International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 3, Issue 4, April 2016

A Secure Parallel Network File System using Protocols

[1] Digambar Waghole [2] Jabbar Tamboli [3] Shrutika Chintal [4] Supriya Vinode
[5] VaishaliShinde
[1][2][3][4][5] Department of Computer Engineering, SavitribaiPhule Pune University, Pune, India,JSPM's RajarshiShahu College of Engineering, Tathawade,Pune.
[1] digambarwaghole@gmail.com [2] jabbar901282@gmail.com [3] shrutikachintal45@gmail.com, [4] supriyavinode@gmail.com [5]svaishu11@gmail.com

Abstract: **- We examine the issue of key foundation for secure numerous correspondences. The issue is enlivened by the multiplication of huge scale dispersed file frameworks supporting parallel access to numerous capacity gadgets. Our work concentrates on the present Internet standard for such file frameworks, i.e. parallel Network File System (pNFS), which makes utilization of Kerberos to establishparallelsessionkeys between clientsandstoragedevices. Our survey of the current Kerberos-based convention demonstrates that it has various impediments: (i) a metadata server encouraging key trade between the customers and the stockpiling gadgets has overwhelming workload that confines the adaptability of the convention; (ii) the convention does not give forward mystery; (iii) the metadata server creates itself all the session keys that are utilized between the customers and capacity gadgets, and this naturally prompts key escrow. In this paper, we propose a mixed bag of confirmed key trade conventions that are intended to address the above issues. We demonstrate that our conventions are fit for decreasing up to give or take 54% of the workload of the metadata server and simultaneously supporting forward mystery and escrow-freeness.**

Keywords**: Authenticated key exchange, Forward secrecy, Network file systems, Parallel sessions**

I. INTRODUCTION

In a parallel file framework, file information is dispersed over various stockpiling gadgets or hubs to permit simultaneous access by numerous undertakings of a parallel application. This is commonly utilized as a part of substantial scale bunch processing that spotlights on elite and solid access to extensive datasets. That is, higher I/O transfer speed is accomplished through simultaneous access to numerous capacity gadgets inside of huge register bunches; while information misfortune is ensured through information reflecting utilizing deficiency tolerant striping calculations. These are generally needed for cutting edge scientific or information concentrated applications, for example, information handling, computerized movement studios, computational user elements, andsemiconductor fabricating. In these situations, hundreds or a great many file framework customers offer information and produce high total I/O load on the le framework supporting petabyte-or terabyte-scale stock. In this work, we explore the issue of secure numerous to-numerous correspondences in vast scale system file frameworks that bolster parallel access to various stockpiling gadgets.

II. RELATED WORK

Some of the earliest work in securing large-scale distributed file systems, We have already employed.

Kerberos for performing authentication and enforcing access control. Kerberos, being based on mostly symmetric key techniques in its early deployment, was generally believed to be more suitable for rather closed, well-connected distributed environments.

On the other hand, data grids and file systems such as, Ocean Store, LegionFS and FARSITE, make use of public key cryptographic techniques and public key infrastructure (PKI) to perform cross-domain user authentication. Independently, Secure File System, also based on public key cryptographic techniques, was designed to enable interoperability of different key management schemes. Each user of these systems is assumed to possess a certified public/private key pair. However, these systems were not designed specifically with scalability and parallel access in mind. With the increasing deployment of highly distributed and network-attached storage systems, subsequent work, such as , focused on scalable security. Nevertheless, these proposals assumed that a metadata server

shares a group secret key with each distributed storage device. Kernel mapping functions for non-linear data sample also uses paillier cryptosystem which is not much secure as compared to AES.

III. SYSTEM ARCHITECTURE

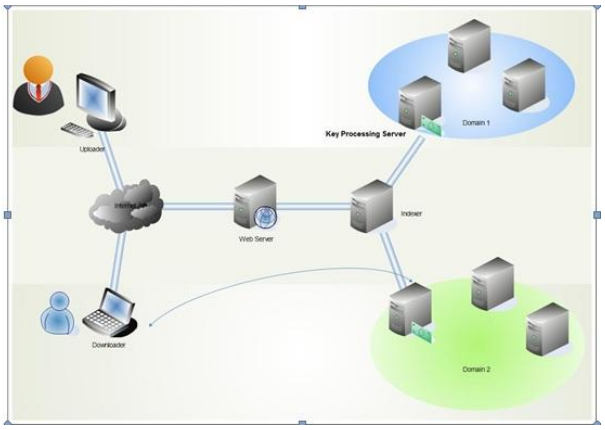

Fig. 1 System Architecture

1) In this architecture the main components are admin and user.
2) Both Admin and User should have project software application installed in his/her machine.
3) Admin or user uploads his data on a server.
4) When Uploading a File, Keys are generated and sent to the receiver's mail.
5) At a receiver side, user can receive the encrypted file and need to decrypt the file using keys.

IV. PROPOSED SCHEME

4.1. RC7 Algorithm

Various algorithms give encryption results which are mostly unbreakable. RC7 is one such algorithm which is an improvement in RC6 by adding to its existing functionalities. The "RC" may stand for either Rivest's Cipher or Ron's code. There have been six RC algorithms so far. To improve the encryption efficiency of the already existing RC6 algorithm, RC7 has been proposed which takes relatively less time for encryption and is comparatively more flexible. Instead of four registers, RC7 makes use of six registers which makes it a better alternative to RC6.

Algorithm:
***Input*:** Plaintext stored in six w-bit input registers "A, B, C, D, E, and F"
Number of rounds "r"
W-bit round keys "S [0 . . . 2r + 1]"
Output: Cipher text stored in A, B, C, D, E, F
Procedure:
B = B + S [0]
D = D + S [1]
F = F + S [2]
for i = 1 to r do
{
t = (B × (2B + 1)) <<<lg w
u = (D × (2D + 1)) <<<lg w
v = (F × (2F + 1)) <<<lg w
A = ((A ⊕ t) <<<u) + S [2i+1]
C = ((C ⊕ u) <<<t) + S [2i+ 2]
E = ((E ⊕ v) <<<t) + S [2i+ 3]
(A, B, C, D, E, F) = (B, C, D, E, F, A)}
A = A + S [2r - 1]
C = C + S [2r]
E = E + S [2r + 1]
RC7 (proposed)is better owing to increased throughput, increased efficiency and decreased encryption time. Its advantages over RC6 are as follows:-
1. Has more secured block cipher.
2. Offers good performance and is more flexible.
3. Makes use of 6 registers instead of 4.
4. It has a block-size of 256 bits.
5. The implementation of the cipher works as per the restrictions provided by the respective systems.

4.2 Speke Algorithm

1. Alice and Bob agree to use an appropriately large and randomly selected safe prime p, as well as a hash function H().
2. Alice and Bob agree on a shared password π.
3. Alice and Bob both construct $g = H(\pi)^2 \bmod p$. (Squaring makes g a generator of the prime order subgroup of the multiplicative group of integers modulo p.)
4. Alice chooses a secret random integer a, then sends Bob $g^a \bmod p$.
5. Bob chooses a secret random integer b, then sends Alice $g^b \bmod p$.
6. Alice and Bob each abort if their received values are not in the range [2,p-2], to prevent small subgroup confinement attack.
7. Alice computes $K = (g^b \bmod p)^a \bmod p$.
8. Bob computes $K = (g^a \bmod p)^b \bmod p$.

Both Alice and Bob will arrive at the same value for K if and only if they use the same value for π. Once Alice and Bob compute the shared secret K they can use it in a key confirmation protocol to prove to each other that they know the same password π, and to derive a shared secret encryption key for sending secure and authenticated messages to each other. Unlike unauthenticated Diffie-Hellman, SPEKE prevents man in the middle attack by the incorporation of the password. An attacker who is able to read and modify all messages between Alice and Bob cannot learn the shared key K and cannot make more than one guess for the password in each interaction with a party that knows it. In general, SPEKE can use any prime order group that is suitable for public key cryptography, including elliptic curve cryptography.

V. MATHEMATICAL MODEL

Let S be a system that describes SPEKE
S={…..} Identify input as I
S={I,..}
Let I ={i1,i2,i3,…..id}
The input will be Blog Attribute .
Identify output as O
S = {I,O,…}
Identify the processes as P
O= The user will get key when he is authenticated.
S={I,O,P,…..}
P={E,D}
E={parameter,id,Attribute of file }
D={parameter,Skid,CTi}
Identify the initial condition as Ic
S={I,O,P,F,s,Ic,…}
Ic=user should always be online and authorized.

Activity: Functions

Let Fs be the Functions used in the Proposed System where Fs={F1,F2,F3,F4,F5,F6}
Let F1 be a rule of S into W such that user logs into the system.
F1(s0)-->{u1,u2,…,un}€W
Let F2 be a rule of U into W such that user inputs a a file attribute .
F2(u0,u1,u2…un)--> (g0,g1,g2…gn) €W
Let F3 be a rule of E into D such that system grabs the values.
F3(g0,g1,g2…gn)-->{X,Y,Z} €D
Let F4 be the rule of A into D such that system receives the value of keys.
F4=(G)-->{Rv} €D
Let F5 be a rule of U into a such that keys is stored into database.
F5(g0,g1,g2…gn)-->{d0| Φd}€A
Let F6 be a rule of U in G such that system recognizes the Generated keys
F6(X,Y,Z)-->{Rx,Ry,Rz}€G
Success Case:
If ({ X,Y,Z})=({d0| Φ}€D)
Then Success.
Failure Case:
If ({ X,Y,Z})≠({d0| Φ}€D)
Then Failure.Validuser = login(uid, passw)
User(Validuser)

Session starts here.

Parse data = load dataset (dataset.csv)
Transform data = Encode dataset (fmkey, scaling key).
Encrypt data = encryption (key, encoded data)
Send to cloud server
Cloud server decrypts data = decrypt (key , encrypted data).
Normalize data = normalize (encoded data)
Train ANN = ANN (normalized data)
Distribute_keys() key distribution to the valid users onlySession ends here.

VI. CONCLUSIONS

We proposed the authenticated key exchange protocols for parallel network file system (pNFS). Our protocols offer three appealing advantages over the existing Kerberos-based pNFS protocol.

1.Forward Secrecy
2.Escrow freeness.
3.Reduce the work load of metadata server.

VII. 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, andJ.J. Wylie. Ursa Minor: Versatile cluster-based storage. In Proceedingsof 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).TheInternet 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 incompletelytrusted environment. In

Proceedings of the 5th Symposium on OperatingSystem Design and Implementation (OSDI). USENIX Association, Dec2002.

[4] M.K. Aguilera, M. Ji, M. Lillibridge, J. MacCormick, E. Oertli,D.G. Andersen, M. Burrows, T. Mann, and C.A. Thekkath. Blocklevelsecurity 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 exchangesecure against dictionary attacks. In Advances in Cryptology– Proceedings of EUROCRYPT, pages 139–155. Springer LNCS 1807.

[8]Parallel virtual file systems (PVFS) version 2. http://www.pvfs.org.

[9] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler,and D. Noveck.Network file system (NFS) version 4 protocol. *The Internet Engineering Task Force (IETF)*, RFC 3530, Apr 2003.

[10]Y. Zhu and Y. Hu. SNARE: A strong security scheme for network attached storage. In Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS), pages 250–259. IEEE Computer Society,
Oct 2003.