

Lab 2: Diffie-Hellman, Public Key and Private Key

Login for Napier network: Login: vsdev1 Password: visual1

Login for Asecuritysite.com: Login: napier Password: napier

We will allocate you a Cloud instance in the forthcoming labs.

1 Diffie-Hellman

No	Description	Result
1	Bob and Alice have agreed on the values: G=2879, N= 9929 Bob Select x=6, Alice selects y=9	Now calculate (using the Windows calculator): Bob's A value ($G^x \text{ mod } N$): Alice's B value ($G^y \text{ mod } N$):
2	Now they exchange the values. Next calculate the shared key:	Bob's value ($B^x \text{ mod } N$): Alice's value ($A^y \text{ mod } N$): Do they match? [Yes] [No]
3	If you are in the lab, select someone to share a value with. Next agree on two numbers (G and N). You should generate a random number, and so should they. Do not tell them what your random number is. Next calculate your A value, and get them to do the same.	Numbers for G and N: Your x value: Your A value:

	Next exchange values.	<p>The B value you received:</p> <p>Shared key:</p> <p>Do they match: [Yes] [No]</p>
--	-----------------------	--

B Private Key

Download openssl from the following link and open-up a console window.

<http://asecuritysite.com/openssl.zip>

No	Description	Result
1	Use: <code>openssl list-cipher-commands</code> <code>openssl version</code>	Outline five encryption methods that are supported: Outline the version of OpenSSL:
2	Using openssl and the command in the form: <code>openssl prime -hex 1111</code>	Check if the following are prime numbers: 42 [Yes][No] 1421 [Yes][No]
2	Now create a file named myfile.txt (either use Notepad or another editor). Next encrypt with aes-256-cbc	Use following command to view the output file: <code>type encrypted.bin</code>

	<pre>openssl enc -aes-256-cbc -in myfile.txt -out encrypted.bin</pre> <p>and enter your password.</p>	Is it easy to write out or transmit the output: [Yes][No]
3	<p>Now repeat the previous command and add the <code>-base64</code> option.</p> <pre>openssl enc -aes-256-cbc -in myfile.txt -out encrypted.bin -base64</pre>	<p>Use following command to view the output file:</p> <pre>type encrypted.bin</pre> <p>Is it easy to write out or transmit the output: [Yes][No]</p>
4	<p>Now Repeat the previous command and observe the encrypted output.</p> <pre>openssl enc -aes-256-cbc -in myfile.txt -out encrypted.bin -base64</pre>	<p>Has the output changed? [Yes][No]</p> <p>Why has it changed?</p>
5	<p>Now let's decrypt the encrypted file with the correct format:</p> <pre>openssl enc -d -aes-256-cbc -in encrypted.bin -pass pass:napien -base64</pre>	<p>Has the output been decrypted correctly?</p> <p>What happens when you use the wrong password?</p>
6	<p>If you are working in the lab, now give you private key to your neighbour, and get them to encrypt a secret message for you.</p>	Did you manage to decrypt their message? [Yes][No]
7	<p>Now encrypt a file with Blowfish and see if you can decrypt it.</p>	Did you manage to decrypt the file? [Yes][No]
8	<p>Now encrypt a file with 3DES and see if you can decrypt it.</p>	Did you manage to decrypt the file? [Yes][No]
9	<p>Now encrypt a file with RC2 and see if you can decrypt it.</p>	Did you manage to decrypt the file? [Yes][No]

C Public Key

Download openssl and open-up a console window.

No	Description	Result
1	<p>First we need to generate a key pair with:</p> <pre>openssl genrsa -out private.pem 1024</pre> <p>This file contains both the public and the private key.</p>	<p>What is the type of public key method used:</p> <p>How long is the default key:</p> <p>How long did it take to generate a 1,024 bit key?</p> <p>Use the following command to view the keys:</p> <pre>Type key.pem</pre>
2	<p>Use following command to view the output file:</p> <pre>type private.pem</pre>	<p>What can be observed at the start and end of the file:</p>
3	<p>Next we view the RSA key pair:</p> <pre>openssl rsa -in private.pem -text -noout</pre>	<p>Which are the attributes of the key shown:</p> <p>Which number format is used to display the information on the attributes:</p>

		What does the <code>--noout</code> option do?
4	Let's now secure the encrypted key with 3-DES: <pre>openssl rsa -in private.pem -des3 -out key3des.pem</pre>	
4	Next we will export the public key: <pre>openssl rsa -in private.pem -out public.pem -outform PEM -pubout</pre>	View the output key. What does the header and footer of the file identify?
5	Now we will encrypt with our private key: <pre>openssl rsautl -encrypt -inkey public.pem -pubin -in myfile.txt -out file.bin</pre>	
6	And then decrypt with our public key: <pre>openssl rsautl -decrypt -inkey private.pem -in file.bin -out decrypted.txt</pre>	What are the contents of <code>decrypted.txt</code>
7	If you are working in the lab, now give you public key to your neighbour, and get them to encrypt a secret message for you.	Did you manage to decrypt their message? [Yes][No]

D Storing keys

We have stored our keys on a key ring file (PEM). Normally we would use a digital certificate to distribute our public key. In this part of the tutorial we will create a crt digital certificate file.

No	Description	Result
1	<p>First download a conf file which will define the defaults for our digital certificate:</p> <p>http://asecuritysite.com/openssl.conf</p> <p>Store this in your openssl folder. Next create the crt file with the following:</p> <pre>openssl req -new -key private.pem -out cert.csr -config openssl.conf openssl x509 -req -in cert.csr -signkey private.pem -out server.crt</pre>	<p>What is the type of public key method used:</p> <p>View the certificate file and determine:</p> <p>The size of the public key:</p> <p>The encryption method:</p>

E Test

Now take the test at:

<http://asecuritysite.com/tests/tests?sortBy=crypto02>