International Journal of Computer Applications (0975 – 8887)
Volume 50– No.22, July 2012

A Proposed Robust Authentication Approach for Secure Data Transmission in Grid Computing Environment

Avijit Bhowmick
Dr.B.C.Roy Engg. College
Jemua Road,Fuljhore
Durgapur,West Bengal,India

Nirmalya Mukhopadhyay
Dr.B.C.Roy Engg. College
Jemua Road,Fuljhore
Durgapur,West Bengal,India

Abhishek Bandyopadhyay
Dr.B.C.Roy Engg. College
Jemua Road,Fuljhore
Durgapur,West Bengal,India

ABSTRACT

Grid is an advanced wide area parallel distributed computing environment where unused processor cycles and underutilized storage of numerous computers are utilized efficiently which act as a supercomputer. Security is the most important concern in Grid computing environment. Among all other security issues authentication is the first step of security requirement for any Grid environment to validate the user. This paper proposes an authentication scheme for Grid environment. The proposed authentication scheme optimises the security required for the entry level user and prevents malicious user from entering into the Grid environment.

General Terms

Authentication, OGSA, Grid node.

Keywords

Grid, authenticator, virtual organization, security.

1. INTRODUCTION

Grid computing [1], being a promising way for distributed supercomputing from its very beginning attracts many attentions worldwide. The term *Grid Computing* originated in the early 1990s as a metaphor for making computer power as easy to access as an electric power grid in Ian Foster 's and Carl Kesselman 's seminal work, "The Grid: Blueprint for a new computing infrastructure" (2004).When talking about Grid Computing Environment, the question of a Common security issue arises in mind of experts or Grid users. In Grid computing users accesses the unused CPU-cycles & memory of the resources attached with that Grid using the Grid portals. The resources are generally dispersed world wide, within different silos, called Virtual Organisations. Lee et al[2] proposed a dynamic procedure by which a remote resource can be used with optimal effectiveness. Lee et al [3][4] also advised a dynamic analysing resources model that can retrieve the information about the CPU usages, number of running jobs at a particular span of time on each Grid node to know the load-balancing factor & according to this information reschedule the jobs, make the Grid much effective.

As like all other Computing environments, Grid computing is also affected by the curse of many security issues. A Computational Grid is a collection of heterogeneous computers and resources spread across multiple administrative domains with the intent of providing users easy access to these resources. There are many ways to access the resources of a Computational Grid, each with unique security requirements and implications for both the resource user and the resource provider. A comprehensive set of Grid usage scenarios is presented and analyzed with regard to security requirements such as authentication, authorization, integrity, and confidentiality [5]. The main value of these scenarios and the associated security discussions is to provide a library of situations against which an application designer can match, thereby facilitating security-aware application use and development from the initial stages of the application design and invocation. A broader goal of these scenarios is to increase the awareness of security issues in Grid Computing. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.

2. THE SECURITY FRAMEWORK FOR A GRID SYSTEM

The Open Grid Services Architecture (OGSA)[6] uses WS-security services for authentication. The Control Architecture for Computational Grids (CACG) is shown below:

Acronyms used:

CSA: CHIEF SECURITY ADMINISTRATOR

SA: SECURITY ADVISOR

SN: SECURITY NEGOTIATOR

VOP: VIRTUAL ORGANIZATION POLICY

AGN: ADMINISTRATOR GRID NODE

CGN: COMPUTE GRID NODE

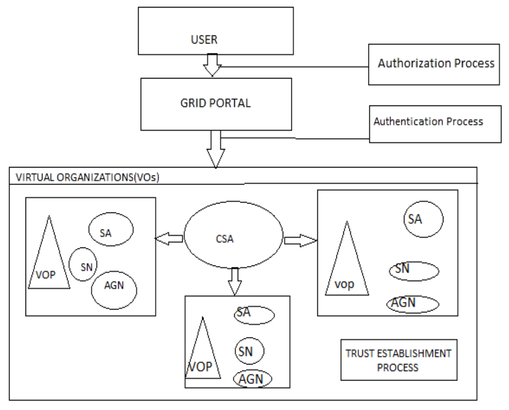


Fig. 1 Block diagram of CACG.

3. DESCRIPTION OF THE PROPOSED AUTHENTICATION SCHEME

To apply the proposed authentication scheme we have classified Grids into two distinct nodes, namely, Administrator Grid Node (AGN) & Compute Grid Node (CGN). A user in CGN requests AGN to process the authenticator operation. After checking, AGN sends acknowledgement to CGN specifying whether the user can continue the process or not.

2.1 Administrator Grid Node (AGN)

To process the user's request AGN needs some tables & operations. To store user information & compare it with the Grid information, AGN needs one database, which is named as Data Base for Generating User Authenticator (DBGUA). The required tables are as follows:

Table: Receive Request: It is used to match the authenticator for the existing users.

Request Command	User-Id	Authenticator

Table: DBGUA: It is used to store the authenticator (produced by the encryption algorithm) for each user.

User-Id	Authenticator

Table: New User Request: This table is used to produce new authenticator for new users. The operation is called new user operation.

New	User-Id	Password	Authenticator

Table: Update User Request: If any user sends update request to AGN for changing his authenticator, AGN uses different attributes to check for authenticator & then if the verification produces no error, it updates the existing authenticator of the user with a new one.

Update	User-Id	Old Password	Old Authenticator	New Password	New Authenticator

Table: Delete User Request: If any user sends delete request to AGN for removing his authenticator, AGN uses different attributes to check for authenticator & then if the verification produces no error, it deletes the existing authenticator of the user.

Delete	User-Id	Password	Authenticator

Table: User Permission Request: If any user needs to access his authenticator, it sends permission request to AGN for getting the chance for accessing.

Permission	User-Id	Password	Authenticator

2.2 Compute Grid Node (CGN)

The user resides in the CGN sends requests to AGN for processing some operations. The acknowledgements received from the AGN are stored in the local database of CGN, namely Compute User Information Data Base (CUIDB).

Table: CUIDB:

User-Id	Authenticator

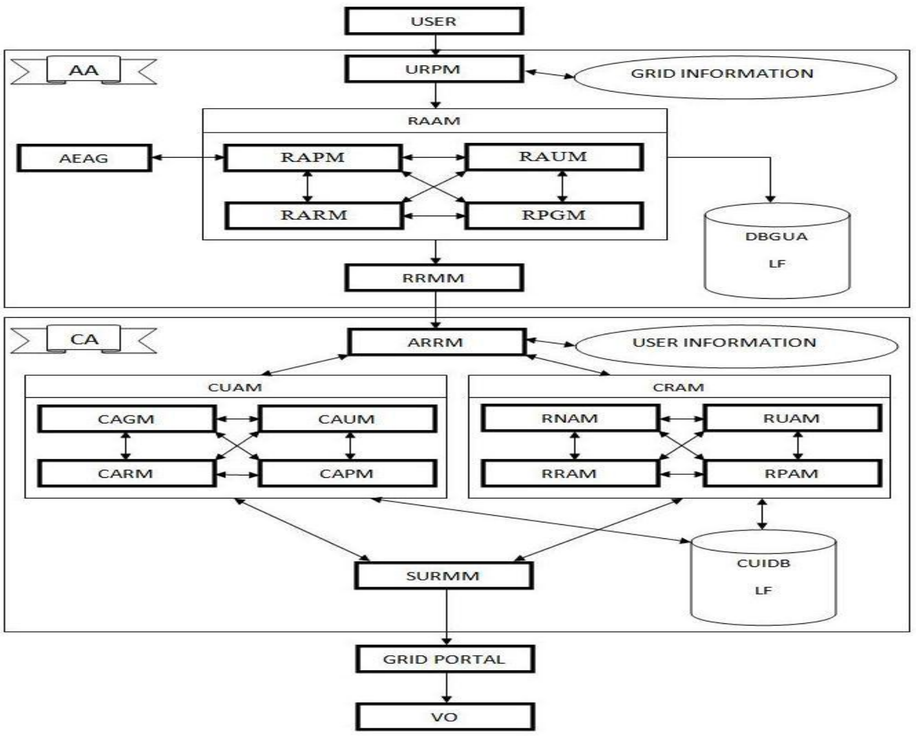


Fig.2 The Architecture of the proposed Authentication scheme

3. THE ARCHITECTURE FRAMEWORK OF THE PROPOSE AUTHENTICATION SCHEME

The acronyms used in the above figure, are described below:

AA: Administrative Authentication: The authentication process used in Administrator Grid Node.

CA: Compute Authentication: The authentication process used in Compute Grid Node.

URPM: User Request Process Module: It resides within AGN & processes the user requests from CGN using the following modules.

RAAM: Remote Authenticator Access Module: It helps user to request for accessing authenticator using the following sub modules.

RAPM: Remote Authenticator Producer Module: It produces authenticator for users. It calls **Authentication Encryption Algorithm Generator (AEAG)** to produce authenticator for a particular user.

RAUM: Remote Authenticator Updater Module: It updates the existing authenticators as per needs.

RARM: Remote Authenticator Remover Module: It deletes the corresponding authenticator if an authentic user wants to delete that.

RPGM: Remote Permission Granting Module: This module is used to provide permissions to access resources as per needs.

RRMM: Returning Result Message Module: It returns the result message to the CGN.

ARRM: Administrator Request Response Module: It returns the user's information as per the needs of AGN.

CUAM: Compute User Authentication Module: It has following four modules:

CAGM: Compute Authenticator Generate Module: It sends new user request to AGN to produce authenticator for the same.

CAUM: Compute Authenticator Update Module: It sends update request to AGN to update existing authenticator.

CARM: Compute Authenticator Remove Module: It sends delete request to AGN to delete existing authenticator.

CAPM: Compute Authenticator Permit Module: It sends permission request to AGN to get permission to access procedures & databases.

CRAM: Compute Receive Authentication Module: It also has the following four modules:

RNAM: Receive New Authenticator Module: It receives new authenticator produced by the new user request & sends it to be stored in the CUIDB.

RUAM: Receive Updated Authenticator Module: It receives the updated version of the existing authenticator.

RRAM: Receive Remove Authenticator Module: It receives the authenticator to be removed.

RPAM: Receive Permit Authenticator Module: It receives the authenticator who got the permission.

SURMM: Send User's Request Message Module: Finally the produced authenticator is sent to the Grid portal.

4. AUTHENTICATION ENCRYPTION ALGORITHM GENERATOR

Related Work: Lee et al presented one algorithm, where they solved some security issues. We, here will modify the existing solution & will provide another solution for user authentication.

Concept: When a user enters his\her User-Id & Password, it comes as plain text; but for security reasons we will convert it into Cipher text. To do so, we must perform the following things:

1. Change the contents of plain text, 2. Network Transmission, 3. Position Exchange, 4. Data Uncertainty & 5. Produce Authenticator.

<u>**Modified Algorithm:**</u>

1. **Change the contents of plain text:**

```
    Input: plaintext (PT) (User-Id=UI, Password=PW), input by the user.

    Do: for N users,

      Create a Symbol Table (ST) to store PT as

      $U_1P_1U_2P_2U_3P_3\ldots U_NP_N$, where N=U+P,

          For all $1\leq i \leq N$,

          Set a key of string by taking
          $KS=U_{i+1}P_{i+2}$ up to N times.

          Convert each semantics of User-Id of
    user $U_i$ into its BCD value.

          Convert each semantics of user $U_{i+1}$ into
          its BCD value.
```

Perform XOR operation as

$U_i'' = ć(U_i')$, where ć is the 1's complement operation , & U_i' is defined as $U_i' = U_i + U_{i+1}$

Convert each U_i'' into its decimal value DU_i''.

Get the Cipher value $CU_i''$for the same as: $CU_i'' = D\ U_i''\%26$

Consider this CU_i'' value as the ASCII of a particular semantics. Replace this semantics with the previous one.

Repeat the same for Passwords to get CP_i''.

End Do.

Output: The plaintext equivalent Cipher text.

2. **Network Transmission:**

For network transmission, we can use TFRC protocol [7], which stands for TCP Friendly Rate Control protocol. We use the following equation (as described by Y. Falik et al.) for transmission through the network:

$$DSR = \frac{K * PS}{D * \sqrt{PLR}}$$

Where, DSR = Data Streaming Rate,

K = Constant factor between .01 to 1.0 depending on the particular network used to transmit the data,

PS = Data Packet Size,

D = End-to-End Transmission Delay &

PLR = End-to-End Packet loss Rate.

Total Data Transmission Time (DTT) may be achieved by the following equation:

$$DTT = \frac{Total\ Amount\ of\ Data\ to\ be\ Transmitted}{DSR}$$

3. Position Exchange:

To exchange the position of the elements in the Symbol Table we use Transposition properties & Permutation properties of Matrices along with each other as below:

We assume that the $N\times M$ matrix is stored in row-major order with zero-based indices. This means that the (n,m) element, for $n = 0,\ldots,N-1$ and $m = 0,\ldots,M-1$, is stored at an address $a = Mn + m$ (plus some offset in memory, which we ignore). In the transposed $M\times N$ matrix, the corresponding (m,n) element is stored at the address $a' = Nm + n$, again in row-major order. We define the *transposition permutation* to be the function $a' = P(a)$ such that:

$Nm + n = P(Mn + m)$ for all $(n,m) \in [0, N-1] \times [0, M-1]$

This defines a permutation on the numbers a = 0 to MN-1.

P can be defined as:

$$P(a) = \begin{cases} MN - 1 & if\ a = MN - 1, \\ Na \bmod MN - 1 & otherwise, \end{cases}$$

Where mod is the modulo operation [8].

Algorithm:

Case 1: (If Symbol Table is in the form of square matrix):

```
for n = 0 to N – 2
    for m = n + 1 to N - 1
        swap A(n,m) with A(m,n)
```

Case 2: (If Symbol Table is in the form of non-square matrix):

```
for each length>1 cycle C of the permutation
    pick a starting address s in C
        let D = data at s
    let x = predecessor of s in the cycle
    while x ≠ s
        move data from x to successor of x
        let x = predecessor of x
    move data from D to successor of s[8]
```

4. Data Uncertainty:

Data uncertainty refers to the discrepancy between data and the spatial characteristic represented by the data. It is the quantitative estimation of error present in the data. It is generated by either systematic error, or random error or statistical error. This uncertainty affects the reliability of applications using the data. Careful methodology can reduce uncertainty by correcting for systematic error and minimizing random error. However, uncertainty can never be reduced to zero. In Authentication, correlated uncertainty is generated because the User-Id & Password is dependent to each other.

We will solve data uncertainty problem using cumulative joint probability distribution.

Execution	Given alphanumeric codes	Produced cipher text
1	A	TX^K3$
2	B	*W14S@
3	C	#L+5!H
4	D	%FA^4Q
5	E	HW$S*1
6	F	J9E0#&
7	G	40!E&~
8	H	`PSU5)
9	I	IC8|YM
10	J	X#D*6Q
11	K	1LE%5&
12	L	FC@1K(
13	M	$G6A*@
14	N	V3(1S^
15	O	:E2A9I
16	P	F9$K1&
17	Q	>WRY!0
18	R	92Y~L&
19	S	X8%O3’
20	T	N7=&S$
21	U	J2(A5^
22	V	4R#D0/
23	W	7:J[2A
24	X	MW8!<9
25	Y	L5E*1%
26	Z	“X0O4*
27	0	TX^4S@
28	1	#D0>WR
29	2	A5^$G6
30	3	!E&4R#
31	4	(1S^:E
32	5	5)@1K(
33	6	9$K0#&
34	7	%FA Y!0
35	8	7=&S!T
36	9	@1 X8%O

Case 1. If both the User-Id & Password are discrete random strings, we will use Probability Mass Function (PMF). So,

$$PMF(U = u \,\&\, P = p) = \begin{cases} PMF(P = p \mid U = u).PMF(U = u) \\ PMF(U = u \mid P = p).PMF(P = p) \end{cases}$$

According to the rules of probability, we can write,

$$\sum_{u}\sum_{p} PMF(U = u \,\&\, P = p) = 1$$

Case 2. If both the User-Id & Password are discrete random strings, we will use Probability Density Function (PDF). So,

$$PDF_{U\,P}(u, p) = \begin{cases} f_{P|U}(p \mid u)\, f_{U}(u) \\ f_{U|P}(u \mid p)\, f_{P}(p) \end{cases}$$

Again, according to the rules, we can write,

$$\int_{U}\int_{P} PDF_{U\,P}(u, p)\, du\, dp = 1$$

5. Produce Authenticator:

An authenticator is a way to prove to a computer system that you really are who you are (called Authentication). The Authenticator is an object that knows how to obtain authentication for a network connection. Usually, it will do this by prompting the user for information. The algorithm is as follows:

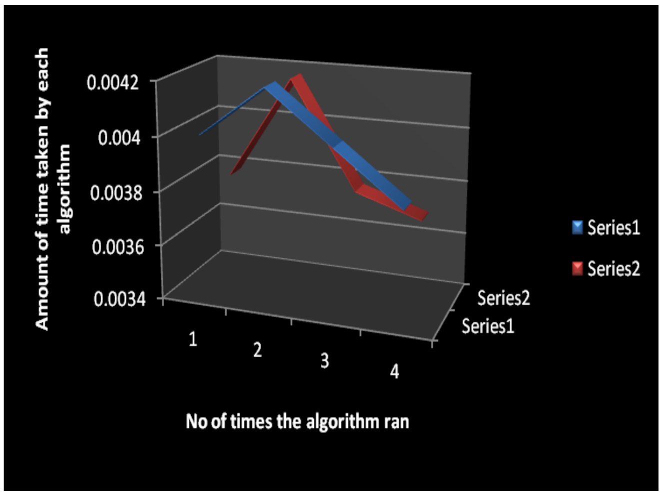
```
Input: User-Id & Password.

Do:     Get the hostname & host address of the
Authentication Server or NULL if not
available.

        Get the port number for the requested
connection.

        Get the prompt string given by the
        requestor.

        Get the Authentication scheme for the
connection.

        Set the protocol for the requested
        connection.

End Do.

Output: Generate the Authenticator for the
        current user who had entered User-Id &
        Password.
```

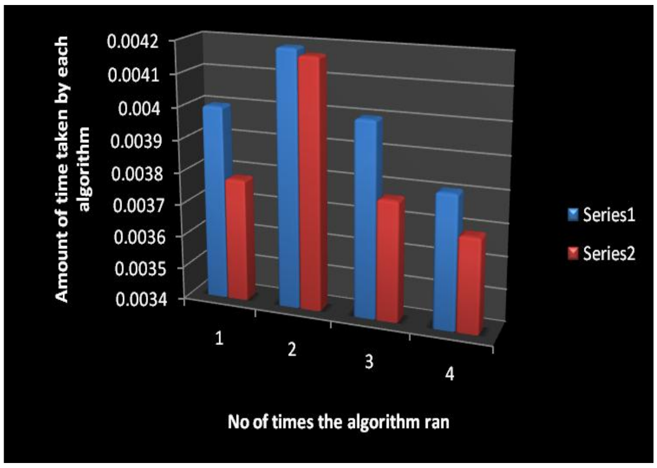
5. EXPERIMENTAL RESULTS

So, For example if we consider a string like "MOUNT007", the equivalent cipher text will be "$G6A*@:E2A9I J2(A5^ V3(1S^ N7=&S$ TX^4S@ TX^4S@%FA Y!0", which is very difficult to break.

Table 2:Time complexity comparison of existing algorithm & my proposed algorithm

Execution	Length of user-id & password in bits	Length of key string in bits	Time required for existing algorithm in ms	Time required for proposed algorithm in ms
1	12,15	22	0.00400	0.00378
2	10,08	29	0.00419	0.00417
3	13,12	22	0.00400	0.00377
4	07,16	20	0.00381	0.00369





6.CONCLUSION

In this paper we have introduced a robust authentication mechanism at the entry points of Grid. After executing and implementing the respective codes for our proposed algorithms, we find that our algorithm is providing a bit better time complexity rather than the existing one. In this paper, we have provided an advanced architecture with algorithm for secure authentication process and experimental results showing better output in context with robustness for data security in Grid

7. REFERENCES

[1] I.Foster, C. Kesselman & S. Tuecke. GRAM: Key concept [Online]. http://www.unix.globus.org/toolkit/docs/3.2/gram/key/index.html

[2] I.Foster, C. Kesselman, Globus: " A metacomputing infrastructure toolkit", International Journal of Supercomputer Application, Vol. 11, No. 2, 1997,

[3] H.M. Lee, C.C. Hsu, M.H. Hsu, "A Dynamic Supervising Model Based on Grid Environment", Knowledge-Based Intelligent information & Engineering System. LNCS, 3682/2005, Springer-Verlag, 2005, pp. 1258-1264.

[4] H.M. Lee, T.Y Lee, C.H. Yang, M.H. Hsu, " An optimal analyzing Resources Model Based on Grid Environment", WSEAS Transactions in Information Science & Apllications, Issue 5, Vol. 3, 2006, pp. 960-966.

[5] H.M. Lee, T.Y Lee, M.H. Hsu, "A Process Schedule Analyzing Model based on Grid Computer", Knowledge-Based Intelligent information & Engineering System, LNAI 4253/2006, 2006, pp. 938-947.

[6] The Open Grid Services Architecture, Version 1.5, http:// www.ogf.org/documents/ GFD.80.pdf

[7] IETF, Real Time Streaming Protocol (RTSP), RFC 2326, April 1998.

[8] Dr. Tony Phillips (May 23, 1999). "ET, phone SETI@home!"..http://science.nasa.gov/newhome/headlines/ast23may99_1.htm. Retrieved 2006-10-06.

[9] M. Lorch, D. Adams, D. Kafura, M. Koneni, A. Rathi, and S. Shah. The prima system for privilege management, authorization and enforcement in grid environments. In Proceedings of the 4th Int. Workshop on Grid Computing - Grid 2003, Phoenix, AZ, USA, Nov. 2003

[10] R. Alfieri et al. (EDG Security Co-ordination Group),"Managing Dynamic User Communities in a Grid of Autonomous Resources", Proceedings of Computing in High Energy and Nuclear Physics (2003).