sshによる安全かつ簡単な遠隔ログイン方法

池田, 大輔
九州大学情報基盤センター研究部

https://doi.org/10.15017/1470405

出版情報：九州大学情報基盤センター広報 : 全国共同利用版. 1 (3), pp.216-223, 2001-10. 九州大学情報基盤センター
バージョン：
権利関係：

sshによる安全かつ簡単な遠隔ログイン方法

池田大輔*

1　はじめに

前号でrsyncによるディレクトリの同期のとり方を紹介しました [1]。この中で、安全にrsyncを使うためにsshを利用する方法も紹介しました。この記事ではrsyncが主題であったため、詳しいsshの使い方は省きました。そこで、本稿でsshについてより詳細に説明します。

ssh (Secure Shell) を用いると、遠隔地のマシン上でコマンドを実行することができます。sshと同時にインストールされるsloginは、ネットワークを介してコンピュータにログインするプログラムです。遠隔ログインはsshコマンドでも可能です。sshと同時にインストールされるscpを使えば、他のマシンへファイルをコピーすることも可能です。つまり、これらssh関連のコマンドはrlogin, rsh, rcpの代替となります。もちろん単なる代替コマンドではなく、セキュリティが非常に強化されています。

現在、情報基盤センター (以下、センターと呼ぶ。) ではtelnetやrloginによる遠隔ログインを禁止はしていませんが、sshによる方法を薦めています。センターが提供する計算機群は多くのユーザが利用しており、セキュリティに無頓着なユーザは他のユーザに多大な迷惑をかける可能性があるからです。

sshの基本は遠隔ログインなどのネットワークを介した計算機資源の利用であるため、計算機資源を使ってよいユーザかどうかの認証が必要となります。sshの認証には、ユーザ鍵を用いないパスワード認証とユーザ鍵を用いるRSA認証があります[1]。パスワード認証でいうパスワードとは、利用したい計算機のパスワード[2]のことです。この認証による遠隔ログインの仕方は2節で、3節でRSA認証による方法を説明します。

また、前回簡単に説明したssh-agentによる方法も、その仕組みまで含めて4節で説明します。これにより、パスフレーズ[3]の入力をせずに遠隔の計算資源を使えるようになります。この機能により、前回説明したrsyncやcvsなどの利用において、認証にsshを用いることができるようになり、安全にこれらのコマンドが使えるようになります。

*情報基盤センター研究部　mailto:daisuke@cc.kyushu-u.ac.jp

[1]実際には、あらかじめ信用する計算機をファイルに列挙するrhosts認証という認証方法もありますが、安全性に問題があるため説明しません。

[2]UNIX系のOSでは/etc/passwdか/etc/shadowなどに記録されている。

[3]sshにおいて、パスワードや暗証番号と同じ役目を負うものです。

1.1 準備

以下、利用者の手元にある計算機を“local”とし、遠隔の計算機を“remote”と表記します。センターの利用者であれば、研究室の計算機が“local”で、センターの計算機が“remote”になります。

sshを“local”と“remote”間で利用するには、どちらの計算機にも自分のアカウントが用意されている必要があります。さらに“remote”上でsshサーバが動いていて、“local”上にsshクライアントがインストールされている必要があります。sshのインストールについては文献[2]を参照してください。使っているOSによっては、バイナリパッケージが用意されていることもあります。現在、センターにおいてサービスを提供している全ての計算機上でsshサーバが稼働しています。

sshは様々な実装が存在します。本稿ではFreeBSD上のOpenSSH 2.3.0を用いて説明します。

2 基本的な使い方

sshはtelnetやrloginと同様に、“どの計算機”へログインするかを指定します。

```
local% ssh remote↵
user@remote's password:  ↵                          ←パスワードの入力
remote%
```

特に準備をしなければ、上記のように“remote”におけるパスワードを入力する必要があります。パスワードを入力すれば“remote”にログインでき、後はtelnetやrloginと全く同様に使えます。

“local”から初めて“remote”へsshを使ってログインする場合は、パスワードを入力する前に以下の警告がでます。

```
local% ssh remote↵
The authenticity of host 'remote' can't be established.
RSA key fingerprint is XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX.
Are you sure you want to continue connecting (yes/no)?  yes↵  ←yesと入力
Warning:  Permanently added 'azure' (RSA) to the list of known hosts.
user@remote's password:  ↵                          ←パスワードの入力
remote%
```

“Are you sure you want to continue connecting”に対し“yes”を入力してから、パスワードの入力という手順になります。

sshでは、ユーザだけでなく接続する先の計算機(この場合は“remote”)の認証も行なわれます。この認証は3節で説明するRSA認証で行なわれます。詳細は後に述べますが、RSA認証では接続対象(この場合は“remote”)のホスト公開鍵とよばれるものが“local”上に必要です。しかし、初めての接続であり、それがないと警告を発せられます。利用者が“yes”を入力すると公開鍵が記録され、次からはこの警告は表示されなくなります。

“local” と “remote” で、ユーザ名が異なる場合は、remote のユーザ名+“@” をホスト名の前に指定して

```
local% ssh user@remote↵
```

とするか、-l オプションにユーザ名を渡して

```
local% ssh -l user remote↵
```

とします。

ssh の代わりに slogin でも同様に遠隔ログイン可能です。

ssh は rlogin だけでなく rsh を置きかえるものでもあります。よって、遠隔ログインではなく、遠隔ホスト (この場合は “remote”) 上でコマンドを実行させることもできます。例えば、ホスト名を表示させる hostname コマンドを “remote” で実行すると

```
local% ssh remote hostname
remote.cc.kyushu-u.ac.jp
```

となります。

2.1　パスワード認証の仕組み

ssh では、ユーザや計算機ごと、あるいは通信のセッションごとに固有の鍵を用いた認証を利用できます。鍵は単なる数値です。ホスト鍵は ssh のインストール時に生成され、セッション鍵は通信を行う時に自動的に生成されます。一方、ユーザ鍵は利用するユーザが明示的に鍵を作らない限り存在しません[4]。

ここまでで説明したパスワード認証による遠隔ログインでは、ユーザ鍵は利用していません。しかし、セッション鍵は存在しており、このセッション鍵を用いて、ユーザのパスワードや通信の内容を暗号化します。鍵を用いた暗号化の仕組みは 3.3 節で説明します。

以上の説明から、パスワード認証においては、入力するパスワードは暗号化されて “remote” 側に渡されることがわかります。暗号化されているので、telnet や rsh と比較すると、安全性は高いといえますが、3 節で説明する RSA 認証では、パスフレーズそのものがネットワーク上を流れません。また、仮にパスフレーズを他人に知られたとしても、ユーザ鍵[5]を格納したファイルの保全が十分であれば、問題ありません。そのため、パスワード認証と比較すると RSA 認証の安全性は高いといえます。

以降で説明する RSA 認証は、本節で説明した方法よりさらに安全度は高く、しかも ssh-agent と同時に利用することでより簡単に使えます。ただ、最初の設定が面倒に感じられるかもしれません。

[4]鍵の作り方は 3.1 節を参照してください。
[5]正確にはユーザの秘密鍵

RSA認証が面倒な場合でも、telnetやrloginを使うのではなく、ここまでで説明したパスワード認証でsshやsloginを使うようにしてください。ここまでの説明で分かるように、基本的な使い方であれば、通常のtelnetやrloginとほとんど同じであり、しかも、これらのコマンドと比較して安全性は非常に高くなります。

3 RSA認証

2節で説明した方法はパスワード認証を使った遠隔ログインです。パスワード認証で入力するパスワードは、接続先計算機のパスワードです。しかし、暗号化されるとはいえ、パスワードがネットワーク上を流れますし、なにより利用のたびにパスワードを入力するのは面倒です。

一方、RSA認証ではパスワードは使わず、パスフレーズとユーザ鍵を用います。パスフレーズは、暗号化したユーザ鍵を元に戻すために必要な文字列です。

3.1 鍵の生成と登録

sshをインストールした際に、鍵を作るためのコマンドssh-keygenもインストールされます。鍵を作るにはssh-keygenを引数なしでを実行します[6]。

```
local% ssh-keygen↵
Generating RSA keys:  Key generation complete.
Enter file in which to save the key (/xxxx/.ssh/identity):↵←鍵ファイル置場
Enter passphrase:  ↵                                    ←パスフレーズの入力
Enter the same passphrase again:  ↵                     ←再入力
```

鍵ファイル置場を尋ねられるところは、そのままリターンキーを押してよいでしょう。別のファイルに鍵を保存したい場合は、オプションで-f filenameとします。次に同じパスフレーズを二度正しく入力します。

通常のパスワードとは違い、パスフレーズの長さに制限はありません。あまり長いとssh利用時に毎回パスフレーズを入力するのが面倒だと思われるかもしれませんが、パスフレーズを入力を"local"にログイン後に1回のみにする方法も説明します(4節)。この方法は、安全性を犠牲にするものではなく、通常のRSAと同じ程度の安全性を有します。よって、充分に長いパスフレーズを利用することをお勧めします。

付属のマニュアルによると、10〜30文字程度の長さがよいとされています。また、英語の平凡な文章は情報量が小さく推測しやすいので避けるべきとあります。

筆者は日本語の適当な長さの文章を、文節ごとに空白で区切り、ローマ字にしたものをパスフレーズに使っています。日本語の文章ですので、覚えるのが非常に簡単であるということ、また、非日本語圏の人間には推測しにくいことなどの利点があると思います。

パスフレーズを後で変更するには-pオプションを使います。パスフレーズを忘れた場合は、新に鍵を作りなおすしかありません。

[6]表示されるメッセージはsshのバージョンによって異なるかもしれません。

鍵ファイルは2つ生成され、それぞれ秘密鍵と公開鍵と呼ばれます。鍵ファイル置場として指定したファイルが秘密鍵で、これに.pubをつけたファイルが公開鍵です。公開鍵はその名の通り公開してよい性質のものですが、秘密鍵は誰にもコピーされてはいけません。秘密鍵の安全には充分気をつけてください。

この公開鍵を"remote"の$HOME/.ssh/authorized_keysというファイルに追加し、鍵に関する設定は終了です。"remote"上に$HOME/.sshがない場合は、自分以外に対するすべて(読み、書き、実行)の権限を落として、

```
remote% cd
remote% mkdir .ssh
remote% chmod 700 .ssh
```

としてください。$HOME/.ssh/authorized_keysがない場合は、新規に作成してください。

公開鍵をauthorized_keysに追加する作業はログインする計算機の数だけ行なう必要があります。簡単にすませるには、以下のようにsshで接続先のコマンドを実行するとよいでしょう。

```
local% cd
local% cat .ssh/identity.pub | ssh remote                    ←次の行へ継続
'mkdir -p .ssh; chmod 700 .ssh; cat >> .ssh/authorized_keys'
```

これにより"local"上の公開鍵が、"remote"上の$HOME/.ssh/authorized_keysの末尾に追加されます。

すでに接続先の計算機上にディレクトリ$HOME/.sshがある場合は、

```
local% cd
local% cat .ssh/identity.pub | ssh remote 'cat >> .ssh/authorized_keys'
```

でよいでしょう。

3.2　利用

ここまでの準備が終われば、2節と同様にsshを起動すれば、RSA認証による遠隔ログインができます。

```
local% ssh remote↵
Enter passphrase for RSA key 'user@local':  ↵          ←パスフレーズの入力
remote%
```

接続先(この場合は"remote")の$HOME/.ssh/authorized_keysに"local"の公開鍵が存在すれば、このように自動的にRSA認証となり、パスフレーズを尋ねます。正しくパスフレーズを入力すると、ログイン完了です。

パスフレーズを間違えた場合は、自動的にパスワード認証となり[7]、パスワードを聞かれます。

3.3 RSA認証の仕組み

RSA認証の仕組みは、次節で述べるパスフレーズ入力を省略する仕組みとも深い関係があります。この節では、この仕組みを簡単に説明したいと思います。

RSA認証では、その名前の由来となるRSA公開鍵暗号方式を用いて安全性を確保します。この暗号方式では、3.1節で述べたように、秘密鍵と公開鍵と呼ばれる、二つの異なる鍵を用います。ペアとなっているこの二つの鍵の一方で暗号化したものは、もう片方の鍵でのみ復号化できます。

また、秘密鍵からは公開鍵は容易に求めることができますが、公開鍵からは秘密鍵は容易に分からないようになっています。よって、公開鍵は広く公開して構いません。

この性質を利用して、通信の暗号化や電子署名が実現できます。例えば、送信相手の公開鍵で暗号化したものを送れば、受けとった相手は自分の秘密鍵で復号化でき、かつ、それ以外の人は秘密鍵がないので、復号化できません。これで暗号化したメッセージを送ることができます。

自分の秘密鍵で暗号化したものを送れば、受けとった相手は公開されてる公開鍵を使って復号化できます。公開鍵によって復号化できたのだから、暗号化したのは、この公開鍵のペアとなる秘密鍵を持つ人となり、署名の役割をはたすことになります。

sshでは、このRSA公開鍵暗号方式を以下のように使います。"local"から"remote"へログインする時の、ユーザの認証を行なう場合を考えます。

"local"からユーザ名とこのユーザの公開鍵を"remote"へ送ります。これらは当然ネットワーク上を流れますが、どちらも公開してよい性質のものです。

次にサーバ側である"remote"では、このユーザの公開鍵が$HOME/.ssh/authorized_keysに登録してあるかどうか調べます。登録してあれば、この公開鍵を使ってランダムに生成した情報(チャレンジと呼びます。)を暗号化し、クライアントに送り返します。

この暗号化されたチャレンジは、対応する秘密鍵でしか復号化できないことに注意してください。また、仮になんらかの方法で復号化できたとしても、ランダムに生成した情報であり、問題ありません。

クライアント側である"local"で、送られたきたチャレンジを秘密鍵を使って復号化します。秘密鍵を開くときにパスフレーズが必要になります。

復号化したものをサーバに送り返し、サーバでは送ったチャレンジと送り返されたチャンレンジが一致するかどうかを調べ、一致すれば、認証が完了します。

ここで、注意すべきは、ネットワークを流れるのは公開鍵、ユーザ名、それにランダムに生成した情報のみで、パスフレーズや秘密鍵は一切流れません。

sshに-vオプションをつけて実行すると、認証の手続きを詳細に表示してくれます。説明した認証の仕組みを確認するのに役立つと思います。また、鍵の生成や登録はきちんとしたつもりでも、ログインに失敗するなどした場合は、どこで認証が失敗しているのか確認できます。

[7]パスワード認証を禁止することもできますが、センターでは禁止していません。

4　パスフレーズ入力の省略

RSA 認証において、特に準備をしなければ 3.2 節の例のようにログインのたびにパスフレーズを入力する必要があります。パスフレーズは長さの制限がないため、長いものを選択すると入力ミスが起こりやすくなりますし、パスワードよりも入力の手間が面倒です。

そこで、"local" にログインした後、パスフレーズを一度入力すれば、以後はパスフレーズ入力なしで ssh が利用できる ssh-agent を利用します。

ssh-agent はユーザの秘密鍵を管理するコマンドです。秘密鍵を ssh-agent の管理下に登録するためには ssh-add コマンドを使います。ユーザが秘密鍵の正当な保有者かどうかは、ssh-add による登録時に、パスフレーズを入力することで確認されます。ssh-agent の管理下にある秘密鍵はメモリ上に保持されます。ssh-agent が起動されていると、秘密鍵が必要な ssh や scp を使うときに、ssh-agent がこれらのコマンドに秘密鍵を渡します。

ssh-agent を起動すると、環境変数 SSH_AGENT_PID と SSH_AUTH_SOCK が定義されます。前者は ssh-agent のプロセス ID であり、後者は UNIX ドメインソケットです。

ssh や scp などは、この環境変数が定義されているかどうかで、ssh-agent が起動されているかどうかを判断します。定義されている場合は、UNIX ドメインソケットを通じてパスフレーズを問い合わせます。

ssh-agent の引数にコマンドを与えて起動すると、このコマンドの子プロセスに環境変数が引き継がれます。そのため、ssh-agent の起動は X Window を起動するときか、ログイン直後に起動するのがよいでしょう。

X Window を利用している場合は

```
local% ssh-agent startx↵
```

とします。startx は X Window を起動する標準のスクリプトであり、自作のスクリプトを指定しても構いません。X Window を利用しない場合は

```
local% ssh-agent tcsh↵
```

などとして、新たにシェルを起動します。こうすることで、これ以降のプロセスに環境変数が継承されていきます。

ssh-agent を起動した後

```
local% ssh-add↵
Need passphrase for /home/daisuke/.ssh/identity
Enter passphrase for username@local:  ↵                    ←パスフレーズ
```

とします。

これにより、鍵が ssh-agent の管理下におかれます。現在の管理下にある鍵を知るには

```
local% ssh-add -l↵
```

としてください。

これ以降、"remote" との間で ssh 関連のコマンドを用いる場合は、パスフレーズの入力は不要です。さらに、接続先の authorized_keys に "local" で作成した公開鍵が登録されている計算機との間の ssh 接続でも、パスフレーズが不要になります。

もしGUI形式のログイン画面になっている場合 (xdm などが起動している場合) は、startx などのXサーバを起動するコマンドは使えません。この場合は$HOME/.xsession の先頭行を

```
#!/usr/local/bin/ssh-agent /bin/sh
```

などと変えてもよいでしょう。ディレクトリは ssh-agent がインストールされたところに変更してください。

3.3 節で説明したように、ssh-agent を使わない RSA 認証では、パスフレーズは秘密鍵を開くためだけに使われ、ネットワークを流れません。同様に、ssh-agent を使う場合でも、秘密鍵を開くためだけに使われ、パスフレーズがネットワークを流れることはありません。つまり、安全性は通常の RSA 認証と同程度であると言えます。

5 おわりに

本稿では説明しませんでしたが、ssh にはポート転送という仕組みもあります。ポート転送により、安全でない通信も ssh が提供する安全な通信路を経由してやりとりすることができます。文献 [3] には、Mew や Netscape Messanger 上の POP を安全にポート転送する方法が詳しく説明されています。

参考文献

[1] 池田 大輔 プログラム開発のための補助ツール (1)–rsync– 九州大学情報基盤センター広報 Vol. 1, No. 2, pp.104–1110, 2001.

[2] 伊東 栄典、SSH: Secure Shell ～おでかけ前に鍵かけて～、九州大学大型計算機センター広報 Vol. 32, No. 2, pp.76–89, 1999. (http://www.cc.kyushu-u.ac.jp/RD/itou/koho/1999.vol.32.no.2/ssh1.html/ssh1.html)

[3] 伊東 栄典、池田 大輔、SSH: Secure Shell (2) ～小荷物を秘かに港で横流し～、九州大学大型計算機センター Vol. 32, No. 3, pp.127–138, 1999. (http://www.cc.kyushu-u.ac.jp/RD/itou/koho/1999.vol.32.no.3/ssh2.html/ssh2.html)