



PRIFYSGOL CYMRU
Y Drindod Dewi Sant
UNIVERSITY OF WALES
Trinity Saint David
SWANSEA - ABERTAWE



Network Security (NS)

Lab -

Digital Signatures

Faculty of Architecture, Computing & Engineering

School of Applied Computing

Mount Pleasant Campus

Swansea

SA1 6ED

The purpose of today's laboratory exercise is to **hack a Webserver using domain fingerprinting**

A **digital signature** is a mathematical scheme for demonstrating the authenticity of a digital message or document.

Digital signatures use a standard, accepted format, called **Public Key Infrastructure (PKI)**, to provide the highest levels of security and universal acceptance. They are a specific signature technology implementation of electronic signature (eSignature).

A valid digital signature gives a recipient reason to believe that the message or document was created by a known sender (see **authentication** section below), that the sender cannot deny having sent the message (see **non-repudiation** section below), and that the message was not altered in transit (see **integrity** section below).

Digital signatures are a standard element of most cryptographic protocol suites (Cyphersuites), and are commonly used for software distribution, financial transactions, contract management software, and in other cases where it is important to detect forgery or tampering of message or document.

What is Public Key Infrastructure (PKI)?

Public Key Infrastructure (PKI) is a set of requirements that allow (among other things) the creation of digital signatures. Through PKI, each digital signature transaction includes a pair of keys: a **private key** and a **public key**. The private key, as the name implies, is not shared and is used only by the signer to electronically signed documents. The public key is openly available and used by those who need to validate the signer's electronic signature. PKI enforces additional requirements, such as the **Certificate Authority (CA)**, a **digital certificate**, end-user enrollment software, and tools for managing, renewing, and revoking keys and certificates.

What is a digital certificate?

A digital certificate is an electronic document issued by a Certificate Authority (CA). It contains the public key for a digital signature and specifies the identity associated with the key, such as the name of an organization. The certificate is used to confirm that the public key belongs to the specific organization. The CA acts as the guarantor. Digital certificates must be issued by a trusted authority and are only valid for a specified time. They are required in order to create a digital signature.

Components of Digital Signature

A digital signature scheme typically consists of three algorithms;

1. A *key generation* algorithm that selects a *private key* uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding *public key*.

2. A *signing* algorithm that, given a message and a private key, produces a **signature**.
3. A *signature verifying* algorithm that, given the message, public key and signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required. First, the authenticity of a signature generated from a fixed message and fixed private key can be verified by using the corresponding public key. Secondly, it should be computationally infeasible to generate a valid signature for a party without knowing that party's private key. A digital signature is an authentication mechanism that enables the creator of the message to attach a code that acts as a signature.

Digital signature arithmetic

To create RSA signature keys, generate a RSA key pair containing a modulus, N , that is the product of two large primes, along with integers, e and d , such that $e d \equiv 1 \pmod{\phi(N)}$, where ϕ is the Euler phi-function. The signer's public key consists of N and e , and the signer's secret key contains N and d .

To sign a message, m , the signer computes a signature, σ , such that $\sigma \equiv m^d \pmod{N}$. To verify, the receiver checks that $\sigma^e \equiv m \pmod{N}$.

Exercise:

1. Choose two prime numbers: $p = 3$ and $q = 11$
2. Compute the modulus: $N = p * q = 3 * 11 = 33$
3. Compute $\phi(N) = (p - 1) * (q - 1) = 2 * 10 = 20$
4. Choose e such that $1 < e < \phi(N)$ and e and n are co-prime. Let $e = 7$
5. Compute a value for d such that $(d * e) \% \phi(N) = 1$. One solution is $d = 3 [(3 * 7) \% 20 = 1]$
6. Public key is $(e, N) \Rightarrow (7, 33)$
7. Private key is $(d, N) \Rightarrow (3, 33)$

As noted earlier, this basic scheme is not very secure. To prevent attacks, one can first apply a cryptographic hash function to the message, m , and then apply the RSA algorithm described above to the result. This approach is secure assuming the hash function is a random oracle.

There are several reasons to sign such a hash (or message digest) instead of the whole document.

For efficiency: The signature will be much shorter and thus save time since hashing is generally much faster than signing in practice.

For compatibility: Messages are typically bit strings, but some signature schemes operate on other domains (such as, in the case of RSA, numbers modulo a composite number N). A hash function can be used to convert an arbitrary input into the proper format.

For integrity: Without the hash function, the text "to be signed" may have to be split (separated) in blocks small enough for the signature scheme to act on them directly. However, the receiver of the signed blocks is not able to recognize if all the blocks are present and in the appropriate order.

Applications of digital signatures

As organizations move away from paper documents with ink signatures or authenticity stamps, digital signatures can provide added assurances of the evidence to provenance, identity, and status of an electronic document as well as acknowledging informed consent and approval by a signatory. The United States Government Printing Office (GPO) publishes electronic versions of the budget, public and private laws, and congressional bills with digital signatures. Universities including Penn State, University of Chicago, and Stanford are publishing electronic student transcripts with digital signatures.

Below are some major reasons for applying a digital signature to data, documents and communications:

1. Authentication

Although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in a financial context. For example, suppose a bank's branch office sends instructions to the central office requesting a change in the balance of an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake.

2. Integrity

In many scenarios, the sender and receiver of a message may have a need for confidence that the message has not been altered during transmission. Although encryption hides the contents of a message, it may be possible to *change* an encrypted message without understanding it. (Some encryption algorithms, known as *nonmalleable* ones, prevent this, but others do not.) However, if a message is digitally signed, any change in the message after signature invalidates the signature. Furthermore, there is no efficient way to modify a message and its signature to produce a new message with a valid signature, because this is still considered to be computationally infeasible by most cryptographic hash functions.

3. Non-repudiation

Non-repudiation, or more specifically *non-repudiation of origin*, is an important aspect of digital signatures. By this property, an entity that has signed some information cannot at a later time deny having signed it. Similarly, access to the public key only does not enable a fraudulent party to fake a valid signature.

Note that **authentication**, **integrity**, **non-repudiation** etc. properties rely on the secret key *not having been revoked* prior to its usage. Public revocation of a key-pair is a required ability, else leaked secret keys would continue to implicate the claimed owner of the key-pair. Checking revocation status requires an "online" check; e.g., checking a "Certificate Revocation List" or via the "Online Certificate Status Protocol". Very roughly this is analogous to a vendor who receives credit-cards first checking online with the credit-card issuer to find if a given card has been reported lost or stolen. Of course, with stolen key pairs, the theft is often discovered only after the secret key's use, e.g., to sign a bogus certificate for criminal intent.

Lab Exercise: Digital Signature Demonstration

In this section, you will learn how to create real life digital signature using the Cryptool 1.4 and a bit of knowledge of digital signature arithmetic. From the section "Components of digital signatures" above, we know that the first step to create a digital signature is to create a pair of keys made of the public key and the private key. However from the "Digital signature arithmetic" section above, we also know the following fact:

To create pair of RSA signature keys (private and public), we need to generate a RSA key pair containing a modulus, N , which is the product of two large prime numbers ($N = p * q$).

Open the Cryptool application and select "Digital Signature/PKI" from the main menu. Then select "PKI" and then "Generate/Import Keys..." as shown in figure 1 below.

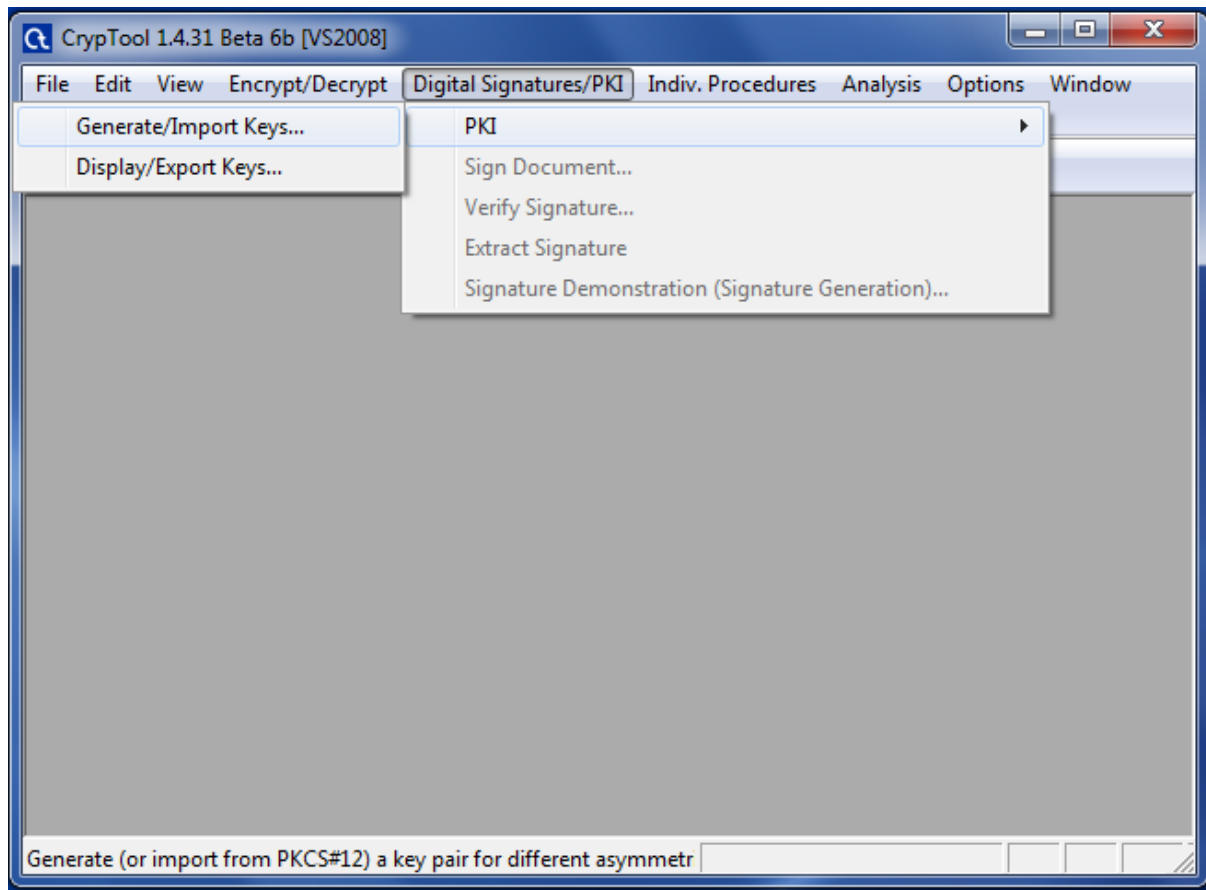


Figure 1: Digital Signature/PKI Menu.

At the main window as shown in figure 2 below, key in the detail for your choice of Asymmetric key. For the algorithm, you can select RSA, DSA or Elliptic Curves. Type your detail for User Data, followed by a PIN number that you can remember. Then click on the “Generate new key pair” button, at the bottom left corner of the screen.

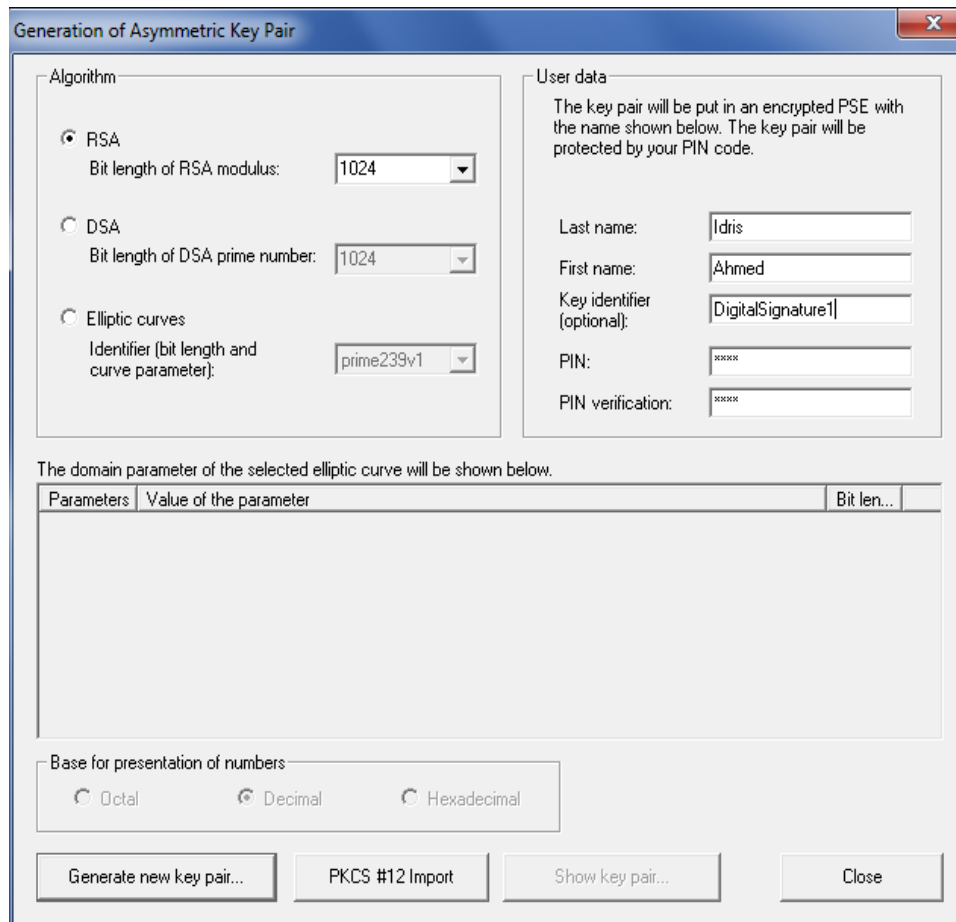


Figure 2: Generation of Asymmetric Key Pair.

Following a successful generation of the RSA key pair, you will get a dialogue screen like figure 3 below.

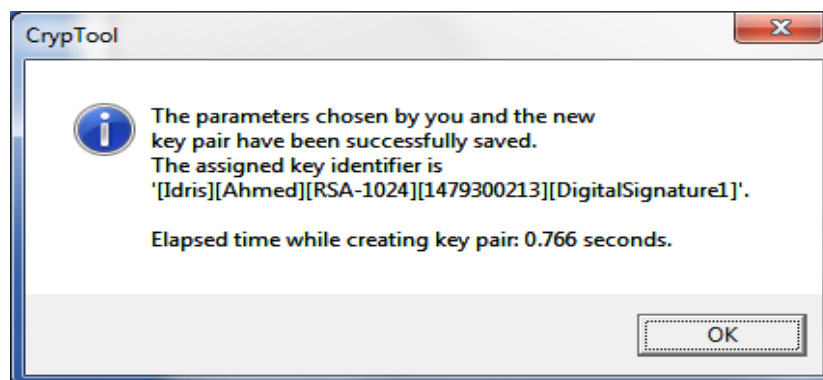


Figure 3: Successful key generation.

Check you PCs desktop, you will notice that a new document is added to the list of icons you already have, like figure 4 below.

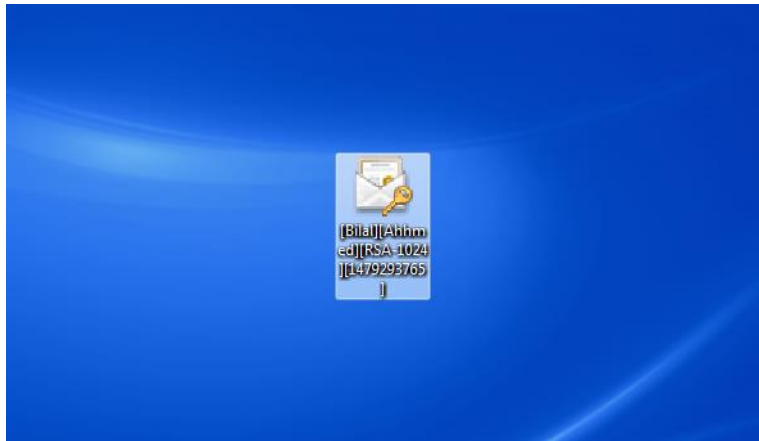


Figure 4: New generated key saved on the computer Desktop.

Do not delete the new generated key pair file. You will need it later during the simulation of how digital signatures are generated. If you are curious, you can double click on the file to see what happens. Doing so will automatically deploy the “Certificate Import Wizard” as shown below in figure 5 below.



Figure 5: Certificate Import Wizard.

The Microsoft Windows operating system stores certificates key pairs locally on the computer or on the computer or device that the user used to request it. The storage location is called the certificate store. You won't need to store the keys generated in this exercise. So click on the “Cancel” button to exit.

Run the Cryptool software. Close the “startingexample-en” window. At the main menu click on “Open File”. Then select the “Digital Signature Sample File.txt”. This is a simple text file given to you by the module leader. For this exercise, you can use this file or any other file, as long as it is a text file. Upon a successful loading of the file, your screen should look like figure 6 below.

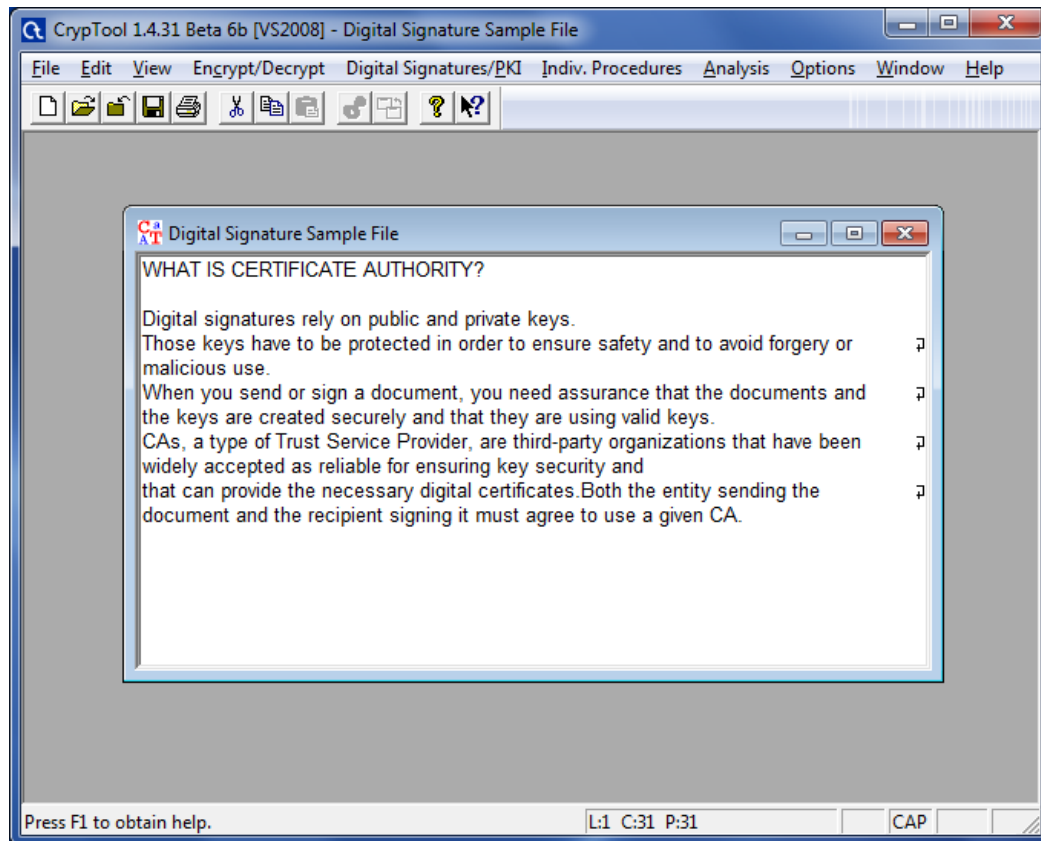


Figure 6: Loading “Digital Signature Sample File”.

With the “Digital Signature Sample File.txt” is loaded as in figure 6 above, go to the main menu, select “Digital Signatures/PKI”, and then click on “Signature Demonstration (Signature Generation)”. Step by step signature generation window will pop up as shown in figure 7 below. The window has two parts; the simulation section (with coloured buttons data flow diagram at the top) and display section at the bottom, showing the content of the “Digital Signature Sample File.txt” in hexadecimal and clear text format.

In order to generate a new digital signature and save it, you need do the following steps:

1. The status of the “Open Document” and the “Document” buttons indicate that they are active, so no need to click on either of the two. It means a file is already loaded into the Cryptool software and ready for simulation. Click on “Select Hash Function”. A pop up window will appear with several radio button options. Select “SHA-1” and click on the “OK” button.
2. Click on the blue “Hash Function” button. Notice that information about the hash function appears at the bottom part of the main window.

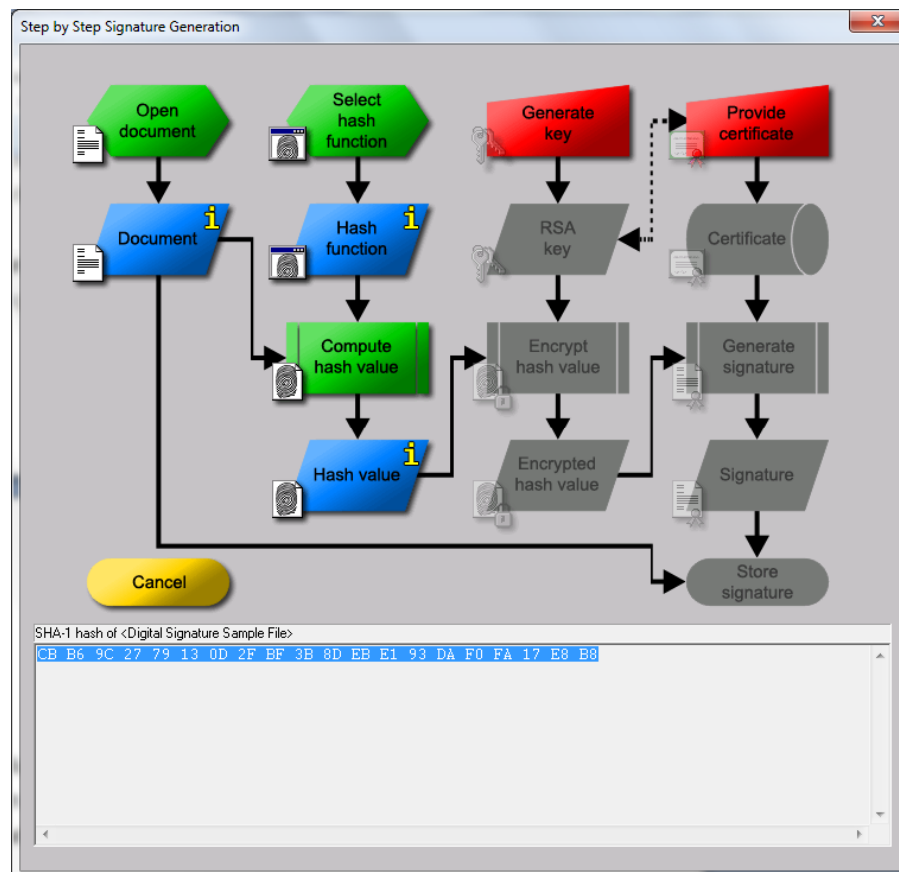


Figure 7: Hash value of “Digital Signature Sample File.txt” using SHA-1.

- Click on the “Compute Hash Value” button, and then click on “Hash Value”. The following hash value is shown in the display window. This is the hash value of the “Digital Signature Sample File.txt” when computed using SHA-1 hashing algorithm.

CB B6 9C 27 79 13 0D 2F BF 3B 8D EB E1 93 DA F0 FA 17 E8 B8

You can verify this hash by going to the <http://onlinemd5.com/> website and uploading the “Digital Signature Sample File.txt”. Select SHA-1 and you will get exactly the same result as shown in figure 8 below.

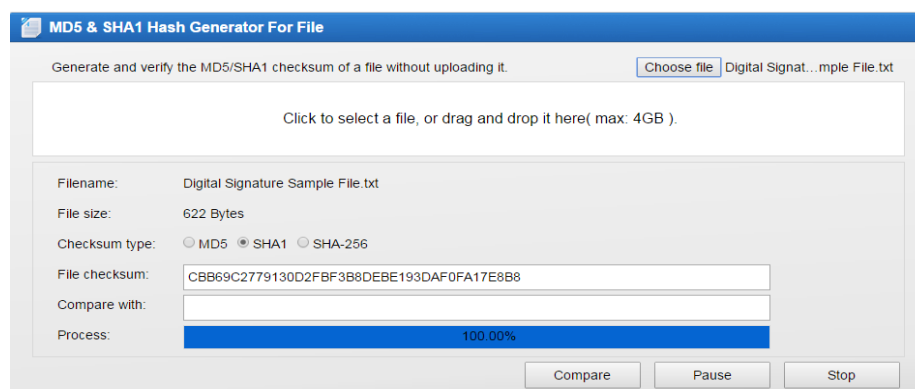


Figure 8: Hash value of “Digital Signature Sample File.txt” using SHA-1.

- Click on the “Generate Key” button. A “Generate RSA Key” window will pop up as shown below in figure 9. Click on “Generate Prime Numbers” Another window will pop up with several default options. Keep the default options. Click on “Generate Prime Numbers” and then click on “Apply Primes”. The original “Generate RSA Key” window is now populated as shown below. Click on “Store Key”. The window will disappear.

Choose two prime numbers p and q. The number $N = pq$ is the public RSA modulus and $\phi(N) = (p-1)(q-1)$ is the Euler phi function. Public key e is coprime to $\phi(N)$. The private key $d = e^{-1} \pmod{\phi(N)}$ is calculated from this.

Prime number entry

Prime number p: 2423163384625797737617864374887

Prime number q: 1434298724673852070184430816748

Generate prime numbers... p and q are prime numbers.

RSA parameter

Length: 304 bit

RSA modulus N: 3475540152245156575672595655541 (public)

$\phi(N) = (p-1)(q-1)$: 3475540152245156575672595655541 (secret)

Public key e: $2^{16}+1$ e does not divide $\phi(N)$.

Private key d: 9391918472963627104561037133166

Store key Cancel

Figure 9: Populated “Generate RSA Key” window.

- From the main window, click “Encrypt Hash Value” button. Then click on “Encrypted Hash Value” button. Notice that information about the hash value is displayed at the display window.
- Click on the “Provide Certificate” button. A new window called “Create Certificate and PSE” will appear. Click on the “Import Certificate and Key” button. Another window will appear as shown below in figure 10. Select your secret key from the list. Type the pin code and then click the “Import” button.
- At the main menu click on the “Encrypt Hash Value” button. Then click on the “Encrypted Hash Value”. This will display the hash value of the encrypted hash in the display window.
- Now, click on “Generate Signature” button. Then click on the “Signature” button. This will display your digital signature in the display window as shown in figure 11 below.
- Press the “Store Signature” button. This will save your digital signature in the repository. If successful a window will pop up as shown in figure 12 below advising you accordingly.

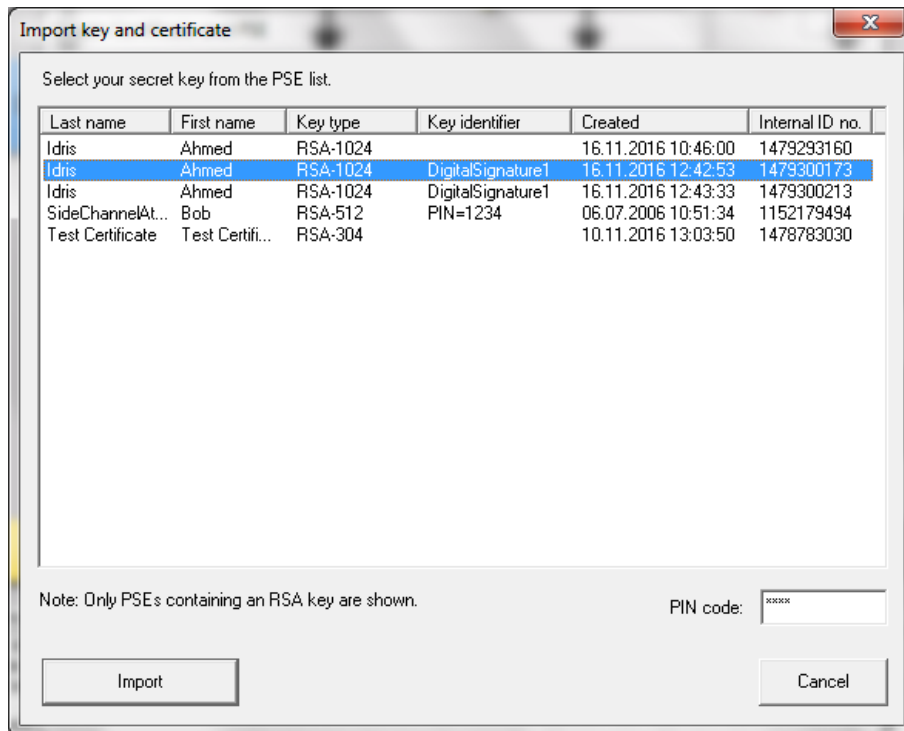


Figure 10: Importing RSA Key Pair.

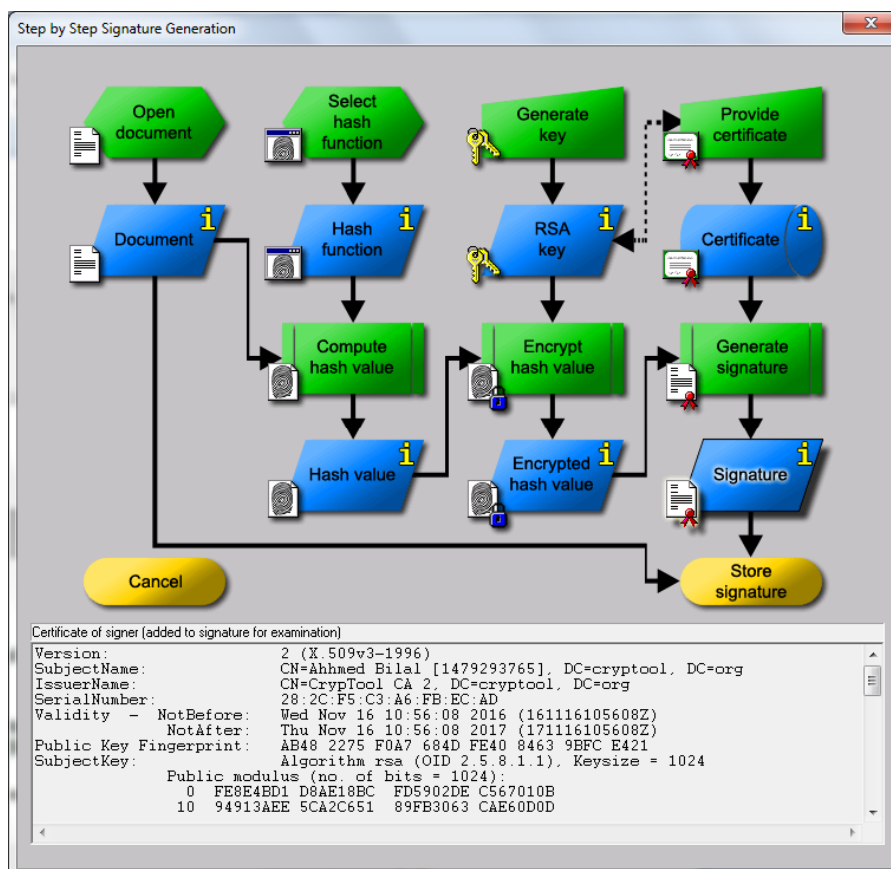


Figure 11: Complete Digital Signature.

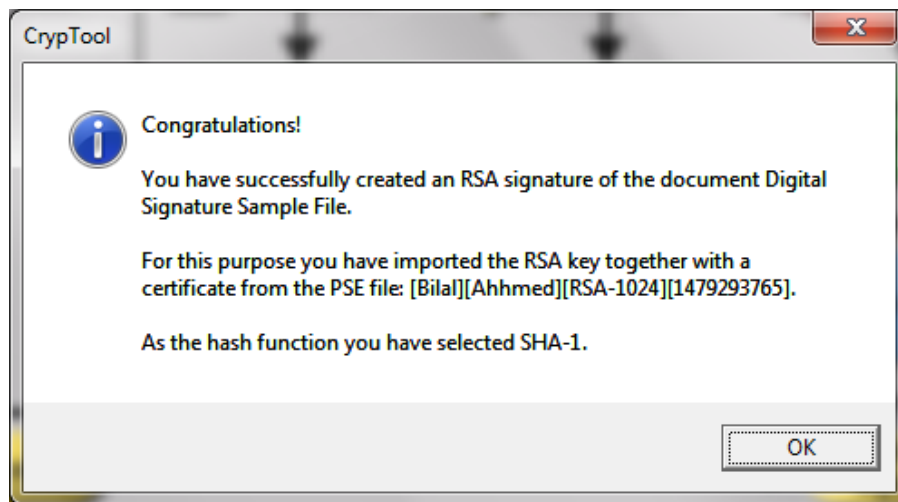


Figure 12: Congratulations Message.

Conclusion

You have successfully created a real life Digital Signature in this lab exercise. The next lab exercise will show you how to use the Digital Signature.