

**Course:**

CE 340 Cryptography & Network Security

**Course Instructor:**

Assoc. Prof. Dr. Süleyman KONDAKÇI

**Project Title:**

Final Project

DSA Implementation in sage

**Submitted By:**

Serdar HALİLOĞLU

20120602019

**Description:**

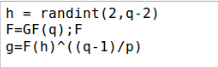
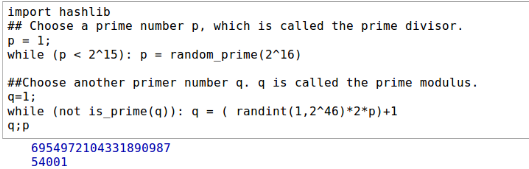
* Generate a digital signature(DS) based on the DSA algorithm.
* Sign a document using the generated DS.
* Verify that the signing is correctly done.

**Outcomes:**

* Learning how to use DSA algorithm in sage
* Learning how to use some functions such as converting integers into asci codes.
* Learning how to sign a document.
* Learning how to verify whether signing is correct or not.

I will explain all the lines, respectively, that I wrote.

1. I used hashlib. I will explain why we need to import this.
2. Generate Components.
   1. Choose a prime number p, which is called the prime divisor.
   2. Choose another primer number q namely prime modulus.
   3. Choose an integer g, such that 1 < g < p, g = h(q–1)/p mod q.   
      p is also called g's multiplicative order modulo q.



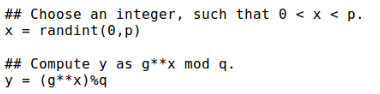
1. Generate Private Components.
   1. Choose an integer x, such that 0 < x < p.

x = randint(0,p)

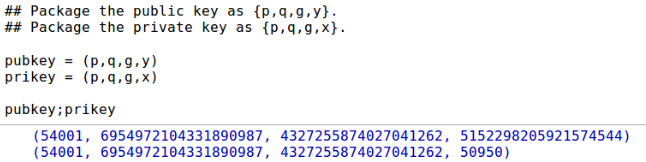
* 1. Compute y as gx mod q.

y = (g\*\*x)%q

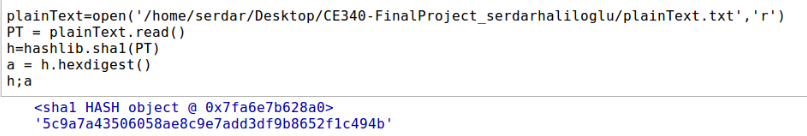
C:\Users\serda\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5.png



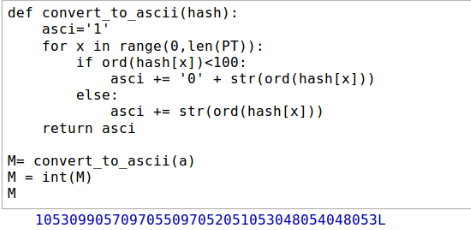
1. Print Public and Private Key
   1. Package the public key as {p,q,g,y}.
      1. pubkey = (p,q,g,y)
   2. Package the private key as {p,q,g,x}.
      1. prikey = (p,q,g,x)
   3. pubkey;prikey



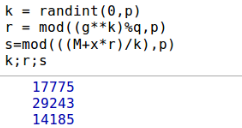
1. Define plainText and open text file such that it is readable.
   1. plainText=open('/home/serdar/Desktop/CE340-FinalProject\_serdarhaliloglu/plainText.txt','r')
   2. Define PT variable for reading .
      1. PT = plainText.read()
   3. hashlib is imported in order to generate the message digest h, using a hash algorithm like SHA1.
      1. h=hashlib.sha1(PT)
      2. a = h.hexdigest()



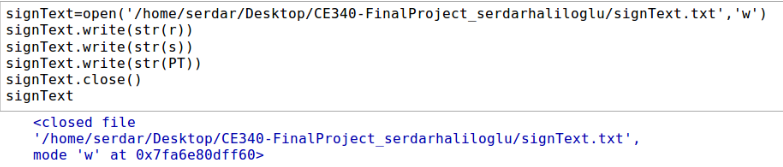
1. Ascii Function and string-integer converting
   1. Define a function called convert\_to\_ascii because we need to convert hex value to ascii value.
   2. Define a variable M, and assign the result of function convert to get the ascii value.
   3. Then convert string to int value.
   4. Print M.



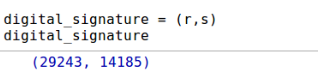
1. Generate k, r, sign(=s)
   1. Generate a random number k, such that 0 < k < p.
      1. k = randint(0,p)
   2. Compute r as (gk mod q) mod p.
      1. r = mod((g\*\*k)%q,p)
   3. Compute s
      1. s=mod(((M+x\*r)/k),p)
   4. Print k,r,s
      1. k;r;s



1. Define signText and open text file such that it is writeable.
   1. signText=open('/home/serdar/Desktop/CE340-FinalProject\_serdarhaliloglu/signText.txt','w')
   2. Write r,s and PT into signText file and close file.
      1. signText.write(str(r))
      2. signText.write(str(s))
      3. signText.write(str(PT))
      4. signText.close()



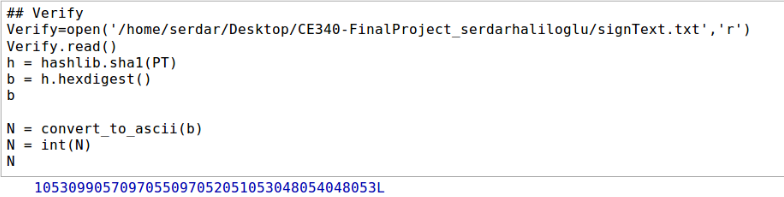
1. Package the digital signature as {r,s} and print.
   1. digital\_signature = (r,s)
   2. digital\_signature



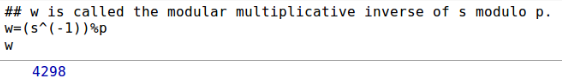
1. To verify a message signature, the receiver of the message and the digital signature follows steps:
   1. Define Verify and open signText file such that it is readable.

Verify=open('/home/serdar/Desktop/CE340-FinalProject\_serdarhaliloglu/signText.txt','r')

* 1. Verify.read()
  2. Generate the message digest h, using the same hash algorithm.
     1. h = hashlib.sha1(PT)
     2. b = h.hexdigest()
     3. b
  3. Define a variable N, and assign the result of function to get its ascii value.
  4. Then convert string to int value.
  5. Print N.
     1. N = convert\_to\_ascii(b)
     2. N = int(N)
     3. N



* 1. Compute w, such that s\*w mod q = 1. w is called the modular multiplicative inverse of s modulo p.
     1. w=(s^(-1))%p;
     2. w



* 1. Compute u1 = N\*w mod p.

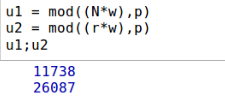
u1 = mod((N\*w),p)

* 1. Compute u2 = r\*w mod p

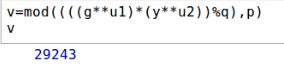
u2 = mod((r\*w),p)

* 1. Print u1,u2

u2 = mod((r\*w),p)



* 1. Compute v = ((gu1\*yu2) mod q) mod p.
     1. v=mod((((g\*\*u1)\*(y\*\*u2))%q),p)
     2. v



* 1. If v == r, the digital signature is valid.

v == r



**User Guide**

All steps are described below, respectively. Please follow the steps to run.

* Extract CE340-FinalProject\_serdarhaliloglu.zip.
* Open sage math with sudo.
* Use notebook() for the browser-based notebook interface.
* Open [http://localhost:8080](http://localhost:8080/) in your web browser.
* Select Upload on Sage Notebook.
* Click Browse Button, then select FinalProject.sws and upload worksheet.
* Change the location of filenames in worksheet.
* After the all steps, click evaluate all in action part.