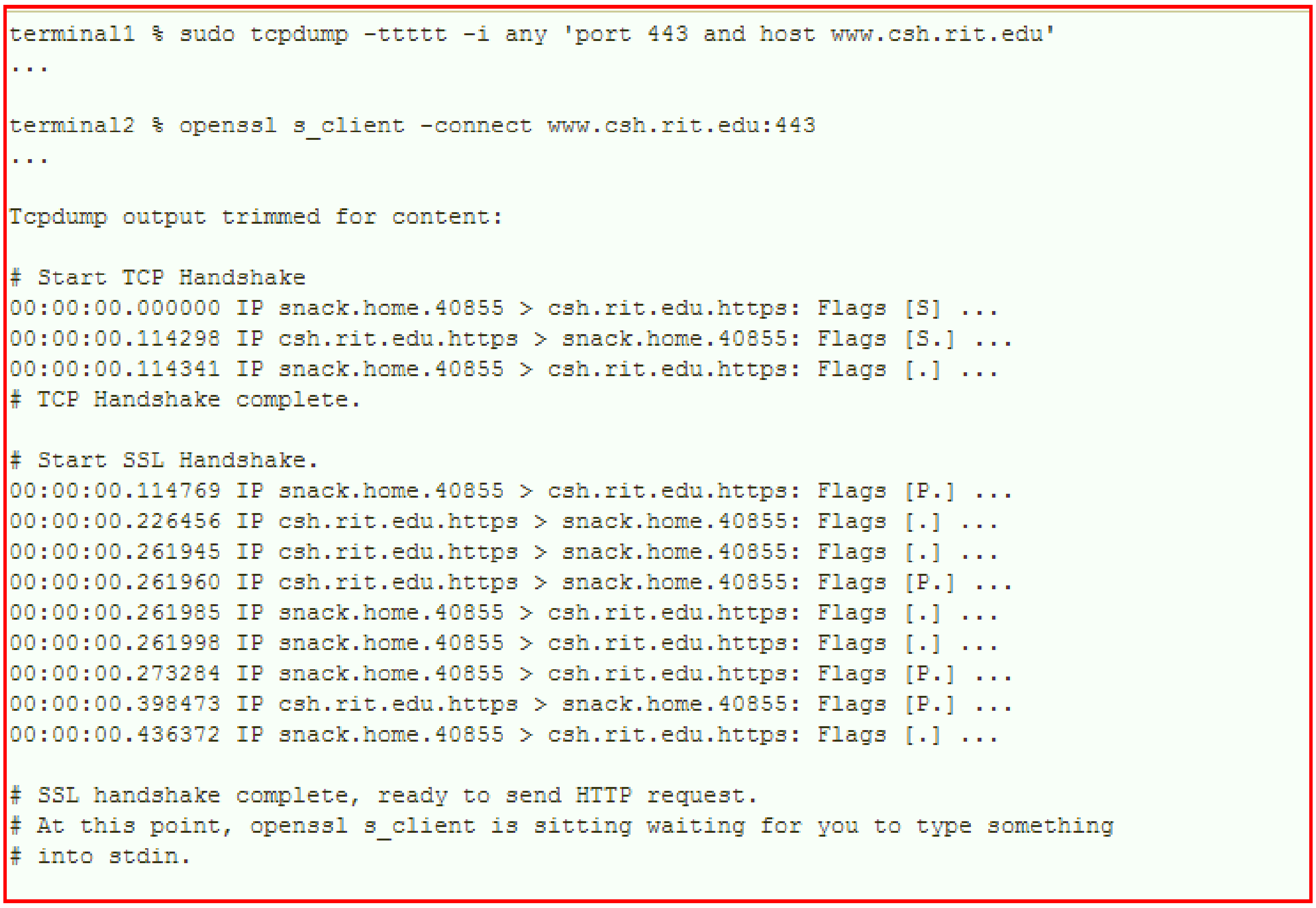
This phase is designed to authenticate the client. Up to three messages can be sent from the client to the server

Message 1: Certificate: The client sends s a certificate message. This includes a chain of certificates that certify the client. This message is sent only if the server has requested a certificate.
Message 2: clientKeyExchange: After sending the certificate message, the client sends the clientKeyExchange message which includes the pre-master secret. The content of this message is based on the key-exchange algorithm used. (which is the fixed Diffie-Hellman Key).
Message 3: Certificate Verify: if the client has sent a certificate declaring that it owns the public key in the certificate, it needs to prove that it was known's the corresponding private key. This is needed to thwart an imposter who sends the certificate and claims that it comes from the client. This is done

by creating a message and signing it with the private key. In using a Fixed Diffie-Hellman, the client sends a Diffie-Hellman certificate in the first message.

C. SSL handshake latency

In work by Jorden Sissel, which he posted on his website on 04 June 2010 when TCP handshake is compared with the SSl hand shake, it is observed that the SSL handshake is longer than TCP handshake. But the question is why? Network and cryptography



```
terminal1 % sudo tcpdump -ttttt -i any 'port 443 and host www.csh.rit.edu'
...

terminal2 % openssl s_client -connect www.csh.rit.edu:443
...

Tcpdump output trimmed for content:

# Start TCP Handshake
00:00:00.000000 IP snack.home.40855 > csh.rit.edu.https: Flags [S] ...
00:00:00.114298 IP csh.rit.edu.https > snack.home.40855: Flags [S.] ...
00:00:00.114341 IP snack.home.40855 > csh.rit.edu.https: Flags [.] ...
# TCP Handshake complete.

# Start SSL Handshake.
00:00:00.114769 IP snack.home.40855 > csh.rit.edu.https: Flags [P.] ...
00:00:00.226456 IP csh.rit.edu.https > snack.home.40855: Flags [.] ...
00:00:00.261945 IP csh.rit.edu.https > snack.home.40855: Flags [.] ...
00:00:00.261960 IP csh.rit.edu.https > snack.home.40855: Flags [P.] ...
00:00:00.261985 IP snack.home.40855 > csh.rit.edu.https: Flags [.] ...
00:00:00.261998 IP snack.home.40855 > csh.rit.edu.https: Flags [.] ...
00:00:00.273284 IP snack.home.40855 > csh.rit.edu.https: Flags [P.] ...
00:00:00.398473 IP csh.rit.edu.https > snack.home.40855: Flags [P.] ...
00:00:00.436372 IP snack.home.40855 > csh.rit.edu.https: Flags [.] ...

# SSL handshake complete, ready to send HTTP request.
# At this point, openssl s_client is sitting waiting for you to type something
# into stdin.
```

For the test, a TCP dump is being used on sniff https traffic and use OpenSSL s_client to connect to the HTTP server over SSLis more than enough to connect. Start TCP dump first, and then run OpenSSL s_client.

No matter how fast your SSL accelerate (hardware load balancer, etc.), if your SSL endpoints are not near the user node, then your first connection will be slow. As shown above, 22ms for the crypto piece of SSL handshake, which means 300ms of the SSL portion above, were likely due to network latency and some other overhead
Once SSL is established, though, it switches to a block cipher which is much faster and the resource overhead is pretty tiny by comparison. In summary, Using SSL incurs 3.5 x latency overhead for each handshake, but afterward, it's fast like plain TCP. [11]

Transport Layer Security

The Internet Engineering Task Force (IETF) created (Transport Layer Security) as the successor to SSL. After SSL 3.0, the next version was introduced as TLS 1.0 in 1999. Then, in 2006, a next version named as

TLS 1.1 was introduced. Then, in 2008, the advanced configuration and bug fixes were included, and TLS 1.2 was introduced. Currently, TLS 1.2 is the latest available Transport Layer Security version. Just as SSL, TLS also provide security services such as confidentiality, integrity, and endpoint authentication. Similarly, encryption, message authentication code, and digital certificates are used to provide these security services. TLS is immune to attacks such as POODLE attack [1]. Technically speaking the latest version of TLS is compatible with SSL 3.0.

III. Comparative Study

A. The difference between SSL and TLS.

• TLS has new advanced features and supports new algorithms when compared to SSL.
• With the attack called POODLE attack, now usage of SSL has become lot vulnerable and, in the new versions of web browsers, SSL will be disabled by default. However, in all browsers, TLS is enabled by default [2].
• TLS supports new authentication and key exchange algorithms suites such as ECDH-RSA, ECDH-ECDSA, PSK, and SRP.
• Message Authentication Code Algorithm suites such as HMAC-SHA256/384 and AEAD are available in latest TLS versions, but not in SSL.
• SSL was developed and edited under Netscape. However, TLS is under Internet Engineering Task Force as a standard protocol and hence is available under RFC.
• There are differences in the implementation of the protocol such as in key exchange and key derivation

B. How is TLS different than SSL in the G-Hadoop Environment?

TSL uses the same key exchange algorithm that SSL uses, which mean the default key exchange, proposed in this model (fixed Diffie-Hellman) can be maintained when switching to TLC from SSL in this paper.

The generation of the cryptographic secret keys is more complex in TLS and SSL. TLS first defines the data expansion function and pseudorandom function.

The data expansion function uses a predefined HMAC to expand a secret into a longer one [8], the extended secret is the concatenation of the hash function values. Each section uses two HMACS a secret and seed. The data expansion function is the chinning of as many sections as required. In order to make the next section dependent on the previous, the section seed is actually the output of the first HMAC of the previous section.

Also, TLS defines the pseudorandom function to be the combination of two data expansion functions that is one using MD5 and the other using SHA-1. This function takes three input, a secret, a label, and seed; the secret is divided into two halves, where each half is used as the secret for each data expansion function. The output of two data expansion function is exclusive-or together [8].
It should be noted that MD5 extra functions are needed to make the two outputs the same size.

Besides TLS uses the pseudorandom function to create the master secret from the premaster secret. Although SSL and TLS use the same process to generate the Pre-master secret, in TLS the master secret is concatenated with the client random number and server random number as the seed. Also to be noted that TLS uses the pseudorandom function to create the key material from the master secret.
This can be proving difficult to guess in an MITM attack [8].

C. Preventing MITM attack in proposed model

By using a double check in this model, especially when TLS was utilized in the model. It will prove difficult for the man in the middle attack to take palace quietly. The reason is that TLS uses HMAC, and the basic idea of HMAC is to concatenate the key and the message and hash them together. This makes it difficult or impossible to find the key [12].

IV: Conclusion

Both TlS and SSL are very similar and hence can be used in our G-Hadoop model. It is the matter of choice and industries standard. For example, if one is using G-Hadoop, over a wide area network TLS can be the right candidate for wide area security because it proves to be harder to crack. TLS uses HMAC encryption; HMAC is known to be a nested MAC, and the implementation of HMAC is more tedious and complex than the simplified MAC with the additional padding feature.
In essence, G-Hadoop can be better secured if the security feature between client and master node is enhanced with the model presented in this paper. This paper has presented a proposed model and Implementation can be made in the language such as JAVA language.

V: Reference

1. Apache Hadoop Project, web page http:www.apache.org
2. Jeffrey Dean, Sanjay Ghemawat, Map Reduce: simplified data processing on large clusters. Commun. ACM 51 (January 2008) 107-113
3. Bing Tang, Mircea Moca, Stephane Chevalier, Haiwu He, Gilles Fedak, Towards Map Reduce for Desktop Grid Computing, In Proceeding of the 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC10, IEEE Computer Society, Washington, DC, USA, 2010, PP, 193-200.
4. Shadi Ibrahim, Hai Jin, Bin Cheng, Haijun Cao, Song Wu, Li Qi, Cloudlet: towards Map Reduce implementation on virtual machines, in: Proceedings of the 18th ACM International Symposium on High-Performance Distributed Computing. HPDC10, ACM, New York, NY, USA, 2009, page. 65-66.
5. Robert Scheafer, Aleksander Byrski, Joanna Kolodzieji, Maciej SmolKa, An agent-based model of hierarchic genetic search, computer. Math. Appl. 64(12) (2012) 292-316.
6. Joanna Kolodziej, Fatos Xhafa, integration of task abortion and security requirements in GA-based meta-heuristic for independent batch Grid Scheduling, computer. Math. Appl 63 (2) (2012) 350-364.
7. Jiaqi Zhaoa, Lizhe Wangb,Jie Taoc, Jinjun Chend,Weiye Sunc, Rajiv Ranjane, Joanna Kołodziejf, Achim Streitc, Dimitrios Georgakopoulose. A security framework in G-Hadoop for big data computing across distributed Cloud data centres. Volume 80, Issue 5, August 2014, Journal of Computer and System Sciences. Pp.994-1007
8. A. Forouzan. Cryptography and Network. 2th-second ed. New York McGraw-Hill 2010
9. G-Hadoop, web page https://sites.google.com/site/ghadoop/research/global-file-system
10. A. Kate. Cryptography and Network Security 2nd-second ed. 7 west Patal Nagar, New Delhi McGraw-Hill 2009.
11. Semicomplete. Web page. http://semicomplete.com/blog/geekery/ssl-latency.html
12. Sack exchange. Web page. http://security.stackexchange.com/questions/20129/how-and-when-do-i-use-hmac