Authenticated Key Exchange Protocols for Parallel Network File Systems

**Abstract:**

In data communication system Key establishment is the major process for providing security to the entire network while transfer the data between the neighbors. Let us consider the secure many-to-many communications, the problem is inspired by the proliferation of large-scale distributed file systems supporting parallel access to multiple storage devices. Our work focuses on the current Internet standard for such file systems, i.e., parallel Network File System. So, Our system should propose a variety of authenticated key exchange protocols that are designed to address the issues during the key establishment.

**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.

Our proposed technique investigates the problem of secure many to many communications in large scale network file systems that support parallel access to multiple storage devices.

Proposed communication model where there are a large number of clients accessing multiple remote and distributed storage devices in parallel. Particularly, 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 the current Internet standard in an efficient and scalable manner.

**Existing System:**

The current design of NFS/pNFS focuses on interoperability, instead of efficiency and scalability, of various mechanisms to provide basic security. Moreover, key establishment between a client and multiple storage devices in pNFS are based on those for NFS, that is, they are not designed specifically for parallel communications. Hence, the metadata server is not only responsible for processing access requests to storage devices (by granting valid layouts to authenticated and authorized clients), but also required to generate all the corresponding session keys that the client needs to communicate securely with the storage devices to which it has been granted access. Consequently, the metadata server may become a performance bottleneck for the file system. Moreover, such protocol design leads to key escrow. Hence, in principle, the server can learn all information transmitted between a client and a storage device. This, in turn, makes the server an attractive target for attackers. Another drawback of the current approach is that past session keys can be exposed if a storage device’s long-term key shared with the metadata server is compromised. We believe that this is a realistic threat since a large-scale file system may have thousands of geographically distributed storage devices. It may not be feasible to provide strong physical security and network protection for all the storage devices.

**Disadvantages:**

* a metadata server facilitating key exchange between the clients and the storage devices has 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.

**Proposed System:**

Our proposed system implements a variety of authenticated key exchange protocols that are designed to address the above issues. We show that our protocols are capable of reducing up to approximately of the workload of the metadata server and concurrently supporting forward secrecy and escrow-freeness. Security component of NFS that provides basic security services.

All this requires only a small fraction of increased computation overhead at the clientandto reduce the workload of the metadata server. On the other hand, the computational and communication overhead for both the client and the storage device should remain reasonably low.

a stronger security model with forward secrecy for three of our protocols such that compromise of a long-term secret key of a client C or a storage device Si will not expose the associated past session keys shared between C and Si. Further, we would like an escrow-free solution, that is, the metadata server does not learn the session key shared between a client and a storage device, unless the server colludes with either one of them.

**Advantages:**

* In the last augmented game, we can claim that the adversary has no advantage in winning the game since a random key is returned to the adversary.
* Our protocols offer three appealing advantages over the existing Kerberos-based pNFS protocol.

**Modules:**

**Cloud Network Formation:**

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 secure with respect to multiple sessions within a time period.

**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. 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

**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 pNFS-AKE-I.

**User Privileges:**

Share Data

The user can share their data into another user in same group the data will translate by path setting data.

Upload Data

The user can upload the file to cloud. And the Admin can allow the data to store the cloud.

Download File

The user also downloads the cloud file by the conditions.

**Server Authentication:**

Accept user

The admin can accept the new user request and also black the users.

Allow user file

The users can upload the file to cloud. And the admin can allow the files to cloud then only the file can store the cloud.

**Date flow diagram**

Level 0

Client registration

Cloud Service Provider

Generate Secrete key

Level 1

User

Output

Cloud Service Provider

Level 2

client

CSP

**TPA**

Verify secrete key

Level 3

Users retrieve data from csp

Client

CSP

**TPA**

Verify secrete key

Decrypt data & download data from CSP

**Use case Diagram**

User

CSP

TPA

Class Diagram

Cloud service provider

Client

Csp name

Maintain user details ()

Available space details

name

address

Contact details

Upload files()

Download file() files()

TPA

Secret key generation ()

Maintain key & ID details

Verify key ()

Encrypt files ()

Decrypt files ()

Public data audting()

Activity diagram

Cloud service provider

client

Login

registration

Upload file

View &Download file

yes

No

Secrete Key generation

Public data auditing

View file details

Client details

TPA

Key verification

**Conclusion:**

Our protocols offer three appealing advantages over the existing Kerberos-based pNFS protocol. First, the metadata server executing our protocols has much lower workload than that of the Kerberos-based approach. Second, two our protocols provide forward secrecy: one is partially forward secure (with respect to multiple sessions within a time period), while the other is fully forward secure (with respect to a session). Third, we have designed a protocol which not only provides forward secrecy, but is also escrow-free.

**References:**

* F. Hupfeld, T. Cortes, B. Kolbeck, J. Stender, E. Focht, M. Hess, J. Malo,J. Marti, and E. Cesario. The XtreemFS architecture – a case for objectbased file systems in grids. *Concurrency and Computation: Practice and Experience (CCPE)*, 20(17):2049–2060. Wiley, Dec 2008.
* Hadoop distributed file system. <http://hadoop.apache.org/hdfs/>. [27] J. Kubiatowicz, D. Bindel, Y. Chen, S.E. Czerwinski, P.R. Eaton, D. Geels, R. Gummadi, S.C. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B.Y. Zhao. OceanStore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 190–201. ACM Press, Nov 2000.
* S. Langella, S. Hastings, S. Oster, T. Pan, A. Sharma, J. Permar, D. Ervin, B.B. Cambazoglu, T.M. Kurc¸, and J.H. Saltz. Model formulation:
* Sharing data and analytical resources securely in a biomedical research grid environment. *Journal of the American Medical Informatics Association (JAMIA)*, 15(3):363–373. BMJ, May 2008.
* A.W. Leung and E.L. Miller. Scalable security for large, high performance storage systems. In *Proceedings of the ACM Workshop on Storage Security and Survivability (StorageSS)*, pages 29–40. ACM Press, Oct 2006.
* A.W. Leung, E.L. Miller, and S. Jones. Scalable security for petascale parallel file systems. In *Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing (SC)*, page 16. ACM Press, Nov 2007.