Asian Journal of Engineering and Applied Technology
ISSN: 2249-068X (P) Vol.6 No.2, 2017, pp.40-50

DOI: https://doi.org/10.51983/ajeat-2017.6.2.819

Integrity and Privacy through Authentication Key Exchange Protocols for Distributed Systems

B. Usharani
Assistant Professor, Department of CSE,
KL University, Andhra Pradesh, India
E-Mail: ushareddy.vja@gmail.com
(Received 10 September 2017; Revised 30 September 2017; Accepted 21 October 2017; Available online 5 November 2017)

Abstract **- Networking is the practice of connecting several computing devices together in order to share resources.In real world, attacks via force and fraud are privacy (unauthorized release of information), Integrity (tampering with data), Service (denial of service).The goals are disallow unauthorized access, allow authorized access, resist DOS attacks .In recent years, many efficient AKE protocols have been proposed to achieve user privacy and integrity in the communications. A communication model is a representation where there are a large number of clients accessing multiple remote and distributed storage devices in parallel. Authenticated key exchange (AKE) protocol allows a user and a server to authenticate each other and generate a session key for the later communications. This paper focuses on how to exchange key materials and establish parallel secure sessions between the clients and the storage devices in the Network in an efficient and scalable manner.**
Keywords: **Authenticated key; Exchange Protocols; Kerberos-based approach; Security**

I. INTRODUCTION

Computer network aims to give users with the ability to distribute data amongst multiple gadgets, whether they are in the same building or across the world. Traditional computer networking relied on Ethernet and fibre optic cables to connect various devices on a network. More modern technology has emerged that allows for wireless connections between electronics. These technologies include Wi-Fi and Bluetooth compatible devices. It is very helpful to understand the role that each of these technologies plays in computer networking.

There are two primary considerations for networks: security and performance. The third consideration is manageability.

1. Network Security is an organization's strategy and provisions for ensuring the security of its assets and of all network traffic.
2. Integrity - ensuring the modification of assets is handled in a specified and authorized manner.
3. Availability - a state of the system in which authorized users have continuous access to said assets

In a parallel application the file data is spread among the various nodes or devices for giving the concurrent access to multiple tasks using parallel file system. This is widely used in large scale cluster computing which is mainly depends upon the reliable fetch as well as high performance to the large amount of datasets. Due to this the bandwidth of I/O is highly achieved by concurrent data fetching with different number of devices in between the maximum number of clusters which are used for computing. During this the loss of data is prohibited or secured using the data mirroring and fault- tolerant striping algorithms are used for data mirroring. There are some examples of highly performance parallel file systems which undergoes in production that uses the General Parallel File Systems.

A. Network Security

Authentication means ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false. If one is not careful, the exchanged messages Q may reveal discernible structure, and can "leak" information about S, enabling a partition attack. Password-based authentication protocols cannot rely on persistent stored information on the client side.

B. Integrity

Integrity means ensures that only authorized parties are able to modify computer system assets and transmitted information. Provisions in the protocol do not allow the contents to be intentionally or unintentionally modified.

During data transmit, data confidentiality is ensured by added message encryption collectively with data transfer protocols that ensure the security of the data being transmitted by the interoperation participants (including data integrity verification). During data storage and processing, their confidentiality, integrity, and privacy are ensured by additional mechanisms of encryption and masking together with well-defined distribution of access in concordance with privileges and permissions

C. Privacy

Privacy is ensured by the absence of sensitive data in the messages being transferred, as well as by the implementation of the required mechanisms of data storage and system access control facilities. Data is encrypted for an intended recipient. System components must not have

underlying potential of unauthorized data acquisition and transfer. The use of weak cryptographic algorithms by the application can lead to privacy violations as data and metadata can be inferred through network activity. In the worst case, weak ciphers can be completely compromised, leading to complete violation of privacy. User privacy can be compromised by attacks such as phishing, which is often the first activity in advance of an advanced persistent threat that can direct to larger scale compromise. Attacks against operating systems have been numerous and virulent, from malware and privacy-compromising advertising to ransom ware and targeted zero-day attacks.

D. Authenticity

Digital certificates are used to grant a verified identity. The certificates are signed by a trusted third party. The TTP is a certification authority (CA). The CA signs the certificate attesting proof the identity has been validated. The browser makes sure the identity assertion from the CA. Authenticity is granted for the server and can be provided for the client.

E. DFS

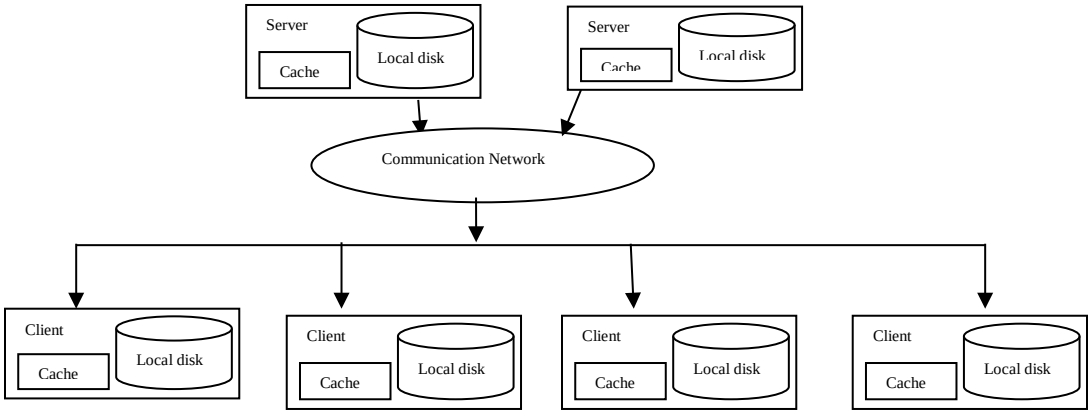

Fig.1 Distributed File System

A distributed file system is a file system distributed across multiple machines i.e. multiple machines share files and storage devices. The main purpose of DFS is sharing of files. Distributed file systems are commonly known as network file systems. [23]

Benefits of DFS

1. Simplified data migration: can move data at background without reconfigure applications and shortcuts and without needing to reeducate users about where they can find their data
2. Increased availability of file server data
3. Security integration: file and folder security is enforced by existing the NTFS file system and shared folder permissions on each target
4. File Sharing
5. Transparency
6. Concurrent File updates
7. File Replication
8. Hardware and Operating System Heterogeneity
9. Fault Tolerance
10. Consistency
11. Security
12. Efficiency

Goals of DFS

1. Access/Network Transparency
2. Availability

F. NFS

The first network file system—called File Access Listener—was developed in 1976 by Digital Equipment Corporation (DEC). NFS was the first modern network file system (built over the IP protocol). It began as an experimental file system developed in-house at Sun Microsystems in the early 1980s. Given the popularity of the approach, the NFS protocol was documented as a Request for Comments (RFC) specification and evolved into what is known as NFSv2[19].The Network File System (NFS) is a client/server application that allows a computer user view and optionally store and revise files on a remote computer as though they were on the user's own computer.

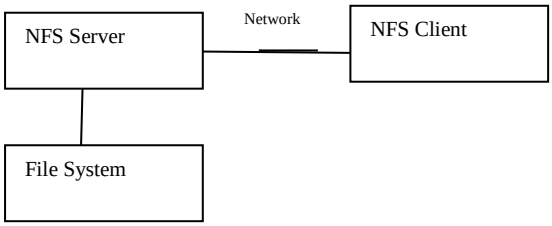

Fig.2 Communication between NFS client and NFS Server

The NFS protocol is one of several distributed file system standards for network-attached storage (NAS)[21].NFS operates on TCP/IP network.NFS enables you to mount a file system on a remote computer and directly access any of the files on that remote file system. Network File System (NFS) [18] is currently the sole file system standard supported by the Internet Engineering Task Force (IETF). The NFS protocol is a distributed file system protocol originally developed by Sun Microsystems that permits a user on a client computer, which may be diskless, to access files over networks in a manner similar to how local storage is accessed. It is designed to be portable across different machines, operating systems, network architectures, and transport protocols. Such portability is achieved through the use of Remote Procedure Call (RPC) primitives built on top of an external Data Representation (XDR) .Among the current key features are file system migration and replication, file locking, data caching, delegation (from server to client), and crash recovery.

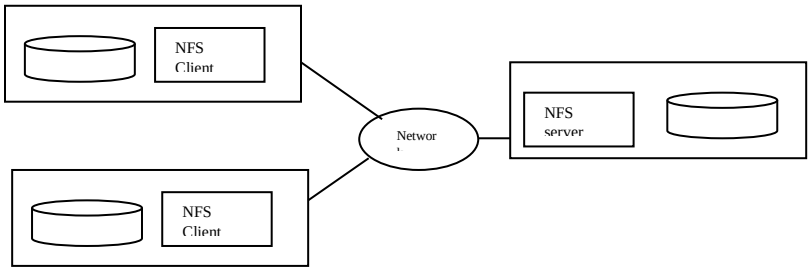

Fig.3 The client-server architecture of NFS

1. Versions of NFS

1. NFSV2-older and widely used version.NFSv2 uses the User Datagram Protocol (UDP) to provide a stateless network connection between the client and server.
2. NFSV3-includes 64 file handles, safe Async writes and robust error handling. NFSv3 can use either UDP or Transmission Control Protocol (TCP) running over an IP network. As of UDP is stateless, if the server goes down unexpectedly, UDP clients go on to saturate the network with requests for the server. For this reason, TCP is the chosen protocol when connecting to an NFSv3 server. Supports 64-bit file sizes and offsets, allocating clients to use more than 2Gb of file data.
3. NFSV4-works through firewalls and through internet, consume state full operations, no need of port map per, supports ACL.

2. NFS Protocols

NFS protocol depends on RPC and port map. An RPC server tells port map which port will be used and the managed RPC number. A client contacts port map to get port number of the desired server.RPC packets are addressed to the corresponding port.

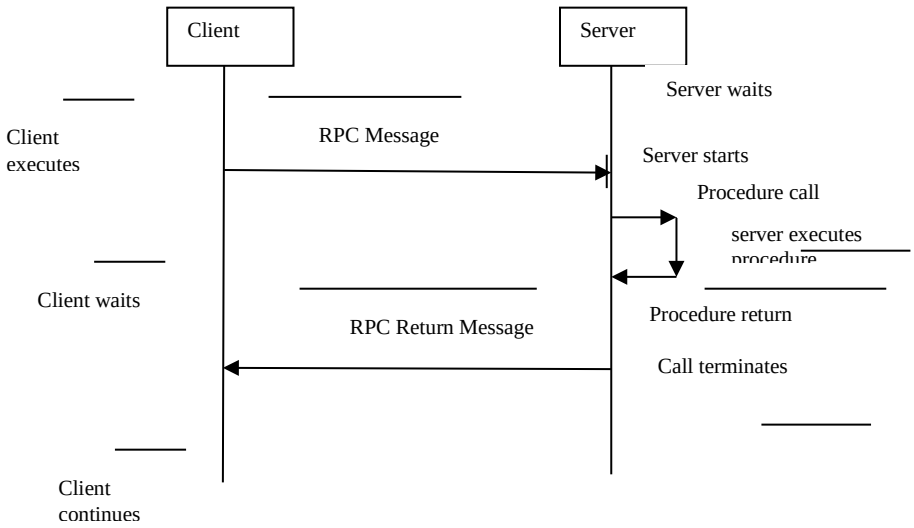

Fig.4 NFS Protocol

NFS protocol design does not depend on transport protocols. It is used with UDP by default but can be used with TCP protocols.NFS is designed as a stateless protocol. So, server maintains no per-client state, and every RPC executes atomically and contains all state essential to perform RPC (.This makes it immune to denial-of-service

attacks. Its design should be robust. There must be no problem when client breaks down or when server reboots. There are no file descriptors; as a substitute, there are file handles, which are 64-bit numbers recognizing files (not filenames).

3. The NFS Protocol life cycle

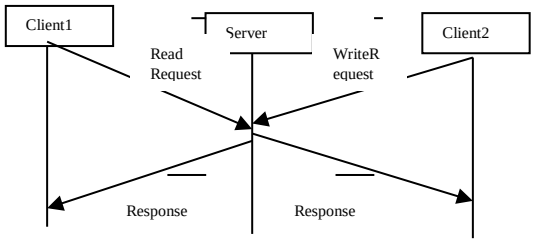

Fig.5 NFS file handles

Read/write consistency is not assured in NFS.
Open/close consistency is assured in NFS.
NFS implementation on clients does pipelining and caching.

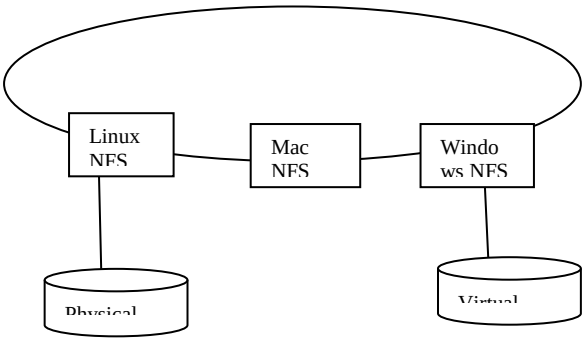

Fig.6 A simple NFS configuration

Figure 6 illustrates a common deployment of NFS within a network of various operating systems, all of which maintain the NFS standard.

In Figure 6, the Linux machine is the NFS server; it shares or exports one or more of its physical, attached file systems. The Mac OS X and Windows machines are NFS clients. Each uses, or mounts, the shared file system. Indeed, mounting an NFS file system give in the same result as mounting a local drive partition—when mounted, applications just read and write files, subject to access control, unconscious to the machinations essential to carry on data.

Here, NFS is a problem, because the limits of the NFS server—are its bandwidth, storage capacity, or processor speed—strangle on the whole performance of the computation. So, NFS is a bottleneck.

4. Practical uses of NFS

1. Share a file or CD with any no: of clients.
2. Sharing the directory
3. Central NFS server stores all user home directories.

5. NFS Problems

When dealing with a distributed file system among multiple nodes, the issue of file integrity becomes the most important. The two main problems fall into the category of either synchronization issues, or reliability issues. NFS no longer assurance write to read consistency with multiple clients, but it does guarantee closed to open consistency because closing and reopening is a much slower process and will make sure all changes are accounted for.

Synchronization: FS does however have close-to-open consistency. This means that the client waits for all write responses from the server before a close is performed. However, close is a system call. There is no CLOSE in the NFS protocol. This is because NFS doesn't care about file descriptors, just file handles. This means that when the client actually calls close (file descriptor), it is likely to be close to return -1 with the error no set to EIO. For this cause it develops into of particular importance to test the close return values for potential write fails on an NFS server. There is one alternative option and that is to enable the Synchronous write flag, but this is infrequently done since it basically removes pipelining resulting in poor performance.

Reliability Issues: On distributed or networked file systems the probability of failure raises proportionally to the number of devices involved. When disk blocks go bad, or devices report read/write errors, there wants to be some sort of mechanism to make sure the data is not lost. The first solution to data corruption was to use logging to spot areas that were compromised. The problem then became, well what happens if the log is corrupted? This gave go up to double logging or redundancy which is now enhanced know today as a RAID, a Redundant Array of Independent Disks.

From a security perspective, the NFS protocol is not encrypted or otherwise protected on the wire. This means that anyone who has access to the NFS network has the ability to capture data that represents the on-disk information stored in virtual machine files. Clearly, this is a significant risk that needs to be mitigated with suitable configuration and management actions.

If the data set grows too large, the NFS server rapidly becomes the bottleneck and significantly impacts system performance because the NFS server sits in the data path between the client computer and the physical storage devices.

G. Authenticated Key Exchange Protocols

A key exchange protocol is said to give key confirmation, if both parties are sure that the intended peers truly hold the session key. A protocol which is an authenticated key exchange with key confirmation protocol is called AKC

protocol [4]. A lot of desirable properties for AKE protocols have been identified:

1. Known-key security: It is reasonable to assume the adversary has the ability to learn the session keys except for the one under attack. A protocol is said to be known-key secure if the compromise of one session key should not compromise other session keys.
2. Forward security: If the static keys of one party or two parties are compromised, the adversary cannot obtain the previously established session keys.
3. Key compromise impersonation resistance: Suppose A's static key is compromised. Clearly, the adversary can arbitrarily masquerade as A in future. However, we want to guarantee that the adversary cannot masquerade as another party B to communicate with party A.
4. Ephemeral key reveal resistance: If the adversary obtains the ephemeral keys of the related sessions, the session key under attack still remains secure. The authenticated key exchange protocols have been established to be surprisingly difficult to design.

Bellare and Rogaway [13] first propose a formal security model for authentication and key distribution. They model the adversary's capability by providing it with oracle queries, e.g. Send, Reveal and Test queries. Since then, there have been several extensions to the model [14], [12], [15]. Choo, Boyd and Hitchcock [16] compare the most commonly used security models for key exchange protocols. All these models attempt to cover as many of these properties as possible.

Key exchange protocols permits two or more parties communication over a public network to establish a common secret key called a session key. Due to their importance in building a secure communication channel, a number of key exchange protocols have suggested over the years for a variety settings. In order to avoid mistakes and impersonations during the process we can use a variety of authentication means. The most commonly used authentication means is based on the following factors: 1. a secret password. 2. Secure device with a secret key. If a protocol have only one authentication factor, it would be risky because the password can be recovered through social engineering and the device can be stolen, open or cloned, even when various tamper-resistant techniques are used to guard it. Apparently combining the two factors in the similar authentication protocol could boost the security since the adversary would have to break the two protections in order to win [17].

The paper "Perfect Forward Secrecy - An authenticated key exchange protocol" gives perfect forward secrecy if disclosure of long-term secret keying material does not compromise the secrecy of the exchanged keys from earlier runs. The property of perfect forward secrecy does not apply to authentication without key exchange.

II. LITERATURE SURVEY

The paper "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment" is designed to support the files and also the I/O workload of desktop computers in a large company or university. It provides availability and reliability through replication; privacy and authentication through cryptography; integrity through Byzantine-fault-tolerance techniques; consistency through leases of variable granularity and duration; scalability through namespace delegation; and reasonable performance through client caching, hint based pathname translation, and lazy update commit [24]. "Block Level Security for Network-Attached Disks Marcos"-This paper propose a practical and efficient method for adding security to network-attached disks (NADs) [25].

The paper entitled "Authenticated Key Exchange Protocols for Parallel Network File Systems" presents large-scale distributed file systems supporting parallel access to multiple storage devices [1].

The paper "Authenticated Key Exchange Secure against Dictionary Attacks" proposes password-based protocols for authenticated key exchange (AKE) despite the use of passwords drawn from a space so small that an adversary might well enumerate, off line, all possible passwords [9].

The paper "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels" presents a key-exchange protocol that satisfies the security with symmetric encryption and authentication functions to provide provably secure communication channels and the communication links are perfectly authenticated, and then translate those using general tools to obtain security [26].

The paper "Simple Password-Based Encrypted Key Exchange Protocols" proposes a technique called SPAKE1: a simple non-concurrent password-based encrypted key exchange based on the multi-dimensional version of password-based chosen basis computational Diffie-Hellman problem, S-PCCDH. [2].

The paper "User Authentication to present security against Online Guessing Attacks" proposed SPAKA+ that strengthens SPAKA protocols against online dictionary attacks [3].

The paper "Key Exchange Protocols: Security Definition, Proof Method and Applications" proposed a compositional method for proving cryptographically sound security properties of key exchange protocols, based on a symbolic logic that is interpreted over conventional runs of a protocol against a probabilistic polynomial-time attacker [4].

The paper "A Secured and Authenticated Message Passing Interface for Distributed Clusters" provide user authentication and authenticate messages transmitted from cluster members to KGC [5].

"J-PAKE: Authenticated Key Exchange Without PKI" - This paper proposed a protocol, called J-PAKE, which authenticates a password with zero-knowledge and then subsequently creates a strong session key if the password is correct. It requires no PKI deployments and protects users from leaking passwords [6].

"Automatically Verified Mechanized Proof of One-Encryption Key Exchange" proposed a protocol One-Encryption Key Exchange (OEKE) .This paper proved the security of OEKE using the tool CryptoVerif [7].

The paper entitled "Password Authenticated Key Exchange by Juggling" proposed a protocol called Password Authenticated Key Exchange by Juggling for security .This protocol achieves mutual authentication in two steps: first, two parties send ephemeral public keys to each other; second, they encrypt the shared password by juggling the public keys in a verifiable way [8].

The paper "Scalable Security for Petascale Parallel File Systems" proposed Maat, a security protocol designed to provide strong, scalable security. Maat can scale to handle file systems with thousands of clients accessing files striped across thousands of network-attached storage devices [10].

III. PARALLEL NETWORK FILE SYSTEM

The PNFS is introduced by the UMich/CITI, IBM, ENC, and Sun. This paper is focusing on maintaining Integrity and Privacy in Authenticated key exchange protocol for Parallel Network File System. PNFS provides parallel file access across distributed servers. PNFS is heavily driven by Panasas, NetApp, EMC, IBM, Sun (now Oracle) among others [22]. The fastest supercomputer in the world and the first computer to reach a peta-flop uses the parallel file system built by Panasas [20]. NFS stands ready to provide super-storage speeds to super-computing machines. PNFS also makes sure that data can be better load balanced to meet the needs of the client.

PNFS divides the file system protocol processing into two parts: metadata processing and data processing. Metadata is information regarding a file system object, such as its name, location within the namespace, owner, permissions and other attributes. The entity that supervises metadata is called a metadata server. Regular files' data is striped and stored across storage devices or servers. Data striping occurs in at least two ways: on a file-by-file basis and, within adequately large files, on a block-by-block basis. Unlike NFS, a read or write of data managed with PNFS is a straight operation between a client node and the storage system itself.

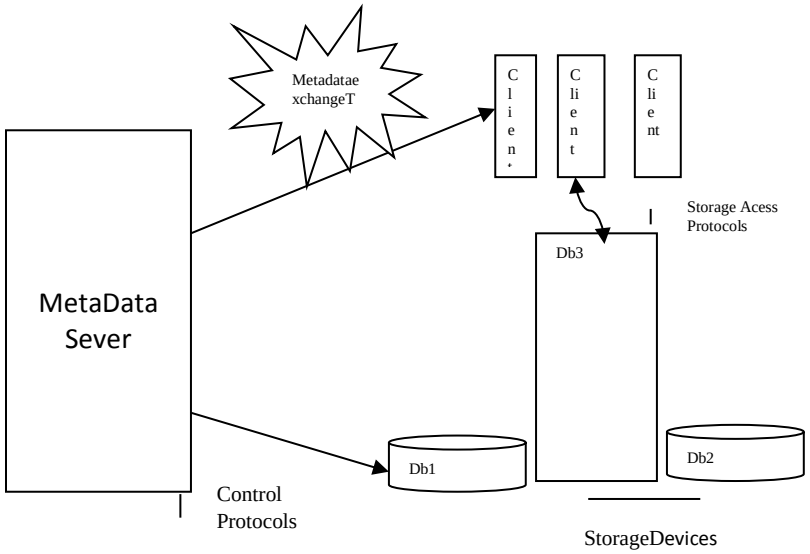

Fig.7 The Conceptual model of PNFS.

A. PNFS Architecture and Core Protocols

The pNFS architecture consists of three main components:

1. The metadata server holds all non data traffic. It is accountable for maintaining metadata that describes where and how each file is stored.
2. Data servers (Storage devices) layup file data and respond directly to client READ and WRITE requests. File data can be striped across a number of data servers.
3. One or more clients are able to access data servers in a straight line based on information in the metadata received from the metadata server.

4. Three types of protocols are used between the clients, metadata server, and data servers:
5. A control protocol is used to synchronize the metadata server and storage devices. Synchronization, such as reorganizing files on media, is hidden from clients. This is not defined by the PNFS specification and differs from vendor to vendor.
6. PNFS protocol is used between clients and the metadata server. The PNFS protocol that transfers file metadata, between the metadata server and a client node. It is used to retrieve, which contain the metadata that describes the location and storage access protocol essential to access files stored on multiple data servers.
7. A set of storage access protocols is used by clients to directly access data servers. The PNFS standard currently defines three categories of storage protocols: file-based (RFC5661), block-based (RFC5663), and object-based (RFC5664). The storage access protocol that specifies how a client accesses data from the associated storage devices according to the matching metadata servers

B. PNFS Performance Evaluation and Advantages

PNFS eliminates the performance bottleneck in traditional NAS systems by allowing the clients to read and write data directly and in parallel, to and from the physical storage devices. The NFS server is used only to control metadata and coordinate access, permitting incredibly fast access to very large data sets from many clients. When a client wants to access a file it first queries the metadata server which provides it with a map of where to get the data and with credentials regarding its rights to read, modify, and write the data. Once the client has those two components, it speaks directly to the storage devices when accessing the data. With traditional NFS every bit of data flows through the NFS server – with PNFS the NFS server is detached from the primary data path allowing free and fast access to data. All the advantages of NFS are maintained but bottlenecks are removed and data can be accessed in parallel allowing for extremely fast throughput rates.

Parallel NFS that combines the advantages of NFS with the massive transfer rates proffered by parallelized input and output (I/O). Using PNFS, file systems are shared from server to clients as before, but data does not bypass through the NFS server. As a substitute, client systems and the data storage system connect openly, providing numerous parallelized, high-speed data paths for massive data transfers.[20].

More specifically, a Read and write operation is a series of protocol operations:

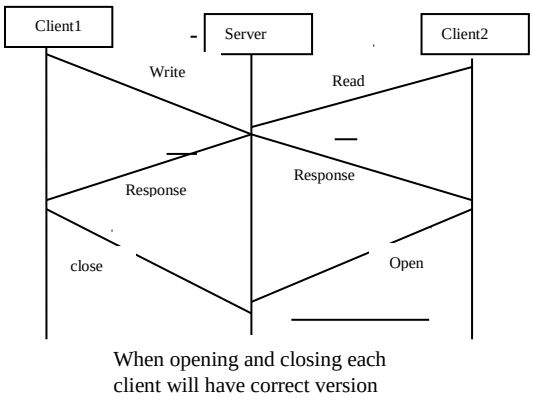

Fig.8 PNFS two Clients handling the Read and write operations.

C. Integrity and Privacy Protection to PNFS

The integrity and the privacy to the distributed parallel network can be achieved by:

1. Parallel Sessions

Parallel sessions are between the clients and the storage devices in the parallel Network File System. This is similar to the situation that once the adversary compromises the long-term secret key, it can learn all the subsequence sessions. If an honest client and an honest storage device complete matching sessions, they calculate the same session key.

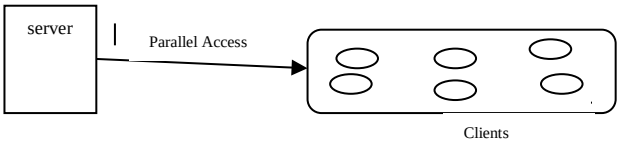

Fig.9 Parallel sessions

2. Authenticated Key Exchange

In an authenticated key exchange, there is the extra goal that the two parties end up sharing a common key known only to them. This secret key can then be used for some time later to provide privacy, data integrity, or both.

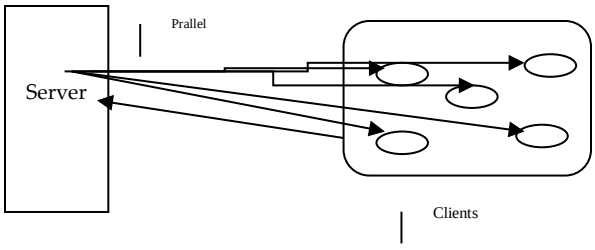

Fig.10 Authenticated Key Exchange

3. Forward secrecy

An authenticated key exchange protocol provides perfect forward secrecy if disclosure of long-term secret keying material does not compromise the secrecy of the exchanged keys from previous runs. The property of perfect forward secrecy does not apply to authentication without key exchange.

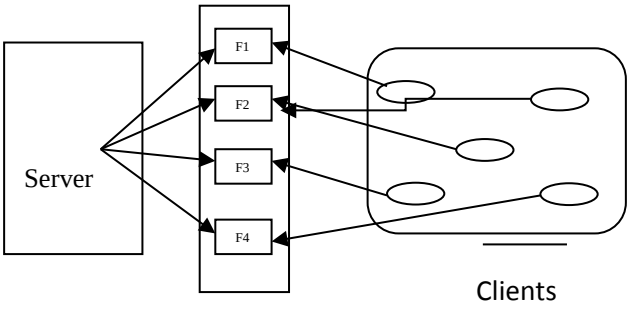

Fig.11 After authenticated key Exchange

4. Server Authentication

The admin can accept the new user request and also block the users. The users can upload the file to the network. And the admin can allow the files to network then only the file can store in storage devices. If the file uploaded by the user is not permitted from the Server means the file cannot be uploaded by the Client.

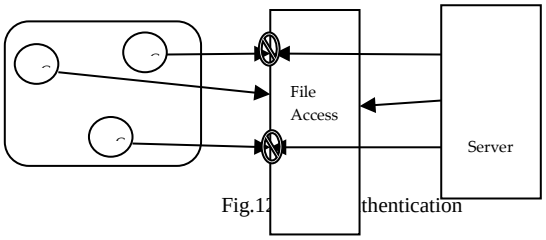

Fig.12 thentication

5. Implementation of AKE Protocols for PNFS

There are three variants of PNFS-AKE protocols

PNFS-AKE-I- This protocol allows the client to generate their own session keys. A session key is pre-computed by the client for each v and forwarded to the corresponding storage device in the form of an authentication token at time t (within v). A Symmetric key encryption is used to protect the confidentiality of secret information used in the protocol [1].

PNFS-AKE-II- The client C and the storage device S_i each now chooses a secret and pre-computes a Dif- fie-Hellman key component. A session key is then generated from both the Dif-fie-Hellman components. Upon expiry of a time period v, the secret values and Dif-fie-Hellman key components are permanently removed, such that in the event when either C or S_i is compromised, the attacker will no longer have access to the key values required computing past session keys [1].

PNFS-AKE-III- Enhance PNFS-AKE-II with a key update technique based on any efficient one-way function, such as a keyed hash function. This protocol achieves full forward secrecy.

Establishment of secure channels between Client and Metadata server

1. Obtain Metadata server's certificate
2. Verified that it is signed by trusted CA.
3. Generate random Session Symmetric key.
4. Encrypt the session key with metadata server's public key.
5. Send Encrypted key to the metadata server

Establishment of secure communication between clients and parallel servers through the help of Metadata server.

PNFS-AKE-I is illustrated as [1]

Phase I-For each validity period v:

(i)C→M : ID_C , $\mathcal{E}(K_{CM}, K_{CS_1}, K_{CS_2}$ ……..$K_{CS_x})$
(ii)M→C: $\mathcal{E}(K_{MS_1}; ID_C, ID_{S_1}, V, K_{CS_1})$....., $\mathcal{E}(K_{MS_x}; ID_C, ID_{S_x}, V, K_{CS_x})$

Phase II-For each access request at time t:

(i)C→M :ID_C ,ID_{S_1} ,.....ID_{S_x}
(ii)M→C: σ_1 ..... σ_n
(iii)C→S_x : σ_x , $\mathcal{E}(K_{MS_x}, ID_C, ID_{S_x}, v, k_{cs_x}), \mathcal{E}(sk_x^0; ID_C, t)$
(iv)S_x→C : $\mathcal{E}(sk_x^0; t+1)$

The diagrammatic representation of the PNFS AKE-I is:

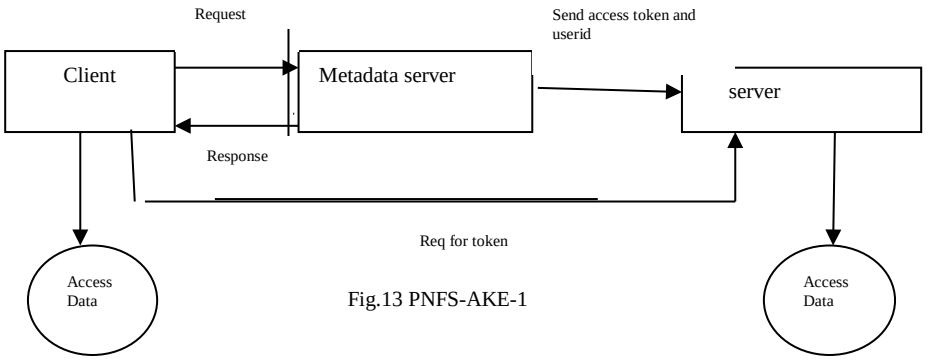

Fig.13 PNFS-AKE-1

PNFS-AKE-II is illustrated as [1]

Phase I-For each validity period v:

(i)$S_X \rightarrow M: ID_{S_x}$, $\varepsilon(K_{MS_1}, g^{s1})$
(ii)$C \rightarrow M : ID_c$, $\varepsilon(K_{CM}; g^c)$
(iii)$M \rightarrow C: \varepsilon(K_{CM}; g^{s1}, ...g^{si})$, $\tau(K_{MS_1}, ID_C, ID_{S_x}, V, g^c, g^{s1})$....., $\tau(K_{MS_x}; ID_C, ID_{S_x}, V, g^c, g^{s1})$

Phase II-For each acess request at time t:

(i)$C \rightarrow M : ID_C, ID_{S_1}, ...ID_{S_x}$
(ii)$M \rightarrow C: \sigma_1 ..... \sigma_n$
(iii)$C \rightarrow S_x : \sigma_x$, $\varepsilon(KMS_x, ID_C, ID_{S_x}, v, g^c, g^{s1})$, $\varepsilon(sk_x^0; ID_c, t)$
(iv)$Sx \rightarrow C : \varepsilon(sk_x^0; t+1)$

PNFS-AKE-III is illustrated as [1]

Phase I-For each validity period v:

(i)$S_X \rightarrow M: ID_{S_x}$, $\varepsilon(K_{MS}, g^{s1})$
(ii)$C \rightarrow M : ID_c$, $\varepsilon(K_{CM}; g^c)$
(iii)$M \rightarrow C: \varepsilon(K_{CM}; g^{s1}, ...g^{si})$, $\tau(K_{MS}, ID_C, ID_{S_x}, V, g^c, g^{s1})$....., $\tau(K_{MS_x}; ID_C, ID_{S_x}, V, g^c, g^{s1})$
(iv)$M \rightarrow S_X : \varepsilon(K_{MS_x}; ID_C, ID_{S_x}, V, g^c, g^{s1})$

Phase II-For each acess request at time t:

(i)$C \rightarrow M : ID_C, ID_S, ...ID_{S_x}$
(ii)$M \rightarrow C: \sigma_1 ..... \sigma_n$
(iii)$C \rightarrow S_x : \sigma_x$, $\varepsilon(sk_x^{j,0}; ID_c, t)$
(iv)$S_x \rightarrow C : \varepsilon(sk_x^{j,0}; t+1)$

Phase1 for each validity period v

a. Each server distribute some key materials to Metadata server .Each Si generate Diffie Hellman key component gsi . This is forwarded to and stored by Metadata server.
b. $S_X \rightarrow M: ID_{S_x}$, $\varepsilon(K_{MS_1}, g^{s1})$
c. Similarly Client generates its Diffie Hellman Key componentgc and send to Metadata server.
d. $C \rightarrow M : ID_c$, $\varepsilon(K_{CM}; g^c)$
e. M sends all key components to C for N storage servers that it may access within a periodv
f. $M \rightarrow C: \varepsilon(K_{CM}; g^{s1}, ...g^{si})$
g. M also sends Client"s Diffei Hellman Components to g^c to each S_i.
h. $M \rightarrow S_X : \varepsilon(K_{MS_x}; ID_C, ID_{S_x}, V, g^c, g^{s1})$
i. After this stage C and S_i are able to agree a Diffei Hellman value g_c^{si} e) C and S_i set $F1(g_c^{si}, IDC, ID_{Si}, v)$ as their initial shared secret state K^0CS_i

Phase2 for each access request at time t

a. C submits an access request M which contains all identities of storage devices Si
   $C \rightarrow M : ID_C, ID_{S_1}, \ldots\ldots\ldots ID_{S_x}$
b. M issues layout σi(Layout contains Client"s identity, File object mapping information and Access permissions)
   $M \rightarrow C: \sigma_1 \ldots\ldots \sigma_n$
c. Cestablish secure session with S_i by computing session key S_k^{ij},z.
   $S_X = \mathcal{E}(K_{MS_x}; ID_C, ID_{S_x}, V, g^c, g^{s1})$
   C sends encrypted layout and identity and time to Si
   $C \rightarrow S_x : \sigma_x , \mathcal{E}(sk_x^{j,0}; ID_c, t)$
d. S_i decrypt encrypted message and check if the layout and ID_c matches the identity of C and if t is within the current validity periodv.
e. If all previous checks pass , S_i replies C with a key confirmation message using key $S_k^{ij,0}$
f. Both C and S_i then set and update their internal shared secret state as $K^j CS_i$

IV. RESULTS

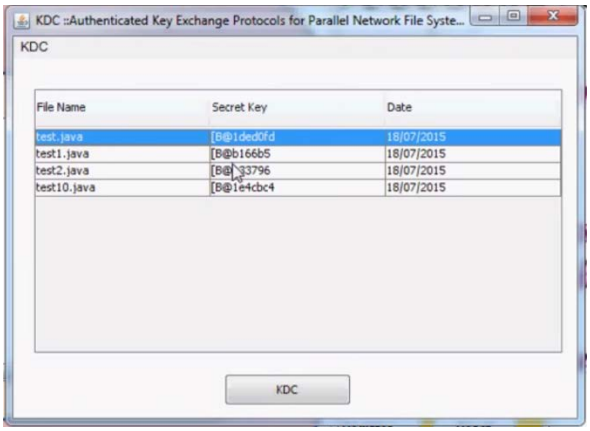

Fig.14 The Secret keys stored at KDC

The initial shared key is then used to derive session keys in the form of a keyed hash chain. The associated session key is forward secured.

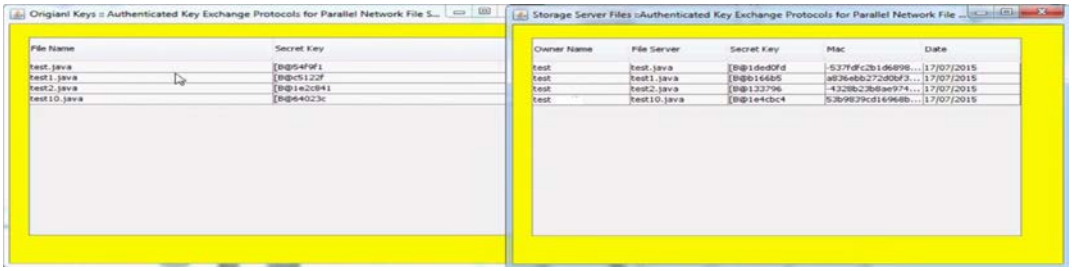

Fig.15 Comparison of the Storage Server Secret keys with the original Keys

For each storage server S_i, M issues a layout S_i. C then forwards the respective layouts, authentication tokens, and encrypted messages to all n storage devices. Secure MAC scheme that takes as input a secret key k and a target message m, and output a MAC tag

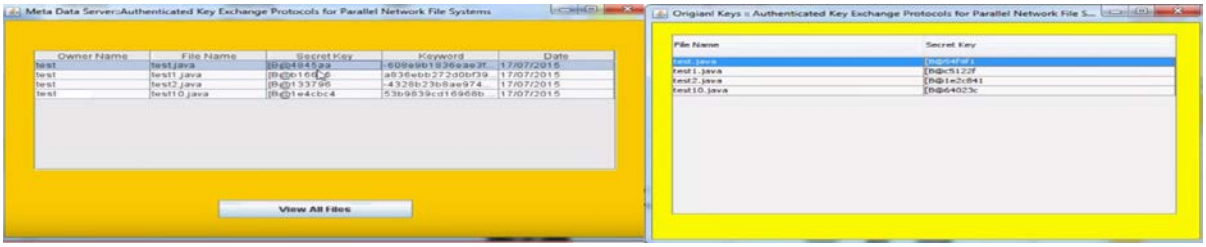

Fig.16 Comparison of the Meta Data Server Secret keys with the original Keys

When client submits an access request to Meta Data Server, the request contains all the identities of storage Servers S_i that Client wishes to access both C and Si to generate and exchange fresh Diffie-Hellman components for each access request at time t.

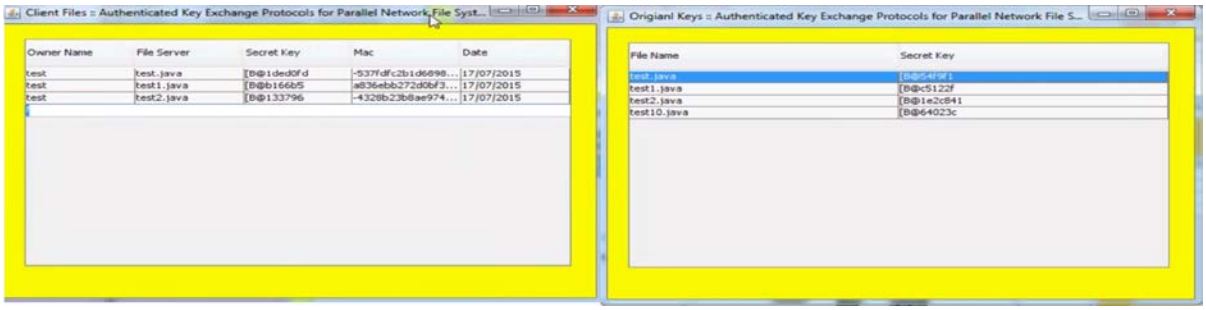

Fig.17 Comparison of the Client keys with the original Keys

Meta Data Server also distributes clients chosen Dif-fie-Hellman component to each Storage server. Hence, both client and storage server are able to agree on a Dif-fie-Hellman value.

V. CONCLUSION

This paper implements the AKE protocols for distributed Systems. Proof has been shown that the network is secure against passive and active attacks. The future work of this paper is to implement these AKE protocols for Wireless Networks for all security services i.e. authentication, authorization, and confidentiality and so on.

REFERENCES

[1] Hoon Wei Lim and Guomin Yang, "Authenticated Key Exchange Protocols for Parallel Network File Systems," *IEEE transactions on parallel and distributed systems*, Vol. 27, No. 1, pp.92-105, January 2016.

[2] Michel Abdalla and David Pointcheval., *Simple Password Based Encrypted Key Exchange Protocols*, Topics in Cryptology – CTRSA 2005, Vol. 3376 of Lectures Notes in Computer Science, San Francisco, CA, USA, Springer-Verlag, Berlin, Germany, pp.191–208, Feb. 14–18, 2005.

[3] A.Sai Kumar and P. Subhadra., "User Authentication to Provide Security against Online Guessing Attacks," PARIPEX - *Indian Journal Of Research* Vol. 2 Issue : 2 , ISSN - 2250-1991, pp. 129-130, February 2013.

[4] Anupam Datta1, Ante Derek1, John C. Mitchell1, and Bogdan Warinschi2., "Key Exchange Protocols: Security Definition, Proof Method and Applications," *International Association for Cryptologic Research (IACR)* 2006/056 , pp. 1-33.

[5] R.S.RamPriya and M.A.Maffina., "A Secured and Authenticated Message Passing Interface for Distributed Clusters." *IIID security and privacy symposium* feb28- mar2 2013 , prabhu Goel Research center for computer and internet security IIT Kanpur , pp.1-2.

[6] Feng Hao1 and Peter Ryan2., "J-PAKE: Authenticated Key Exchange Without PKI", *International Association for Cryptologic Research (IACR)*, pp 1-24, 2010/190.

[7] Bruno Blanchet., "Automatically Verified Mechanized Proof of One-Encryption Key Exchange" *International Association for Cryptologic Research (IACR)*, pp 1-24, 2012/173.

[8] Feng Hao⋆1 and Peter Ryan2.,"Password Authenticated Key Exchange by Juggling" , *IEEE P1363: Research Contributions*, pp.1-12 ,April 2008.

[9] *Authenticated Key Exchange Secure Against Dictionary Attacks* M. Bellare, D. Pointcheval, and P. Rogaway, Advances in Cryptology Eurocrypt '00, Lecture Notes in Computer Science Vol. , B. Preneel ed., Springer-Verlag, 2000, pp.1-16.

[10] "Scalable security for petascale parallel file systems," A. W. Leung, E. L. Miller, and S. Jones, *in Proc. ACM/IEEE Conf. High Perform. Network Compute*, pp.1-12, Nov. 2007.

[11] "The Kerberos version 5 GSS-API mechanism," J. Linn, Internet Eng. Task Force (IETF), RFC 1964, Jun. 1996.[12] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In EUROCRYPT, pp.139–155, 2000.

[12] M. Bellare and P. Rogaway, *Entity authentication and key distribution*. In D. R. Stinson, editor, CRYPTO, volume 773 of Lecture Notes in Computer Science, pp. 232–249. Springer, 1993.

[13] M. Bellare and P. Rogaway, *Provably secure session key distribution: the three party case*. In STOC, pp. 57–66. ACM, 1995.

[14] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," In B. Pfitzmann, editor, EUROCRYPT, Vol. 2045 of Lecture Notes in Computer Science, pp. 453–474. Springer, 2001.

[15] K.-K. R. Choo, C. Boyd, and Y. Hitchcock. "Examining indistinguishability-based proof models for key establishment protocols," In B. K. Roy, editor, ASIACRYPT, Vol. 3788 of Lecture Notes in Computer Science, pp. 585–604. Springer, 2005.

[16] D. Pointcheval and S. Zimmer, "Multi-Factor Authenticated Key Exchange," *in Proceedings of Applied Cryptography and Network Security*, pp. 277-295, 2008.

[17] C. Adams, The simple public-key GSS-API mechanism (SPKM). *The Internet Engineering Task Force (IETF)*, RFC 2025, Oct 1996.

[18] https://www.ibm.com/developerworks/library/l-network-filesystems/index.html

[19] https://www.ibm.com/developerworks/library/l-pnfs/index.html

[20] http://searchenterprisedesktop.techtarget.com/definition/Network-File-System

[21] https://storagegaga.wordpress.com/category/nfs/

[22] https://en.wikipedia.org/wiki/Clustered_file_system#Distributed_file_systems

[23] A. Adya, W.J. Bolosky and M. Castro, "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment", *Appears in 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, pp.1-14 December 2002.

[24] K. Aguilera, Minwen Ji and Mark Lillibridge,"Block Level Security for Network-Attached Disks Marcos," HP Systems Research Center∗ Palo Alto, CA, pp.1-18.

[25] Ran Canetti and Hugo Krawczyk,"Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", pp.451-472.