Keep your secrets under control - Cryptography for All

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

# Keep your secrets under control - Cryptography for All"

#### Disclamer

By joining this course or using material from the course you agree that all materials of the course are solely provided "as is" without any guarantee of functionality. All methods are provided for training purposes. Course developed by non security expert by explaining the practical use of **OpenSSL** software library in R Statistical Software. There is a risk that you can loose your information in case you are not properly follow the instructions or by loosing your private key or alike. Make sure you know what you do at any time when using the material of the course. Always test your knowledge on dummy data before applying it to real data... Memory areas for your private key, secrets, or passphrase are not securely cleaned after use. You are using the material of the course at your own risk!

#### Lecture 6 - generate your private key and store it

Welcome back to the course. In this lecture we will focus on several things:

* understanding how to create our private key using **openssl** library
* getting to know structure of the **Private Key**
* how to store your **Private Key** as a File in the Computer File System

The MUST remember takeaway for this lecture is the understanding that **Loosing Private Key == Loosing your data**

We will go very slowly with every step so you can follow it in your own pace. For those who already advanced in R and perhaps don't need such detailed explanations feel free to jump to the end of this lecture to review the code chunk you need to use to create the key.

Open your R Studio and create a new script. Start loading a package with a command library(openssl). Inside this library we have function that can create our private key... this function is named rsa\_keygen(). By typing **?keygen** you can read help for this function

# load openssl library  
library(openssl)  
  
# read function documentation  
?keygen

## starting httpd help server ...

## done

Now we can simply type private underscore key and generate our key by typing:

# generating private key  
private\_key <- rsa\_keygen()

Let's understand what happened now. First of all Look to the Environment. YOu have just generated an object named 'private\_key'. I want to draw your full attention now: this is a **UNIQUE** key. If you run command again this will be **ANOTHER** key. **So be really careful!!!**

Secondly, lets understand what kind of object we just generated... If you look to environment again you will find that this is a list of 4. You can easily see what is inside of this list by using str() function. Just type summary(private\_key) or str(private\_key) ... this way you will see a summary and structure on what is in the list

# understand what is in the key  
str(private\_key)

## List of 4  
## $ type : chr "rsa"  
## $ size : int 2048  
## $ pubkey:List of 5  
## ..$ type : chr "rsa"  
## ..$ size : int 2048  
## ..$ ssh : chr "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDNpOEaN ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] 5e 39 aa 94 ...  
## ..$ data :List of 2  
## .. ..$ e:Class 'bignum' raw [1:3] 01 00 01  
## .. ..$ n:Class 'bignum' raw [1:257] 00 cd a4 e1 ...  
## $ data :List of 8  
## ..$ e :Class 'bignum' raw [1:3] 01 00 01  
## ..$ n :Class 'bignum' raw [1:257] 00 cd a4 e1 ...  
## ..$ p :Class 'bignum' raw [1:129] 00 ee 25 9f ...  
## ..$ q :Class 'bignum' raw [1:129] 00 dd 0f 7a ...  
## ..$ d :Class 'bignum' raw [1:257] 00 ae e4 b0 ...  
## ..$ dp:Class 'bignum' raw [1:128] 10 bb ca 0c ...  
## ..$ dq:Class 'bignum' raw [1:128] 6f db 87 49 ...  
## ..$ qi:Class 'bignum' raw [1:129] 00 aa 0d db ...

summary(private\_key)

## Length Class1 Class2 Mode   
## 1192 key rsa raw

you can access each element just by adding dollar sign and the element of the list. For example if I type private\_key$type i will get a type of the key:

private\_key$type

## [1] "rsa"

notice we have there some important elements. one of them is a **public key** on this we will talk in the next lecture and data. What we can see as well that data list of the **public key** is also a part of the data of the **private key**...

If you want to see what is in this long number you can of course simply print it:

private\_key$data$n

## [b] 25960157534257223496252635798322862017550119113103074092425251349540629806663148956582470618812511431120804697226064799436380979404475850526388003034591167278314283442260677395221831170584636550988310611183782587639637154226121762804735015421244600743136065157898578619013485040641461797968342180500930406875074946345269920458330203869553861055714429144474184914958927005727736314544465482260753387748277441737486300166981290599433998612546517491032261315652128433606072767338887944405598626357264578557314095421200233560025577286616992101327951850319632468792603611247466002832685141890276225093793503080740216521621

You can of course study your private key further but we will now concentrate on the way we can **keep and retain** our key. In fact we should store this key somehow. In fact this will be very important as we can loose the access to our information if we accidentally delete our private key!!! So to store the key we can use another function named **write\_pem()** to actually write our password to the file.

to use this function we must supply the private key we just generated, the name of the file and the password [optional].

Note that we can provide here not only the file name but also a path where we want to keep our file... It is possible to set up a password as well. This way we will restrict our system to read our key from file... this will obviously further increase security

This way we can simply write our file by executing the function **write\_pem()**. I will now provide password to be 'udemy'

write\_pem(x = private\_key, path = "private.key", password = "udemy")

After executing this function you should go to the Files tab of the project and you will find your file there. You can do whatever you want with this key but I will make the following. I will save my private key to the usb key. For that I will add a path to the usb key to the function...

#write\_pem(x = private\_key, path = "USB/private.key", password = "udemy")

Once I am TOTALLY sure that my private key is saved to the file I must delete the private key from the environment. To do this we can run function rm and provide names of the elements we want to remove.

# delete the private key  
rm(private\_key)

If you are using R studio you can also delete everything from the environment using a brush icon.

We now have one last and important learning objective: is about loosing our private key... Well, it's the same effect as you simply forget your password and there is no one who can help you to reset it. Risk is to have a data you will never decrypt that is it! Remember this was yourself who generated it so it's up to you how to keep this file in a way you never loose it... For the purpose of the course I will be using a usb stick where I will keep my private key for my private secrets!!!

At this time let's officially conclude this lecture by making a quick summary:

* understand how to generate private key We can generate our UNIQUE private key by using command rsa\_keygen and set number of bits we want for our key
* look what is the structure of private key This way we will create a list with important parameters including part of it will be a public key
* write our key to the file In order to store our key outside of computer we can store it to the file. We can still protect our private key file with a password. Once our private key is in the file we must delete our private key from the r studio environment
* risks of loosing keys = risk of loosing your data BE very CAREFUL to not loosing your key file. It's not a simple file but a UNIQUE file make sure you have this under control!!!

And before we end this lecture I will just summarise the code you need to generate your private key, write it to the file and delete it from R environment

library(tidyverse)

## Loading tidyverse: ggplot2  
## Loading tidyverse: tibble  
## Loading tidyverse: tidyr  
## Loading tidyverse: readr  
## Loading tidyverse: purrr  
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## filter(): dplyr, stats  
## lag(): dplyr, stats

openssl::rsa\_keygen(bits = 2099) %>%   
write\_pem(path = "private.key", password = "udemy")

That is all for this lecture where we discussed about private keys how to create, what they contain and how to manage them... I will be looking forward to welcome you to the next lecture where we will talk about using private key to generate public key

##### ################ ////////////// some more research, optional read...////////////

Let us also make an experiment and try to play with bits parameter. First we will try to specify bits parameter to be **2000**. We call it key2000. After generating this key we will look the structure...

key2000 <- rsa\_keygen(bits = 2000)  
str(key2000)

## List of 4  
## $ type : chr "rsa"  
## $ size : int 2000  
## $ pubkey:List of 5  
## ..$ type : chr "rsa"  
## ..$ size : int 2000  
## ..$ ssh : chr "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAA+wDuSJOoK ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] a9 28 b6 f3 ...  
## ..$ data :List of 2  
## .. ..$ e:Class 'bignum' raw [1:3] 01 00 01  
## .. ..$ n:Class 'bignum' raw [1:251] 00 ee 48 93 ...  
## $ data :List of 8  
## ..$ e :Class 'bignum' raw [1:3] 01 00 01  
## ..$ n :Class 'bignum' raw [1:251] 00 ee 48 93 ...  
## ..$ p :Class 'bignum' raw [1:126] 00 fb 3a ef ...  
## ..$ q :Class 'bignum' raw [1:126] 00 f2 ce b7 ...  
## ..$ d :Class 'bignum' raw [1:251] 00 cb 94 46 ...  
## ..$ dp:Class 'bignum' raw [1:125] 61 a6 01 a1 ...  
## ..$ dq:Class 'bignum' raw [1:125] 6f 5f 6c 6f ...  
## ..$ qi:Class 'bignum' raw [1:125] 1a 95 a7 0e ...

now we will do the same for bits 5000 ...

key5000 <- rsa\_keygen(bits = 5000)  
str(key5000)

## List of 4  
## $ type : chr "rsa"  
## $ size : int 5000  
## $ pubkey:List of 5  
## ..$ type : chr "rsa"  
## ..$ size : int 5000  
## ..$ ssh : chr "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACcgDBKC7y2 ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] 47 fe db bd ...  
## ..$ data :List of 2  
## .. ..$ e:Class 'bignum' raw [1:3] 01 00 01  
## .. ..$ n:Class 'bignum' raw [1:626] 00 c1 28 2e ...  
## $ data :List of 8  
## ..$ e :Class 'bignum' raw [1:3] 01 00 01  
## ..$ n :Class 'bignum' raw [1:626] 00 c1 28 2e ...  
## ..$ p :Class 'bignum' raw [1:313] 0e 94 be 6d ...  
## ..$ q :Class 'bignum' raw [1:313] 0d 3f 44 9e ...  
## ..$ d :Class 'bignum' raw [1:625] 2a 18 a2 da ...  
## ..$ dp:Class 'bignum' raw [1:313] 02 31 22 1f ...  
## ..$ dq:Class 'bignum' raw [1:313] 09 af 59 d1 ...  
## ..$ qi:Class 'bignum' raw [1:313] 0d e7 0d 52 ...

what differences can you see? You see the lenght of the elements is different...

length(key2000$data$n)

## [1] 251

length(key5000$data$n)

## [1] 626

* dsa\_keygen
* ec\_keygen

ecKey521 <- ec\_keygen(curve = "P-521")  
ecKey384 <- ec\_keygen(curve = "P-384")  
ecKey256 <- ec\_keygen(curve = "P-256")

structure of these file

str(ecKey256)

## List of 4  
## $ type : chr "ecdsa"  
## $ size : int 256  
## $ pubkey:List of 5  
## ..$ type : chr "ecdsa"  
## ..$ size : int 256  
## ..$ ssh : chr "ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzd ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] d3 1a a2 89 ...  
## ..$ data :List of 3  
## .. ..$ curve: chr "P-256"  
## .. ..$ x :Class 'bignum' raw [1:32] 80 98 51 41 ...  
## .. ..$ y :Class 'bignum' raw [1:32] 5d f0 b4 2e ...  
## $ data :List of 4  
## ..$ curve : chr "P-256"  
## ..$ x :Class 'bignum' raw [1:32] 80 98 51 41 ...  
## ..$ y :Class 'bignum' raw [1:32] 5d f0 b4 2e ...  
## ..$ secret:Class 'bignum' raw [1:32] fd 74 46 2f ...

str(ecKey384)

## List of 4  
## $ type : chr "ecdsa"  
## $ size : int 384  
## $ pubkey:List of 5  
## ..$ type : chr "ecdsa"  
## ..$ size : int 384  
## ..$ ssh : chr "ecdsa-sha2-nistp384 AAAAE2VjZHNhLXNoYTItbmlzd ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] 4e a0 d6 92 ...  
## ..$ data :List of 3  
## .. ..$ curve: chr "P-384"  
## .. ..$ x :Class 'bignum' raw [1:48] 7c 1a d4 59 ...  
## .. ..$ y :Class 'bignum' raw [1:48] 8f a1 02 61 ...  
## $ data :List of 4  
## ..$ curve : chr "P-384"  
## ..$ x :Class 'bignum' raw [1:48] 7c 1a d4 59 ...  
## ..$ y :Class 'bignum' raw [1:48] 8f a1 02 61 ...  
## ..$ secret:Class 'bignum' raw [1:48] 76 12 6c e8 ...

str(ecKey521)

## List of 4  
## $ type : chr "ecdsa"  
## $ size : int 521  
## $ pubkey:List of 5  
## ..$ type : chr "ecdsa"  
## ..$ size : int 521  
## ..$ ssh : chr "ecdsa-sha2-nistp521 AAAAE2VjZHNhLXNoYTItbmlzd ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] 88 31 40 ce ...  
## ..$ data :List of 3  
## .. ..$ curve: chr "P-521"  
## .. ..$ x :Class 'bignum' raw [1:66] 00 df b5 4f ...  
## .. ..$ y :Class 'bignum' raw [1:66] 00 3d 46 9d ...  
## $ data :List of 4  
## ..$ curve : chr "P-521"  
## ..$ x :Class 'bignum' raw [1:66] 00 df b5 4f ...  
## ..$ y :Class 'bignum' raw [1:66] 00 3d 46 9d ...  
## ..$ secret:Class 'bignum' raw [1:66] 00 59 d5 61 ...

* dsa\_keygen

dsaKey1024 <- dsa\_keygen(bits = 1024)  
dsaKey2024 <- dsa\_keygen(bits = 2024)

structure of these file

str(dsaKey1024)

## List of 4  
## $ type : chr "dsa"  
## $ size : int 1024  
## $ pubkey:List of 5  
## ..$ type : chr "dsa"  
## ..$ size : int 1024  
## ..$ ssh : chr "ssh-dss AAAAB3NzaC1kc3MAAACBAJQWGstOEl1sTT8s3 ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] 54 04 7e 13 ...  
## ..$ data :List of 4  
## .. ..$ p:Class 'bignum' raw [1:129] 00 94 16 1a ...  
## .. ..$ q:Class 'bignum' raw [1:21] 00 fe 41 3a ...  
## .. ..$ g:Class 'bignum' raw [1:128] 70 fa 17 72 ...  
## .. ..$ y:Class 'bignum' raw [1:128] 70 2a 38 29 ...  
## $ data :List of 5  
## ..$ p:Class 'bignum' raw [1:129] 00 94 16 1a ...  
## ..$ q:Class 'bignum' raw [1:21] 00 fe 41 3a ...  
## ..$ g:Class 'bignum' raw [1:128] 70 fa 17 72 ...  
## ..$ y:Class 'bignum' raw [1:128] 70 2a 38 29 ...  
## ..$ x:Class 'bignum' raw [1:21] 00 9b e0 70 ...

str(dsaKey2024)

## List of 4  
## $ type : chr "dsa"  
## $ size : int 2048  
## $ pubkey:List of 5  
## ..$ type : chr "dsa"  
## ..$ size : int 2048  
## ..$ ssh : chr "ssh-dss AAAAB3NzaC1kc3MAAAEBAPFxRGtB6MA/MehZ/ ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] 50 3e 8b c4 ...  
## ..$ data :List of 4  
## .. ..$ p:Class 'bignum' raw [1:257] 00 f1 71 44 ...  
## .. ..$ q:Class 'bignum' raw [1:21] 00 fb fb a4 ...  
## .. ..$ g:Class 'bignum' raw [1:256] 6f 8b e4 a1 ...  
## .. ..$ y:Class 'bignum' raw [1:256] 03 3a b0 9c ...  
## $ data :List of 5  
## ..$ p:Class 'bignum' raw [1:257] 00 f1 71 44 ...  
## ..$ q:Class 'bignum' raw [1:21] 00 fb fb a4 ...  
## ..$ g:Class 'bignum' raw [1:256] 6f 8b e4 a1 ...  
## ..$ y:Class 'bignum' raw [1:256] 03 3a b0 9c ...  
## ..$ x:Class 'bignum' raw [1:20] 5f 99 9c 9a ...

##### // Some more research -- > trying to generate exact same private key...

set.seed(1)  
myPrivateKey1 <- rsa\_keygen()  
set.seed(1)  
myPrivateKey2 <- rsa\_keygen()  
  
# are they equal?  
identical(myPrivateKey1, myPrivateKey2)

## [1] FALSE

Keys are really different...

str(myPrivateKey1)

## List of 4  
## $ type : chr "rsa"  
## $ size : int 2048  
## $ pubkey:List of 5  
## ..$ type : chr "rsa"  
## ..$ size : int 2048  
## ..$ ssh : chr "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDAKjj4p ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] b0 6d a8 5e ...  
## ..$ data :List of 2  
## .. ..$ e:Class 'bignum' raw [1:3] 01 00 01  
## .. ..$ n:Class 'bignum' raw [1:257] 00 c0 2a 38 ...  
## $ data :List of 8  
## ..$ e :Class 'bignum' raw [1:3] 01 00 01  
## ..$ n :Class 'bignum' raw [1:257] 00 c0 2a 38 ...  
## ..$ p :Class 'bignum' raw [1:129] 00 fd 89 f0 ...  
## ..$ q :Class 'bignum' raw [1:129] 00 c2 07 c3 ...  
## ..$ d :Class 'bignum' raw [1:256] 45 38 f0 7c ...  
## ..$ dp:Class 'bignum' raw [1:129] 00 ba 2a a1 ...  
## ..$ dq:Class 'bignum' raw [1:129] 00 b0 7f 5d ...  
## ..$ qi:Class 'bignum' raw [1:129] 00 d8 42 de ...

str(myPrivateKey2)

## List of 4  
## $ type : chr "rsa"  
## $ size : int 2048  
## $ pubkey:List of 5  
## ..$ type : chr "rsa"  
## ..$ size : int 2048  
## ..$ ssh : chr "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCxEpS0U ..."  
## ..$ fingerprint:Classes 'hash', 'md5' raw [1:16] bb 54 89 36 ...  
## ..$ data :List of 2  
## .. ..$ e:Class 'bignum' raw [1:3] 01 00 01  
## .. ..$ n:Class 'bignum' raw [1:257] 00 b1 12 94 ...  
## $ data :List of 8  
## ..$ e :Class 'bignum' raw [1:3] 01 00 01  
## ..$ n :Class 'bignum' raw [1:257] 00 b1 12 94 ...  
## ..$ p :Class 'bignum' raw [1:129] 00 da bc a3 ...  
## ..$ q :Class 'bignum' raw [1:129] 00 cf 3c ea ...  
## ..$ d :Class 'bignum' raw [1:256] 06 48 d8 8a ...  
## ..$ dp:Class 'bignum' raw [1:128] 5a ad f3 71 ...  
## ..$ dq:Class 'bignum' raw [1:128] 56 79 b1 b0 ...  
## ..$ qi:Class 'bignum' raw [1:129] 00 c5 4d c5 ...