

Paper 359-2011

**Tunneling Your Way to a More Secure SAS/CONNECT®**

Greg McLean, Statistics Canada, Ottawa, Ontario, Canada

**ABSTRACT**

For many years now SAS/CONNECT software has allowed SAS® users to make use of remote servers for data processing. However, in our modern day processing environments, security has become a very important issue that cannot be neglected.

Within the SAS/CONNECT environment we have several encryption technologies available that will allow us to communicate securely between a client SAS session and a remote SAS session. Choosing the appropriate encryption technology will depend on the level of security that is required. Many of the common encryption algorithms that are utilized include RC2, RC4 DES, TripleDES and AES.

The purpose of this paper is to explore the use of a particular encryption technology referred to as Secure Shell (SSH) Tunneling within the SAS/CONNECT environment. One of the great advantages of SSH is its flexibility and ease of use. SSH can be obtained through several commercial vendors as well as freeware (such as OpenSSH).

The intended audience for this paper are SAS developers with a medium to advanced knowledge of SAS/CONNECT within the Microsoft Windows environment.

**INTRODUCTION**

As surprising as this may sound, it is very common for organizations not to use encryption on their computer networks. These organizations are either not concerned or unaware of the potential threats by hackers, employees, and any other interested parties. Often firewalls are used to act as the primary barrier against attacks. Without encryption, data travels from one machine to another often as open text. These transmissions may include user ids, passwords, as well as sensitive data.

Therefore, the use of encryption when using SAS/CONNECT is both easy and recommended. The following table compares several encryption technologies that are available to us when using SAS/CONNECT software. Although all of these methods are viable strategies for encryption, we will focus on the encryption technology referred to as Secure Shell (SSH).

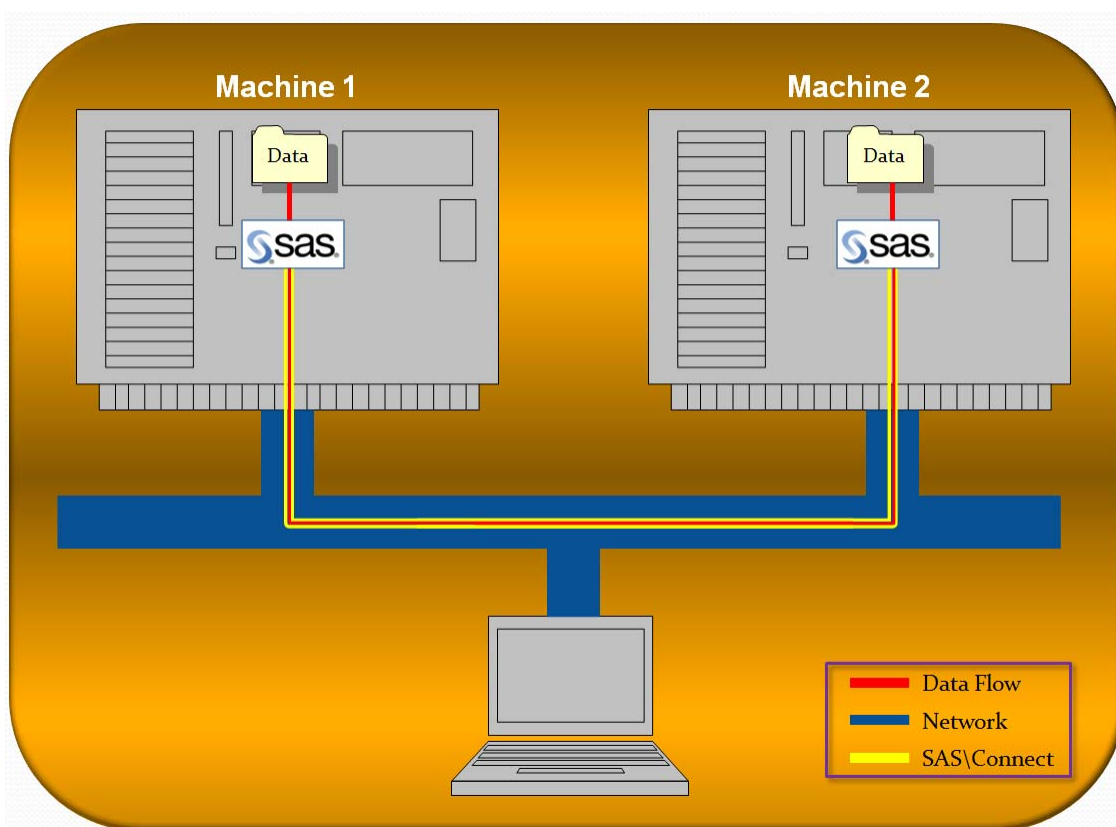
Features	SAS Proprietary	SAS/SECURE	SSL	SSH
License required	No	Yes	No	No
Encryption	Yes	Yes	Yes	Yes
Authentication	No	No	Yes	No
Encryption Level	Medium	High	High	High
Algorithms Supported	SAS Proprietary fixed encoding	RC2, RC4, DES, TripleDES, AES	RC2, RC4, DES, TripleDES, AES	Product Dependent
Installation Required	No (Part of Base SAS)	Yes	Yes	Yes
OS Supported	UNIX Windows z/OS	UNIX Windows z/OS	UNIX Windows z/OS* OpenVMS*	UNIX Windows z/OS
SAS Version Support	8 and later	8 and later	9 and later	8.2 and later

**Table 1- Encryption options within SAS**

In the following sections we will detail the steps required to set up a SSH Tunnel and show how we can make use of it using SAS/CONNECT software. In particular, we will illustrate using open source software called “OpenSSH”. This software is freeware and does support many of the most popular encryption algorithms.

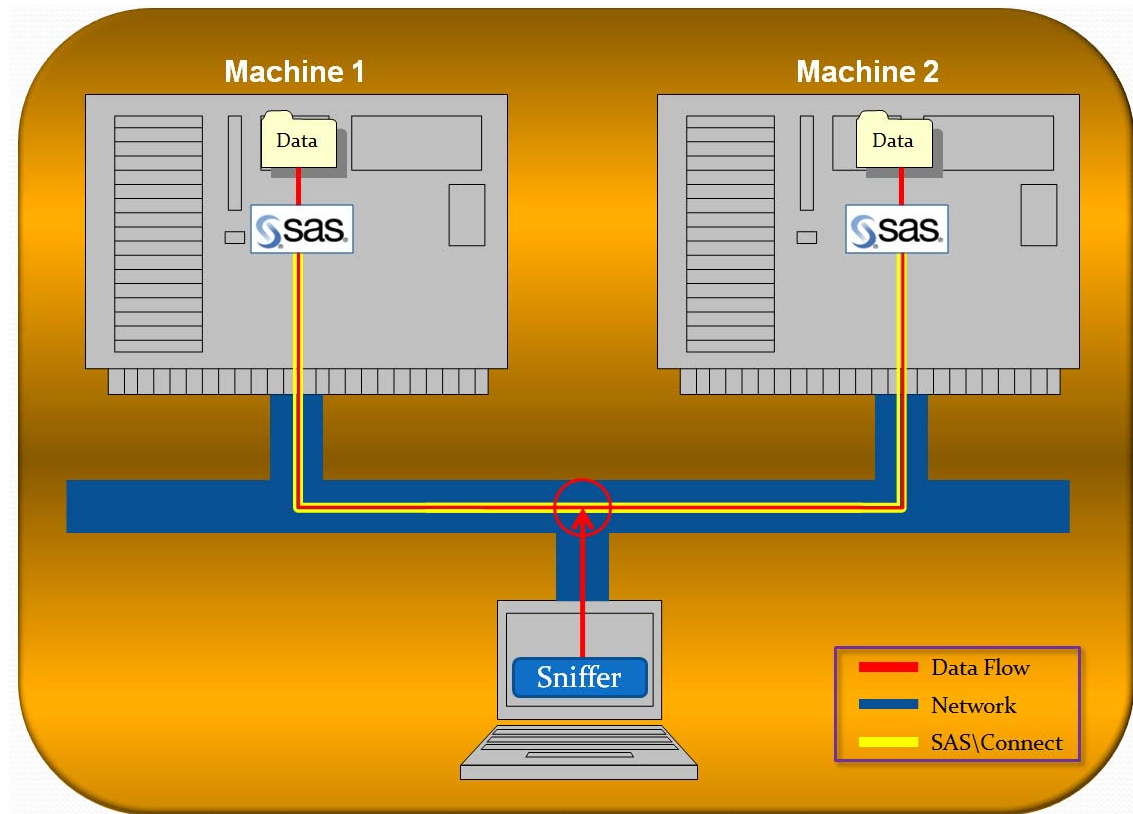
## SAS/CONNECT

Before getting into the details of using SSH, a brief review of SAS/CONNECT may be beneficial. Simply put, SAS/CONNECT software allows a SAS session on one machine, referred to as the client, to “spawn” or start SAS sessions on one or more remote machines, referred to as the server(s). These machines may or may not be of the same operating system type, as long as the necessary SAS software is installed and configured (including SAS/CONNECT). Once a connection has been made, users may be able to remote submit code to these connected machines or move data to and from using special SAS/CONNECT procedures (PROC UPLOAD and PROC DOWNLOAD). Of course there is a lot more functionality available within the SAS/CONNECT product; however, for the purposes of discussion, the above mentioned functionality is sufficient.



**Figure 1- SAS/CONNECT between two machines**

The problem that we have when using SAS/CONNECT over a network is that any data or information passed between the connected machines is typically unencrypted and available to any persons who have the expertise and desire to intercept. **Figure 2** below illustrates how a special program called a “sniffer” is able to gain access to data and information that is passed between machines.



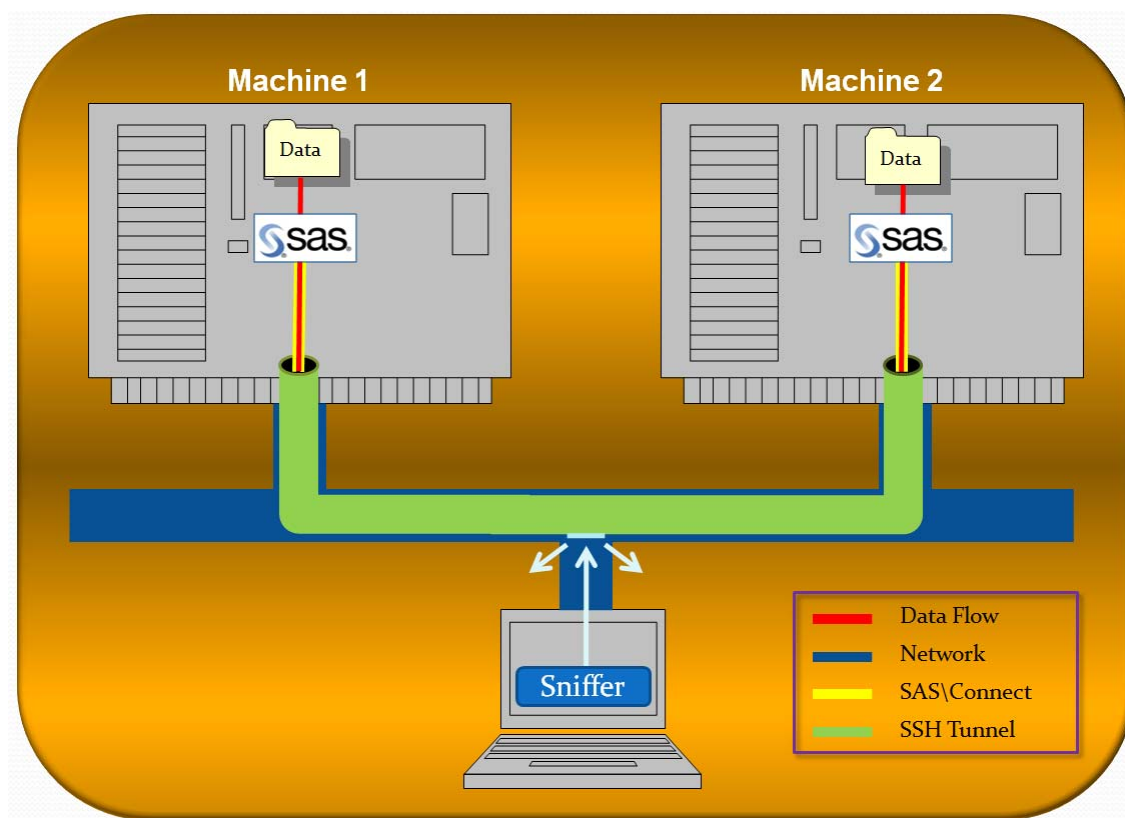
**Figure 2 – Vulnerability of transmitted data**

Therefore, the solution to this problem is to encrypt the data and information that is transmitted between these machines. And of course the solution that we wish to consider is called Secure Shell (SSH).

### SECURE SHELL (SSH) - TUNNELING

In particular we wish to focus our attention on a SSH feature called “tunneling”. The concept of a SSH Tunnel is quite simple and easy to understand without having to discuss the underlying details of this encryption technology. SSH Tunneling is often referred to as “*Port Forwarding*”. A virtual tunnel is created between machines, which allow data and information to travel in an encrypted format, thus preventing any interception by undesired recipients.

Once we successfully setup a SSH tunnel, we will show how we can then make use of it within the SAS environment (using SAS/CONNECT). The successful use of this encryption technology will then allow SAS/CONNECT to function in a secure manner. This is illustrated in **Figure 3**.



**Figure 3- SAS/CONNECT using SSH Tunnel**

Several commercial SSH Software packages are available, as well as an open source version called “OpenSSH”. For illustration purposes, we shall discuss the syntax and use of OpenSSH in this paper. The following section will describe in detail the steps required to successfully setup and use a SSH tunnel within a SAS/CONNECT context.

## STEPS TO SETUP AND USE A SSH TUNNEL


Before performing the following steps, ensure that the Foundation SAS system, including SAS/CONNECT, has been installed and configured on both the client and server machines.

### SAS Spawner Setup on Server

The first order of business is to setup a SAS Spawner on the machine that we will refer to as the server. The SAS Spawner is responsible for launching SAS sessions on the server as requested from client SAS sessions. Typically when instantiating a SAS Spawner service we do not normally specify a port number. However, in this case we will.

1. Start a Command (DOS) window and navigate to the folder that contains the “Spawner.exe” program. A typical location for this file is:


“C:\Program Files\SAS\SAS 9.1”	(SAS 9.1.3)
“C:\Program Files\SAS 9.2\SASFoundation\9.2”	(SAS 9.2)

 **Note:** Under Vista or Windows 7 you may have to open a Command window as an “Administrator” due to the increased security in these versions of the Microsoft Operating systems:

1. **Start ► All Programs ► Accessories ► Command Prompt**
2. Select **Run as administrator** (right mouse click).


2. In the Command (DOS) window, type the following command and hit enter to create the SAS Spawner service:

```
spawner -i -service 4321 -name "SpawnerSSH" -security
```

 **Note:** We have chosen an unused port (4321) on the server machine and we have named the Windows service "SpawnerSSH".


3. Now that the SAS Spawner Service has been created, it must be started. To start the SAS Spawner Service type the following command in the Command window:

```
Net Start "SpawnerSSH"
```

 **Note:** We have chosen the name "SpawnerSSH" for our example. Any name may be used; however it must be the same name that was used in step 2.

4. Now that the SAS Spawner is setup on the server machine, it would be a good idea to test the SAS connection. Start a SAS session on the client machine and submit the following code to determine if the SAS Spawner is functioning correctly.

```
%LET myremote=SERVER1 4321;  
OPTIONS REMOTE=myremote;  
SIGNON;  
SIGNOFF;
```


 **Note:** In the above test program we used the name of our test server (SERVER1) as well as the port number that we used when creating the SAS Spawner service in Step 2. Ensure that you use the correct name or alias of your remote server. A successful SAS/CONNECT session can be verified in the SAS log on the client machine.

### **SSH Setup on Client Machine**

We now need to install and configure the OpenSSH software on the client machine.


1. Download the OpenSSH software ("setupssh381-20040709.zip") from the following internet location.

```
http://www.filewatcher.com/m/setupssh381-20040709.zip.2410307.0.0.html
```

 **Note:** This is an older version of OpenSSH and several newer versions are available. Also note that the OpenSSH software (freeware) is available for download from several other internet sites.


2. Unzip the downloaded file ("setupssh381-20040709.zip") and run the executable file called "setupssh.exe". This program will then install the SSH client. Simply select all of the default settings during installation.
3. In a Command (DOS) window, navigate to the following folder "C:\Program Files\OpenSSH\bin" (default location where OpenSSH software is installed) and type the following commands:

```
mkgroup -d >> ..\etc\group
```

 **Note:** This step could take several seconds.

4. In the same Command (DOS) window, at the same folder location type the following command to add a user:

```
mkpasswd -d -u user1 >> ..\etc\passwd
```

 Note: “user1” is an example of a user id on the domain. Make sure that you specify a valid user id on your domain.

5. In the same Command (DOS) window, type the following command to start the SSH service:

```
Net Start “opensshd”
```

### **SSH Setup on Server Machine**


We now need to install and configure the same OpenSSH software on the server machine. Basically the setup is the same as performed in the previous section (**SSH Setup on Client Machine**). Login to the server machine and perform the following steps:

1. Copy the previously downloaded file (“setupssh381-20040709.zip”) to the server machine and run the executable file called “setupssh.exe”. This program will then install the SSH software. Simply select all of the default settings during installation.
2. In a Command (DOS) window, navigate to the following folder “C:\Program Files\OpenSSH\bin” (default location where OpenSSH software is installed) and type the following commands:

```
mkgroup -d >> ..\etc\group
```

3. In the same Command (DOS) window, at the same folder location type the following command to add a user:

```
mkpasswd -d -u user1 >> ..\etc\passwd
```

 Note: “user1” is an example and should be a valid user id on your domain.

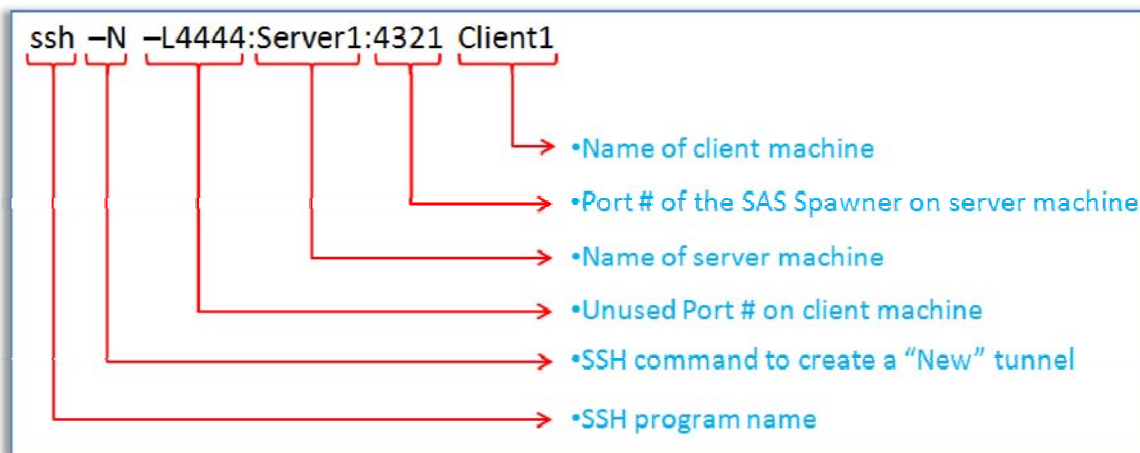
4. In the same Command (DOS) window, type the following command to start the SSH service:

```
Net Start “opensshd”
```

### Creating the SSH Tunnel

Now that we have the SSH software running as services on both the client and server machines, we can now create the SSH Tunnel by performing the following:

1. On the client machine, start a Command (DOS) window and navigate to the following folder "C:\Program Files\OpenSSH\bin" (default location where OpenSSH software is installed) and type the following commands:

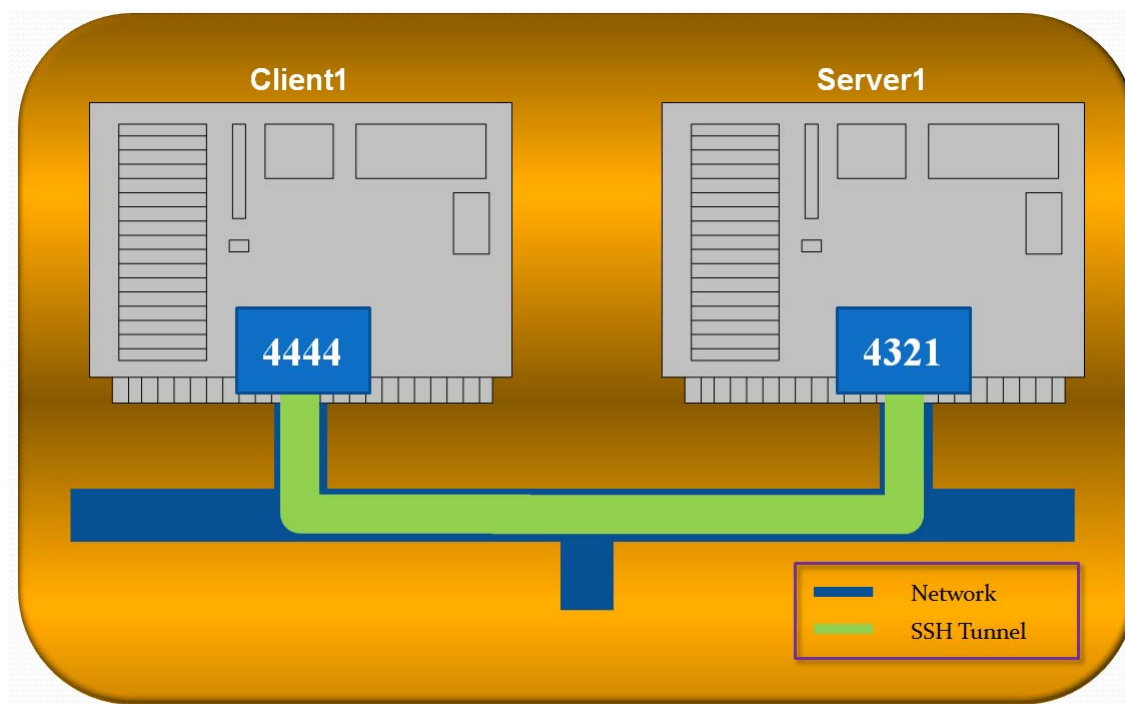


2. You will be prompted and asked if you would like to continue. Select "Yes".
3. You will then be prompted for your network password. Type your valid password and hit the Enter Key.
4. It will appear as if your command window is doing something (**Figure 4**). Simply minimize this window. This will be an indication that your SSH Tunnel has been successfully implemented.



*Figure 4- SSH Tunnel Connection*

The SSH Tunnel that would be created following the preceding steps is illustrated in **Figure 5**.



*Figure 5 – SSH Tunnel*

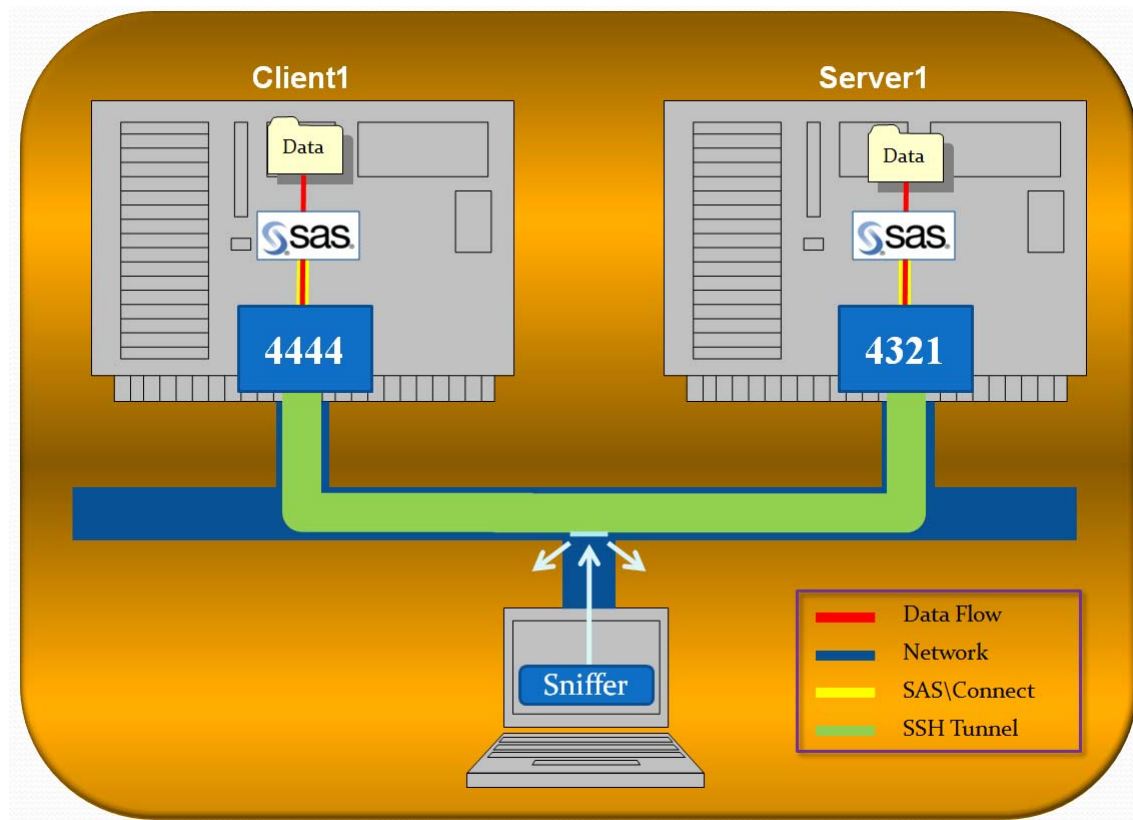
## USING SAS/CONNECT WITH SSH TUNNEL

Now that we have the SSH Tunnel in place we can use SAS/CONNECT software to communicate from the client machine to the server machine using a safe and encrypted medium. Start a SAS session on the client machine and run the following SAS code to remotely connect to a SAS session on the server machine using the configured SSH Tunnel.

```
%let mynode=localhost 4444;  
OPTIONS remote=mynode;  
SIGNON;
```

The difference, compared to traditional SAS/CONNECT statements, is that we actually point to a port (4444) on the client machine and not the SAS Spawner Port (4321) on the server machine. Also note that we are using the term “localhost” and not the actual name of the client machine. The SSH Tunnel will automatically forward all data between the client machine to the server machine in an encrypted fashion.





**Figure 6 – SAS/CONNECT through SSH Tunnel**

Once we have a valid SAS/CONNECT session established between the two machines, we can then use various SAS/CONNECT statements and procedures to move data accordingly. The following is an example that illustrates how we can move data from the server machine to the client machine (by submitting code on the client machine). And of course since we are moving the data through the SSH Tunnel, all communication is encrypted and safe.

```
LIBNAME DATAOUT "C:\Temp";          /*Client Machine */
RSUBMIT;
  LIBNAME DATAIN "C:\Data";          /*Server Machine */
  PROC DOWNLOAD IN=DATAIN
                OUT=DATAOUT;
  RUN;
ENDRSUBMIT;
```

**Example 1 – Transferring data between machines**

## SUMMARY

Just as a bank uses a secure transportation service such as Brinks and not a general courier service to transport money, we too must ensure that our electronic assets (data and information) are safe while being transmitted.

Although there are several software technologies available to be used within SAS, Secure Shell (SSH) proves to be one of the most flexible and easy to use solutions that ensures a high level of encryption. The level and strength of encryption is only dependent on vendor implementation, thus allowing greater flexibility and choice.

So next time you use SAS/CONNECT think about SSH Tunneling technology to ensure that your data and information are safe.

## REFERENCES

SAS Institute Inc. 2011, SAS Online Help and Documentation

SAS Institute Inc. 2011, "Encryption: Comparison", SAS Knowledge Base, Product Documentation, SAS 9.2 Documentation

SAS Institute Inc. 2011, "Providers of Encryption", SAS Knowledge Base, Product Documentation, SAS 9.2 Documentation

SAS Institute Inc. 2011, "Usage Note 20612: Installation of the PC spawner in the Windows Vista Enterprise environment might fail due to User Account Controls", SAS Knowledge Base, Samples & SAS Notes

Glenn Horton. "Sample 25240: Using Secure Shell (SSH) with SAS Products", SAS Knowledge Base, Samples & SAS Notes

Dave Dean, Project Leader, System Engineering Division, Statistics Canada

The OpenBSD project ([www.openssh.com](http://www.openssh.com))

## CONTACT INFORMATION

For information on topics covered in this paper please contact:

	Statistics Canada	Statistique Canada
<b>Greg McLean</b>		
Project Leader – SAS Technology Centre System Engineering Division R.H. Coats Building, 14 <sup>th</sup> Floor, Section Q		
Ottawa, Ontario, Canada K1A 0T6 (613) 951-2396 Fax (613) 951-0607 Greg.Mclean@statcan.gc.ca		
		

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.