ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

Secure File System Architecture- Based Session Key Generation in pNFS

Sreedevi R

PG Student [CS], Dept. of C.S, Mar Baselios Institute of Technology and Science, Nellimattom, Kerala, India

ABSTRACT: Defining a new model in large scale distributed file systems for solving problems such as heavy workload on metadata server, absence of forward secrecy (exposure of past session keys by the compromise of long-term secret key), server compromise and loss of session keys (key- escrow); nowadays the system uses Kerberos for establishing parallel session keys. The analysis on the existing Kerberos –based protocol displays that it has a number of limitations. A new version of the Network File System (NFS) standard has established the Internet Engineering Task Force (IETF) standards: parallel NFS (pNFS). The newer version of NFS does not require any changes to the existing applications. The primary goal here is to design an efficient and secure authenticated key exchange protocol that meets the specific requirements of pNFS, such as scalability to metadata server, forward secrecy and escrow- freeness. pNFS concurrently moves data over parallel paths to the compute nodes. The new method proposes a variety of authenticated key exchange protocols. These are designed to address the issues which were faced by the existing system. We show that our protocols are efficient to handle and reducing the workload of the metadata server and concurrently supporting forward secrecy and escrow-freeness. Only a small fraction of increased computation overhead at the client is what all required.

KEYWORDS: Network File System; pNFS; Forward Secrecy; Key Escrow; Authenticated Key Exchange

I. INTRODUCTION

A new era in high- performance systems and applications often requires data sets far larger than system memories or a single interconnect can accommodate. Compute clusters can process huge amounts of data, but storage I/O often becomes the bottleneck. The NFS (Network File System) Protocol originally developed by Sun, is still the workhorse storage protocol for large- scale and clustered computing. NFS allows sharing of files between a numbers of machines on a network. Part of the popularity of NFS is its ease of use. Mount the file system and it accesses files as if they were local rather than on the network. The NFS client sees the directories and files. When the client requests a file the NFS server sends it over the network to the client. Today's NFS server stores not only the files, but also the file metadata, including the file's length, type (.doc, .pdf, etc), and creation date and which blocks on the storage system contain the file. Metadata is nothing but data about data. In particular, metadata server contains the file rights of all the files contained in the block storage along with the pointers to all those files. Yet the simplicity of standard NFS comes at a price, it is difficult to break a file into pieces for parallel delivery multiple storage systems to multiple clients over the network. For large files, or for large number of clients, the network becomes a bottleneck. In this work discussing a major new technology refresh for NFS that will be released as the NFS v4.1 standard. This is the first performance improvement to NFS in many years. The new version of NFS i.e. NFS v4.1 contains a new technology, called parallel NFS (pNFS). This new IETF (Internet Engineering Task Force) standard, parallel NFS (pNFS), with wide industry backing, is the first industry standards-based solution for high- performance I/O. Panasas has deep expertise in parallel file systems and expects to be one of the first vendors to support pNFS. Network Appliance, IBM, EMC, and Sun are also likely to implement the new standard, as they are key members of the IETF pNFS committee. pNFS aims to increase the performance of Distributed systems. pNFS, an integral part of NFSv4.1, promises to bridge the gap between the performance requirements of large, parallel applications and their Interoperability and security requirements. The pNFS standard enables high- speed parallel data transfers without requiring any changes to applications or operating systems. The pNFS- enabled storage separates data and metadata. The storage systems have the data. The metadata is stored on a pNFS metadata server that acts as a control node. The metadata server keeps track

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

of where the data is stored. Current NFS protocols force clients to access all files on a given file-system volume from a single server node, which can become a bottleneck for scalable performance. As a standardized extension to NFSv4.0, however, pNFS provides clients with scalable end-to-end performance and the flexibility to interoperate with a variety of clustered storage service architectures. The pNFS protocol enables clients to directly access file data spread over multiple storage servers in parallel. As a result, each client can leverage the full aggregate bandwidth of a clustered storage service at the granularity of an individual file. A standard protocol also improves manageability of storage client software and allows for interoperability across heterogeneous storage nodes. Finally, the pNFS protocol is backward-compatible with the base NFSv4.0 protocol. This allows interoperability between old and new clients and servers. In traditional distributed systems, there is only one metadata server. Metadata server monitors the access to the files requested by the clients in Distributed File System. The metadata server is the backbone of the parallel network file system since it handles the requests of the clients and directs them to the respective block storage. The metadata server facilitates flexibility as the clients can directly access the data from the block storage. In file systems upto NFSv4.0, the data movement was handled by data servers. Hence, data had to be moved from the block storage to the data servers and from the data servers to the clients. However, using a metadata server in NFSv4.1, data movement is halved and block storage access is given directly to the clients. This substantially reduces the data movement and increases efficiency and speed of data access. High-performance data centers have been aggressively moving toward parallel technologies like clustered computing and multi-core processors. While this increased use of parallelism overcomes the vast majority of computational bottlenecks, it shifts the performance bottlenecks to the storage I/O system. To ensure that compute clusters deliver the maximum performance, storage systems must be optimized for parallelism. Defining a new model in large scale distributed file systems for solving problems such as heavy workload on metadata server (performance bottleneck), absence of forward secrecy (exposure of past session keys by the compromise of long- term secret key), server compromise and loss of session keys (key- escrow); nowadays the system uses Kerberos for establishing parallel session keys. The analysis on the existing Kerberos –based protocol displays that it has a number of limitations. The newer version of NFS does not require any changes to the existing applications. The primary goal here is to design an efficient and secure authenticated key exchange protocol that meets the specific requirements of pNFS, such as scalability to metadata server, forward secrecy and escrow-freeness. pNFS concurrently moves data over parallel paths to the compute nodes. The new method proposes a variety of authenticated key exchange protocols. These are designed to address the issues which were faced by the existing system. We show that our protocols are efficient to handle and reducing the workload of the metadata server and concurrently supporting forward secrecy and escrow-freeness. Only a small fraction of increased computation overhead at the client is what all required.

II. SYSTEM MODEL

A. Overview:

Petabyte-scale file systems are often extremely large, containing gigabytes or terabytes of data that can be spread across hundreds or thousands of storage devices. Hence, the cost of security operations can be very high. Recent security proposals for large-scale file systems have been focussing on the use of hybrid symmetric and asymmetric key cryptographic techniques, in order to strive for a balance between security and performance. However, key management issues, such as distribution, renewal and revocation of keys, have not been explicitly addressed. We know that key management can be very challenging and costly in large-scale systems, and can have significant impact on the scalability of the systems. We then propose a secure file system architecture which makes use of lightweight key management techniques. Our approach not only addresses essential key management concerns, it also improves existing proposals with stronger security and better usability. Cryptographic key management, such as generation, distribution, storage, renewal and revocation of keys are the foundations for securing a system. It can be very challenging, particularly for petabyte-scale, open distributed file systems because the cost of managing keys may be far higher than the cost of executing security mechanisms or protocols in which the keys are used. Files in such systems are often extremely large, containing terabytes of data, which can be spread across thousands of devices and accessed by thousands of clients. While it is possible that the authenticated public key of each storage device is made available to all clients within the system, it can be very difficult and costly for the clients to manage such large sets of public keys, particularly in terms of key renewal and revocation. Most prior work on large-scale file systems, particularly

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

those based on object storage devices or network-attached disks, presupposes that all keys required performing security protocols are held by all the relevant parties. For example, a set of security protocols designed to provide strong and scalable security for petabyte-scale file systems by making use of various state-of-the-art techniques, such as extended capabilities, automatic revocation and secure delegation. Each client knows the authenticated public key of all other entities in the system. This is a rather strong assumption which has significant impact on the overall performance and scalability of the system. Each storage device within a file system possesses a public/private key pair, of which the public component is assumed to be available to all clients in the system. A client must then, using the relevant storage device public keys, establish shared symmetric keys with storage devices on which the client stores its files. (This is necessary so that the client can subsequently access its files in a secure and efficient manner.) This process of establishing a shared key has to be repeated thousands of times if the client has many large files stored across thousands of storage devices. A more worrying concern is that if a storage device is compromised or corrupted, and thus so as its private key and the symmetric keys that the device shares with its clients, revocation and replacement of these keys can be extremely tedious and costly. Similarly, when new storage devices are added to the system, it may also be costly to distribute new public keys to all the clients. Other recent proposals, such as, also assumed the existence of all necessary authenticated public keys and did not address issues related to key distribution, key renewal and revocation either.

Here the secure file system architecture (SFSA) which not only addresses the aforementioned key management issues, but also has stronger security and better usability in comparison with the recent proposals. In large-scale and highly distributed file systems, storage devices can be vulnerable to various attacks and may encounter hardware or software failure, and thus this may lead to exposure of secret cryptographic keys or data stored on the devices. However, this has not been explicitly taken into consideration in the existing security model, although it has significant impact on not only key management, but also the data protected by the relevant keys. Hence, we first propose a stronger security model by considering forward secrecy, an essential security property for shared key establishment between two parties. In our security model, we assume that a long-term secret key stored on a storage device can be corrupted or exposed, and that even when the key is revealed to an adversary, past session keys derived using the long-term key are still protected from the adversary. Our proposal of SFSA then employs lightweight key management techniques suitable for petabyte-scale object storage file systems. In our approach, clients make use of only short-lived cryptographic keys in order to negotiate shared symmetric keys with storage devices. Thus this obviates many difficulties in public key management, such as key revocation and renewal. We develop an authenticated key agreement protocol using the metadata information. Our protocol is not only lightweight, but also shown to be secure in the stronger security model.

B. Related Work:

Earlier work in securing large-scale distributed file systems, for example, employed Kerberos for performing authentication and enforcing access control. Kerberos, being based on symmetric key cryptography, is usually regarded as a very efficient approach. However, it is generally believed to be more suitable for rather closed, well-connected distributed environments. On the other hand, data grids and file systems such as, OceanStore, LegionFS and FARSITE, make use of public key cryptographic techniques and public key infrastructure (PKI) to perform cross-domain user authentication. Each user of these systems is assumed to possess a certified public/private key pair. However, these systems were not designed specifically with usable and scalable security in mind.

With the increasing deployment of highly distributed and network-attached storage systems, focussed on scalable security. Nevertheless, these proposals assumed that a metadata server shares a group secret key with each distributed storage device. The group key is used to produce capabilities in the form of message authentication codes. However, compromise of the metadata server or any storage device allows the adversary to impersonate the server to any other entities in the file system. This issue can be alleviated by requiring that each storage device shares a different secret key with the metadata server. Nevertheless, such an approach restricts a capability to authorising I/O on only a single device, rather than larger groups of blocks or objects which may reside on multiple storage devices. The protocol that facilitate (i) authenticated key establishment between clients and storage devices, (ii) capability issuance and renewal, and (iii) delegation of the key. The authenticated key establishment protocol allows a client to establish a shared

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

(session) key with a storage device by performing the following steps: generates a public/private key pair; Certifies the newly created public key, from the metadata server; Creates and transports a session key to the storage device;

In order to access files on specific storage devices, the client then obtains a short-term capability from the metadata server. The capability specifies the client's I/O request and access permissions, which allow the storage devices to evaluate whether or not the client is authorised to access the requested data.

III. PROPOSED SYSTEM

A. *Secure File System Architecture (SFSA)*

Our proposal of secure file system architecture (SFSA) is based on the lightweight key management and improved usability in our approach. In SFSA, a client does not make use of long-term cryptographic keys when accessing data on storage devices. All is assumed, from the client's perspective, is that it needs to share a secret, i.e. password, with its metadata server. The client then interacts with the storage devices using only short-term cryptographic keys, which in turn, will be destroyed at the end of a session. We also assume that each storage device pre-distributes some keying material to the metadata server when it is first added to the system. Figure 1 gives an overview of our security architecture.

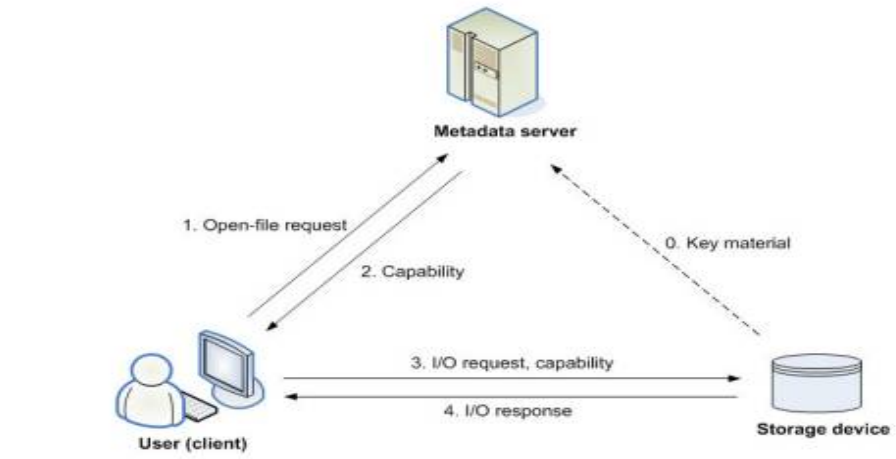

Figure 1: Architectural view of Secure File System Architecture

B. *Security Model*

The security model used in existing proposals assumes that both a metadata server and storage devices are trusted entities. The former is trusted to act as a reference monitor and issue valid capabilities, while the latter are trusted to store data and only perform I/O operations upon authorised requests. On the other hand, no implicit trust is placed on clients.

A large-scale file system may have thousands of storage devices which are geographically distributed. Thus it may not be feasible to provide strong physical security and network protection for all the devices. This implies that storage devices are at a higher risk of being compromised compared to a metadata server, which can usually be physically secured and protected. Furthermore, storage devices may encounter hardware or software failure, causing the data stored on them no longer accessible. We believe that these risks are serious and have significant impact on the design of security protocols and their associated key management.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

We, therefore, raise the security bar of protocols for large-scale object storage systems by considering forward secrecy, which has already been an important part of the security requirements for most modern security protocols. The idea of forward secrecy is that previous encrypted data is securely locked in the past. For example, if a key establishment protocol performed between a client and a storage device provides forward secrecy, past session keys shared between the client and the storage device will not be exposed even if one of their long-term private key (or secret component) has been compromised. This is to ensure that all data encrypted under past session keys and exchanged between the two parties will not be revealed. This is clearly a very desirable security property, particularly if we assume that cryptographic keys stored in a storage device can be vulnerable to exposure.

Our authenticated key agreement protocol is based on the asynchronous mail-sms-split key exchange technique. In our security model, we assume that the entire communication network is managed by an adversary, who may schedule interactions arbitrarily, and who may inject, modify and drop messages arbitrarily. We say that the authenticated key agreement protocol between a client, a metadata server and a storage device is secure if all the following conditions are satisfied.

(i) The client and the storage device are mutually authenticated.
(ii) No useful information about the session key between the client and the storage device is revealed to the adversary during a successful protocol run.
(iii) The exposure of the current session key does not leak any information about past session keys.
(iv) The exposure of either the client's, metadata server's or storage device's long-term credential does not leak any information about past session keys, thus achieving forward secrecy.

C. *Key Management*

In SFSA, we employ a lightweight and "just-in-time" key management approach, in which the client is not required to possess long-term public keys of storage devices, but makes use of only ephemeral key materials in order to establish a session key with each storage device. These short-lived key materials are generated as needed, and destroyed at the end of a security session, and there-fore the client can avoid long-term key management and its associated issues. Furthermore, the client does not perform any asymmetric cryptographic operations during key establishment with the storage devices. More importantly, our simplified key management approach suits perfectly the stronger security model that ensures forward secrecy.

D. *Pre- Distribution Of Keys*

In our approach, each storage device distributes key material to the server through an authenticated channel. This can also be achieved during the registration process. For each file registration, a storage device's key material is distributed to a client as part of the capability issued to the client. This way, the task of obtaining and validating the storage device's key material has been transferred from the client to the server. We remark that this eases significantly the problem of revoking the key material if the associated secret component is compromised. This is because it is much easier and sensible for only the server to obtain an up-to-date revocation list on a timely basis, instead of relying on all clients to update their respective revocation lists regularly and check for revoked key materials.

E. *Session Key Generation*

In our SFSA approach, a fresh shared key is used between the client and a storage device for each new security session. Both the client and the storage device share a key- material, from which a value can be computed and used to derive the session key.

F. *Deletion Of Keys*

At the end of each file access session between the client and the storage device, all session keys are deleted from the memory upon completion of a protocol run. This implies that the client uses only short-term fresh session keys. Hence,

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

key revocation is not a major concern from the client's perspective. The deletion of session key is essential in order to achieve forward secrecy.

G. *Authenticated Key Exchange*

In our simplified key exchange approach, users make use of only key materials and it is possible for a client and a storage device to agree on a session key based on the key materials in a lightweight manner. Our 3-party authenticated key exchange protocol in SFSA between a client, a metadata server and a storage device is shown in Figure 1.

H. *Security Analysis*

Each session key of a protocol run is computed based on the metadata information. This implies that, in principle, the computed session key should be indistinguishable from a randomly generated bit string of similar size. Therefore, in the event of exposure of a current session key, the adversary has only a negligible probability of discovering any past session keys through a brute-force search. On certain rare occasion, the adversary may have access to the client's machine or the storage device, thus their long-term credentials are exposed. Similarly if the metadata server's long-term private key is compromised, it is obvious that past session keys are still protected, even though the adversary can now impersonate the server to the clients and storage devices. Hence, we conclude that requirement on forward secrecy is also satisfied.

IV. RESULTS

We implemented the SFSA in java. The codes were compiled with NetBeans and run on Intel(R) Pentium(R) CPU, 2.16 GHz processor, 2.00 GB memory and under Windows 8.1 in 64-bit mode. We used typical choices of cryptographic algorithms AES encryption. The implementation of the algorithms/schemes makes use of optimisation techniques to improve their performance, for example, pre-computation of key generation.

Table 1: A comparison of computational cost for one protocol run in milliseconds

	Kerberos-based protocol	**SFSA**
Server	1.316	1.227
Client	0.085	1.440
Device	0.070	1.420
Total	1.470	4.086

Table 1 shows the actual computation times (in ms) incurred by different entities participating in the existing Kerberos- based protocols and the SFSA protocol. Each computation time is the average time for a protocol run over 1000 iterations.

V. CONCLUSION

Many recent security proposals for network file systems which employ hybrid symmetric and asymmetric key cryptographic techniques did not make use of realistic key management assumptions. In this paper, we considered and addressed key management issues, which could have caused significant efficiency and scalability issues too many existing proposals. In our approach, we adopted lightweight key exchange techniques for session key establishment between a client and a storage device. We also improved the usability of existing security proposals by making use of short-lived credential or key materials. This way, users are not required to manage potentially a large amount of long-term public keys of storage devices. Moreover, we raised the security bar of large-scale network file systems by introducing forward secrecy to protect data exchanged in past security sessions.

To summarise, our authenticated key exchange protocol makes use of pre- computed key materials of metadata information techniques and it does not rely on computationally expensive public key encryption. Moreover, our approach has the luxury of using a fresh session key between a client and a storage device for each new security session (rather than re-using a session key for many different security sessions).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

REFERENCES

1. 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.
2. G. A. Gibson, D. F. Nagle, K. Amiri, J. Butler, F. W. Chang, H. Gobioff, C. Hardin, E. Riedel, D. Rochberg, and J. Zelenka, "A cost-effective, high-bandwidth storage architecture," ACM SIG-PLAN Notices, vol. 33, no. 11, pp. 92–103, Nov. 1998.
3. J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West, "Scale and performance in a distributed file system," ACM Trans. Comput. Syst., vol. 6, no. 1, pp. 51–81, Feb. 1988.
4. 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 object-based file systems in grids,"Concurrency Comput.: Prac. Exp., vol. 20, no. 17, pp. 2049–2060, Dec. 2008.
5. 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 bio-medical research grid environment,"J. Am. Med. Informatics Assoc., vol. 15, no. 3, pp. 363–373, May 2008.
6. O. Tatebe, K. Hiraga, and N. Soda, "Gfarm grid file system,"New Generation Comput., vol. 28, no. 3, pp. 257–275, Jul. 2010.

BIOGRAPHY

Sreedevi R is a M. Tech Student in the Computer Science and Engineering Department, Mar Baselios Institute of Technology and Science, A P J Abdul Kalam Technological University. She received Bachelor of Technology (B. Tech) degree in 2015 from MBITS, Nellimattom, India. Her research interests are Computer Security, Cryptography etc.