

公開鍵暗号のすすめ

~SSHを便利&セキュアにつかおう~

2020/0903追記

秋山 正和

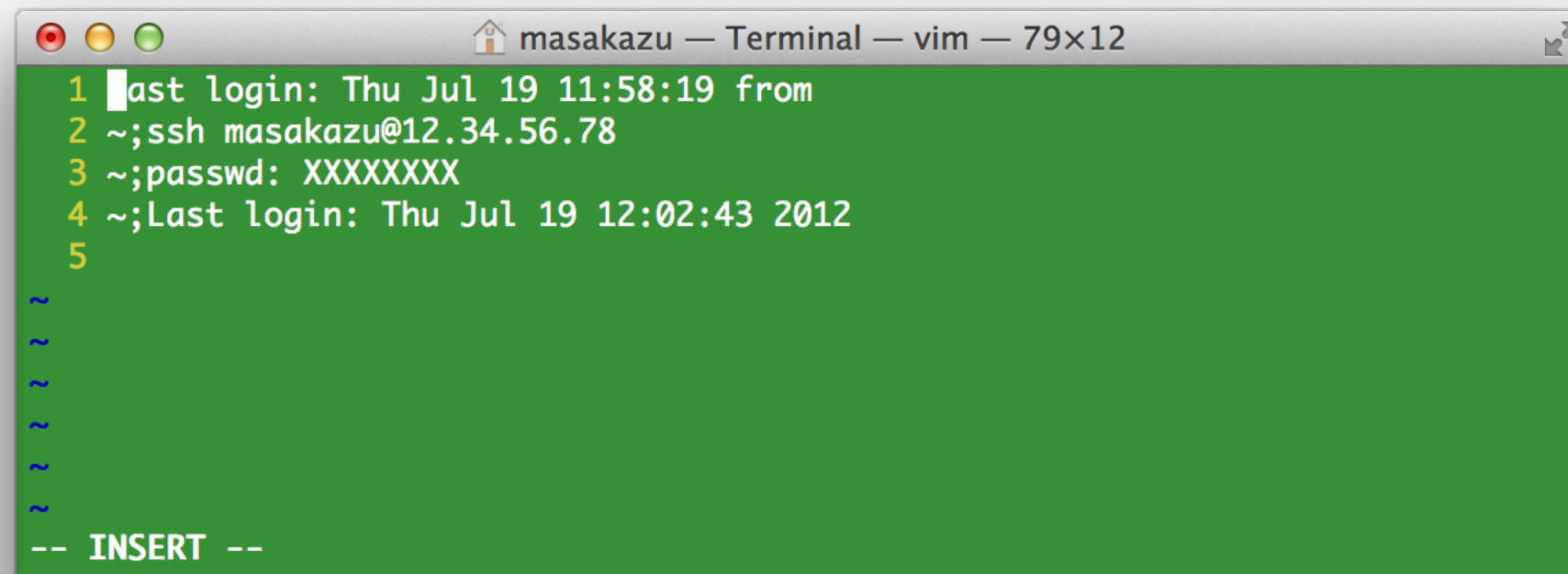
明治大学 MIMS

はじめに

~SSHでのパスワード認証とは~

アカウント: masakazu

クライアント(あなたのPC)

A terminal window titled 'masakazu - Terminal - vim - 79x12' with a green background. It shows a sequence of commands and outputs: 1. A cursor at the prompt '~'. 2. Command: '~;ssh masakazu@12.34.56.78'. 3. Command: '~;passwd: XXXXXXXX'. 4. Output: '~;Last login: Thu Jul 19 12:02:43 2012'. 5. A cursor at the prompt '~'. Below the prompt are several tilde characters '~'. At the bottom, it says '-- INSERT --'.

```
1 | last login: Thu Jul 19 11:58:19 from
2 | ~;ssh masakazu@12.34.56.78
3 | ~;passwd: XXXXXXXX
4 | ~;Last login: Thu Jul 19 12:02:43 2012
5 |
~
~
~
~
-- INSERT --
```

アカウント: masakazu

パスワード: XXXXXX

ipアドレス: 12.34.56.78

SSHサーバー(入りたいPC)

ログインする際, 上記のように打つ
と
パスワードが暗号化されサーバーに送
られる. 認証されればログインするこ
とができる.

SSHでのパスワード認証の問題点1

アカウント: masakazu

パスワード: XXXXX

ipアドレス: 12.34.56.78

SSHサーバー(入りたいPC)

サーバーはSSHパスワード認証のために常にパスワードが正しいか否かを精査する必要がある。

悪意のあるものがサーバーに対して攻撃を仕掛ける場合がある。その際、攻撃者はアカウント名とパスワードがセットになった対をサーバーに対して複数回（1~1000回/秒）送りつける。これはサーバーに大変な負荷をかけることになる。

SSHでのパスワード認証の問題点2

アカウント: masakazu

パスワード: XXXXX

ipアドレス: 12.34.56.78

SSHサーバー(入りたいPC)

アカウントとパスワードが合致する可能性は低いだろうか？

日本人によくある名前（1000個ぐらい）と4文字以下の26文字のアルファベットの組み合わせの場合、一秒に1000回攻撃すれば53日でパスワードが解かれる計算になる。また、一般に攻撃者はよく使われる名前とパスワードの対を辞書式に持っており、効率的に攻撃をしかけてくる。

そこで . . . 公開鍵暗号を使おう！

整数同士の掛け算は容易であるが、素因数分解は困難であることが知られている。その困難さに立脚して開発された暗号系(RSA暗号)をSSHでは使うことができる。

(<http://ja.wikipedia.org/wiki/%E5%85%AC%E9%96%8B%E9%8D%B5%E6%9A%97%E5%8F%B7>)

公開鍵暗号方式では公開鍵と秘密鍵がポイントになる。

公開鍵暗号を使おう！

利点：

1. サーバーを公開鍵暗号のみ受け付けるモードにすると、パスワード総当たり攻撃を受けなくなる。したがってサーバーの負荷は激減する。
2. 暗号化の強度が選べる。暗号化強度をデフォルトの2048bitからさらに高めることができる。（ちなみに最低でも768bitなので安心（300bitぐらいたと効果的な素因数分解アルゴリズムならば解かれてしまう危険性があった。））
3. **パスワードを毎回打たなくても良い！**

欠点：

1. 最初の一回だけ設定が面倒かもしれない。
2. 秘密鍵を盗まれると誰でもサーバーにログインできちゃう。

公開鍵暗号を使おう

I. サーバーで公開鍵と秘密鍵をつくる

ここからは実際の状況(ここではMacを使うがLinux系でもUnix系でも基本的には同じ)を仮定しながら話をすすめる。

サーバーにログインしssh-keygenコマンドで公開鍵と秘密鍵のペアを作る。手順

- ① サーバーにSSHでログインする。
- ② 初めてログインする場合は出るメッセージ。yesとする。
- ③ R.S.A.暗号を用い、鍵のbit数を2048にする。(通常この設定で十分。)
- ④ 鍵を保存する場所。特に問題はないのでエンターを押す。
- ⑤ パスフレーズ^{*}を打つ。5文字以上

```
masakazu — Terminal — ssh — 87x31
Last login: Wed Aug  1 15:58:34 on ttys001
~;ssh masakazu@133.50.104.2
The authenticity of host '133.50.104.2 (133.50.104.2)' can't be established.
RSA key fingerprint is 3c:d5:d0:02:6d:d4:64:b9:f9:c1:11:7a:d8:f8:b6:d3.
Are you sure you want to continue connecting (yes/no)? yes ②
Warning: Permanently added '133.50.104.2' (RSA) to the list of known hosts.
Password:XXXXXX
Last login: Wed Aug  1 15:59:02 2012
~;ssh-keygen -t rsa -b 2048 -m PEM
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/masakazu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): #####
Enter same passphrase again: #####
Your identification has been saved in /Users/masakazu/.ssh/id_rsa.
Your public key has been saved in /Users/masakazu/.ssh/id_rsa.pub.
The key fingerprint is:
21:2c:f8:61:54:33:2c:3a:fb:e5:e3:38:17:5c:7d:6b masakazu@MacMiniServer.local
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .0+             |
|     0...0            |
|    ..+.0 ..         |
|   00 0  .... .      |
|    0.. .S   . .     |
|     .  +     E       |
|    . 0 . .          |
|    0.+              |
|   .+..              |
+-----+ ⑧
~;
```


公開鍵暗号を使おう

I. サーバーで公開鍵と秘密鍵をつくる

⑥ ~/.ssh/にid_rsaという秘密鍵が作成された。

⑦ ~/.ssh/にid_rsa.pubという公開鍵が作成された。

⑧ 特段、このメッセージは気にしなくても良い。

パスフレーズは設定しなくても(単にエンターを押せば設定されない)いいが秘密鍵を誰かに盗まれた時、ちょっと安心度が増すので設定することをおすすめする。

```
masakazu — Terminal — ssh — 87x31
Last login: Wed Aug  1 15:58:34 on ttys001
~;ssh masakazu@133.50.104.2
① The authenticity of host '133.50.104.2 (133.50.104.2)' can't be established.
RSA key fingerprint is 3c:d5:d0:02:6d:d4:64:b9:f9:c1:11:7a:d8:f8:b6:d3.
Are you sure you want to continue connecting (yes/no)? yes ②
Warning: Permanently added '133.50.104.2' (RSA) to the list of known hosts.
Password:XXXXXX
Last login: Wed Aug  1 15:59:02 2012
~;ssh-keygen -t rsa -b 2048 -m PEM
③ Generating public/private rsa key pair.
④ Enter file in which to save the key (/Users/masakazu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): #####
⑤ Enter same passphrase again: #####
Your identification has been saved in /Users/masakazu/.ssh/id_rsa. ⑥
Your public key has been saved in /Users/masakazu/.ssh/id_rsa.pub. ⑦
The key fingerprint is:
21:2c:f8:61:54:33:2c:3a:fb:e5:e3:38:17:5c:7d:6b masakazu@MacMiniServer.local
The key's randomart image is:
+---[ RSA 2048]-----+
|      .0+             |
|     0...0            |
|    ..+.0 ..         |
|   00 0  ....  .     |
|    0.. .S  . .      |
|     .  +   E         |
|    . 0 .  .         |
|     0.+             |
|    .+..             |
+-----+
~;
```


公開鍵暗号を使おう

II. サーバーで公開鍵の設定をする.

- ① “authorized_keys”という名前の空のファイルをつくる。(あれば作らなくて良い.)
- ② 権限を600(あなたはread+writeできるが他の人は見ることも出来ない)に設定する.
- ③ 作成した公開鍵を authorized_keysに登録する.
- ④ もう公開鍵はいらないので消す。(消さなくてもいい)
- ⑤ ちゃんと出来てるか確認.

以上



```
f72589212@wz10004:~/ssh — ssh — 53x11
.ssh;ls
id_rsa id_rsa.pub
① .ssh;touch authorized_keys
② .ssh;chmod 600 authorized_keys
③ .ssh;cat id_rsa.pub >> authorized_keys
④ .ssh;rm id_rsa.pub
.ssh;ls
⑤ authorized_keys id_rsa
.ssh;
```

公開鍵暗号を使おう

III. クライアントで秘密鍵の設定をする.

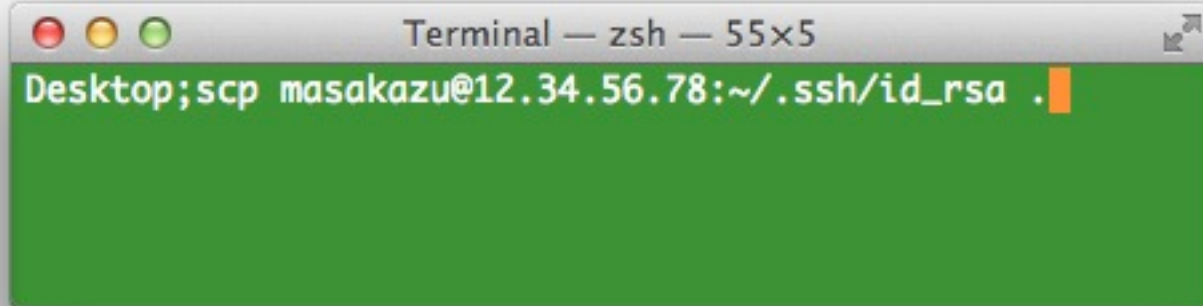
① 秘密鍵であるid_rsaファイルを安全な経路を通してクライアントのpc(ここでは~/Desktopに)に移す. 例えばscp, sftpなどの手段でOK. 管理体制の整ったUSBメモリーorCD-Rに入れて持ち運んでもOK. (ただし絶対にこのファイルを他人に見られてはいけない!!)

② 秘密鍵が増えた時のことも考えて名前をわかりやすいものに変更する.

③ ホームディレクトリの下での.sshフォルダにファイルに移す. (.sshフォルダがなければ作成)

④ ちゃんとあるか確認.

①

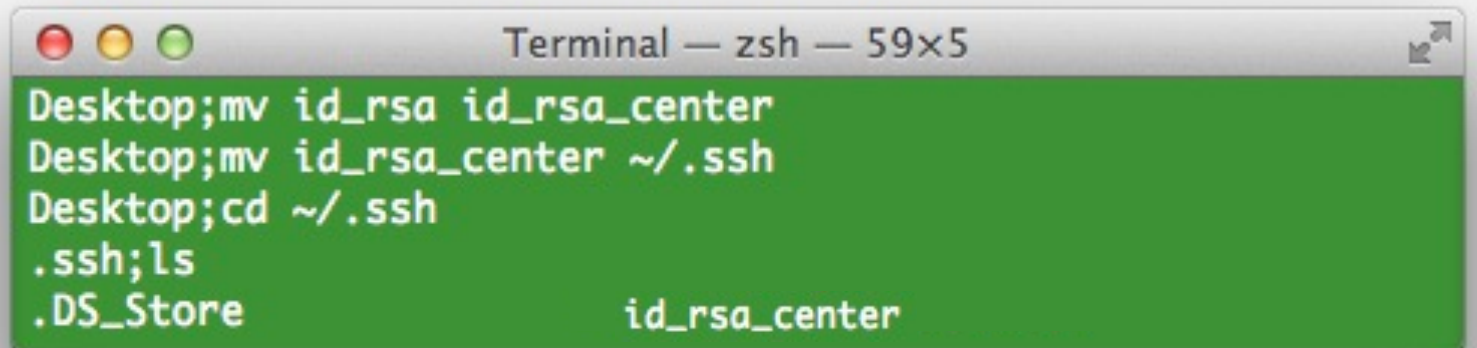


```
Terminal — zsh — 55x5
Desktop;scp masakazu@12.34.56.78:~/id_rsa .
```

②

③

④



```
Terminal — zsh — 59x5
Desktop;mv id_rsa id_rsa_center
Desktop;mv id_rsa_center ~/.ssh
Desktop;cd ~/.ssh
.ssh;ls
.DS_Store          id_rsa_center
```

公開鍵暗号を使おう

IV. サーバーにログインする.

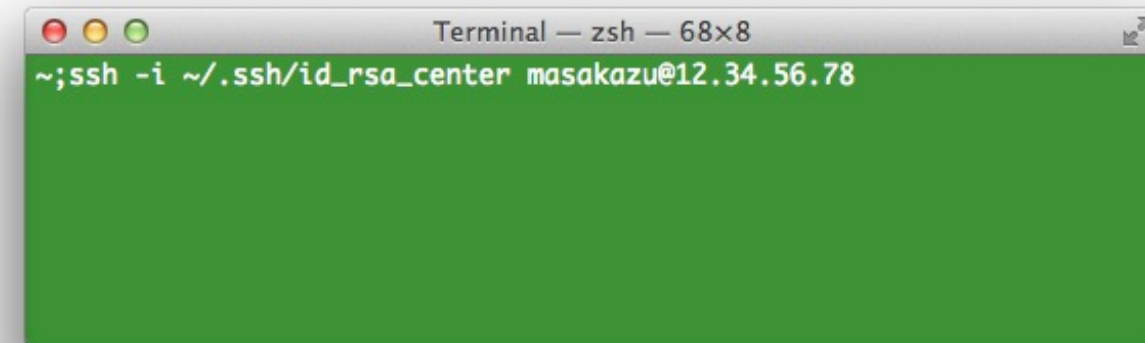
ここまでの設定がうまくいっていれば、あとはログインするだけ.

① 鍵を指定 (id_rsa_center) してサーバー (12.34.56.78)にログインする.

② Macの人は図のような画面が出てくるので、I.-⑤で設定したパスフレーズを打つ. キーチェーンに保存させても良い.

②' Linux or Un*xの人は①のように行くとエラーとなる場合がある. その場合は図のように打ち秘密鍵をpcに登録させる.

①

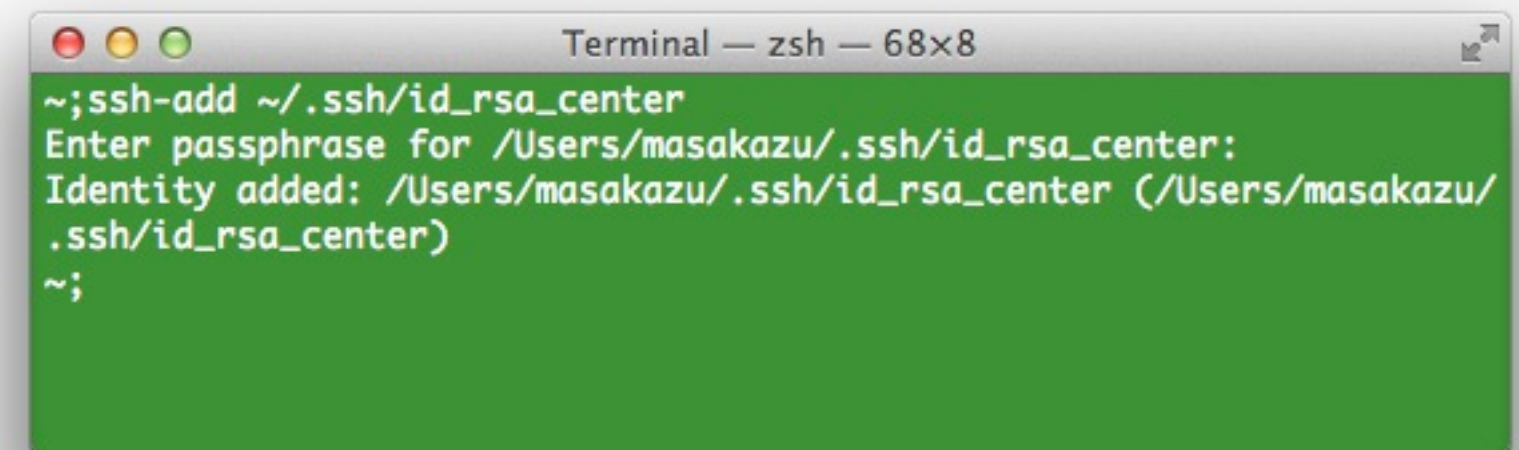


```
Terminal — zsh — 68x8
~;ssh -i ~/.ssh/id_rsa_center masakazu@12.34.56.78
```

②



②'



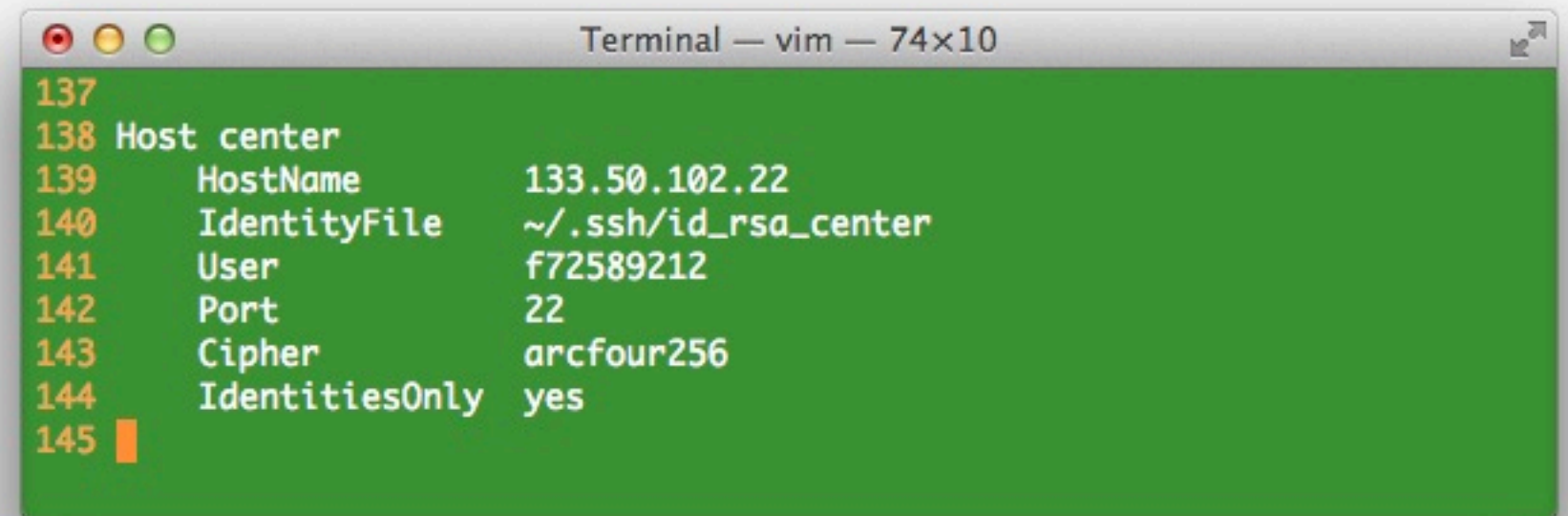
```
Terminal — zsh — 68x8
~;ssh-add ~/.ssh/id_rsa_center
Enter passphrase for /Users/masakazu/.ssh/id_rsa_center:
Identity added: /Users/masakazu/.ssh/id_rsa_center (/Users/masakazu/.ssh/id_rsa_center)
~;
```


公開鍵暗号を使おう

V. サーバーにログインする2.

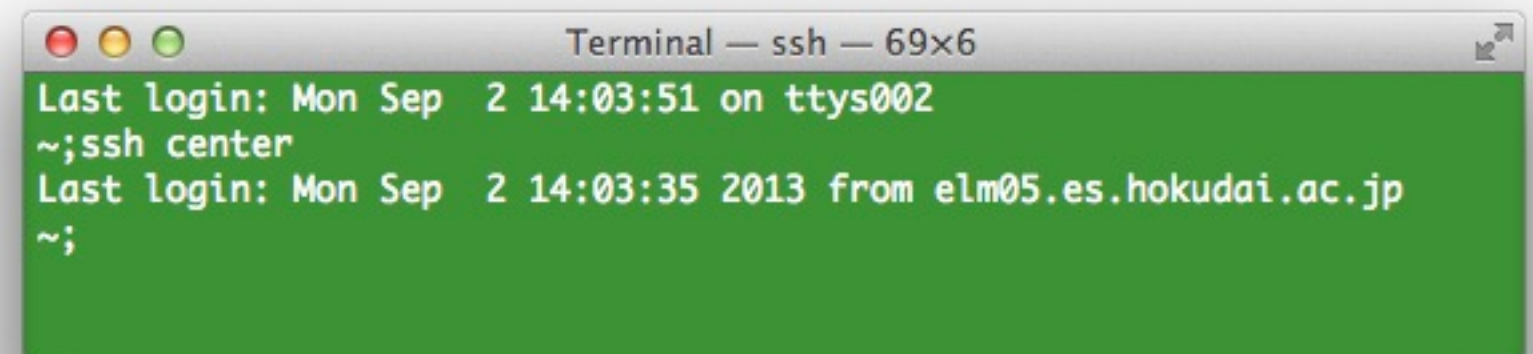
pcをログアウトするまでは、秘密鍵の情報がメモリ空間に蓄えられるので、いちいちパスワードを打たなくてもログインできるはずだ。基本的には以上で終了である。

さらに便利に使うためには、sshフォルダの中にconfigファイルを作成しそこへ右のように書き込む。(行数は関係ないのであしからず)ここではユーザ名、Port番号そして暗号化方式なども詳しく設定できる。



```
Terminal — vim — 74x10
137
138 Host center
139     HostName      133.50.102.22
140     IdentityFile  ~/.ssh/id_rsa_center
141     User          f72589212
142     Port          22
143     Cipher        arcfour256
144     IdentitiesOnly  yes
145
```

次回以降は単に“ssh center”と打つだけで、ログインすることができる。



```
Terminal — ssh — 69x6
Last login: Mon Sep  2 14:03:51 on ttys002
~;ssh center
Last login: Mon Sep  2 14:03:35 2013 from elm05.es.hokudai.ac.jp
~;
```

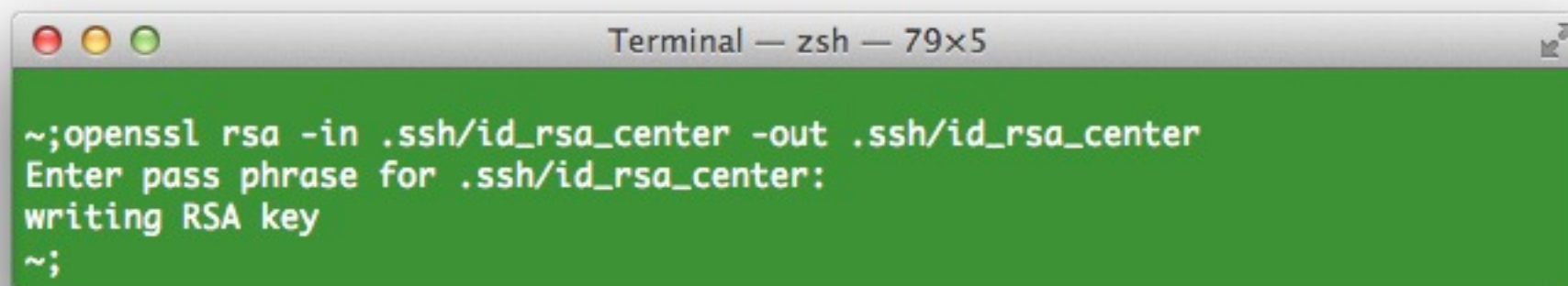

Chips

ssh-addはsshセッションの際、秘密鍵をメモリ空間に蓄えることによって、2回目にはパスフレーズを打たなくても良いように設定してくれる。

Macの人はssh-addの代わりにキーチェーンが働くことによって、上記のことを行ってくれるので便利である。しかも、ログアウトしても次回ログインする際に自動的にssh-addを行う。

ただし、configファイルにIdentitiesOnlyの設定(これがなんのことなのかわからない人はこのChipsは無視して良い)を書き込むと、毎回キーチェーンがパスフレーズを聞いてくる。

それを回避するために、次のように打ち込み秘密鍵ファイルを登録する方法がある。こちらだと上記のような設定でも問題なく動くが、鍵情報の暗号化を解除したまま保存していることになるので、若干セキュリティは落ちる。(20200903追記)



```
Terminal — zsh — 79x5
~;openssl rsa -in .ssh/id_rsa_center -out .ssh/id_rsa_center
Enter pass phrase for .ssh/id_rsa_center:
writing RSA key
~;
```

Chipsの参照記事

3. IdentitiesOnly を設定した際に、毎回パスフレーズを聴かれない様にする
上記のオプションを打つのが面倒なので、config の設定を行うと、
パスフレーズ設定をしている ssh 鍵が、Macのキーチェーンを参照しなくなり、
毎回キーチェーンが起動するようになる。



※ 「パスワードをキーチェーンに保存」にチェックを入れても、効かない。
キーチェーンを使わないで済む様に、鍵ファイルにパスフレーズを書き込む

```
$ openssl rsa -in org.key -out new.key  
# <パスフレーズの入力
```

(これは Apache で SSL の自動起動設定をする時に使うヤツ)

(当然、鍵のパーミッションは400で)

ここまで設定すると、キー登録しているホストへは一発で接続でき、普通にユーザー名とパスワードでログインしたいサーバーへは

```
$ ssh user@192.168.1.100  
user@192.168.1.100s password:
```

と普通にログインできるようになる。

めでたし。

<http://d.hatena.ne.jp/ozuma/20130510/1368114329>