Secure connections between GNU/Linux- and Windows-computers using ssh - with special consideration of the Sharp Zaurus

Detlev Reymann
detlev et reymann dot org

Version 1.0, April 2004

Preface

Why this How-To? There were two major reasons to write it.

The first one was the fact, that I bought one of this really nice toys named Sharp Zaurus (in my case it is a SL C 760) some weeks ago. This little machine is running my favorite operating system Linux and brings a lot of the software with it, which is also on my desktop-computer. But from my point of view the configuration of ssh which comes as standard configuration could be improved a little bit on the Zaurus.

The second is the development of Korganizer/Platform-independent (Ko/Pi) (http://www.pi-sync.net/) which brings a full-featured PIM-Software to the Zaurus and - what is even better – it includes the possibility to synchronize your calenders between Zaurus and your desktop-PC. Because this synchronisation is based on ssh this How-To might be useful for users of Ko/Pi who want to use this feature.

Following the discussions in different Zaurus-related forums I got the impression, that there are a lot of Zaurus-users who are new to Linux or do not have this Linux-expert-freak-experience. This How-To is not written for experts who only want to fine-tune their ssh-settings. It is mainly for the „normal“ user who simply wants a running system. Anyway, if you are one of those experts and you find mistakes or you do have ideas how to improve the How-To please drop my an email.

By the way, if there are native english speaking people who want to help fine-polishing the english of this text feel free to do so.

Content

1 Introduction: Basics of ssh

Ssh stands for ***s***ecure ***sh***ell, and that shows the intention behind this software. Ssh has been developed to allow secure connections between two computers. In „early“ days of computing and networking you did not have to think about security aspects when you connected computers. Nowadays, especially if you are connected to the Internet, you have to.

The standard way to connect two computers and to send commands from one computer to another was the usage of telnet. Only the knowledge of a user name and a password kept other users out. The major problem of telnet is that the connection between both computers is not encrypted. With special software it is no problem to listen to the connection and to get username and password. Especially because most systems explicitly ask for a login and for a password. So imagine somebody recording the whole traffic and using software to search for those patterns „login:“ and „password:“. Voila, there the bad boys are. Don't say that this is not relevant for you – for example a major part of the spam mail is meanwhile coming from not secured personal computers with internet connection.

Ssh works different. The traffic between the partners is encrypted via keys. To keep it simple: Each of the computers has a key, which has a secret and a public part. Every partner has the public part of the key of the other. The sender encrypts his traffic with his key, checks wether the recipient is known, the partner has the public part and is able to decrypt the message.

To be a little bit more precise, when we talk about ssh-connections, the computer which starts the connection is the ssh-client (it is using the service of the other) and the computer to which the connection is going is the ssh-server (it delivers the ssh-service). This is important, because you need special software on a computer to be a client and also special software to be a ssh-server. Guess what - because the Zaurus is running a real operating system, there is free software for both and most Zauri come with this software preinstalled. At least the new ones.

So to summarize:

- Ssh is based on keys,
- Ssh encrypts traffic between computers,
- Ssh has a client part and a server part.

Meanwhile ssh is more than only connecting computers and to get a command prompt on the server. Based on the priciples described above you can transfer files between the computers, you can even have a graphical login and so on.

I will only cover the file-transfer, all the rest can be found on the internet.

2 Configuration of a ssh-client

Even if you are only interested in the configuration of a Windows-computer, please read the chatper about Zaurus and GNU/Linux-computers first. I wanted to avoid too much text been written twice, so there might be some explanations in the Zaurus and GNU/Linux-text which help understanding the text about Windows-computers.

2.1 Zaurus and GNU/Linux

2.1.1 Software-installation

To see, whether the ssh-client-software is already installed on your system look into the list provided by „Add/Remove Software“ on your Zaurus or use the package-manager on your desktop-system. Or open a console and type `ssh -V` (capital letter V!). On my Debian-PC I see something like

```
$ > ssh -V
OpenSSH_3.8p1 Debian 1:3.8p1-1 ...
```

On my Zaurus I see:

```
> ssh -V
OpenSSH_3.7.1p2, SSH protocols 1.5/2.0 ...
```

If you don't, you have to install the software. For your Linux-desktop follow the instructions of the distribution how to install software. On my Debian-system an `apt-get install ssh` would do for example. For the Zaurus look for OpenSSH. A link to this is available for example on http://www.killefiz.de/zaurus/. Be sure to take the most recent version which is available. You will get a compressed tar-archive containing three installable ipks. You should install all three of them.

2.1.2 Key-generation

I recommend ssh-connections via key-authentication. To get this working you need your encryption-key in a subdirectory called `.ssh` in your home-directory. As described in the introduction this key consists of two parts, one secret and one public part.

To create your key open a console and type in:

```
ssh-keygen -t dsa
```

to create a key of the type dsa. You will be asked whether the file should be saved as .ssh/id_dsa. Press the enter-key to confirm the default. The next step is a critical one. You are asked for a passphrase. In the case you protect your key with a passphrase you will will be asked for this passphrase everytime you use your keys. This enhances security a lot. In the case somebody gets your key it is still not usable for him when you keep your passphrase secret. On the other side, you can automate your connections to a higher degree without a passphrase. But in this case really nobody should have access to your keys and that should mean even not to send the public part of the key via unencrypted mail and that there should be a high level of security on the side of the administration of the server you want to connect to! Keep in mind that it is really not a problem to get access to your home-directory even of your linux-box if there is physical access to this computer (as much as I love Knoppix, Gnoppix, Morphix and Co. - the disadvantage of them is that you are able to boot those and instantly have root access to all partitions of the host!). I think it is not neccessary to explain, how insecure the operating system windows is. So really take time to think about that! The securest way to use ssh I know in the moment would be to save your key on a smartcard and not on your computer but we keep it simple here. If you want to sync KoPi-calenders you can use ssh either based on password-authentication or based on keys without a passphrase.

So type in your passphrase or leave it empty – you will be asked two times. Afterwards you will find a file named id_dsa and a file named id_dsa.pub in the subdirectory .ssh. The first one is the secret key and the second one the public part of the key. The last one should be transfered to the server you want to connect to. Please avoid to send this file via unencrypted mail or leave this file on a disc in your office etc.!

You could also create a second key of the type rsa doing the same as described above but using the parameter rsa instead of dsa, the command would be: `ssh-keygen -t rsa`

2.1.3 Establish a connection

Before you can establish a ssh-connection from your client to a ssh-server the server has to be prepared for that by the sys-admin. If you are the sys-admin, please have a look to the chapter about server-configuration to get information how to configure the server.

You should test the connection to your server first on the command-line. Type in

`ssh -v hostname_or_host-ip` (replace the latter with the ip-number or the qualified hostname of the ssh-server)

The -v parameter is normally not neccessary, but gives you a lot of messages, which could be very helpful to analyse problems. This may be very useful for your first attempts and may even help to understand what is going on. Take for example the following output (the column on the right tries to explain what happend on the most important parts of the output):

Command and output	***Explanation***
ssh -v 192.168.0.1	Start a ssh-session in verbose mode with the server with the ip 192.168.0.1.
[scipping some output]	

Command and output	*Explanation*
Enabling compatibility mode for protocol 2.0	There are still ssh-servers using the old protocol version 1.0. If you get an according message you have to create a special key! This server uses protocol version 2.
The authenticity of host '192.168.0.1' can't be established. RSA key fingerprint is 13:ab:68:34:gf: and some more signs ... Are you sure you want to continue connecting (yes/no)	As described the key authentication used by ssh is based on a public and a secret part of an encrypted key. This is also true for the server. You have to compare this fingerprint with the fingerprint you received from the system-administrator of the remote system. Type yes and hit the enter-key only if you really can be sure, that this key-fingerprint is ok.
Warning: Permanently added '192.168.0.1' (RSA) to the list of known hosts.	This means that the fingerprint is permanently saved in a file named known_hosts in the subdirectory .ssh. If you said yes, you should not see this message again.
[scipping some output]	
Authentications that can continue: publickey, password Next authentication method: publickey Trying private key: /home/kashmir/.ssh/identity Trying private key: /home/kashmir/.ssh/id_rsa Offering public key: /home/kashmir/.ssh/id_dsa Next authentication method: password kashmir@192.168.0.1's password: Permission denied, please try again. kashmir@192.168.0.1's password: Permission denied, please try again. kashmir@192.168.0.1's password: No more authentication methods to try	Client and server try first athentication via keys, you see the order in which they try. Because none of them work, the server asks for a valid password but in the end even that does not work.

If the server knows your key and is already registered in your known_hosts file than you simply should get the following if you don't use the -v parameter and if you did not define a passphrase:

```
ssh 192.168.0.1
kashmir@server:~>
```

The last line is already the command prompt from the server! If you defined a passphrase you will be asked for that before you get the command prompt. To stop the connection simply type `exit` and you will be back at the command prompt of your locale computer.

2.1.4 Finetunig

There are several options you can add (and you have to add for your Zaurus) to your ssh command besides the -v verbose-option. The default configuration always tries to connect to the server using the user-name you currently have on the client-computer. If you think of a ssh-connection between your desktop-computer as client and your Zaurus as server you will probably not have a name on your desktop which is „zaurus“. You can force a different user-name for the remote computer, using the parameter -l.

So the user kashmir on his desktop could connect to his Zaurus (with the default ip 192.168.139.201) and be user zaurus there by:

```
ssh -l zaurus 192.168.129.201
```

If you are as lazy as I am, you can simplify the process by creating a file named `config` in

the subdirectory .ssh. Use a text-editor to create it and add the following lines to that file. The lines beginning with # are comments, they are not neccessary but they may help to remember what this file is good for:

```
# Define connection from desktop-computer to zaurus
# Host ist the shortname you can use for ssh and scp commands
Host zaurus
# This is the DNS of the partner or the IP
HostName 192.168.129.201
# the user-name on the zaurus
User zaurus
```

You can add different configurations for other ssh-connections in this file as well.

As a result of the lines described above you only have to type

```
ssh zaurus
```

to connect to your zaurus without giving the different user-name and the ip-number. That's nice or? Ssh will look for a file called config in the .ssh-directory, will look whether there is an entry Host zaurus, will use the ip given in the line HostName and will log in as user zaurus. If you want to connect to a server on the internet and the name of this server can be resolved to an ip-number via a dns-service or if you resolve the ip in your /etc/hosts-file you can put this hostname in the line beginning with HostName. There are a lot more options for the config-file, we will leave that to the experts.

An interesting aspect of ssh is the possibility to add a command which will be executed on the remote computer to the end of the command-line invoking ssh.

For example

```
ssh zaurus ls
```

would execute the ls command on your Zaurus immideately after establishing the connection. Play around with that possibility.

2.1.5 File transfer

Probably in most cases you will not only be interested in ssh-connections to have access on the command-line level. Even more interesting is the possibility to transfer files via ssh.

There is a command-line tool called scp which can be used for that purpose.

If you followed the instructions of the foregoing chapters including the creation of a config-file in the .ssh-subdirectory the command

```
scp abcd.txt zaurus:xyz.txt
```

would copy the file abcd.txt from the current working directory of your client-computer to the file named xyz.txt in the home-directory of the Zaurus. The command

```
scp zaurus:xyz.txt abcd.txt
```

would copy from the Zaurus to your desktop.

To make file-transfers easier, there is a nice graphical interface for Desktops running GNU/Linux. But you must have a working ssh-connection, so this is just the cream on top! You have to install a program named *gftp* on your desktop according to the instructions how to install software in your distribution (for example `apt-get install gftp` on Debian-Systems).

Start gftp. First select „FTP“ and „Options“ on the menu, select the tab named „SSH“ and make sure „use SSH2 SFTP funktion“ is checked. With the original Sharp-Rom on my Zaurus I had to change the setting of SSH2 sftp-server path to /usr/libexec to get the system working, with the latest Cacko-Sharp-Rom that seems not to be neccesary. On most Linux-systems the sftp-server will be installed in /usr/lib/; in some cases it might be neccessary to ask your the administrator of your ssh-server for details.

To start a connection, go to the prompts below the menu-entries of gftp, type `zaurus` in the prompt after „computer“, type `zaurus` in the prompt after „user“, select `SSH2` in the drop-down list on the top to the right and press the return-key. You should see the directory-listing of the Zaurus in the right panel and your local files on the left. You can select files using the mouse and transfer them via the arrow-buttons. Have a look to the gftp-manuals for details. If you know how to use a file-manager, usage of gftp should be easy.

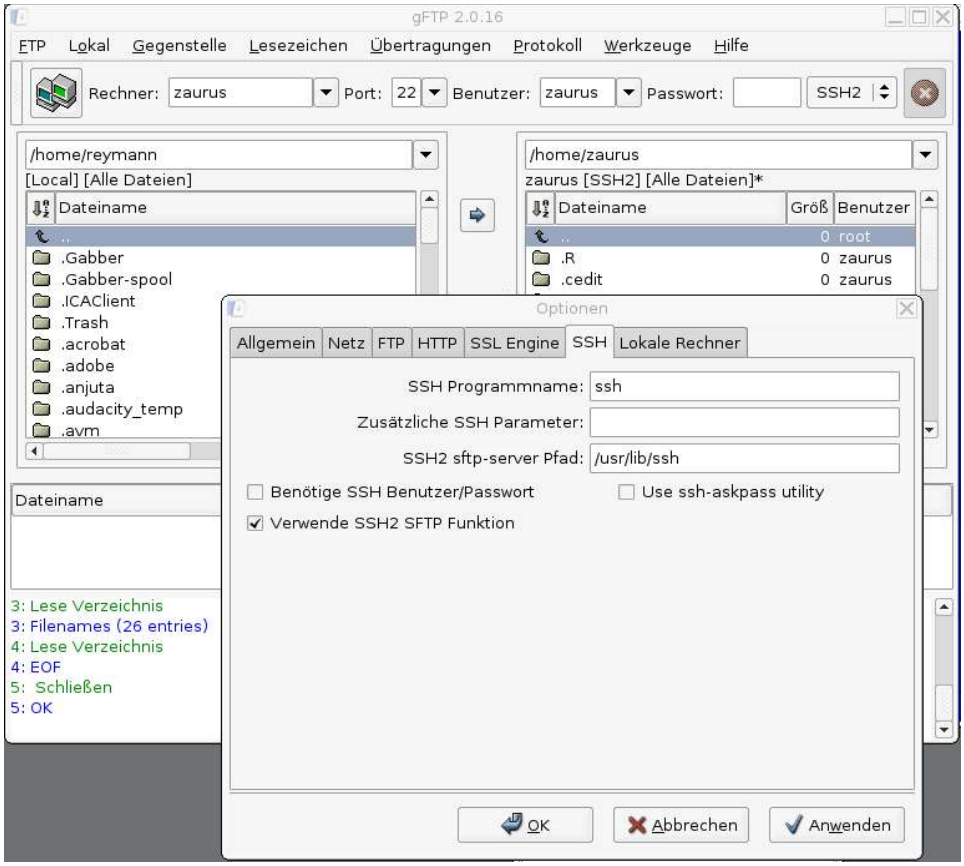

Figure 1: Screenshot of gftp with Options-Tab showing a connection from a GNU/Linux-desktop as client to a Zaurus as server

2.2 Windows

I will only describe two different solutions to install free (at least free for non-commercial use) ssh-client-software on Windows-systems, each of them including a graphical interface for file-transfer. One is the combination of putty (equivalent to ssh) and Winscp (similar to gftp, see above), the other one has been released for non-commercial use by SSH Communications Security, Inc.. They sell the next generation of this client-software and also server-software.

If you want to use this software to connect your windows-box to your zaurus and you want to sync from the windows-computer to your Zaurus (or to another computer) , you have to trick a

little bit due to the fact that KoPi assumes, that there is an executable which is called ssh and anotherone that is called scp (like it is on Linux). Perhaps in the future we will have the possibility to tell KoPi the names and the directories of the appropriate executables.

2.2.1 SSHSecureShellClient

First of all, you have to download the software. Go to ftp://ftp.ssh.com/pub/ssh (you can use your webbrowser for that). Look for a file called SSHSecureShellClient-3.2.9.exe or perhaps a later version. After the installation start the Secure Shell Client from the menu or from the desktop-link. Select „Global Settings“ -> „User Authentication“ -> „Keys“ and klick on the button „Generate New...“. You can leave the default settings as they are. You are prompted for a filename, a comment and a passphrase. See the chapter about the configuratiopn on Linux-systems for deeper explanations. You could for example say id_dsa for the name, a comment as you like and a passphrase or none (see above). Select „Finish“. For use with KoPi you have to click on the „Configure“-button, because KoPi uses the command line client. Click OK on the following dialog

If the server allows password-authentication, you can continue directly, if not, the admin of the server has to install the key (see below). The default location for your keys is „Documents and Settings\Application Data\SSH\UserKeys\“. There you will find id_dsa.pub. I had to modify the content of this file to be accepted on a Debian-Linux system. The original content of this file was something like:

```
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "kashmir@windows-box [1024-bit dsa, kashmir@windows-box, Thu Apr
08 2\
004 21:38:09]"
AAAAB3NzaC1kc3MAAACBAIDt0G1v5/FoeWjgd3soLcyq1o7XAjQ0f7c4sIIfFEePs+FDjM
... and some more lines with a lot more funny letters ...
t2T9GKmZEcQfRKUylA==
---- END SSH2 PUBLIC KEY ----
```

I had to change it so it looks like:

```
ssh-dss AAAAB3NzaCA... and a lot more funny letters ...t2T9GKmZEcQfRKUylA==
```

That means I had to strip the comments, had to add ssh-dss to the begin of the line and had to remove the linefeeds.

If you press the Enter or the Space key in the program-window of the Secure Shell Client a dialog box appears and you see a the prompts for host and user name already containing your previously saved values. If you click „Connect“ you will see a dialog asking whether to store the server key locally. Say Yes if you are sure. You can create profiles in this program for connections to different ssh-servers. Look into the documentation for details.

To use this software with KoPi you have to go to the directory containing the program files (you have been asked for this during installation). Look for the files called ssh2.exe and scp2.exe. Copy them to files named ssh.exe and scp.exe. Test whether this works by opening a command line and type for example ssh -v -l zaurus `192.168.129.201` if you copied and added the public-key to .ssh/authorized_keys-file on your Zaurus.

If that all works, it should not be a problem to start the Secure File Transfer Client. You can use the profiles created within the Secure Shell Client. The result should look like this:

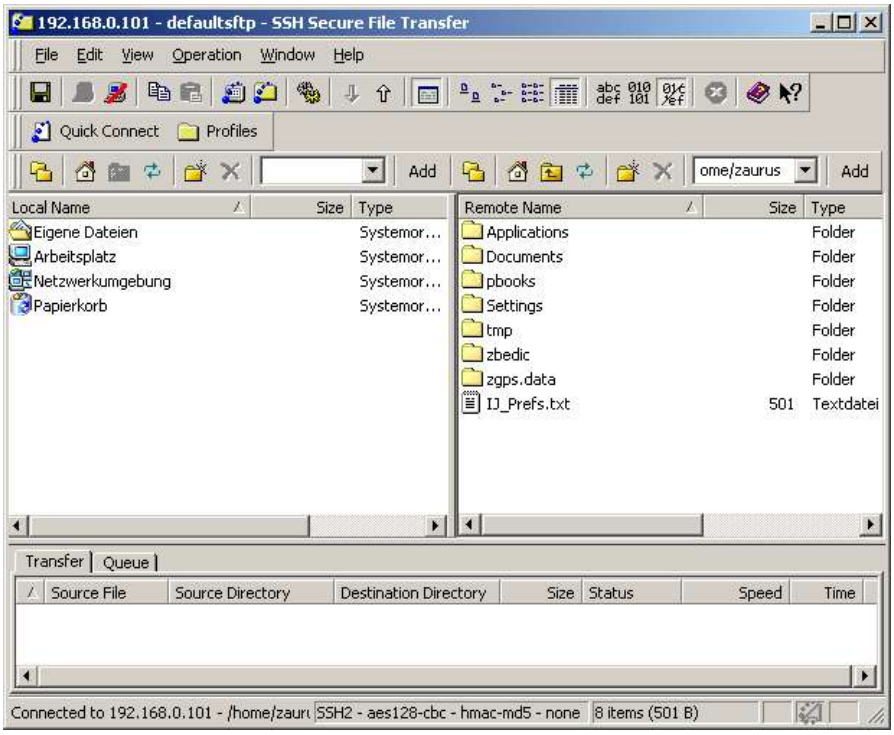

Figure 2: Screenshot of Secure File Transfer Client with connection from a Windows-client to a Zaurus-server

2.2.2 WinScp and putty

First you have to download putty. Go to the downloadpage of the putty-website http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. The easiest way to install it is to download the windows-installer (in the moment writing this documentation it was putty-0.54-installer.exe). If you leave the defaults as they are, you will find some program files in C:\Program Files\putty.

The file pscp.exe is equivalent to scp and the file plink.exe is equivalent to ssh. To use putty with KoPi you have to copy pscp.exe to scp.exe and I recommend to copy plink.exe to ssh.exe. Again the suggestion to add a configuration-option for KoPi to give the path and the name of the ssh-executable. You also have to add the putty-directory to your PATH. Go to Start->Settings->System and change or add an Evironment-Variable PATH to include this directory. After this you should be able to see some output from the programm if you open a console-window and type in ssh and press the enter-key. You should already be able to connect to your Zaurus by typing ssh -l zaurus 192.168.129.201 if your Zaurus allows password-authentication.

If you want to use key-authentication you have to create your key via the program puttygen. Select SSH2 dsa for the type of key to generate at the bottom of the program and press Generate. You have to move your mouse and after a while key-generation finishes. Use the button „Save public key“ and the button „Save privat key“ to save your keys. Read the text above about using a passphrase or not. Please, really read the text – it is important! Save the privat key as id_dsa.ppk and the public key as id_dsa_pub.

Transfer the public-key to the server and the sys-admin should add this key to the file .ssh/authorized_keys. I had to modify the content of this file to be accepted on a Debian-Linux system in the same way as desrcibed for the Secure Shell Client. Look into the chapter above

for details.

Now you should be able to sync from Windows-KoPi to your Zaurus! Because in the moment there is no possibility to tell KoPi to use a special key for authentication (parameter -i for scp) the combination putty/Winscp only allows password-authentication (or I did not find the right configuration?; if so, drop me a mail).

To have the possibilty of file-transfer using a program with a nice interface you should download WinSCP. Go to http://winscp.sourceforge.net/ and download the winscp-installer. There is an international one including support for different languages and a smaller one which is only in english. Install winscp using the installer. Important! You can disable the Installation of puttygen and pageant in the page with the options for the userdefined installation because you already have them. For KoPi we need the scp equivalent, so we need the full putty-installation and not only the parts which would come with winscp.

Start winscp and configure it to use the keys generated by putty. That should be straight forward. The result could look like this:

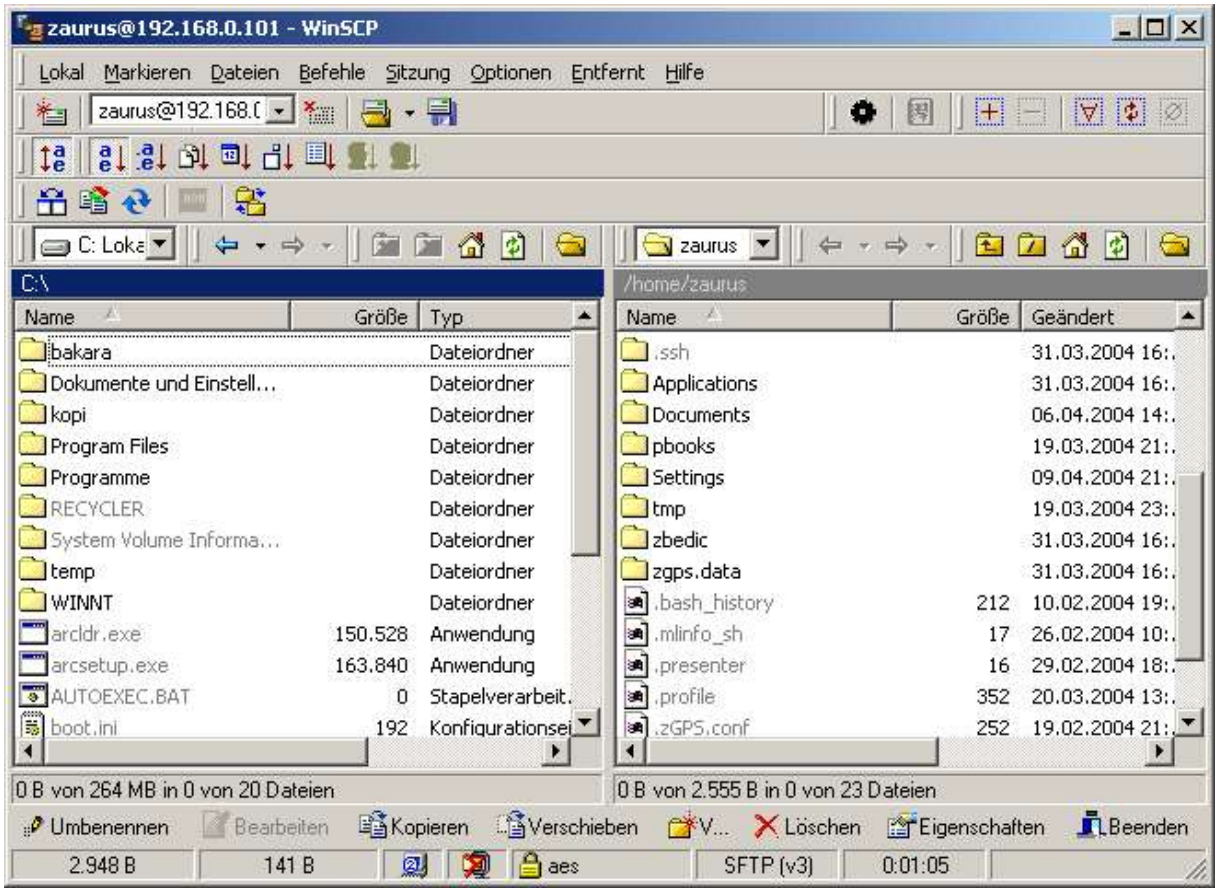

Figure 3: Screenshot of WinSCP with connection from a Windows-client to a Zaurus-server

3 Configuration of a ssh-server

Even if you are only interested in the configuration of a Windows-computer, please read the chapter about Zaurus and GNU/Linux-computers first. I wanted to avoid too much text been written twice, so there might be some explanations in the Zaurus and GNU/Linux-text which help understanding the text about Windows-computers.

3.1 Zaurus and GNU/Linux

3.1.1 Software-installation

The server program is normally started during the init of the system. In most Linux systems you will find scripts in the directory /etc/init.d/. This may be different depending to your distribution but it is the case on my Debian-System. On the Zaurus those scripts can be found in /etc/rc.d/init.d/. So the first thing you should do is to look, whether you find a script named ssh in this directory. If it is there, the ssh-server is installed on your system.

If not follow the instructions how to install the software in the chapter about ssh-clients.

In most cases the installation of ssh will also enable your computer to start the ssh-server during the bootup. This is done by a file named S45ssh in /etc/rc2/ (on my Debian-box) or a file called S17ssh in /etc/rc.d/rc5/ on the Zaurus. What do those files do? In fact those files are so called symbolic links to the above described script ssh. During the bootup-process of a linux-machine there are different runlevels. Each runlevel stands for a different state of the computer, the higher the runlevel the more services are available (single user, multiuser, networking, graphical windowing system), only runlevel 6 is an exception, this level will shutdown the system. For every runlevel the system looks into the appropriate directory (rc.0 to rc.6). If there exist scripts starting with the letter S the script where the symbolic link points to will be run with the parameter start when the system enters this runlevel, each script starting with the letter K will be run with the parameter stop when the system leaves this runlevel. Test whether you understood the text by searching for the script which stops the ssh-server in the correct runlevel-directory – got it?.

As I already said, normally those scripts are created, when you install the server-software. So usually you don't have to care about creating these scripts.

To test, whether the ssh-server is running you could do the following. Open a console. Become root (typing su and eventually the root-password) and type (on your Zaurus) `/etc/init.d/ssh stop` or `/etc/rc.d/init.d/ssh stop (on a Debian-system)`. Start the server again with the same command but `start` as parameter.

3.1.2 Server-configuration

There is a directory /etc/ssh/ where you can find the keys which are used by the ssh-server and a configuration file called sshd_config. There is one option, which should be changed on the Zaurus(as far as I think). Search for a line containing `PermitEmptyPasswords yes`. This allows connections to your zaurus without having a password. Since everybody knows, that the default user on a Zaurus is named zaurus, this in fact enables connections to your Zaurus for everyone and makes the whole security of ssh more or less useless.

Please change this to `PermitEmptyPasswords no`. You have to stop and to restart the server as described above to activate these settings. When your configuration is tested and user-access based on keys is working you could also change the line PasswordAuthentication to „no“.

Btw., I think these default settings are so „relaxed“ because the Zaurus is not seen as a network computer by Sharp. But I don't agree. You can have the security of ssh without any loss on convenience. So why not using it? So I also created a password for your normal zaurus user and the root-account on my Zaurus.

If you have a working ssh-server on your Zaurus you could (and should) look whether the insecure telnet-server is still offering connections. Open the file `/etc/inetd.conf`. All lines except the line starting with `ssh` should start with the letter # (which prevents those services to be offered by your Zaurus because they change this line into a comment); whether you should leave the lines starting with `netbios` the way they are depends on the fact whether you want to use the samba-abilty of the Zaurus. Because there are nice possibilities for file-transfer based on ssh and I dont see any use in samba in my case they are also uncommented. But I think some of the Intellisync-synchronisation is based on samba – if you use Intellisync you should probably not touch these lines.

3.1.3 Key-generation for the server

As far as I see, nowadays the keys which are used for the encryption process are generated automatically when the server starts the first time or upon software-installation. Neither on the Zaurus nor on my desktop I had to create them manually. Look in the directory /etc/ssh whether there a files called ssh_host_dsa_key and ssh_host_rsa_key. If not, you have to create them. Open a console and type in:

```
ssh-keygen -t dsa -G /etc/ssh/ssh_host_dsa_key
ssh-keygen -t rsa -G /etc/ssh/ssh_host_rsa_key
```

to create two different types of keys. Be patient, this may take a while, that really means a long while! Please write down the RSA fingerprint which is displayed after this key-generation! Clients will have to check this fingerprint to verify the connection. If you need the fingerprint later and forgot to make a note, you can get by as root by typing `ssh-keygen -l` and entering the host-key (usually `/etc/ssh/ssh_host_dsa_key`).

3.1.4 Configuration of user-access

The next and last step to enable ssh-connections to this server is the preparation of the access for those users who are allowed to connect via ssh.

There is the possibility to connect from a ssh-client to a server and to rely the authentication only on the user-name and the valid password. This is still a higher level of security compared to telnet, because the traffic between client and server is encrypted. But you really should use the key-based authentication!

What you have to do is the following: You have to change into the HOME-directory of the user. Let's assume a user called kashmir (that's the name of my dog) shall be able to access the server from his remote computer. There has to be an account for this user on the server. If there is no account, create one, e.g. via the „adduser“ command or the tools provided by your distribution. In most cases you will have a HOME-directory for the user as a subdirectory of /home, in our example it would be /home/kashmir. As root open a console and change into this directory:

```
#> cd /home/kashmir
```

Create a directory called .ssh (the dot is important, because this creates a hidden directory.

```
#> mkdir .ssh
```

Make shure that this directory is owned by the user and that it is only readable for this user.

```
#> chown kashmir.kashmir .ssh
#> chmod 700 .ssh
```

Change into this directory.

```
#> cd .ssh
```

There you need the public part of the user key (see the description in the section above). This file will be something like id_dsa.pub. You have to create a file named authorized_keys which contains a line with this key. You could create it via:

```
#> cat id_dsa.pub >> authorized_keys
```

The real critical part is to be sure, that this key is really the public key of the user! If there is enough paranoia, you should receive the key via the „adidas-net“ (You do not know this net: it means copying the key to a diskette, putting on your running-shoes and bringing it physically to the admin of the server).

On my Zaurus the adduser-command does not work. Because the default user for most applications is zaurus anyway, you have to execute the same commands for the user zaurus. The HOME-directory is /home/zaurus and the group is zaurus.qpe. So the whole thing would be:

```
#> cd /home/zaurus
#> mkdir .ssh
#> chown zaurus.qpe
#> chmod 700 .ssh
#> cd .ssh
#> cat id_dsa.pub >> authorized_keys
```

You should be ready.

3.2 Windows

Windows does not have any ssh-software by default. There is a port of the OpenSSH-Server for the Windows-environment which is freely available on http://sshwindows.sourceforge.net/download/ Go to the directory with the latest release and you will find something like setupssh371-20031015.zip (The release may change in the future). As the authors say themselves: „OpenSSH for Windows is a free package that installs a minimal OpenSSH server and client utilities in the Cygwin package without needing the full Cygwin installation“. You can download this package and install it on your Windows computer; keep in mind, that you need administrator privileges to do this.

The package mentioned above contains the server- and the client-software.

This ssh-software is a port of the Linux/Unix software to Windoof (an insider-joke for german speaking people) via Cygwin. Normally the Cygwin-package tries to install a complete Unix/Linux like environment on a windows-machine. This may cause some irritations for people only living in the windows-world (for example concerning the naming of drives).

If you do not really need a ssh-server on your windows-box perhaps stay with only the client-software. Installation and usage of this may be more simple. By the way it does not make to much sense to install a secure server-software on the base of a so unsecure operating-system. And because this ssh-server is based on Cygwin the authors themselve recommend not to use this software on production-systems.

After installing the software – let's say in `C:\Program Files\OpenSSH\` you will find the configuration files in a subdirectory of this directory called `etc`.

You will have to create a .ssh-directory for the user according to the desription for a linux-

box. But you have to use `C:\Documents Settings\username` as the home directory. All other explanations from the chapter about a Gnu/Linux ssh-server should we valid for this software too. The ssh-server process can be stopped and started via „Start, Settings, system, Administration, Services or by opening a command line and typing `NET STOP OPENSSH` or `NET START OPENSSH`

The ducumentation which comes with this software and which is in the subdirectory docs gives detailed informations how to alter the configuration.

To be honest – up to now I had no success to configure the ssh-server on my Windows 2000 computer in a way that authentication via keys work. Only password-authentication is ok. But my enthusiasm to fiddle with configuration details on this system is not the same as on my Linux-computers ...