

Assignment 0x01

Frank Kaiser – 1742945, Jan Martin – 1796943

November 19, 2017

1 Task 1

Snow is white.

```

[aro@arch-aro ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aro/.ssh/id_rsa): /home/aro/.ssh/id_rsa_psi
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aro/.ssh/id_rsa_psi.
Your public key has been saved in /home/aro/.ssh/id_rsa_psi.pub.
The key fingerprint is:
SHA256:E/pB4YMB3Mhgfoa0LrcTl1MzLw23/T/LMFPctS9sgDo aro@arch-aro
The key's randomart image is:
+---[RSA 2048]---+
|.O+..+|.
|..B..+|.
|..=..+|.
|..O..O+O|.
|...S..+|.
|..*..O+..+|.
|..*..B..+|.
|..*..+|.
|..+..+|.
+---[SHA256]---+

```

Figure 1: ssh-keygen

```

[aro@arch-aro ~]$ ssh-copy-id kaiser@88.99.184.129
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 2 key(s) remain to be installed -- if you are prompted now it is to install the new keys
kaiser@88.99.184.129's password:

Number of key(s) added: 2

Now try logging into the machine, with: "ssh 'kaiser@88.99.184.129'"
and check to make sure that only the key(s) you wanted were added.

```

Figure 2: save public key on server

2 Task 2 - Public-Key Authentication in SSH

2.1 GNU/Linux

To generate the ssh-key pair I used the command `ssh-keygen` which generates by default a 2048 bit long rsa key.

To copy the key on the server `ssh-copy-id user@88.99.184.129` was used. Since I already had a public key for another server and the exercise was to create one, both keys got uploaded to the server as seen in the picture 2.

The following log-in worked without the user password for the server. Only the optional password for the private ssh key was required. The keys on the server are stored in `~/.ssh/authorized_keys`. See figure 3

```

[aro@arch-aro ~]$ ssh kaiser@88.99.184.129
Linux psi-introsp 4.9.0-4-amd64 #1 SMP Debian 4.9.51-1 (2017-09-28) x86_64

Last login: Fri Nov 17 20:32:43 2017 from 188.194.245.11
kaiser@psi-introsp:~$ cd .ssh/
kaiser@psi-introsp:~/.ssh$ ls
authorized_keys  known_hosts
kaiser@psi-introsp:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCOuXvB7Y614gRvtXefF15YnBgILx2XYV5BR346fvmPGrKsrla0Z5MAK5VQJUsmAXTe0IIImyBy01VOT
0IMKrsPervscl+HKKURMEfAlG6q1d8f6CwLW4s9y5G0DX/DE6d2Z/7B1d4LopTm13BT+00veLw6fKufqmwKv5GnlZV/Lhr2UPJhD-clutq8InWNTp
E+770oFhmCSaye4+BCGWTf99waFqB51806wsgRe51kMor05lmsGusfZ4M1f6EaSoz/N756F2rtkDh70JY1lyh0b+aduN/rtoDhexDq+PvIEDTpkAa
+B057ECTHkVbuUnZeQsGLxLHN+3LTmVvgmQ8Pask5RVze9HJMVf3eVyp8V2LYnQD/Tv73KUSZVZc327TFN56LKc693svHFYkTctZmc9l+5LN/9uaw
UDFNSxLwTata29G0vKAK2dX0mKUBSQM/LH8NorWYw7u3L4XCNzU57Fhcwktf3lctL14mCwmN+Xr/80PffdyZypn/vnu0XdrRrogU5og6y1f337bxxNB
86J4ZU1UDP25HqELV8pppCuf7c500dCgWfXwVKW6w46eDeu64dXoazpc9WwIa/s1Na6Bw9MHj0dPj7b9s45VEGH185Ee+W09MMFKA1Eb661aOMT
30vJ0zYv0pQe= frank-phillip.kessler@stud.uni-bamberg.de
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAO92nThVnQdJK0f4Ech3Croox1Gha20vdJ3SpFZPMBZuJNz7ae7mKeGmTzGyae0LSlyh99ka03t1rIP2
Niw3PILVKKX09Bd+K+G4H2pXuy8Zgdqfgk1dM1K9WLB4t1PZu501MWTJhgmWBY19XzqbHzkcmzXzMY052IDBD/+ELCKHL4tyZy58sA3icSq3X0rMHL
93N4TFx8757tq/9H/kHo7yqBL709R231CqJ21LzCuOfs0J5pyIozTtV+ukzTHUWIZ/gT0PB/entk1ZDXUxCKbhcDpd9/IWSUpmw4+T0BJgSLDnk3420
r/mnC3CKKcD70G0K2+uXyB09 aro@arch-aro
kaiser@psi-introsp:~/.ssh$

```

Figure 3: location of keys on server

```

[aro@arch-aro ~]$ cd .ssh/
[aro@arch-aro .ssh]$ ls
id_rsa  id_rsa.pub  id_rsa.psi  id_rsa.psi.pub  id_rsa.pub  known_hosts
[aro@arch-aro .ssh]$ cat id_rsa.psi.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAQ9znThVngbJK0f4EchJcRooxIghaZDvdJjSpFZPwB2uJNz7ae7mkeGmIzGyae0l5jlyhp9ka03tjriP2
Nne3PILVAXO298dK+G4R2pXuy8epqfGk1ldHIXQwQL84t1PZuS01HMTJHgmMBV19XzqbMakcmIzKZAY02z1BBD/+ELCKHLtyZy585A3icq3X8fPwE
93N4tFx877td/9H/KHo7yqBL7099R23lqZl1Cu0Fs0J5pyIo2TtV+Iuk2ThuWIZ/gT0PB/entkiZDXUjxCkhhQcpnd9/IW5uPmw4+TQbJgsIDmk3429
sRYmC3CMKcJ070GOKZ+tuYB89 aro@arch-aro
[aro@arch-aro .ssh]$ pwd
/home/aro/.ssh

```

Figure 4: location of keys on local machine

```

martin@psi-introsp:~/.ssh$ ls
authorized_keys  known_hosts
martin@psi-introsp:~/.ssh$ ls -l
total 8
-rw-r--r-- 1 martin martin 398 Nov 17 21:22 authorized_keys
-rw-r--r-- 1 martin martin 215 Nov  6 18:00 known_hosts
martin@psi-introsp:~/.ssh$

```

Figure 5: permission check

The ssh-keys on your local machine are by default stored in `~/.ssh/id_rsa.pub` for the public part and the private one in `~/.ssh/id_rsa`. (Provided you did create a rsa key) See figure 4.

2.2 Windows

On a windows machine the same tools from openSSH were not available. That's why the procedure was a little bit different. Here PuTTY was used.

To generate the key the tool `putty-gen` was used. Then we logged into the server via ssh and created the file `authorized_keys` in `~/.ssh/` and pasted the key into it using nano. Lastly, we checked that only we have write access to the file by `ls -l`

In figure 6 you can see the successful login by using the ssh-key pair.

```

login as: martin
Authenticating with public key "rsa-key-20171117"
Passphrase for key "rsa-key-20171117":
Linux psi-introsp 4.9.0-4-amd64 #1 SMP Debian 4.9.51-1 (2017-09-28) x86_64

Last login: Fri Nov 17 20:55:56 2017 from 185.53.42.56
martin@psi-introsp:~$

```

Figure 6: windows ssh-key login

3 Task 3

The source code for the exercise can be found in the following file: [Vigenere Cipher](#)

To compile: `gcc -Wall vigenere_cipher.c -o "output-name" Optional flag: -DDEBUG`

Below is the source code readable:

```
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int getRotation(char c) {
    /* According to ascii 'A' transfers to 65
    and Z to 90. By subtracting 65 of the char we get
    the rotation. */
    return c - 65;
}

int main(int argc, char *argv[]) {
    char key[256];
    char word[256];

    printf("Type in the key\n");
    fgets(key, sizeof(key), stdin);

    printf("What do you want to encrypt?\n");
    fgets(word, sizeof(word), stdin);

    // Number of char that were not uppercase
    int cntSkipped = 0;
    // length of the key - 1 to know when to start from 0
    int keyLength = strlen(key) - 1; // remove \n
    int keyPosition = 0; // index of the key word

    for (int i = 0; i < strlen(word); i++) {
        /* Set keyPosition to 0 when end of key is reached
        i is subtracted by the number of skipped chars
        so the % operator works as intended */
        if ((i > 0) & ((i - cntSkipped) % keyLength == 0)) {
            keyPosition = 0;
        }
        // ignore lowercases and whitespaces
        if (!isupper(word[i])) {
            cntSkipped++;
            continue;
        }
    }
}
```

```

        int rotation = getRotation(key[keyPosition]);
        keyPosition++;

#ifdef DEBUG
        printf("%i ", rotation);
#endif

        word[i] = ((word[i] - 'A' + rotation) % 26) + 'A';
    }
    printf("%s", word);

    return 0;
}

```

4 Task 4

Snow is white.