PROTOCOLS USING KEY EXCHANGE FOR PARALLEL NETWORK FILE SYSTEM

N. Sireesha[1], Bathala. Muni Hema Kumar[2]
[1]*Dept of MCA, SKIIMS, Kapugunneri, Affiliated to S.V.University, Tirupati*
[2]*Assistant Proffessor, Dept of MCA, SKIIMS, Kapugunneri, Affiliated to S.V.University, Tirupati*

***Abstract*- The key establishment difficulty is the most important issue and we learn the trouble of key organization for secure many to many communications for past several years. The difficulty is enthused by the propagation of huge level dispersed file systems behind parallel admission to manifold storage space plans. Our task focal points on the present Internet ordinary for such file systems that is parallel Network File System [pNFS], which creates employ of Kerberos to set up similar session keys flanked by clients and storage strategy. (a) a metadata server make possible key swap over sandwiched between the clients and the storage devices has important workload that put a ceiling on the scalability of the procedure; (b) the procedure does not make available frontward confidentiality; (c) the metadata server produces itself all the assembly keys that are used between the clients and storage devices, and this intrinsically shows the way to key escrow. Demonstrate that our procedures are competent of plummeting up to roughly 54% of the workload of the metadata server and concomitantly at the bottom of onward confidentiality and escrow freeness.**

***Index Terms*- Parallel sessions, authenticated key exchange, net- work file systems, forward Secrecy, key escrow.**

I. INTRODUCTION

In a parallel file system, file data is distributed across multiple storage devices or nodes to allow concurrent access by multiple tasks of a parallel application. This is typically used in large-scale cluster computing that focuses on high performance and reliable access to large datasets. That is, higher I/O bandwidth is achieved through concurrent access to multiple storage devices within large compute clusters; while data loss is protected through data mirroring using fault-tolerant striping algorithms.

In this work, we investigate the issue of the secure many-many communications in the large scale network file systems which support parallel fetch to multiple storing devices. That we considering the communication model where there are a large number of the clients accessing multiple remote and distributed storage devices in parallel. Particularly, we tries to focus on how to exchange the key materials and establishment of the parallel secure sessions between clients and storage devices in the parallel Network File System (pNFS).

II. MODULES

Clients:
In this module, client browse a file encrypt and upload to the router. Generates a mac address for the particular file while uploading and Sends Meta data to Meta data server.

Router:
Receive encrypted data from client. Exchange all files key by assigned date from KDC and Send updated keys to corresponding files, Send key exchange details to Meta data server to replace old keys. Capture key and file attackers and block. Generate log about key exchange and View all files and keys, Decrypts the data and sends to the receiver

Key Distribution Server (KDC):
Generate New Keys and exchange keys in the server on the date period

Meta data server:
Data owner send Meta data to keep copy of the file and Send log about file integrity and Capture all attackers

Receivers:
Request secret key and available files in the router, Request and receive decrypted files.

Attacker:
Type-1: Attacks secret key
Type-2: Injects malicious data and corrupts original File.

III. EXISTING AND PROPOSED SYSTEMS

A. Existing System

Parallel Network File System (pNFS), which makes use of Kerberos to establish parallel session keys between clients and storage devices our review of the existing Kerberos-based protocol shows that it has a number of limitations.

Drawback of Existing System:

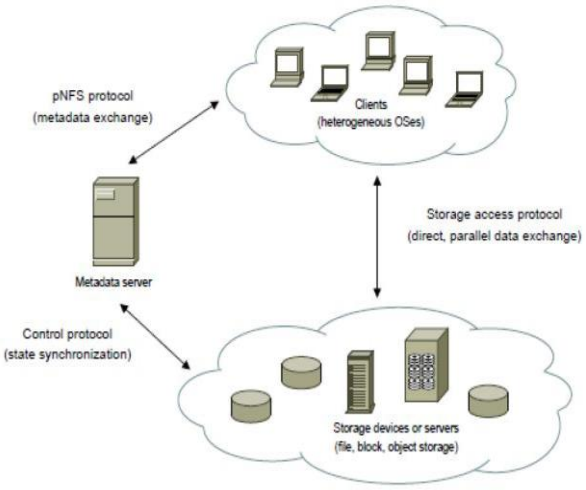
- Devices have heavy workload that restricts the scalability of the protocol.
- The protocol does not provide forward secrecy.
- The metadata server generates itself all the session keys that are used between the clients and storage devices, and this inherently leads to key escrow.

B. Proposed System:

In this project, we propose a variety of authenticated key. Exchange protocols that are designed to address the existing issues. We show that our protocols are capable of reducing the workload of the metadata server and concurrently supporting forward secrecy and escrow-freeness. All this requires only a small fraction of increased computation overhead at the client. We focus on how to exchange key materials and establish parallel secure sessions between the clients and the storage devices in the parallel Network File System (pNFS) the current Internet standard.

Advantage of Proposed System:

- Secure authenticated key exchange protocols.
- Efficiency and security.



IV. SECURITY CONSIDERATION

Previous descriptions of NFS listening carefully on Straightforwardness and competence, and were intended to employment well on intranets and restricted networks. Afterward, the afterward versions aspire to get better access and presentation within the Internet environment. However, safekeeping has then become a greater concern. Among many other sanctuary issues, user and server authentication within an open, dispersed, and cross-domain environment are a difficult matter. Key management can be tedious and luxurious, but an important aspect in ensuring security of the system. Moreover, data time alone may be critical in high performance and parallel applications, for example, persons associated with biomedical in sequence sharing, financial data processing and analysis and drug simulation & discovery. Hence, distributed storage devices pose greater risks to various security threats, such as illegal modification or stealing of data residing on the storage devices, as well as interception of data in transit between different nodes within the system. NFS (since version 4), therefore, has been mandating that implementations support end-to-end authentication, where a user (through a client) mutually authenticates to an NFS server. Moreover, consideration should be given to the integrity and privacy (confidentiality) of NFS requests and responses. The RPCSEC GSS framework is currently the core security component of NFS that provides basic security services. RPCSEC GSS allows RPC protocols to access the Generic Security Services Application Programming Interface (GSS-API). The latter is used to facilitate exchange of credentials between local and remote communicating parties, for example between a client and a server, in order to establish a security context. The GSS-API achieves these through an interface and a set of generic functions that are independent of the underlying security mechanisms and communication protocols employed by the communicating parties. Hence, with RPCSEC GSS, various security mechanisms or protocols can be employed to provide services such as, encrypting NFS traffic and performing integrity check on the entire body of an NFSv4 call. Similarly, in pNFS, communication between the client and the metadata server are authenticated and protected through RPCSEC GSS. The metadata server grants access permissions (to storage devices) to the client

according to pre- defined access control lists (ACLs). The client's I/O request to a storage device must include the corresponding valid layout. Otherwise, the I/O request is rejected. In an environment where eavesdropping on the communication between the client and the storage device is of sufficient concern, RPCSEC GSS is used to provide privacy protection.

A. Parallel Sessions

Parallel secure sessions between the clients and the storage devices in the parallel Network File System (pNFS),the current Internet standard—in an efficient and scalable manner 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 compute the same session key. Second, two our protocols provide forward secrecy: one is partially forward securing with respect to multiple sessions within a time period.

B. Authenticated Key Exchange

Our primary goal in this work is to design efficient and secure authenticated key exchange protocols that meet specific requirements of pNFS. The main results of this paper are three new provably secure authenticated key exchange protocols as shown in Fig.1. We describe our design goals and give some intuition of a variety of pNFS authenticated key exchange6 (pNFS-AKE) protocols that we consider in this work.

C. Forward Secrecy

The protocol should guarantee the security of past session keys when the long-term secret key of a client or a storage device is compromised. However, the protocol does not provide any forward secrecy. To address key escrow while achieving forward secrecy simultaneously, we incorporate a Diffie-Hellman key agreement technique into Kerberos-like.

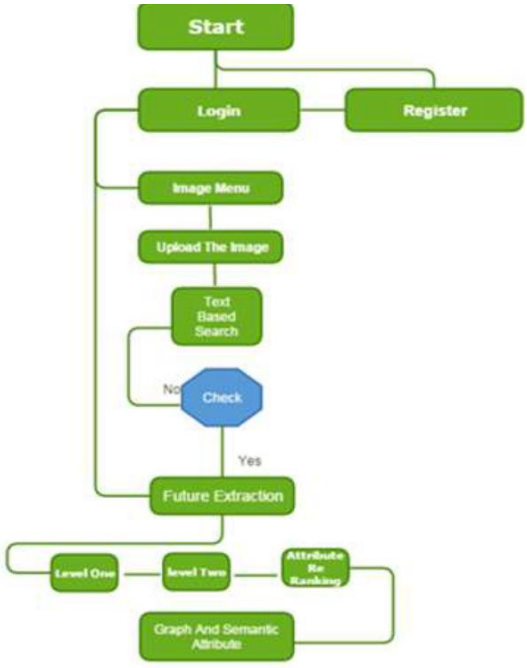
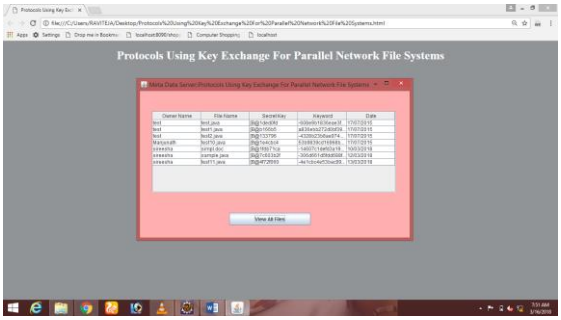


Fig: 1 Flow Diagram for Proposed System

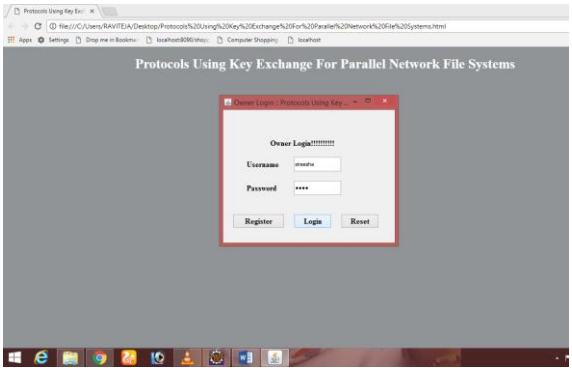
V. SCREEN SHOTS:

Storage Server:

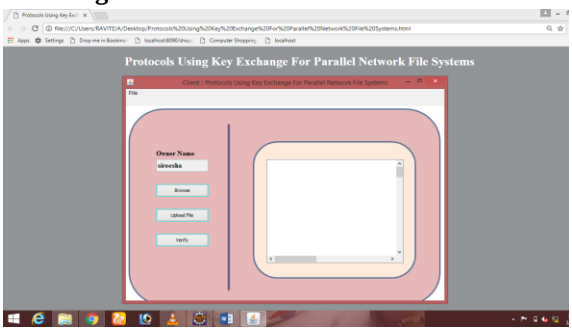
Meta Data Server:



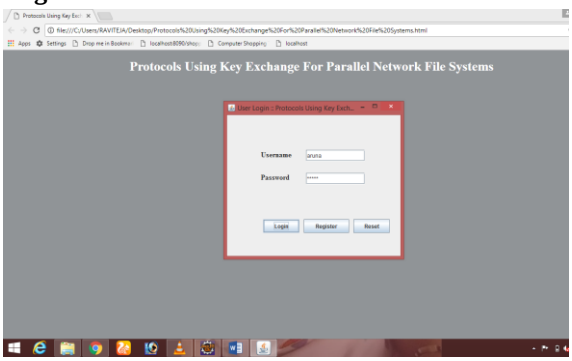
Login for Owner:



Home Page for Owner:



Login for End User:



Home Page for End User:

VI. CONCLUSION

We planned three genuine key swaps over protocols for parallel network file system (pNFS). Our procedures present three attractive compensations over the obtainable Kerberos based pNFS procedure. Primary, the metadata server implementing our procedures has much subordinate workload than that of the Kerberos based move toward. Subsequent, two our procedures make available frontward confidentiality: one is incompletely frontward protected [with admiration to manifold assemblies within an occasion era], at the same time as the additional is completely onward protected [with admiration to an assembly). Next, we have intended a procedure which not only make available onward confidentiality, other than is too escrowing gratis.

REFERENCES

[1] M. Abd-El-Malek, W.V. Courtright II, C. Cranor, G.R. Ganger, J. Hendricks, A.J. Klosterman, M.P. Mesnier, M. Prasad, B. Salmon, R.R. Sambasivan, S. Sinnamohideen, J.D. Strunk, E. Thereska, M. Wachs, and J.J. Wylie. Ursa Minor: Versatile cluster-based storage. In *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST)*, pages 59–72. USENIX Association, Dec 2005.

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

[3] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI)*. USENIX Association, Dec 2002.

[4] M.K. Aguilera, M. Ji, M. Lillibridge, J. MacCormick, E. Oertli, D.G. Andersen, M. Burrows, T. Mann, and C.A. Thekkath. Blocklevel security for network-attached disks. In *Proceedings of the 2nd International Conference on File and Storage Technologies (FAST)*. USENIX Association, Mar 2003.

[5] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58. ACM Press, Apr 2010.

[6] Amazon simple storage service (Amazon S3). http://aws.amazon.com/ s3/.

[7] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against

dictionary attacks. In *Advances in Cryptology – Proceedings of EUROCRYPT*, pages 139–155. Springer LNCS 1807, May 2000.
[8] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology – Proceedings of CRYPTO*, pages 258–275. Springer LNCS 3621, Aug 2005.

AUTHOR PROFILE:

AUTHOR 1:

N.Sireesha received Graduate Degree in B.Sc Computer Science from Sri Venkateswara University, Tirupati in the year of 2012-2015. Pursuing Master of Computer Applications from Sri Kalahastiswara Institute of Information and Management Sciences, Srikalahasti, Affiliated to Sri Venkateswara University, Tirupati in the year of 2015-2018.

AUTHOR 2:

B. Muni Hema Kumar Working as an Assistant Professor in Dept. of Computer Science , Sri Kalahastiswara Institute of Information and Management Sciences, Srikalahasti (AP) India. Received Master of Computer Applications from Sri Venkateswara University, Tirupati.