I.J. Computer Network and Information Security, **2014, 9, 1-11**
Published Online August 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2014.09.01

SysProp: A Web-based Data Backup, Synchronization and System Administration

Salekul Islam and Mohammad Amanul Islam
United International University, Dhaka, Bangladesh
Email: salekul@cse.uiu.ac.bd, aman.cse.bd@gmail.com

***Abstract*—**From the inception of computer based computing, preventing data loss or data corruption is considered as one of the difficult challenges. In early days, data reliability had been increased by replicating data in multiple disks, which were attached with the same system and later located inside the same network. Later, to avoid potential risk of single point of failure, the replicated data storage has been separated from the network from which the data has been originated. Thus, following the concept of peer-to-peer (P2P) networking, P2P storage system has been designed, where data has been replicated inside multiple remote peers' redundant storages. With the advent of Cloud computing, a similar but more reliable Cloud-based storage system has been developed. Note that Cloud storages are expensive for small and medium enterprises. Moreover, users are often reluctant to store their sensitive data inside a third-party's network that they do now own or control. In this paper, we design, develop and deploy a storage system that we named *SysProp*. Two widely used tools—Web applications and UNIX daemon—have been incorporated in the development process of *SysProp*. Our goal is to congregate benefits of different storage systems (e.g., networked, P2P and Cloud storages) in a single application. *SysProp* provides a remotely accessible, Web-based interface, where users have full control over their data and data is being transferred in encrypted form. Moreover, for data backup, a powerful UNIX tool, rsync has been used that synchronize data by transferring only the updated portion. Finally, *SysProp* is a successful demonstration of the concept that UNIX daemons can be remotely executed and controlled over the Web. Hence, this concept might be exploited to build many system administrative applications.

***Index Terms*—**Storage system, data backup, synchronization, Web service, remote system admin.

I. INTRODUCTION

Devices like laptops, smart phones, tabs or any other mobile devices are now an associated part of our lives at work, at home, or even while traveling. Tremendous growth of the intranet and the Internet based services bring a revolution in our daily lives. A wide range of service requirements have been created that are based on the sophisticated online applications. Following this pattern, an increasing need has been observed to make reliable, regularly used Web-based services including system administration (e.g., Webmin [1]), data backup, remote backup synchronization and storage management (e.g., Dropbox [2]), etc. While developing these services, integration between open source system tools and Web-based workgroup tools are becoming popular in the developer communities. Since the Web technology continues to penetrate in sharing contents and connecting remote devices, different types of service provisioning, remote access computing are becoming increasingly prevalent. While each of these applications performs different tasks, they all have some common properties: discovering participants' nodes, accessing files or directory inside the remote nodes, etc.

To facilitate further discussion we would like to explain two commonly used terms—backup and synchronization—that are being used frequently in explaining data storage systems. In general, data backup means keeping a copy of a file or directory on a separate storage device so that it can be restored if the original copy is lost or damaged. It is quite understandable that if one simply copies the whole file or directory, the backup time will be directly proportional to the size of data. Hence, for a very large data, the backup process will be cumbersome and time consuming due to the bandwidth limitations of typical connections. To speed up the backup process and efficiently utilize the bandwidth, synchronization method sends only those portions of a file or data which have been changed since the last backup to the target system. The smarter the comparison method (that finds out the updated portions) is the lesser the backup time is needed [3].

From the inception of computer systems, plenty of resources (both hardware and human effort) had to engage in keeping backup copy of sensitive data. In early days, data had to store manually in extra disks. Later, automated networked storage systems such as RAID1 (Redundant Arrays of Inexpensive Disks) [4] saved plenty of man-hours. However, such system is not free from accidental failure of the network, which makes it vulnerable to the single point of failure and sensitive data might be lost or unavailable. An alternative solution is to use a peer-to-peer (P2P) storage system, where a number of remote hosts (peers) contribute the unused portions of their storages. Thus, a large-scale storage is built from many small-scale, unreliable, low-availability distributed hosts [5]. The reliability and availability of a P2P storage system depends on the availability of participating peers, and ensuring that peers storing others' backup data will not deny the restore service in times of need. To overcome the limitations of the P2P storage systems, Cloud-based storages have been designed. In such storage systems, data is stored in virtualized pools of storages, which are generally hosted by third parties, i.e., the Cloud storage providers [6]. However, despite the rapid growth of Cloud based storages, there are significant, persistent concerns about users' control and security over their data that they store inside the Cloud storages [7]. The main objective of this study is to research the existing data backup systems and accumulate the benefits of these systems in a single model.

In this study, we develop a Web-based storage system that we named *SysProp*, which provides data backup and synchronization over remote nodes. *SysProp* has been developed by a successful integration between two commonly used but unrelated technologies: Linux/UNIX based daemon and Web services. Note that a daemon [8] is a computer program that runs as a background process, rather than being under the direct control of an interactive user. The daemon process we have developed can be controlled through Web services from a remote place. The daemon scripts, basically execute over multiple nodes (machine). These nodes can exist on a private or public Cloud as physical or virtual platform. *SysProp* implements a number of administrative services such as database administration, data compression and system status monitoring, and backup management services such as remote data backup, data synchronization. Integration between Web application tools and operating system daemon requires strong level of security during communication over LAN and the Internet. Considering this challenge, SSL based Web communication, use of digital certificates, key based password, SSH and other security tools have been discussed and practiced while developing *SysProp*.

The rest of the paper is organized as follows: section II summarizes the existing work and the present trend of storage management, section III presents the requirements and a high-level architecture of the developed services, section IV explains the *SysProp* system overview including the implemented services, application software architecture, the application daemon, security methods used etc. Section V describes the development process including the required programs and packages, system components at different ends, security key configuration and an example service. The limitations and future extensions of the system have been presented in the next section, and finally section VII concludes the paper.

II. Literature review

Researchers are working on various networked data storage services for many years. Remote data backup and synchronization is also another area of interest for many researchers. Recently a new trend has been emerged in Web-based system administration tools. Hence in these areas, a wide range of research work had been carried out and many commercial tools are available as well. In the following we mention some significant developments in related topics.

Networked storage is always preferred due to its simplified management nature by centralizing storage under a consolidated manager interface. Storage Area Networks (SAN) and Network Attached Storage (NAS) are two established networked storage techniques [9]. For example, RAID1 (Redundant Arrays of Inexpensive Disks) [4] is a widely deployed mirroring-based NAS technique, which stores the same data in more than one disk drive. Mirrors are usually used to guard against data loss due to drive failure. Note that although NAS and SAN have different storage abstractions (e.g., NAS offers file system functionality and SAN provides a simple, un-typed, fixed-size memory blocks) the difference is becoming blur and may soon not even be a recognizable difference.

However, these networked storage systems suffer from some limitations, including single point of failure (in case the network is down the stored data is not accessible), not cost-effective, needs extra maintenance attention, etc. Therefore, a peer-to-peer (P2P) on-line storage and backup system could be designed [10] that exploits the unused disk spaces and other resources of peers. Often, instead of designing a pure P2P model, a hybrid approach that incorporates residential gateways [11] or a Cloud storage provider [11] could be developed. A hybrid approach offloads the peers and also increases the reliability as well. A cooperative, P2P backup system, Friendstore [12] has been proposed that differs from the pure P2P proposals in one key aspect: a node can choose a subset of peer nodes to store its backup data. Thus, each user (node) can stores it backup data to trusted nodes only by selecting her friends or colleagues.

P2P storage systems suffers due to its lack of reliability, which could be much improved by Cloud computing [6] based storage systems. The illusion of infinite and elastic storage, the pay for use business model, reliability, separation of storage system from resources make storage Cloud an attractive model to large enterprise customers, Small and Medium Enterprises (SME) and even personal users. However, lack of control over data [7], security [13][14] of users' data and high price [15] are the major challenges that the storage Cloud is facing. In spite of these limitations, in the recent years a number of commercial storage Clouds have been successfully deployed—Dropbox [16][2], Google Drive [17] and Amazon's Scalable Storage Service (S3) [18]—are a few to mention.

For UNIX-like operating systems, a very popular utility software and network protocol called `rsync` [19], is being used for synchronizing files and directories from one location to another. `rsync` minimizes data transfer by using the '`rsync` algorithm' which provides a very fast method for bringing remote files into sync. Although `rsync` is a powerful tool for data backup and synchronization its use is confined to UNIX-like operating systems and a minimum user proficiency in UNIX is needed.

III. System Requirements

In the previous section, we broadly classified the data storage and synchronization systems into three categories: networked storage systems, P2P models and Cloud-based storages. All these methods have their own merits and limitations as well. While determining the requirements of the *SysProp* system, we gather the merits of the previous three models. In the following, we summarize the requirements that the *SysProp* system should focus in:

1. Separation of working and data backup networks: To avoid any single point of failure the backup/storage network should be separated from the user's own network. This is a drawback of SAN/NAS model that has been solved in the Cloud-based storages.
2. Remote operations: The designed system should be accessible from remote locations so that users may backup their data while traveling or working remotely. All the commercial data backup solutions (e.g.,
3. Dropbox, Google Drive and Amazon S3) support such remote operations.
4. Full control over data: Cloud-based storage systems fail to provide full control to the users over their data that they store inside the Cloud. This is considered as a major drawback of the Cloud and the P2P storage systems. Hence, the *SysProp* system should not store data inside a third-party's network over which the owner of the data has no control or access.
5. Friendly user interface: To make the designed storage system popular a friendly user interface (e.g., a Web-based user interface) should be built that is even accessible from a remote location.
6. Secured and efficient data transfer: Data must be transmitted in encrypted form. Moreover, to make the backup system faster, instead of rewriting the whole data only the updated portion should be synchronized (this method is followed by `rsync` during synchronization).

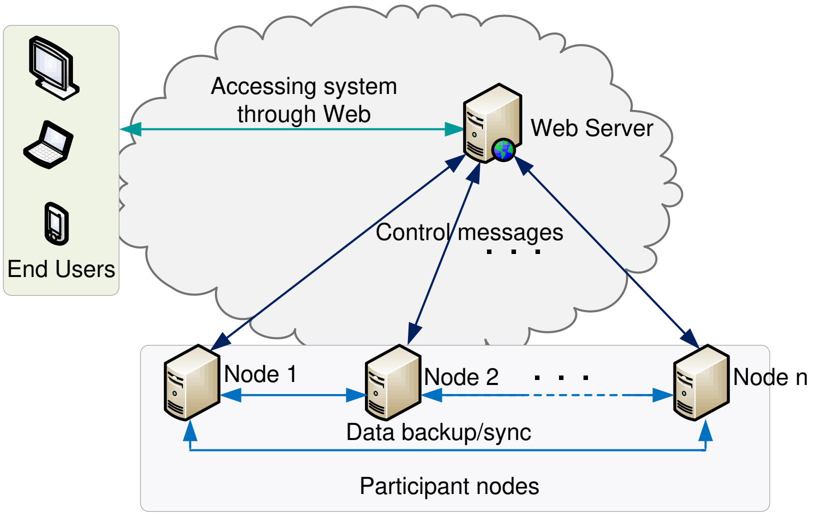


Fig 1: High-level architecture of *SysProp* system

Before explaining the implementation details of the *SysProp* system, in Fig. 1 we present the high-level system architecture. The end users can access the system through a Web browser and any end device with Internet connectivity could be used. The user's input should be converted to control messages by the Web server. In this system, any host machine could be a participating node, which might be the source or the destination node. Data from a source node should be backed up through synchronization to the destination node. Note that any host with a global IP address, might be used as a source/destination node. Therefore, the same node might be the source node for a backup operation, while the destination node in another operation. Moreover, the participant nodes might be in the same network or in different networks. However, the owner of data will have full control over data while both nodes are maintained and controlled by the same network admin. Finally, data should be transferred in encrypted format and also be stored in encrypted format at the destination node.

IV. SysProp System Overview

The *SysProp* system is composed of a central Web server and a number of distributed participants nodes. Fig. 2 shows the interactions between the Web server and the participant nodes. A node is a networked entity that implements one or more application services. A node might be a PC, a server or even a supercomputer. Since *SysProp* provides the Web-based access of different shell scripts/daemon running inside nodes, Linux/UNIX machines have been selected as participant nodes. Note that each node operates independently and a node can be uniquely identified by its network address or its corresponding host/domain name (by considering specific filter rules behind the firewall). A participant node uses its network address or domain name to connect with the Web server. To communicate with the Web server, each participant node should have HTTP components. The

Web server controls the shell scripts/daemon running inside a node through the HTTP component. The Web server connects each node's HTTP component using SSI (Server Side Includes) and URL (Uniform Resource Locator) redirection.

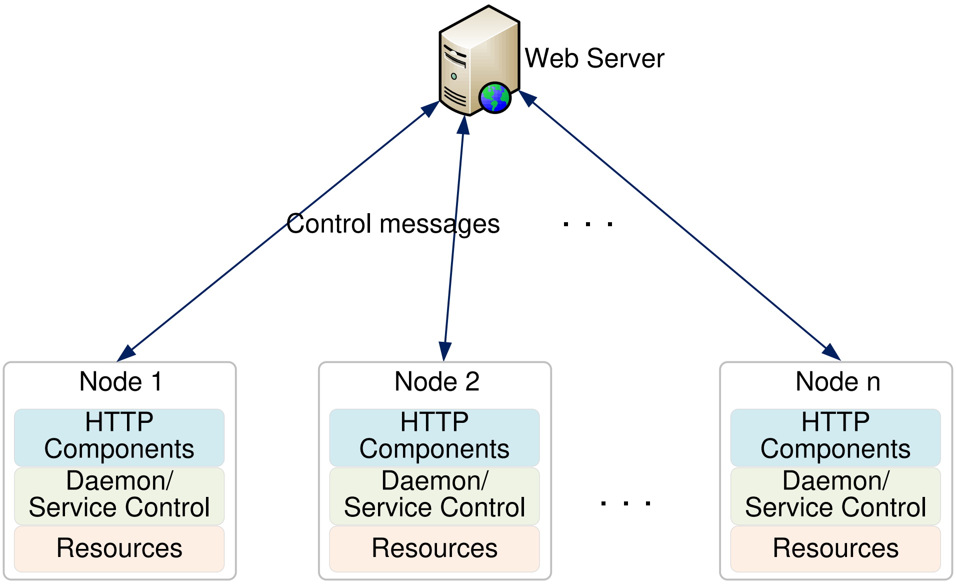


Fig 2: Interaction between participant nodes and the Web server

A. *Implemented services*

A node is a physical or virtual OS platform that implements shell based application as per service requirements. *SysProp* might be considered as a communication platform to administering different services running on remote participant nodes. In our prototype implementation, we have developed a few services including remote data backup and synchronization. However, there exists plenty of scope in developing a wide range of services such as file transfer, database management, network management services, etc. In the *SysProp* system the implemented services can be divided into two categories:

- Data Archive Operation: *SysProp* has implemented a few commonly used data backup operations on Linux/UNIX based physical or virtual platforms, such as data compress, database dump with specific selection process.
- Data Synchronization Operation: It is possible to synchronize a data directory with permission and ownership for different types of services which are running commonly on connected multiple nodes over the network as remote backup or remote sync operation.

B. *Application sofware architecture*

The *SysProp*'s software architecture is composed of five layers. Note that a layer may provide different functions to different entities and not all five layers are present in both entities (i.e., in the Web server and participant nodes). The different layers present in application software architecture in the Web server and participant nodes are shown in Fig. 3 and Fig. 4 respectively. Similar layering of Web applications could be found in [20].

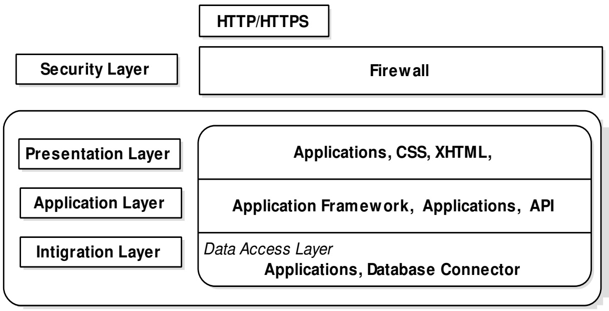


Fig 3: Web server's application architecture

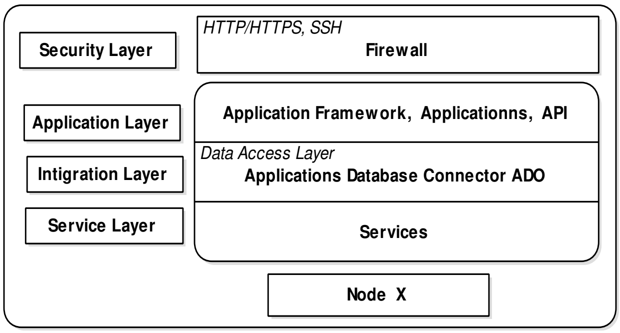


Fig. 4: Participant nodes' application architecture

In the following we briefly discuss the five layers:

Security layer: This layer describes that Web access and system access encapsulate the minimal and essential primitives that are common to network and Web-based communication. It includes building blocks to enable key mechanisms for Web application and authorized system privileges, including communication transports (including firewall and NAT traversal), the secure access to Web server and connected nodes, encrypted data transfer between specified nodes and associated security primitives.

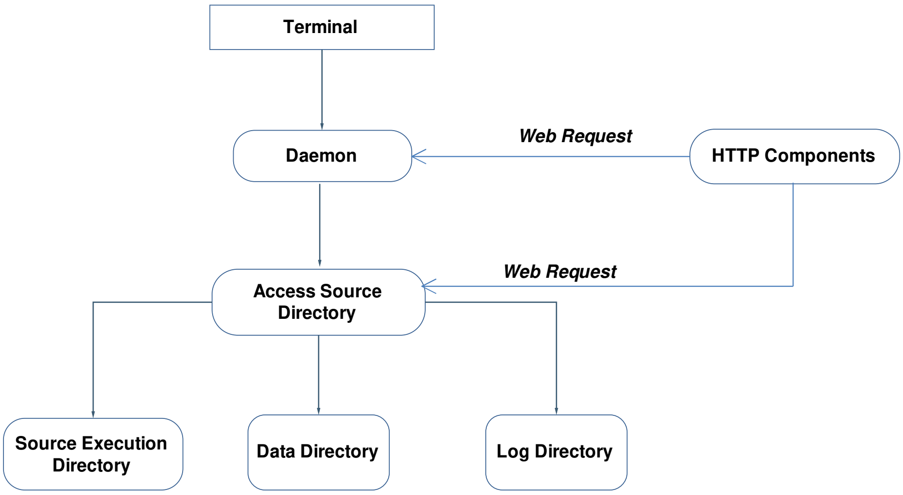


Fig 5: The application architecture of the daemon

Presentation layer: The presentation layer represents the user interface on Web form to do the tasks easier at different levels of client devices for the user's system administration, system access, make operation from anywhere. Present user's standard input and output on activities which are accessed through other layers.

Application layer: The applications layer includes implementation of integrated applications, such as response to remote application execution and sending request as well. This layer provides access to integration layer's data for the execution of self or remote applications through URL, hyper link, HTTP GET or POST methods etc.

Integration layer: The integration layer is sometimes included in the integrated applications and services. This layer intercepts input parameters from user for service layer operation including resulting output validation through application and presentation layers.

Service layer: The service layer represents the actual activities inside the participant nodes based on the request from remote users through the web servers. Identifying the tasks, performing on those, submitting the status accordingly are the big roles of this layer.

C. Application Daemon

Maintaining security while performing systematic operational tasks on the system is extremely important and daemon is an approach for the tasks to provide access control to the system services. *SysProp* implements a shell based application which depends on the daemon for performing the core tasks. Fig. 5 illustrates the daemon's application architecture. Basically the daemon is responsible for integrating the user access and the source execution. Note that on our application platform, the end user can perform her tasks directly on the terminal or remotely through Web access. Because of the terminal access facility, end user through the daemon can directly access the source directory, schedule their tasks and perform other related operations. However, when the end user accesses through Web, the daemon has to process Web requests coming from the HTTP components and then communicate with the source execution directory. This HTTP component works as a Web API written using PHP scripts.

D. Securing control messages

Security of *SysProp*'s control messages lies on primarily two secured communications: secured communication between two participant nodes and secured communication between the Web server and the participant nodes. Fig. 6 shows the techniques we use in securing these two communications.

1) Secured communication between participant nodes

To secure communication between two participant nodes, SSH with PKI (Public Key Infrastructure) has been deployed between the links of two participant nodes. PKI system uses two keys: a public key that is known by everyone and a private key that is kept secret and known by the owner of the key only. Public key based authentication, one of the most secure methods, is considered as more secured than password based authentication since it is less vulnerable to brute-force or dictionary attacks. In our prototype, we use OpenSSH [21], an open source implementation of the SSH connectivity tools. Later we have explained the process of generating a set of keys, and using them for logging into participant nodes during remote communication over the network using SSH service. Note that SSH is being used to secure the session setup and initial communications between two participant nodes and not to secure data transmission. Secured data transmission is discussed in section IV.E.

2) Secured communication with the Web server

The communication links between the remote user's end device and the Web server, and the Web server and a participant node must be protected. To protect these links

SSL with PKI has been implemented in *SysProp*. Later we have explained the process of generating a set of keys, and using them for implementing Web SSL communication. Due to the use of public key based SSH and SSL, the following security requirements can be attained:

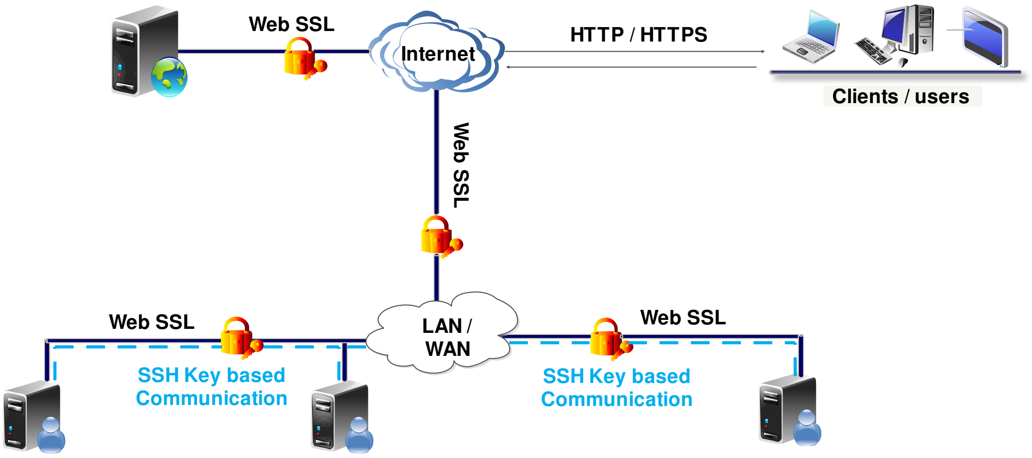


Fig 6: Using SSH and SSL to secure control messages

- Confidentiality: guarantees that the contents of a message are not disclosed to unauthorized individuals.
- Authentication: guarantees that the sender is who he or she claims to be.
- Authorization: guarantees that the sender is authorized to send a message.
- Data integrity: guarantees that the message was not modified accidentally or deliberately in transit.

E. *Securing data transfer*

During the data transmission over the Internet, *SysProp* application always transfers data files, file content and folders as encrypted format. To process this encryption mechanism, *SysProp* uses cipher blowfish algorithm using an open source tools called `EncFS` [22]. Fig. 7 shows how `EncFS` is being used in our system. It creates a user specified virtual encrypted file system, which encrypts user data in a directory inside the source node. Later, at the destination node, it creates a similar encrypted directory, which is decrypted and mounted to another directory. Thus, plain data is visible at the destination node. This process must supply a password to encrypt both filenames and file contents and establishes a permanent mount point for a directory during the system boot up. The same password must be supplied to the source and the destination nodes simultaneously. There will be some options to the user allowing some control over the usage of the algorithm including key size, block size settings etc. Due to the use of `EncFS` based encrypted data transfer, data confidentiality, and data integrity could be achieved.

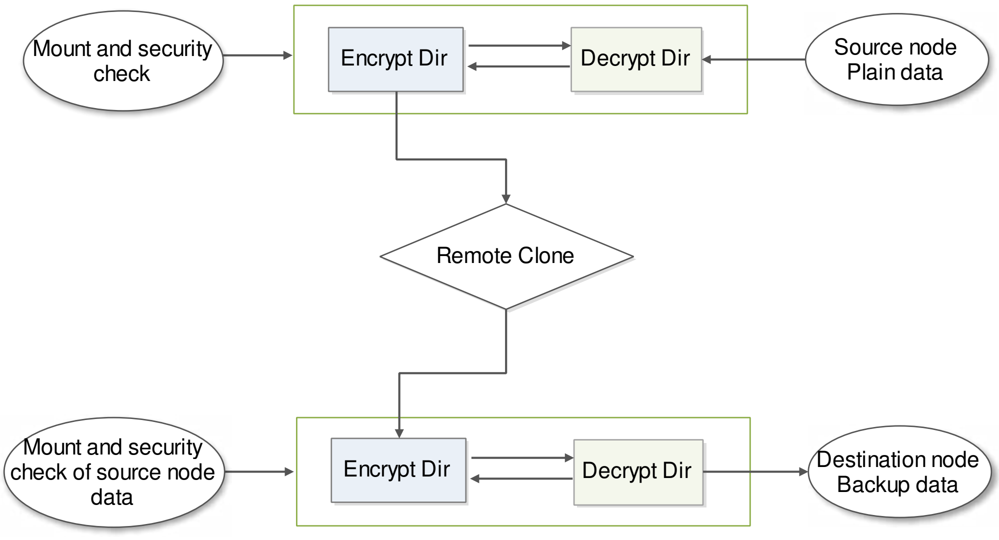
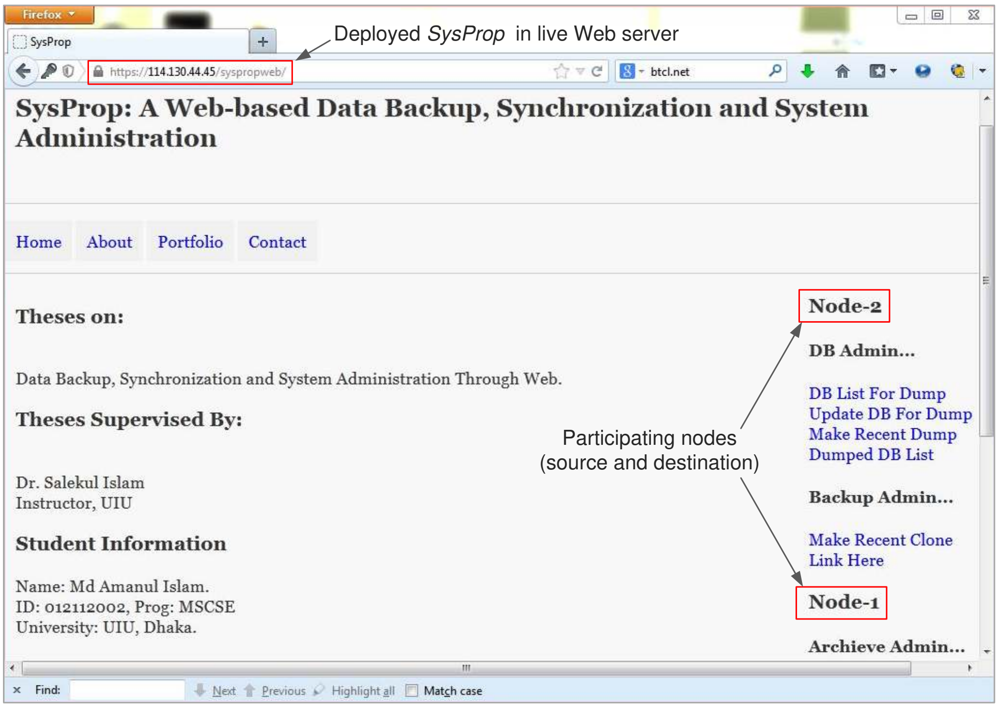


Fig 7: Using EncFS for encrypted data transfer


Firefox
SysProp
https://114.130.44.45/syspropweb/
btcl.net
Deployed SysProp in live Web server
Home
About
Portfolio
Contact
Participating nodes (source and destination)
Find:
Next
Previous
Highlight all
Match case

SysProp: A Web-based Data Backup, Synchronization and System Administration

Theses on:

Data Backup, Synchronization and System Administration Through Web.

Theses Supervised By:

Dr. Salekul Islam
Instructor, UIU

Student Information

Name: Md Amanul Islam.
ID: 012112002, Prog: MSCSE
University: UIU, Dhaka.

Node-2

DB Admin...

DB List For Dump
Update DB For Dump
Make Recent Dump
Dumped DB List

Backup Admin...

Make Recent Clone
Link Here

Node-1

Archieve Admin...

Fig 8: A screenshot of the *SysProp* prototype that we have deployed

V. Developing SysProp

Developing a distributed system such as *SysProp* needs a series of tasks: developing different components, configuring a number of machines, integrating the whole system, deploying and testing the system. We have developed a prototype of *SysProp* Web application, and then deployed the prototype in a live server. The screenshot of this deployed application is shown in Fig. 8. In this section, we discuss different steps of *SysProp* system development and deployment.

A. Required programs and packages

Our implementation requires the following open source tools/programs to be installed in the Web server and the participant nodes:

- PHP Standard Edition 5.1 or later release [23],
- Apache server version: Apache 2.0 (UNIX) [24] or later releases, which are available for the Solaris Operating Environment, Microsoft Windows, Linux, Mac OS X and other operating systems to build and deploy Web server applications,
- iptables for firewall setup process.

The following open source tools/programs to be installed in the participant nodes:

- fuse, fuse-encfs, fuse-lib to implement encryption and decryption process,
- bash 4.0 or later releases in Linux/UNIX based operating systems for building shell script.

In addition to these programs, we use a number of commonly used packages including httpd, mod_ssl, openssl, php, bash, `rsync`, openssh-server, vsftpd, etc.

B. SysProp components at the Web server

For accessing the Apache server, access security should be granted. Since our application works as a single resource on the Apache server, we are using .htaccess file to authenticate the access to the server and the application resources as well. To accomplish the authentication procedure, the following configuration settings are stored in the httpd.conf file:

```
 # Authentication settings
AccessFileName .htaccess
<Directory "/var/www/html/SysPropweb/">
AuthType Basic
AuthName "To Login With Valid
Authetication-Theses Team"
AuthUserFile
/var/www/html/SysPropweb/.htpasswd
Require valid-user
</Directory>
```

In the AuthUserFile, we add a user admin with a specific password using the htpasswd command. For example,

```
#htpasswd -c
//var/www/html/SysPropweb/.htpasswd
admin
  ask new password with
  retype request
```

This authentication procedure works as the application's security feature as well. Note that there exist plenty of scopes to enhance the application's access security by using other security tools, but since our focus is if the application is working, we use Apache's built in authentication module.

The Web server on receiving any user request checking a service's status or performing any task forwards the request to the related participant's Web API (i.e., the HTTP component shown in Fig. 5). The Web server application redirects the user's request to the participant node's Web API using the following simple PHP code:

```
<?php
//Call to execute participant nodes web
API
$response =
file_get_contents('http://participant_nod
e_ip/nodes_web_api/file.php');
echo "<pre>$response</pre>";
?>
```

User's any request to test input to modify any data has been resolved through ftp access. Here, these functionalities can be solved by any other standard programming functionalities such as database connectivity for users input to the database. Following is the PHP scripts running inside the Web server that accesses to the participant's node:

```
<?php
if(isset($_POST['submit'])) {
$ftpstream = @ftp_connect('
participant_node_ip ');
$content = $_POST['contents'];
//Login to the FTP server
$login = @ftp_login($ftpstream, 'user',
'password');
if($login) {
//Connected to FTP server, create a
temporary file
$temp = tmpfile();
fwrite($temp,$content);
fseek($temp,0);
//Upload the temporary file to server
@ftp_fput($ftpstream, 'file.ext', $temp,
FTP_ASCII);
echo "Writing done"; }
else {
echo "<br> Cannot login";
      }
}?>
// Text Area Formatting
<h1>Alter TEXT File</h1>
<form name="edit" method="post">
<textarea name="contents" cols="20"
rows="7">
<?php
//open a file to read previous input
$file =
"ftp://file:password@participant_node_ip/
file.ext";
$open = fopen($file, "r");
$size = filesize($file);
$count = fread($open, $size);
echo($count);
?>
```

C. SysProp components in the participant node

The main part of the *SysProp* system, executed inside the participant node, is divided into two loosely coupled tasks:

1. The service operational scripts running from a specific directory with a centralized daemon control
2. A Web API which integrates the functionalities between the executed scripts and the web server.

Note that most scripts are written with bash shell on Linux platform and the Web APIs are written using PHP language.

1) Directory organization

For better organization of the development process we have to create different directories inside the participant node machine. Table 1 shows the organization of the directories.

Table 1: Directory organization inside the participant node

Name of the directory	Files stored
`PrOp`	The main working directory that stores all the application scripts. The other subdirectories are created inside `PrOp`.
`bin`	Includes all the sources scripts related to different types of system administrative tasks including data backup, database dump, remote data backup or synch operation.
`log`	Stores the events of all types of execution errors to be used in troubleshooting.
`data`	Keeps the data which are the inputs from the users through the Web. Since the end user's web request is generating from the username `apache`, the directory ownership and necessary permissions with read/write options should be set for the user `apache`.

2) Daemon script configuration and settings

In our application, we have used case statements in a script named *SysProp* to group some commands related to system administrative tasks (i.e., data backup and data archive). Also the scripts have worked as daemon which can be used to control the specified tasks on different run level. This script is always being executed from the directory, /etc/init.d/*SysProp*. As we have mentioned earlier, the path of different variables for application management have been assigned following the subdirectory structure. Following is the description of the bash script with necessary comments:

```
#! /bin/bash
#
# Define PATH Details
BASEDIR=/usr/local/PrOp
# Source scripts in the PATH
BINDIR=$BASEDIR/bin
# Optional configuration scripts in the
PATH
CONFDIR=$BASEDIR/conf
# Log details in the PATH regarding
source execution
LOGDIR=$BASEDIR/logs
# PATH to store backup data.
DEST=/var/data/
# Making log data in context of SysProp
execution
exec &> $LOGDIR/SysProp.log
# Start 'case' statement
case "$1" in
# For Archive Operation as System Admin
Task
        tarmake)
        echo "Making Archive Data:"
        $BINDIR/MakeArchive.sh
        echo "Note: All The Data Have
Destined to $DEST"
        ;;
# For DB Dump Operation as System Admin
Task
        make)
        echo "Making Database Dump:"
        $BINDIR/MakeDBdump.sh
        echo "Note: All The Data Have
Destined to $DEST"
        ;;
# For Remote Backup or data Sync
Operation
        clone)
        echo "Making Remote Backup:"
        #/bin/sh $BINDIR/MakeClone.sh
        $BINDIR/MakeClone.sh
        ;;
  *)
# Make Individual Operation Using Daemon
"SysProp"
        echo "Usage: $0
{make|tarmake|clone}"
        exit 1
esac

exit 0
```

D. Security key configuration

1) Key generation and distribution for SSH communications

SSH-based communications have been deployed to secure the links between two participant nodes. To make this communications secured but transparent to the end users password less SSH communications during data transfer have been implemented. Therefore, we need to do following tasks:

- Generate a pair of private and public keys for a new system user
- Share this newly generated public key among the existing participant nodes

2) Key generation and distribution for SSL communications

SSL communications that use key-based secure communications and digital certificate-based authorization have been implemented to secure user's Web request over the Internet or over the LAN. To implement secured SSL communications the following tasks need to be completed:

- Generate the necessary keys, the Certificate (CRT) and the Certificate Signing Request (CSR)
- Install and distribute the keys, CRT and CSR

E. Example of implementing system operation: Remote backup using synchronization

SysProp implements a number of system operations. In this section we discuss in detail how remote data backup using synchronization has been accomplished.
Here we present how we write the scripts to perform data backup using synchronization. It has two components: the Web API part (deployed inside the participant node) and the shell script (which is executed inside the participant node and triggered by the Web request coming from the Web API). In the following we present the PHP scripts that acts as the Web API:

```
<?php
# execute daemon script with backup
action.
$output = shell_exec("/etc/init.d/SysProp
clone");
echo "<pre>$output</pre>";
?>
```

In the *SysProp* scripts, we express the remote backup operation using the word clone in the case statement. According to the defined action on *SysProp* scripts to make a backup operation the shell scripts, MakeClone.sh will be executed. Here using the `rsync` tool, Mysqldump file from the source node to /Sys_Backup directory inside the remote destination node has been synchronized. The description of MakeClone.sh scripts has been given in the following:

```
#!/bin/bash
# Defined directory for application
platform
DATADIR="/usr/local/PrOp/data"
LOGDIR="/usr/local/PrOp/logs"
echo $DATE
# Defined directory for logs
exec &> $LOGDIR/MakeClone.log
# Defined directory in a variable for
sources and
# data contain directory
CPDIR0="/var/data/archive"
CPDIR1="/var/data/mysqldump"
CLONEDIR="/Sys_Backup/en_118.179.212.141/
"
# Command source in a variable
DATE="$(which date)"
CHOWN="$(which chown)"
RSYNC="$(which rsync)"
# Define the host for backup operation
a="118.179.212.141"
b="114.130.44.42"
#Transfer local system to remote system
sudo $RSYNC -azv $CPDIR0
/Sys_Backup/118.179.212.141_de/data/ ||
/bin/true
sudo /usr/bin/ssh $b "mkdir -p
${CLONEDIR}" || /bin/true
sudo      $RSYNC      -azv      $CLONEDIR
"$b:${CLONEDIR}" || /bin/true
```

VI. Discussion

The *SysProp* system has been developed to meet the requirements stated in section III. Hence, the benefits of *SysProp* are quite understandable. However, the implemented *SysProp* system has some limitations. In the following we discuss the limitations of the *SysProp* system, how those limitations could be overcome and also how this system could be extended in future to provide more services:

Selecting a participating node: The present implementation of *SysProp* statically provides the addresses of the source and destination participant nodes. Although with little extension it is possible to provide two separate lists of source and destination nodes. Note that a participant node (source/destination) needs a piece of application component to be copied in specific directories. In the future extension, this application code might be dynamically available and also downloadable from the Web server.

Storing in more than one node: To increase the availability and reliability, data might be backed up in more than one destination node. The user may select more than one destination node during data synchronization.

Increasing security level: In the present prototype, all the security keys have been assigned manually beforehand. A better security model might be deployed where all the participating nodes will be authenticated using digital certificates. Moreover, at the starting of a data transfer session, symmetric keys will be established dynamically between the source and destination nodes, which will be only valid for that session.

Optimizing data synchronization: Since we use `EncFS` for encryption and `rsync` for data transmission, optimization in data transfer depends on the underlying algorithms of these two. However, the implementation is modular enough to deploy some other encryption and/or synchronization techniques if needed.

Implementing more administrative services: Although we have implemented data archive and backup services only, the use of *SysProp* system is not limited to these services. Any daemon-based service that could be controlled over the Web is a potential service to be integrated in our design.

VII. Conclusion

The contribution of the study lies in successfully integrating benefits from different storage management methods by incorporating various open-source tools and applications. This model is simple with Web-based user-friendly interface. At the same time the solution is cheaper than the existing commercial solutions. Moreover, the end user retains her full control over her data. The prototype demonstrates a concept of accessing and controlling daemons from remote places through Web services. Hence, this model might be exploited to access any daemon-based services, including data restoration, file transfer, DNS, etc. In future, we would like to carry out a performance study by deploying other encryption techniques and synchronization schemes.

References

[1] Webmin. http://www.webmin.com/. [Online]. http://www.webmin.com/.

[2] Dropbox. [Online]. https://www.dropbox.com/.

[3] David Cane, David Hirschman, Philip Speare, Lev Vaitzblit, and Howard Marson, "File Comparison for Data Backup," U.S. Patent 6,101,507, August 8, 2000.

[4] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," ACM Computing Surveys, vol. 26, no. 2, pp. 145-185 , 1994.

[5] Charles Blake and Rodrigo Rodrigues, "High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two," in Proceedings of 9th Workshop on Hot Topics in Operating Systems, 2003.

[6] Michael Armbrust and et al., "Above the Clouds: A Berkeley View of Cloud," University of California at Berkeley, UCB/EECS-2009-28, 2009.

[7] Richard Chow and et al., "Controlling Data in the Cloud:Outsourcing Computation without Outsourcing Control," in ACM Workshop on Cloud Computing Security (CCSW), 2009, pp. 85-90.

[8] Devin Watson, "Linux Daemon Writing HOWTO," http://www.netzmafia.de/skripten/unix/linux-daemon-howto.html, 2004.

[9] Garth A. Gibson and Rodney Van Meter, "Network attached storage architecture," Communications of the ACM, vol. 43, no. 11, pp. 37-45, 2000.

[10] Laszlo Toka, Matteo Dell'Amico, and Pietro Michiardi, "On Scheduling and Redundancy for P2P Backup," Arxiv preprint arXiv:1009.1344, 2010.
[11] Serge Defrance and et al., "Efficient peer-to-peer backup services through buffering at the edge," in IEEE International Conference on Peer-to-Peer Computing (P2P), 2011, pp. 142-151.
[12] Dinh Nguyen Tran, Frank Chiang, and Jinyang Li, "Friendstore: cooperative online backup using trusted nodes," in 1st ACM workshop on Social network systems, 2008, pp. 37-42.
[13] Lori M. Kaufman, "Data Security in the World of Cloud Computing," IEEE Security & Privacy, vol. 7, no. 4, pp. 61-64, 2009.
[14] Xi Chen,Yong Li,"Efficient Proxy Re-encryption with Private Keyword Searching in Untrusted Storage", International Journal of Computer Network and Information Security, vol.3, no.2, pp.50-56, 2011.
[15] J. Broberg and Z. Tari, "MetaCDN: Harnessing storage clouds for high performance content delivery," in Sixth International Conference on Service-Oriented Computing, Sydney, 2008.
[16] Idilio Drago and et al., "Inside Dropbox: Understanding Personal Cloud Storage Services," in ACM conference on Internet measurement, 2012, pp. 481-494.
[17] Google Drive. [Online]. https://drive.google.com.
[18] Amazon S3, Cloud Computing Storage for Files, Images, Videos. [Online]. http://aws.amazon.com/s3/.
[19] `Rsync`. [Online]. http://rsync.samba.org/.
[20] Microsoft Patterns & Practices Team, "Chapter 21: Designing Web Applications," in Microsoft Application Architecture Guide (Patterns & Practices).: Microsoft Press, 2009.
[21] OpenSSH. [Online]. http://www.openssh.com/.
[22] EncFS Encrypted Filesystem. [Online]. http://www.arg0.net/encfs.
[23] PHP: Hypertext Preprocessor. [Online]. http://php.net/.
[24] The Apache HTTP Server Project. [Online]. http://httpd.apache.org/.

AUTHORS' PROFILES

Salekul Islam is an Assistant Professor at United International University (UIU), Bangladesh. Before joining UIU, he worked as an Assistant Professor at North South University, Bangladesh. Earlier, he worked as an FQRNT postdoctoral fellow at INRS, a constituent of the Université du Québec. He has a Bachelors degree from Bangladesh University of Engineering and Technology in Computer Science and Engineering, a Masters degree and a PhD both in Computer Science from Concordia University, Canada. His present research interests are in Cloud computing, virtualisation, future Internet. His also carried out research in IP multicast especially in the area of security issues of multicasting.

Mohammad Amanul Islam completed Master's in Computer Science and Engineering from United International University (UIU), Bangladesh in 2014. He is working in a software company where he is engaged in developing mobile banking software.

How to cite this paper: Salekul Islam, Mohammad Amanul Islam,"SysProp: A Web-based Data Backup, Synchronization and System Administration", IJCNIS, vol.6, no.9, pp.1-11, 2014. DOI: 10.5815/ijcnis.2014.09.01