SSH の使い方

– より安全にネットワークを利用するために –

赤坂 浩一 *　　平野 彰雄 *

1　はじめに

インターネットを利用して遠隔地から計算機を利用する場合には、 telnet などを使ってリモート計算機にログインしますが、この時、端末から入力された ID とパスワードは、そのまま平文 (暗号化されない文字列) として、通信経路に流されています。

インターネットはご存じの通り、複数の計算機を物理的に接続して成り立っていますので、ある計算機からある計算機までの通信経路には、中継する計算機が存在します。途中の経路に悪意のあるネットワーク管理者が存在すれば、通信データを簡単に覗き見することでき、 ID やパスワードを盗み出すことも可能です。 (**図 1**)

今回、紹介する **SSH**(Secure SHell) を利用することにより、通信データ (ID やパスワードを含めて) を暗号化して、安全に計算機を利用することが可能になります。 SSH は、クライアントサーバ形式のプログラムですので、 SSH を利用する場合は、リモート計算機とローカル計算機の双方に SSH を導入する必要があります。

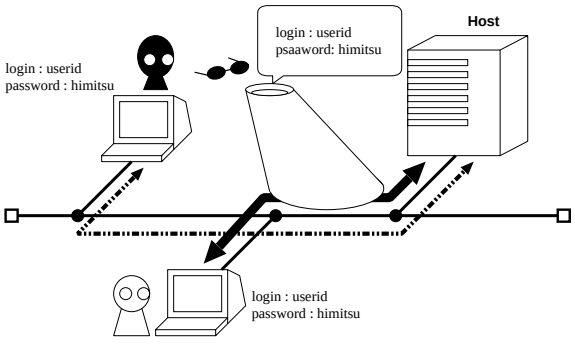

図 1. 盗聴の危険性

それでは、平成 11 年 7 月より本センターの汎用 UNIX システム (以下、 sakura) に導入した SSH の使い方を簡単に説明します。本稿では、例として UNIX 系 OS からの利用方法と Windows パソコン (以下、 PC) からの利用方法について紹介します。

なお、 SSH には、 SSH1 と SSH2 の二種類がありますが、 sakura にインストールしたものは、 SSH1(ssh-1.2.27) です。また、 SSH の公式ホームページは、 `http://www.ssh.fi/` です。

2　SSH とは

SSH は、ネットワークに接続された二つのホスト間に安全な通信経路を提供するプログラムです。強力な認証機能で IP アドレスの偽装などを防ぐことができ、通信データを暗号化することで内容が盗聴されないようにすることができます。

SSH での暗号化の仕組みについては、今回は詳しくは説明しません。

SSH では、計算機の認証とユーザの認証に、 RSA の認証プロトコルを利用します。 RSA の認証は、公開鍵暗号方式で「公開鍵」と「秘密鍵」の二つの鍵を使います。公開鍵暗号方式の利点は、「公開鍵」は復号することが困難なので、通信経路にそのまま流しても、誰かに盗聴されても困りません。そのかわり、暗号 ⇔ 復号に要する処理が複雑なため、非常に時間がかかってしまいます。

そこで、 SSH では、計算機とユーザの認証に RSA を利用して、信頼のおける計算機とユーザであることを確認できると、その通信で使用する「共通鍵」を動的に生成して、「共通鍵」を公開鍵暗号方式で交換し、その後の通信は、「共通鍵」を利用

* あかさか ひろかず , ひらの あきお (京都大学大型計算機センター)

した共通鍵暗号方式で通信データを暗号化します。「共通鍵」は使い捨ての鍵です。

共通鍵暗号方式は、暗号 ⇔ 復号に要する処理が比較的簡単なため、時間は短くなりますが、暗号 ⇔ 復号に同じ鍵を利用するので、そのまま通信経路に流すと盗聴される恐れがありますが、使い捨てる「共通鍵」を毎回、生成し公開鍵暗号方式で交換するので、安心して利用することができます。

SSH では、公開鍵暗号方式と共通鍵暗号方式の二種類の暗号方式を利用して、安全な通信経路を提供しています。

3 UNIX からの利用方法

SSH は、多くの UNIX 系 OS で動作が確認されています。ここでは、お使いのワークステーション (WS) から sakura を利用する方法を説明します。

3.1 ソースコードの入手

SSH のソースコードは、次の FTP サイトから入手することができます。

- 一次配布元：

`ftp://ftp.cs.hut.fi/pub/ssh/`

- 国内：

`ftp://ftp.kyoto.wide.ad.jp/pub/security/ssh/`

古いバージョンはセキュリティに問題があると考えられますので、常に最新のバージョンのソースコードを入手してください。本稿執筆中の最新バージョンは、`ssh-1.2.27.tar.gz` です。

また、上記の FTP サイト以外からも入手することができますが、SSH は暗号プログラムのため、国外からの入手・利用には注意が必要です。特に米国では暗号プログラム輸出は禁止されているようなので、米国サイトからソースコードを入手しないようにしてください。

3.2 インストール

お使いの WS から sakura を SSH で利用するためには、SSH クライアントだけを用意すれば利用することができます。お使いの WS を遠隔地から SSH で利用するためには、SSH サーバを用意する必要があります。ここでは、SSH サーバについては省略します。

入手したソースコードは、SSH のファイル一式がアーカイブされていますので、`tar` や `gzip` コマンドを使って次のように展開します。

```
%  gzip -dc ssh-1.2.27.tar.gz | tar -xvf -
```

展開すると `ssh-1.2.27/` のディレクトリが作られ、その中にファイルが展開されています。

作成されたディレクトリに移動し、`INSTALL` ファイルを読むとインストール方法が書かれています。SSH は autoconf 対応なので、`configure` スクリプトを実行すると自動的に `Makefile` を生成してくれます。あとは、`make` コマンドを実行すれば、コンパイルできます。

手順としては、次のようになります。

```
%  cd ssh-1.2.27
%  ./configure
%  make
```

無事、コンパイルが終了すると次のようなプログラムおよびユーティリティが作成されます。

クライアントプログラム	`ssh,slogin,scp`
サーバプログラム	`sshd`
その他ユーティリティ	`ssh-keygen,ssh-agent,` `ssh-add,ssh-askpass,` `make-ssh-known-hosts`

作成されたプログラムやユーティリティは、特に指定しない限り、`sshd` は `/usr/local/sbin/` に、それ以外は `/usr/local/bin` ディレクトリにインストールされます。

それでは、root になってインストールします。

```
#  make install
```

なお、サーバプログラムの `sshd` もインストールされるので、この時、次のような RSA ホスト鍵と設定ファイルが生成されます。

`/etc/ssh_config`	SSH の設定ファイル
`/etc/sshd_config`	`sshd` の設定ファイル
`/etc/ssh_host_key`	ホスト鍵 (秘密鍵)
`/etc/ssh_host_key_pub`	ホスト鍵 (公開鍵)

以上でインストールは完了です。お使いの WS で SSH サーバを起動しない場合は、これ以上の作業は必要ありません。

3.2.1 SSH の設定ファイル

SSH の設定は、/etc/sshd_config ファイルで、サーバ (sshd) の設定、/etc/ssh_config ファイルで、ssh クライアントのデフォルトの設定が定義されています。

ssh クライアントの設定ファイルは、各ユーザの~/.ssh/config に同じような内容で置くことができ、こちらが優先して参照されます。

特にここでは内容について紹介しませんので、詳しく知りたい方は、ssh(1) および sshd(8) のマニュアルを参照してください。

3.3 基本的な使い方

SSH をインストールするとクライアントプログラムとして、slogin,ssh,scp のコマンドが使えるようになります。これらのコマンドはそれぞれ、rlogin,rsh,rcp といったコマンド (いわゆる R コマンド) の代わりに使用することができ、sshd(サーバ) が動作しているリモート計算機 (sakura) に 対して実行することによって、相当する R コマンドと同様な利用ができます。**表 1**は、SSH と R 系コマンドの対応を示しています。

ssh コマンド (slogin,scp も同様) を使って、初めて、リモート計算機 (sakura) に接続する場合には、**図 2**のように、その旨のメッセージを表示して許可を求めてきますので、“yes” と答えると、リモート計算機 (sakura) の RSA 鍵が、お使いの WS のファイル (~/.ssh/known_hosts) に書き込まれ保存されます。

次回以降の接続には、この鍵がチェックされ、もし入れ替わっていれば警告メッセージが出力されます。これにより、DNS が書き換えられて悪意のあ

表 1. SSH と R 系コマンドの対応

rlogin(リモートログイン)	slogin
rsh(リモートシェル)	ssh
rcp(リモートコピー)	scp

るホストへ接続されるような事態を防ぐことができるようになっています。

なお、もし接続するリモート計算機で、sshd が動作していない場合は、自動的に、rlogin,rsh,rcp にフォール・バックしてくれるので、特に接続先を気にせず、rlogin,rsh,rcp の代わりに slogin,ssh,scp を使うことができます。

3.3.1 SSH の認証

図 2からも分かるように、ssh は rsh と違って接続時にパスワードを尋ねてくるので、~/.rhosts のようなファイルを事前に用意しておかなくても、**パスワードによる認証**でリモート計算機から利用することができます。

パスワードなしで利用するためには、次の三つの方法があります。

- .rhosts による認証

  rsh の場合と同じように、~/.rhosts ファイルや /etc/hosts.equiv ファイルによる認証を許す方法ですが、この認証は危険なので、通常は許可されていません。

- RSA によるホストの認証

  ~/.rhosts または /etc/hosts.equiv と RSA によるホスト認証を組み合わせて、パスワードなしでコマンドを実行させることが可能です。

```
%  ssh sakura.kudpc.kyoto-u.ac.jp -l w55037
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)?  yes
Host 'sakura.kudpc.kyoto-u.ac.jp' added to the list of known hosts.
Creating random seed file ~ /.ssh/random_seed.  This may take a while.
w55037@sakura.kudpc.kyoto-u.ac.jp's password:
```

図 2. ssh コマンドでの接続例

サーバ側の ~/.rhosts や /etc/hosts.equiv にクライアント・ホストが登録されていることに加えて、クライアント・ホストの RSA 鍵がサーバ側の~/.ssh/known_hosts または /etc/ssh_known_hosts に登録されていることが条件になります。

rsh は許したくないが、 ssh ではパスワードの入力を省略したいという場合は、~/.rhosts や /etc/hosts.equiv の代わりに、~/.shosts や /etc/shosts.equiv という名前のファイルを用意します。形式は、~/.rhosts や /etc/hosts.equiv と同じです。

- RSA によるユーザ認証

  RSA によるユーザ認証については、次に詳しく説明します。

3.3.2 RSA によるユーザ認証

これは、ユーザ自身を RSA で認証する方法で、まず、**図 3**のように ssh-keygen コマンドを実行して、自分の鍵 (公開鍵と秘密鍵) を作ります。

秘密鍵は、~/.ssh/identity に、公開鍵は、~/.ssh/identity.pub に作成されます。

ssh-keygen コマンドを実行すると、パスフレーズの入力を求められるので、適当なパスフレーズを入力します。パスフレーズには、通常のパスワードを入力するのではなく、長めの覚えやすい文字列を入力します。空白を含んでも構いません。パスフレーズを忘れると、以降のアクセスできなくなるので忘れないようにしてください。確認のため、再度パスフレーズの入力が求められます。

これで、自分の鍵が作成できました。「秘密鍵」は他人に漏れると「秘密」の意味がなくなるので、しっかりと管理しておきましょう。

次に、「公開鍵」を sakura (接続先となるホスト計算機) の ~/.ssh/authorized_keys に適当なコマンドを使ってコピーします。この例では、 scp コマンドを使っています。

なお、紙面の関係上 “ → ” で折り曲げています。

```
%  scp ~/.ssh/identity.pub →
   w55037@sakura:~/.ssh/authorized_keys
```

この authorized_keys ファイルが .rhosts ファイルに相当しますので、複数の計算機から、 sakura にアクセスする場合は、それぞれの計算機で作成した「公開鍵」を適当な名前で送り、このファイルに cat コマンドなどを利用して追記します。

この authorized_keys ファイルがあると、 ssh コマンドの実行時に次のようにパスフレーズの入力が求められるようになります。

```
%  ssh sakura.kudpc.kyoto-u.ac.jp -l w55037
Enter passphrase for RSA key 'akasaka@sanga':
```

ここでは、通常のログイン時に使用するパスワードでなく、設定したパスフレーズを入力します。

```
%  ssh-keygen
Generating p:  .............++ (distance 212)
Generating q:  .........++ (distance 166)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key (/home/akasaka/.ssh/identity):
Enter passphrase: _____________________
Enter the same passphrase again: _____________________
Your identification has been saved in /home/akasaka/.ssh/identity.
Your public key is:
1024 33 10637207147305651931462011980567492548831337952051079089779689235569537
0
47821023734729840827161366679811011532053741803345459048108109822030879040578898
72869269933978072690978482438131840818486070293916371141870325808041511443581077
31622008928633082623793629978571941775682401358376879201972631507729016123737 ak
asaka@sanga
Your public key has been saved in /home/akasaka/.ssh/identity.pub
%
```

図 3. ssh-keygen コマンドによる鍵の生成

3.3.3 パスフレーズ入力の省略

パスフレーズを毎回入力する手間を省くためには、 ssh-agent というプログラムを使用します。このプログラムは、プロセスのメモリ空間にユーザの秘密鍵を保持しておき、 ssh と通信して送り出してくれるエージェントプログラムです。

ssh-agent コマンドを次のように実行すると、バック・グラウンドで動作し、 SSH_AUTH_SOCK と SSH_AGENT_PID の環境変数に値を設定します。

```
%  eval `ssh-agent`
Agent pid 2245
```

次に、 ssh-add コマンドを使って、ユーザの秘密鍵を登録します。次のように実行して、パスフレーズを入力すると秘密鍵が登録されます。

なお、紙面の関係上、出力される文字列を後ろをカットしています。

```
%  ssh-add
Need passphrase for /home/akasaka/.ssh/identity
Enter passphrase:
Identity added: /home/akasaka/.ssh/identity
```

登録されている秘密鍵は、 ssh-add コマンドの “-l” オプションで確認することができます。

これで、後は普通に ssh コマンドを実行すれば、パスワードもパスフレーズも問い合わせされることなく、実行することができます。

ssh-agent に登録した秘密鍵を (ssh-agent のメモリから) 消去するには、次のように、 ssh-add コマンドに “-D” オプションを指定して実行します。

```
%  ssh-add -D
All identities removed.
```

また、 ssh-agent 自体を終了するには、次のように、 ssh-agent コマンドに “-k” オプションを指定して実行します。

設定された SSH_AUTH_SOCK と SSH_AGENT_PID の環境変数も消去されます。

```
%  eval `ssh-agent -k`
Agent pid 2245 killed
```

3.3.4 パスフレーズの変更

設定したパスフレーズを変更するには、次のように、 ssh-keygen コマンドに “-p” オプションを指定して実行すると、秘密鍵ファイルの問い合わせがあり、古いパスフレーズを入力し正しければ、新しいパスフレーズの入力を求められます。

```
%  ssh-keygen -p
Enter file key is in (/home/akasaka/.ssh/identity):
```

なお、パスフレーズを変更しても、「公開鍵」をリモート計算機に送り直す必要はありません。

3.4 SSH の応用的な使い方

ssh の基本的な使い方は、 rlogin,rsh,rcp といったコマンドの代わりに使用することですが、ここでは、 ssh コマンドを使用することによって、得られる付加的な機能について紹介します。

3.4.1 安全な X コネクションの確立

X ウィンドウで、例えば、 “xhost +” などとしてどこからでもその X サーバに接続できるようになっていると非常に危険です。 X ウィンドウ・サーバは出力だけでなく、入力もつかさどっているため、 X サーバに接続できるクライアントは、キーボード入力を得ることも可能なので、悪意のある人のプログラムが X サーバに接続して、キーボードから打鍵される内容を盗むことも可能となります。

ssh(slogin) でリモート計算機にログインすると、接続した計算機で X ウィンドウ・クライアントが利用できる場合には、 DISPLAY 環境変数がセットされます。このとき、自分の計算機が “myhost”、接続する計算機が “sakura” の場合、通常は DISPLAY 変数の値は、 “myhost:0.0” のようになりますが、ssh を利用すると、 “sakura:1.0” のような値がセットされます。

図 4のように、 DISPLAY 環境変数 (“sakura:1.0”) のディスプレイ番号 1 では、 ssh が待ち構えていて、 X コネクションを受けて SSH コネクション上へ流してくれます。このことにより、 X プロトコルによる通信を安全に行うことができます。

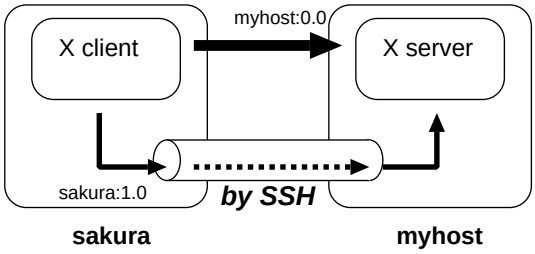

図 4. 安全な X コネクション

ただし、安全な通信経路を確保することは、多少の負荷がかかるので、 Web ブラウザのようなクライアントを立ち上げる場合、少し動作が遅くなります。そのような場合は、もし、通信が安全でなくてもよいならば、 ssh 実行時に “-x” を指定すると、DISPLAY 環境変数のセットを抑止することができます。また、クライアントごとに使い分けたい場合には、セッション開始後に DISPLAY 変数 (または、“-display” オプション) で使い分けることができます。

3.4.2 SSH のポート・フォワーディング

次に、 SSH のポート・フォワーディングについて紹介します。 SSH は、 rlogin や rcp だけでなく、**図 5**のように任意のポートを使用した通信を安全なものとすることができます。

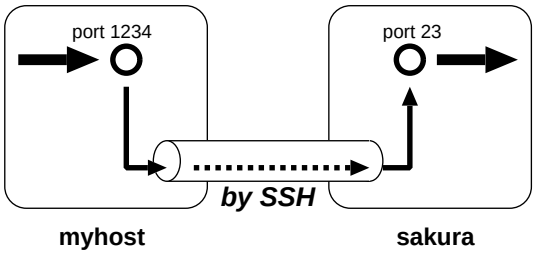

図 5. ポート・フォワーディング

例えば、お使いの WS(myhost) の 1234 番ポートをリモート計算機 (sakura) の 23 番 (telnet) ポートへ接続するには、次のように実行します。

```
%  ssh -L 1234:sakura.kudpc.kyoto-u.ac.jp:23 →
        sakura.kudpc.kyoto-u.ac.jp -l w55037
```

```
%  telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.


UNIX(r) System V Release 4.0 (sakura)

login:
```

図 6. ローカルのポート・フォワーディング

リモート計算機 (sakura) のパスフレーズもしくはパスワードを入力して接続が完了したのちに、**図 6**のように、別の端末 (ウィンドウ) からお使いの WS(myhost) の 1234 番ポートへ telnet すると、リモート計算機 (sakura) へ接続されます。

これとは逆に、リモート計算機 (sakura) 側のポートをフォワードすることも可能です。お使いの WS(myhost) から次のように実行します。

```
%  ssh -R 1234:myhost.kudpc.kyoto-u.ac.jp:23 →
        sakura.kudpc.kyoto-u.ac.jp -l w55037
```

同様にリモート計算機 (sakura) のパスフレーズもしくはパスワードを入力して接続が完了したのちに、**図 7**のように、リモート計算機 (sakura) で、localhost の 1234 番ポートへ telnet すると、お使いの WS(myhost) へ接続されます。

```
%  telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.


IRIX System V.4 (myhost)

login:
```

図 7. リモートのポート・フォワーディング

このようにして、他のアプリケーション間の通信を SSH コネクションを利用して、中継することができます。

4　PC から利用方法

PC から SSH を利用するためのソフトウェアは、いくつか開発されているようですが、ここでは、**TTSSH** を紹介します。

TTSSH は、PC で利用できる **Tera Term Pro** を拡張して、SSH を利用できるようにするソフトウェアです。

Tera Term Pro については、本センター広報「Windows パソコン活用ガイド (5) –sakura,kuma を利用するために –」(Vol.31 No.1) で紹介していますので、そちらを参照してください。

Tera Term Pro,TTSSH ともフリーソフトウェアとして提供されています。

4.1　各ソフトウェアの入手

本稿執筆中の最新バージョンは、Tera Term Pro version 2.3 (`ttermp23.zip`) と TTSSH version 1.4 (`ttssh14.zip`) となっています。

また、それぞれのファイルは、zip 形式で圧縮されて提供されていますので、zip 形式のファイルを展開するためのツールが必要となります。

ポピュラーな展開ツールとして、フリーソフトウェアの **Lhasa** (`lhasa011.exe`) がありますので、こちらを利用するのが良いでしょう。

各ソフトウェアは、次の URL から入手してください。

- Tera Term Pro
  `http://www.vector.co.jp/authors/VA002416/`
- TTSSH
  `http://www.zip.com.au/~roca/ttssh.html`
- Lhasa
  `http://www.digitalpad.co.jp/~takechin/`

4.2　各ソフトウェアのインストール

まず、展開ツールである Lhasa をお使いの PC にインストールします。Lhasa (`lhasa011.exe`) は、自己解凍形式ですので、適当なフォルダにファイルを置き、実行 (ダブルクリック) するとインストールすることができます。

Lhasa のインストールが完了したら、今度は、Tera Term Pro (`tterm23.zip`) を Lhasa で展開し、インストールします。

一般的な DOS/V パソコンの場合は、特に指定しない限り、`C:¥Program Files¥TTERMPRO` のフォルダにインストールされます。

次に、TTSSH (`ttssh14.zip`) を Lhasa で展開すると、**図 8**のように `Libeay.txt`,`libeay32.dll`, `Readme.txt`,`ttssh.exe`,`TTXSSH.dll` の五つのファイルが生成されます。

図 8. TTSSH のファイル群

TTSSH のインストールは、上記の五つのファイルを Tera Term Pro のフォルダに移動すれば完了です。なお、`Readme.txt` ファイルは、Tera Term Pro のフォルダ内にも同一ファイル名で存在しますので、どちらかのファイル名を変更しておくと良いでしょう。

以上で、インストールは完了です。

4.3　TTSSH での認証方式

まず、TTSSH での認証方式について簡単に紹介します。

認証方式には、次の三つがあります。

- パスワード認証
- RSA 認証
- ホスト RSA 認証付き rhosts 認証

デフォルトでは、パスワード認証を利用するように設定されています。この場合、リモート計算機での認証には通常のログインに使うパスワードを使用します。この方法でも、安全な通信経路による利用が可能です。

ここでは、 RSA 認証を利用する方法について、もう少し詳しく紹介します。

4.3.1 RSA による認証方式

RSA 認証を利用する場合には、「公開鍵」と「秘密鍵」をお使いの PC に用意する必要があります。残念ながら TTSSH には、鍵を生成するツールが含まれていないので、別途入手が必要となります。

近くに利用できる UNIX 環境がある場合には、UNIX 上の ssh-keygen コマンドで、次のようにファイル名やコメントを指定して、鍵を作成することができます。

```
%  ssh-keygen -f pc.key -C my-pc
```

しかし、鍵ファイルを安全に PC に転送できる環境が無い場合は、 PC 上で鍵を作成することをお勧めします。

鍵の生成には、コマンドライン・ベースの SSH ソフトウェア (**ssh-1.2.14-win32bin**) の中にある、鍵生成プログラムを利用します。次の FTP サイトからファイルを入手し、 Lhasa で展開します。

- ssh-1.2.14-win32bin

  ftp://ftp.cs.hut.fi/pub/ssh/contrib/ →

  ssh-1.2.14-win32bin.zip

展開すると、 gmp.dll,ssh.exe,scp.exe, ssh-keygen.exe,zlib.dll の五つのファイルが生成されます。この中の ssh-keygen.exe コマンドで、 RSA 鍵を作成します。

ユーザの RSA 鍵とホスト RSA 鍵は本質的な違いはなく、どちらも ssh-keygen.exe コマンドで作成することができますが、ホスト RSA 鍵のパスフレーズには空白文字列を設定する必要があります。また、 ssh-keygen.exe を利用する場合は、必ず、 “-C” オペランドでコメントを指定しないとエラーとなります。

図 9に、 ssh-keygen.exe を使用してホスト RSA 鍵を生成する手順を示します。

ホスト RSA 鍵を作成する場合は問題がないのですが、筆者の環境では、鍵ファイル名 (host-key) を入力して Enter キーを押すと、一回目のパスフレーズの入力フィールドに勝手に Enter キーが入力されるため、パスフレーズを入力することができません。

ユーザの RSA 鍵を生成する場合は、次のように、 “-f” オプションでファイル名を指定し、 “-N” オプションでパスフレーズを指定して実行してください。

```
ssh-keygen.exe -f identity -N "ssh no passphrase"
```

作成したユーザの「公開鍵 (identity.pub)」と「秘密鍵 (identity)」の RSA 鍵ファイルは、Tera Term Pro のフォルダに移動しておきます。

```
C:¥>ssh-keygen -C my-host-name
Initializing random number generator...
Generating p:  .................++ (distance 334)
Generating q:  ...++ (distance 38)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key ($HOME/.ssh/identity):  host-key
Enter passphrase: 何も入力せずに改行
Enter the same passphrase again: 何も入力せずに改行
Your identification has been saved in host-key.
```

図 9. ssh-keygen コマンドでのホスト RSA 鍵の生成

また、「公開鍵 (identity.pub)」ファイルの中身は、他人に覗き見されても構いませんので、 FTP などの適当な方法で TTSSH を利用して接続するリモート計算機に転送し、~/.ssh/authorized_keys ファイルに追記しておきます。

これで、 RSA 認証を利用するための鍵の準備が完了しました。

4.4 TTSSH の使い方

PC から SSH を利用して、リモート計算機 (sakura) を使うために、 Tera Term Pro と TTSSH をインストールしました。

TTSSH は、 Tera Term Pro の拡張モジュールですので、 SSH を利用する場合は、インストールしたフォルダの ttssh.exe ファイルから実行します。実行すると**図 10**のような画面が表示されます。

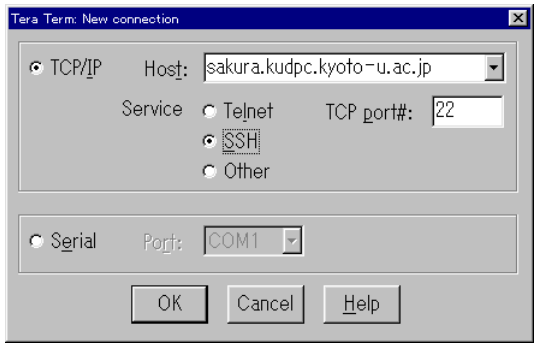

図 10. TTSSH 拡張された Tera Term Pro

通常の ttermpro.exe ファイルから実行した場合との違いは、 Service の欄に SSH が追加されていることで、 SSH を選ぶことにより SSH での利用が可能となります。

また、環境変数 (TERATERM_EXTENSIONS) を設定することにより、 ttermpro.exe ファイルの実行時に拡張モジュールの TTSSH を組み込んで起動することもできます。

Windows95/98 の場合は、 C:¥autoexec.bat ファイルに次の一行を追記します。

```
set TERATERM_EXTENSIONS=1
```

これで、 ttermpro.exe ファイルを直接実行しても拡張モジュールの TTSSH が組み込まれます。

4.4.1 TTSSH の環境設定

使い始める前に、 TTSSH の環境設定を行います。**図 10**の画面は、ここでは、「**Cancel**」をクリックして閉じておきましょう。

Tera Term Pro のメニューバーの「**Setup**」をクリックすると、**図 11**のようなメニューが表示されます。

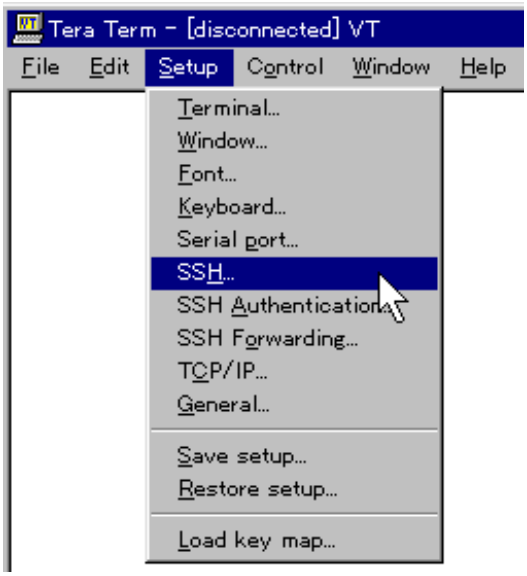

図 11. 「Setup...」メニュー

メニューの「**SSH...**」では、**図 12**のように、暗号方式の選択と、既知ホスト鍵を保存するファイル名を設定します。

暗号方式や既知ホスト鍵を保存するファイル名は、デフォルトの設定のままでも構いません。

メニューの「**SSH Authentication...**」では、**図 13**のように、ユーザ名や認証方式の選択と、RSA 秘密鍵を保存しているファイル名を指定します。

ユーザ名には、接続する SSH が動作しているリモート計算機 (sakura) のログイン名を指定します。**図 13**の例では、 sakura のログイン名を指定しています。

認証方式は利用するに認証方式を選択します。**図 13**の例では、 RSA 認証を利用する設定にしています。 **Private key file** の欄には、先ほど作成した「秘密鍵 (identity)」ファイル名を指定します。

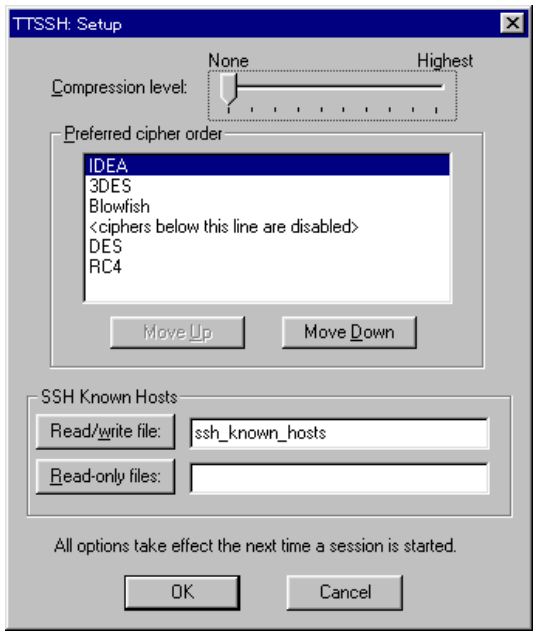

図 12. SSH の Setup ウィンドウ 1

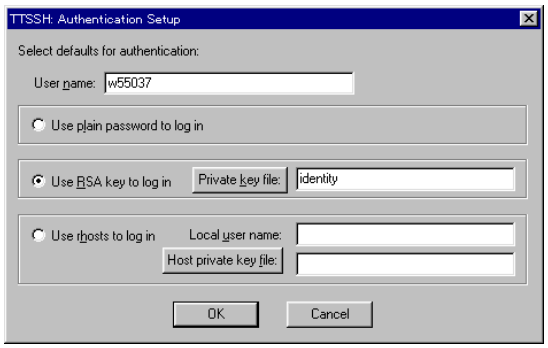

図 13. SSH の Setup ウィンドウ 2

図 12と**図 13**のように設定ができたら、OKキーを押して画面を閉じます。これで、 TTSSH の環境設定は完了です。

忘れないうちに、 Tera Term Pro のメニューバーの「**Setup**」をクリックし、メニューから「**Save setup...**」を選択して環境設定を保存しておきましょう。

4.4.2 TTSSH による接続

Tera Term Pro のメニューバーの「**File**」をクリックし、メニューから「**New connection...**」を選択して画面を開き、**図 10**のよう Service で **SSH** を選択し、 **Host** の欄に、接続するリモート計算機 (sakura.kudpc.kyoto-u.ac.jp) を指定して、OKをクリックします。

初めて接続するリモート計算機 (sakura) の場合には、**図 14**のような画面が表示されます。

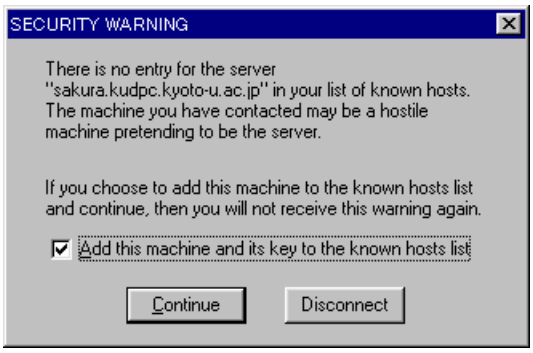

図 14. 既知ホストの確認

チェックボックスにマークして、Continueキーを押すと既知ホスト鍵ファイルにリモート計算機 (sakura) のホスト RSA 鍵が書き込まれます。

後は、設定した正しいパスフレーズを入力して接続が完了すると、安全な通信経路による利用が可能となります。

4.5 TTSSH によるポート・フォワーディング

SSH のポート・フォワーディング機能については、 **3.4.2**で紹介しましたが、 Tera Term Pro 拡張モジュールの TTSSH も、このポート・フォワーディング機能をサポートしています。

この機能を利用するためには、 Tera Term Pro のメニューバーの「**Setup**」をクリックして、メニューの「**SSH Forwarding...**」で設定します。

4.5.1 ポート・フォワーディング機能の応用

ここでは、例としてお使いの PC から、リモート計算機 (sakura) の POP(Post Office Protocol) サービスをポート・フォワーディング機能を使って、安全な通信経路で利用する方法を紹介します。

POP サービスをポート・フォワーディング機能で利用するための概念図を**図 15**に示します。

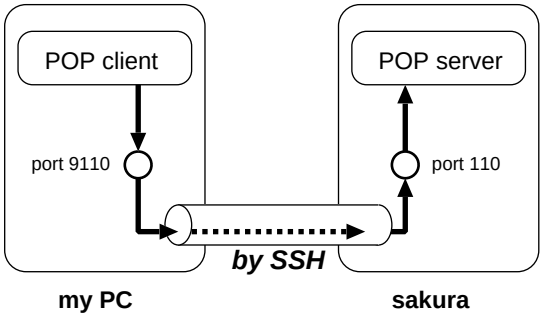

図 15. POP のポート・フォワーディング

図 15では、お使いの PC の 8110 番のポートとリモート計算機 (sakura) の 110 番 (POP サービス) のポートの間を SSH により安全な通信経路を確保しています。

それでは、ポート・フォワーディング機能の設定を行います。TTSSH 拡張モジュールを組み込んだ Tera Term Pro を起動し、メニューバーの「**Setup**」をクリックし、メニューから「**SSH Forwarding...**」を選択すると、**図 16**のような画面が表示されます。

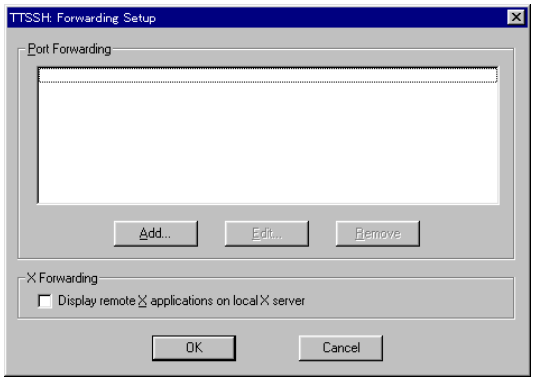

図 16. TTSSH のフォワーディング Setup

図 16の画面で、`Add...` をクリックすると、ポート・フォワーディングするリモート計算機名とポート番号、それを受取るお使いの PC でのポート番号を指定する画面が表示されます。

お使いの PC の 8110 番ポートとリモート計算機 (sakura) の 110 番ポート (POP サービス) の間をポート・フォワーディングするためには、**図 17**の

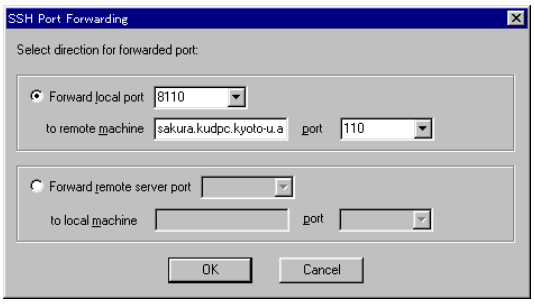

図 17. POP のためのフォワーディングの設定

ように、「**Forward local port**」を選択し欄に、8110を記入し、「**to remote machine**」の欄に、sakura.kudpc.kyoto-u.ac.jpを記入し、「**port**」の欄に、110を記入します。

これで、リモート計算機 (sakura) から POP で受取るメールは、安全な通信経路で利用することができます。今度はリモート計算機 (sakura) の SMTP サーバにメールを送る場合も同じように安全な通信経路を利用するための設定を行います。

図 17の画面は、`OK` をクリックして閉じ、再び、**図 16**の画面で、`Add...` をクリックします。

リモート計算機 (sakura) の SMTP サーバは、25 番ポートでサービスしていますので、このポートとお使いの PC の 8025 番ポートの間をポート・フォワーディングするための設定を行います。

図 18のように、「**Forward local port**」を選択し欄に、8025を記入し、「**to remote machine**」の欄に、sakura.kudpc.kyoto-u.ac.jpを記入し、「**port**」の欄に、25を記入します。

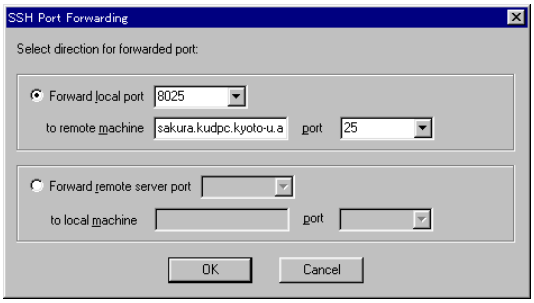

図 18. SMTP のためのフォワーディングの設定

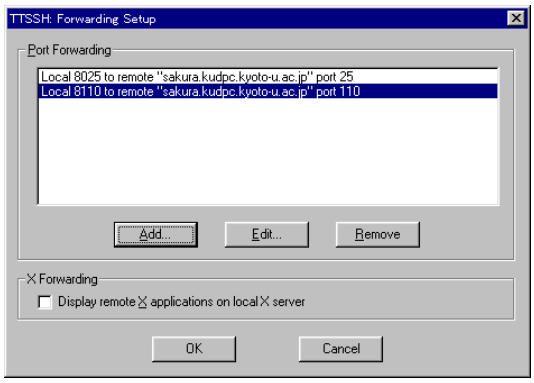

図 19. 設定された二つのフォワーディング

OK をクリックして画面を閉じると、**図 19**のように、メールを受信するための POP(110 番) とメールを送信するための SMTP(25 番) の二つのポート・フォワーディングが設定されたことを確認できたら、OK をクリックします。

これで、 TTSSH での環境設定は完了ですので、忘れないうちに、 Tera Term Pro のメニューバーの「**Setup**」をクリックし、メニューから「**Save setup...**」を選択して環境設定を保存しておきましょう。

これで、ポート・フォワーディング機能を利用して、安全な通信経路でメールを利用する環境が整いました。実際にポート・フォワーディング機能を利用するためには、リモート計算機 (sakura) にログインする必要があります。

4.4.2で紹介した要領でリモート計算機 (sakura) に接続してください。接続が完了した時点から、ポート・フォワーディング機能が有効となっています。また、ポート・フォワーディング機能を利用している間は、ログアウトしないでください。

次にメーラー側の設定を紹介します。ここで例としているメーラーは、 **AL-Mail32** ですが、 POP と SMTP のポート番号が設定できるメーラーであれば、どのようなメーラーでも構いません。

POP と SMTP のサービスをポート・フォワーディング機能により、お使いの PC のポートに設定していますので、**図 20**のように、「サーバ情報」の「**POP3 サーバ名**」と「**SMTP サーバ名**」の欄に、localhostと記入し、**高度な設定** をクリックして、**図 21**のように、「**POP3 ポート番号**」の欄に、8110と「**SMTP ポート番号**」の欄に、8025を記入します。

図 20. AL-Mail32 のオプション画面

これで、メーラーの設定は完了です。OK をクリックして、画面を閉じます。

後は、通常と同じようにメールの送受信を行うことで、ポート・フォワーディング機能を利用して安全な通信経路でメールを利用することができます。

図 21. Al-Mail32 の高度な設定画面

5 おわりに

本センターでは、現在、提供している全ての計算機で SSH をサービスするには至りませんが、今後、順次導入を考えています。

紙面および時間の関係で、十分な解説となりませんでしたが、今後は下記の URL を充実させていく予定ですので、本稿と合わせてご覧ください。

この記事に関して、ご意見・ご質問などございましたら、プログラム相談室までご連絡ください。

http://www.kudpc.kyoto-u.ac.jp/Services/SSH/