Introduction to Digital Signature

What is a digital signature?

You can use a digital signature for many of the same reasons that you might sign a paper document. A digital signature is used to authenticate digital information — such as form templates, e-mail messages, and documents — by using computer cryptography. Digital signatures help to establish the following assurances:

- Authenticity The digital signature helps to assure that the signer is who he or she claims to be.
- Integrity The digital signature helps to assure that the content has not been changed or tampered with since it was digitally signed.
- Non-repudiation The digital signature helps prove the origin of the signed content to all parties. "Repudiation" refers to the act of a signer denying any association with the signed content.

To make these assurances about a form template, you must digitally sign your form template. You can also enable digital signatures for your form template so that your users can make the same assurances about the forms that they fill out. In either case, the following requirements must be met in order to digitally sign a form or form template:

- The digital signature is valid.
- The certificate associated with the digital signature is current (has not expired).
- The signing person or organization, known as the publisher, is trusted.
- The certificate associated with the digital signature is issued to the publisher by a trusted certificate authority (CA).

Digital signatures are the digital equivalent of regular ink signatures. Just like ink signatures signal your approval or involvement in a paper document and its contents, a digital signature does the same on digital documents. And they do it far better than ink signatures can.

Digital signatures use a Public Key Infrastructure (PKI), a standard format that provides high security and acceptance to your document. This combination of a public key and a private key is what makes a digital signature so safe.

Through the PKI and the processes involved in creating electronic signatures and storing digitally signed documents, you can be sure that your signature cannot be forged, and once signed. The document cannot be altered.

This makes a digital document with an e-signature secure enough to be valid worldwide, including anywhere within the United States and the European Union.

Furthermore, A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document. It's the digital equivalent of a handwritten signature or stamped seal, but it offers far more inherent security. A digital signature is intended to solve the problem of tampering and impersonation in digital communications.

Digital signatures can provide evidence of origin, identity and status of electronic documents, transactions or digital messages. Signers can also use them to acknowledge informed consent.

Digital signatures are based on public key cryptography, also known as asymmetric cryptography. Using a public key algorithm, such as RSA (Rivest-Shamir-Adleman), two keys are generated, creating a mathematically linked pair of keys, one private and one public.

Digital signatures work through public key cryptography's two mutually authenticating cryptographic keys. The individual who creates the digital signature uses a private key to encrypt signature-related data, while the only way to decrypt that data is with the signer's public key.

If the recipient can't open the document with the signer's public key, that's a sign there's a problem with the document or the signature. This is how digital signatures are authenticated.

Digital signature technology requires all parties trust that the individual creating the signature has kept the private key secret. If someone else has access to the private signing key, that party could create fraudulent digital signatures in the name of the private key holder.

What are the benefits of digital signatures?

Security is the main benefit of digital signatures. Security capabilities embedded in digital signatures ensure a document is not altered and signatures are legitimate. Security features and methods used in digital signatures include the following:

Personal identification numbers (PINs), passwords and codes. Used to authenticate and verify a signer's identity and approve their signature. Email, username and password are the most common methods used.

Asymmetric cryptography. Employs a public key algorithm that includes private and public key encryption and authentication.

Checksum. A long string of letters and numbers that represents the sum of the correct digits in a piece of digital data, against which comparisons can be made to detect errors or changes. A checksum acts as a data fingerprint.

Cyclic redundancy check (CRC). An error-detecting code and verification feature used in digital networks and storage devices to detect changes to raw data.

Certificate authority (CA) validation. CAs issue digital signatures and act as trusted third parties by accepting, authenticating, issuing and maintaining digital certificates. The use of CAs helps avoid the creation of fake digital certificates.

Trust service provider (TSP) validation. A TSP is a person or legal entity that performs validation of a digital signature on a company's behalf and offers signature validation reports.

Other benefits to using digital signatures include the following:

Timestamping. By providing the data and time of a digital signature, timestamping is useful when timing is critical, such as for stock trades, lottery ticket issuance and legal proceedings.

Globally accepted and legally compliant. The public key infrastructure (PKI) standard ensures vendor-generated keys are made and stored securely. Because of the international standard, a growing number of countries are accepting digital signatures as legally binding.

Time savings. Digital signatures simplify the time-consuming processes of physical document signing, storage and exchange, enabling businesses to quickly access and sign documents.

Cost savings. Organizations can go paperless and save money previously spent on the physical resources and on the time, personnel and office space used to manage and transport them.

Positive environmental impact. Reducing paper use also cuts down on the physical waste generated by paper and the negative environmental impact of transporting paper documents.

Traceability. Digital signatures create an audit trail that makes internal record-keeping easier for business. With everything recorded and stored digitally, there are fewer opportunities for a manual signee or record-keeper to make a mistake or misplace something.

How Do Digital Signatures Work?

1. The digital signing software

To properly use a digital signature, you can’t just get a JPEG of your signature and paste it on a Word document. You need an electronic signature app to do the job.

Electronic signature solutions, like:

1. Adobe Sign - Tracking and document management
2. Secured Signing - With video confirmation
3. DocuSign - Handles multiple recipients
4. OneSpan Sign - For large and small organizations
5. SignEasy - Reusable templates
6. Signaturely

Make your digital signatures effective by becoming a TSP and certifying the document for you, keeping it safe.

Signaturely, for example, uses ISO 27001 and FIRMA certified data centers managed by Amazon. This allows Signaturely to access AWS data centers to securely store all your data on the cloud, ensuring only your signer's eyes can access it. The data you send to or from Signaturely is also encrypted in transit through 256-bit encryption. Signaturely also gives you the power further to protect your data through 2-Factor Authentication (2FA) to ensure you are the only person accessing your Signaturely account. Electronic signature platforms like Signaturely also handle all parts of the digital signing process for you, ensuring everything about the digital signing process is valid and legally binding.

2. Signing up for a platform for electronic signatures

Since e-signatures are only valid when using the right software, you'll need to choose one that works for you. There are a few options available for digitally signing documents, but you can get started for free by creating a an account.

Any E-Signature application offers a forever-free account allowing you up to three sent documents for free per month.

Start by creating a new account with your name, email address, and password, or sign up with your Google login for an even faster process. Within seconds you'll be able to access platform to create your new document.

3. Create or upload your documents

E-signature applications like Adobe, SecureSign, or Signaturely varies on the processes in signing your document, these processes differs but you will still get the same result. You can get started immediately with the contracts you have. There are no unnecessary processes so that you can set up your contract immediately.

Simply upload your document, and use the editor to add the signature fields. That's it.

You can either upload them directly from your computer or import them directly by connecting your DropBox, Google Drive, OneDrive, or Box accounts.

When the document has been uploaded, simply open it with the editor to add the signature fields, positioning them exactly where someone would sign if they were using an ink signature.

4. Send signature requests

When your document is fully digitized and ready to be signed, it's time to send a signature request to your signees. This process can be completed entirely in-app, letting do the heavy lifting for you.

All you need to do is to add the signees' names and email addresses. If your contract needs to be signed by people in a specific order, you can have the app send the document to them one after the other as each individual signs the agreement.

The e-signature app will then guide your signees through each step of the signing process, starting with creating their e-signature and continuing through the whole signing process step by step until each signature has been added to the document.

5. Wait for your digital documents to return

It is understandable that it can be nerve-wracking to wait for a document to be returned. The longer you wait, the more questions you ask yourself, like: "Have they seen it yet?" When did they see it? Should I call them and ask why they haven't signed the document?

Other application uses dashboard, so you can easily track your documents as they progress. It lets you know who has signed, when, and who has yet to sign. Other apps will remind the signee that their signature is still required, so you don't have to get directly involved. This gentle reminder is usually enough to nudge someone into signing without pressuring them into it.

Using powerful encryption and security keys to protect your document

When your signees open a document, their only option will be to sign it. They can't alter, edit, or change anything on the document. When a user signs the document, the signature records a timestamp. Once all users have signed, the document automatically locks itself, preventing further edits. You and your signees can know that the document you're signing cannot be altered in any way.

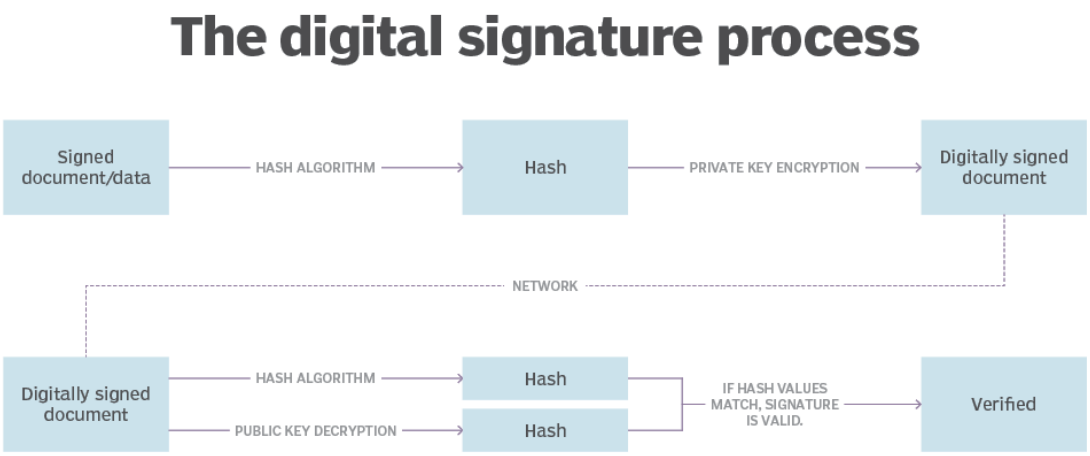
The digital signature signing software, such as an email program, is used to provide a one-way hash of the electronic data to be signed.

A hash is a fixed-length string of letters and numbers generated by an algorithm. The digital signature creator's private key is then used to encrypt the hash. The encrypted hash -- along with other information, such as the hashing algorithm -- is the digital signature.

The reason for encrypting the hash instead of the entire message or document is a hash function can convert an arbitrary input into a fixed-length value, which is usually much shorter. This saves time as hashing is much faster than signing.

The value of a hash is unique to the hashed data. Any change in the data, even a change in a single character, will result in a different value. This attribute enables others to use the signer's public key to decrypt the hash to validate the integrity of the data.

If the decrypted hash matches a second computed hash of the same data, it proves that the data hasn't changed since it was signed. If the two hashes don't match, the data has either been tampered with in some way and is compromised or the signature was created with a private key that doesn't correspond to the public key presented by the signer -- an issue with authentication.



Sample signing scenario

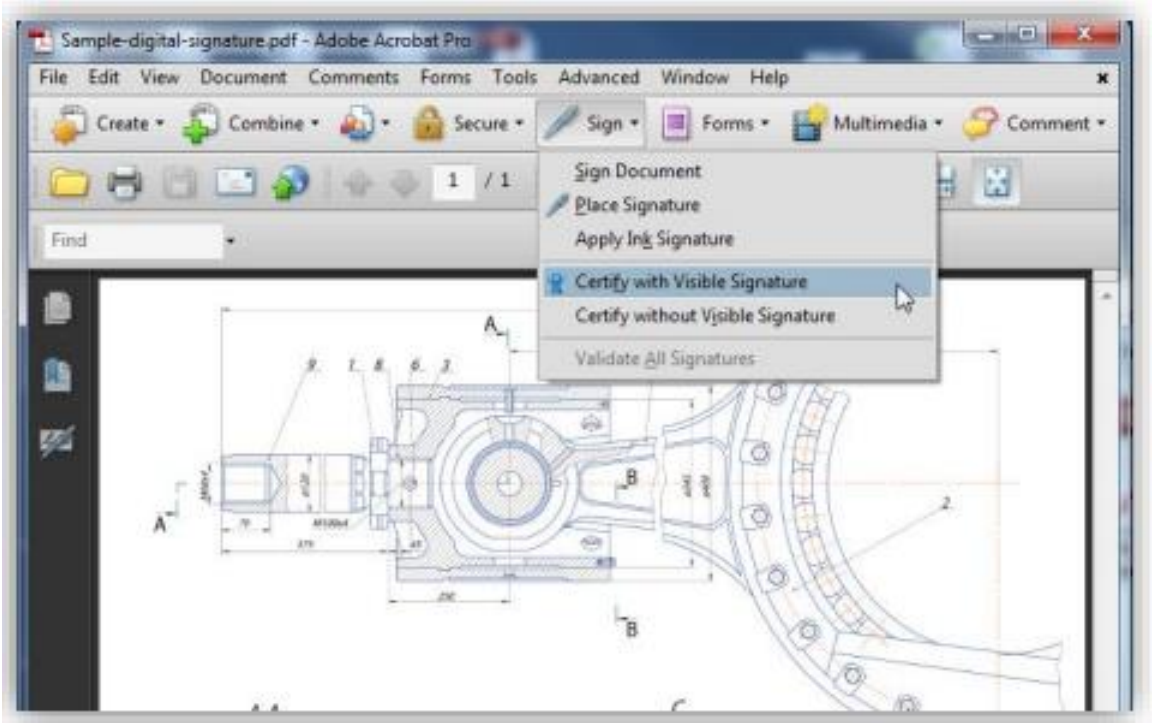
Scenario:

Leslie has a drawing she needs to submit to Joe. There are a number of programs that can be used to apply the signature (e.g., Adobe LiveCycle, BlueBeam, etc.), but in this example, we're going to use Adobe Acrobat. Per this project's specifications, Leslie needs to certify the document and insert her Professional Engineer seal. There are two components to the signature process – creating the signature and validating it. We'll start with the first part – Leslie applying the digital signature.

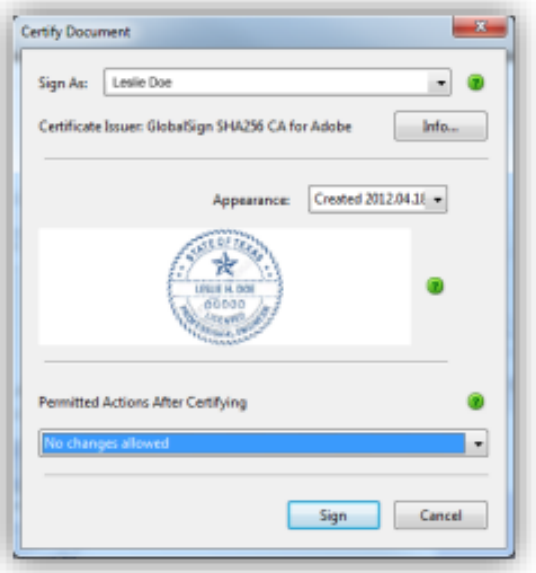
Applying the signature

These are the steps Leslie will go through to apply the digital signature.

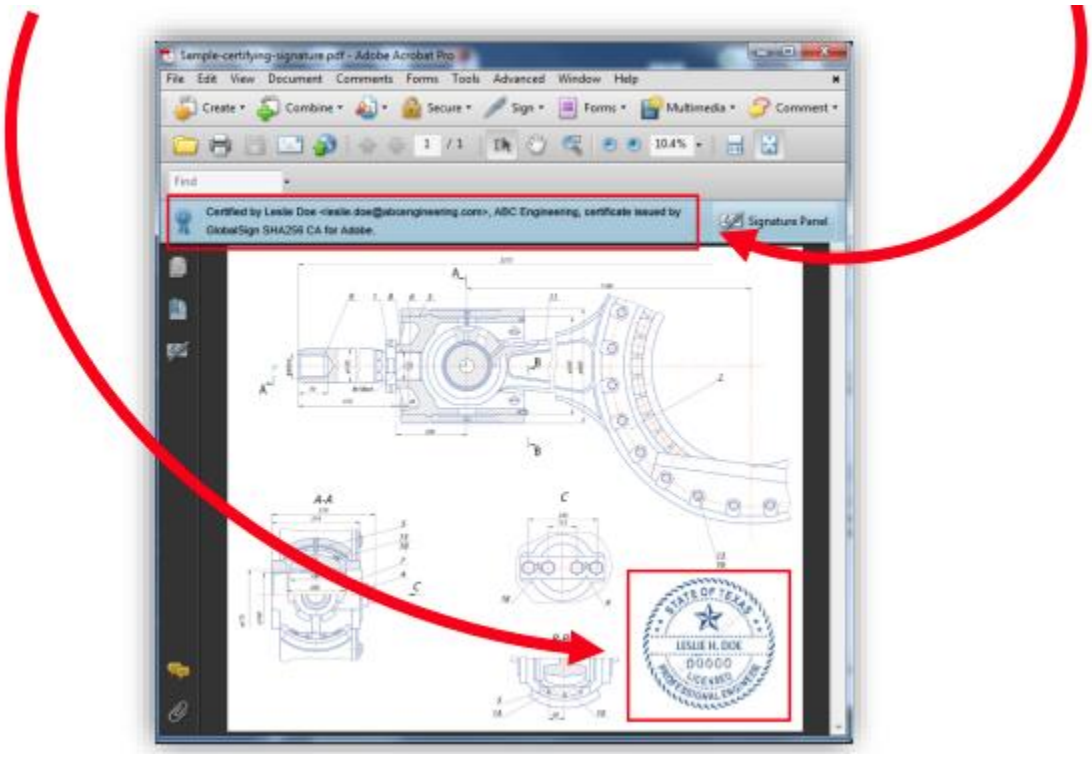
1. Leslie opens her drawing in Acrobat. She clicks “Certify with Visible Signature”.



2. After she chooses where she would like the visible signature to appear, she selects the certificate she wants to use to sign the document, chooses how she wants the signature to appear (her PE seal), and disallows any changes to be made to the document after the signature is applied.



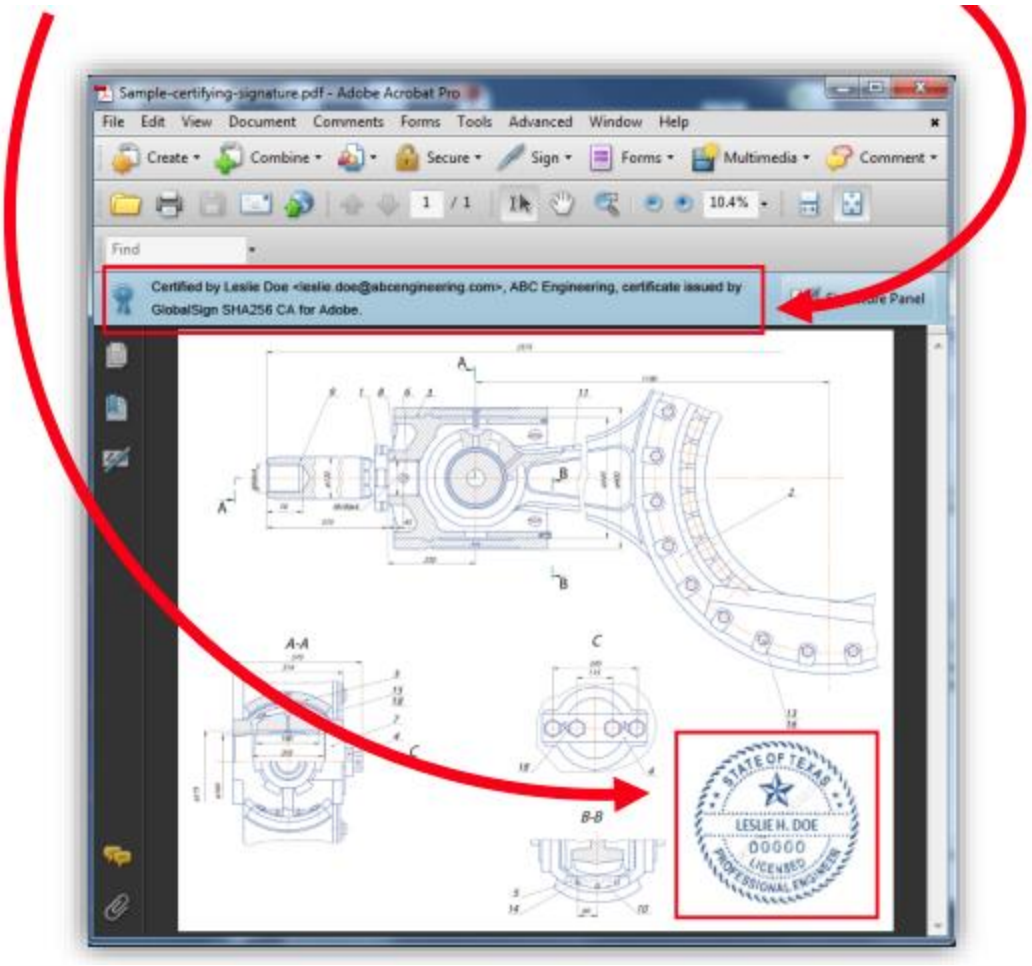
3. Finally, she enters her password and the signature is applied. The document now includes two key trust indicators - a notice at the top of the document stating that it has been certified by Leslie, whose identity was verified by a third party CA (in this case GlobalSign) and her PE seal. The document is ready to send to Joe.



Verifying the signature

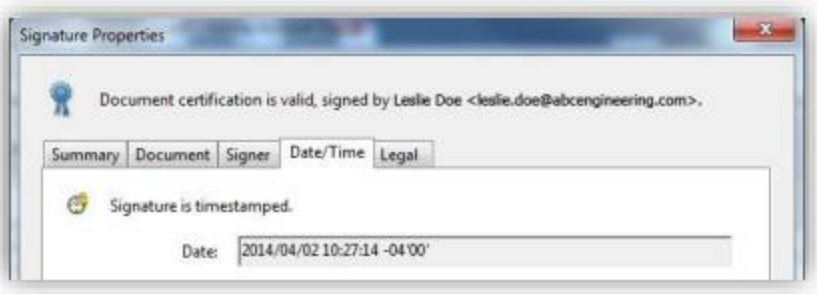
These are the steps Joe will go through to verify Leslie's signature. Note: Adobe Reader automatically verifies the signature, so Joe doesn't actually need to do anything beyond open the document in Reader. Here we'll walk you through what to look for in a digitally signed document and show you how you can find details about the digital signature.

1. Joe opens the PDF in Adobe Reader and sees the same two trust indicators explained above - the notice at the top of the document and Leslie's engineering seal



2. Clicking the seal verifies Leslie's signature and reaffirms that no changes have been made to the document since she signed it.

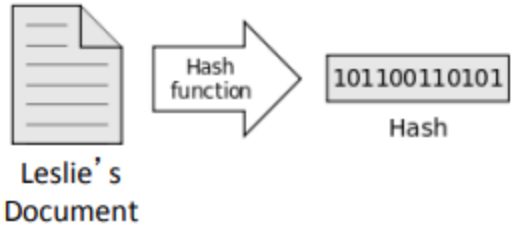
3. Joe can view "Signature Properties" for more information, including a timestamp of when the document was signed.



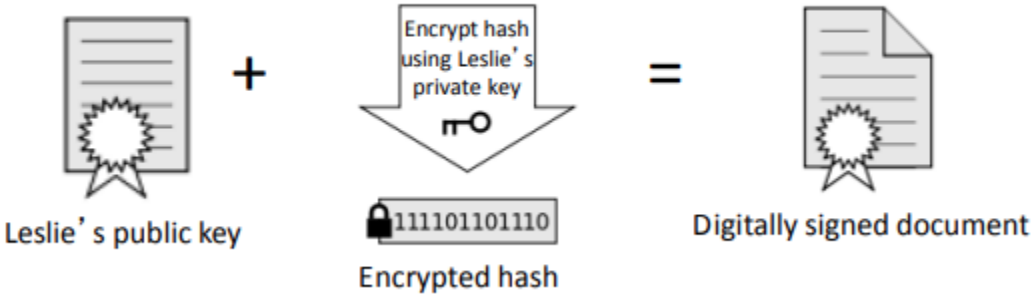
Behind the scenes of the signing process

A. Applying the Signature

1. When Leslie clicks “sign” in Adobe Acrobat, a unique digital fingerprint (called a hash) of the document is created using a mathematical algorithm. This hash is specific to this particular document; even the slightest change would result in a different hash.



2. This hash is encrypted using Leslie’s private key from her digital certificate. The encrypted hash and Leslie’s public key are combined into a digital signature, which is appended to the document.

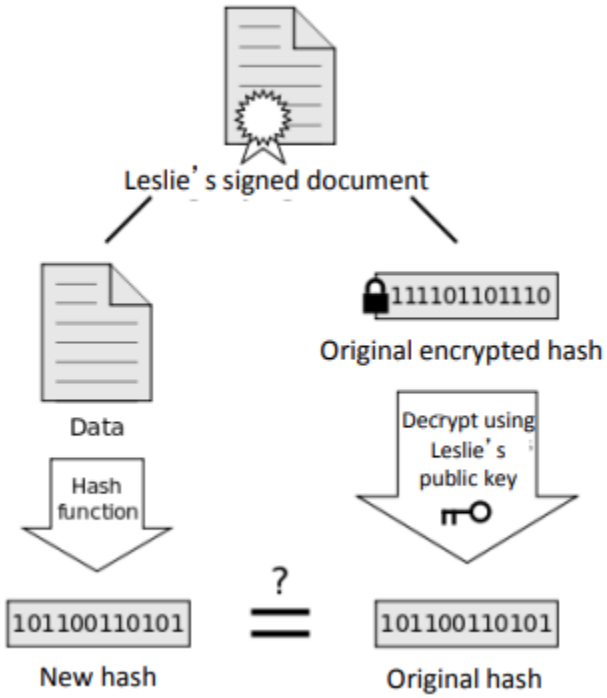


3. Leslie can now share the digitally signed document with Joe

B. Verifying the Signature

1. When Joe opens the signed PDF, Adobe Reader automatically uses Leslie's public key (which was included in the digital signature with the document) to decrypt the document hash.

   Reader calculates a new hash for Leslie's document. If this new hash matches the decrypted hash from Step 1, Reader knows that the document has not been altered and displays the message, "The Document has not been modified since this signature was applied."



Reader also checks the validity of the certificate Leslie used to apply the signature (i.e., that it has not been revoked) and verifies that the public key used in the signature belongs to Leslie.

Other signing scenarios

We ran through a basic scenario, in which Leslie simply needed to send a certified PDF stamped with her PE seal to Joe. There are a number of options when applying digital signatures to fit your specific workflow, document type, or any applicable government regulations.

- Digital version of handwritten signature
- Instead of a PE seal, Leslie could have included an image of her handwritten signature.

- Multiple signatures within one document
- Leslie chose to not allow any changes to be made to the document after she applied her signature, but she could have allowed other digital signatures to be applied.
- Sign multiple pages of the same document
- Leslie could have added her PE seal to multiple pages to the document.

Sources:

How to Create a Self-Signed Digital Certificate in Microsoft Office 2016 (groovypost.com)

What is Digital Certificate? | A Technology Overview from Comodo

PowerPoint Presentation (globalsign.com)

What is a Digital Signature? (techtarget.com)

What Is a Digital Signature (and How Does it Work) | Signaturely

The difference between a digital signature and digital certificate » AET Europe

Cryptography

NOVEMBER 6, 2015 by ed harmoush leave a comment

Do you do online banking? Do you work from home? Do you use VPNs to access company resources? All these would not be possible without Cryptography.

Cryptography is the art of keeping secrets, specifically through any form of communication.

Cryptography has existed for thousands of years, but has become increasingly more important in recent history due to the explosion of the Internet and the need for data privacy and secure online communications.

But what does "secure communication" even mean? Typically, it refers to (at least) these four concepts:

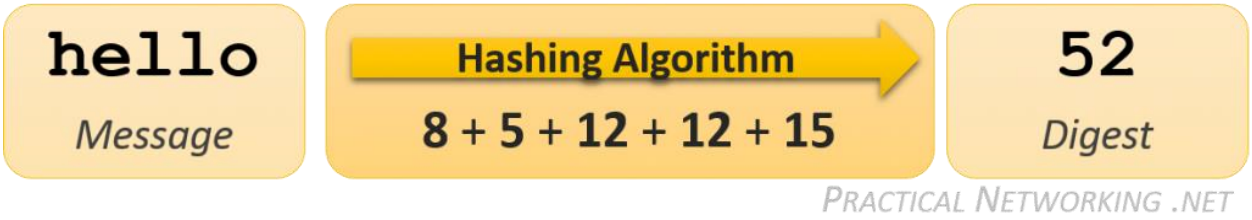
- **Confidentiality** – Assuring only the intended recipients in communication have access to the message.
- **Integrity** – Assuring that the message cannot be modified in transit without the other party being made aware.
- **Authentication** – Assuring the other party is indeed who they claim to be.
- **Anti-Replay** – Assuring the message cannot be maliciously re-transmitted.

Hashing Algorithm

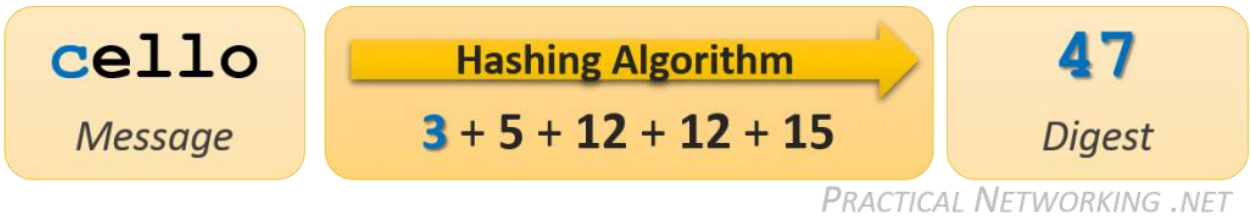
The first concept we need to discuss in our exploration of Cryptography is that of a **Hashing Algorithm**.

A Hashing Algorithm is a mathematical **formula that takes a Message of arbitrary length as input and produces as output a representational sample** of the original data.

For instance, a rudimentary example of a hashing algorithm is simply adding up all the letter values of a particular message. (A=1, B=2, C=3, etc…):



The result of a hashing algorithm is called a message **Digest** (or sometimes Checksum, or Fingerprint). The result of our example hashing on the original message of `hello` was `52`. If someone were to change our original message and process it through the same hashing algorithm, the result would be different:



By comparing the message digests of each calculation, it is easy to determine that our message has changed.

Obviously, the Hashing algorithm used in the example is full of flaws. There are many words that when processed through the example algorithm that might result in the same Digest. Had the original Message been changed to `cellt`, the resulting digest would still be 52, and we would be unaware that the original Message had been altered.

In reality, a legitimate hashing algorithm must maintain four qualities before it is approved for industry usage:

1. **It is mathematically impossible to extract the original message from the digest.**

It should be impossible to reverse the hashing algorithm and recover the original Message knowing just the resulting Digest. In fact, Hashing is sometimes referred to as *one-way encryption*: the message can be encrypted but is impossible to decrypt. This is accomplished using one-way functions within the hashing algorithm.

In a way, our example Hashing algorithm satisfied this condition. It is impossible to derive `hello` knowing only a resulting digest of `52`. Mostly because there could be thousands of messages that result in the identical digest.

2. **A slight change to the original message causes a drastic change in the resulting digest.**

Any minor modification – even as small as changing a single character – to the original Message should greatly alter the computed digest. This is sometimes referred to as the Avalanche effect.

It is possible because a Hashing algorithm is not simply one calculation. It is a series of calculations, done iteratively over and over. As a result, a small change in the beginning, creates an exponentially bigger and bigger change in the resulting digest. Just like a snowball tumbling down a mountain forming an avalanche.

3. **The result of the hashing algorithm is always the same length.**

It is vital for the resulting Digest to not provide any hints or clues about the original Message – including its length. A digest should not grow in size as the length of the Message increases.

In our example Hashing algorithm, the longer the word, the bigger the resulting digest would be as we are adding more and more letters together. However in an industry approved hashing algorithm, hashing the word `hello` would produce a digest the same size as hashing the entire library of congress.

4. **It is infeasible to construct a message which generates a given digest.**

With our example hashing algorithm, if given the digest of `52` , it would not be overly difficult to generate a list of words that might have been the original message. This is what this attribute is trying to prevent.

In a proper hashing algorithm, this should be infeasible — short of attempting every possible combination of messages until you found a match (aka, brute-forcing the algorithm). But even this becomes infeasible given a large enough digest size.

In the next article in this series, we will look at exactly *how* Hashing Algorithms are used to detect modified messages. But for now, we will continue to look at additional aspects of Hashing Algorithms.

Digest Lengths

Below is a table with commonly seen, industry recognized hashing algorithms:

Algorithm	Digest Length
MD5	128 Bits
SHA or SHA1	160 Bits
SHA256	256 Bits
SHA384	384 Bits

Each of these Hashing algorithms satisfy the four cryptography hashing algorithm properties, as described above. The primary difference between each of them is the size of the resulting digest.

As with passwords, it is typically considered that a hashing algorithm which results in a longer digest tends to be regarded as more secure.

Hashing Demonstration with Linux

To take it a step further, I would like to demonstrate how you can use a hashing algorithm from a standard Linux terminal. Note that if you are unfamiliar with Linux, feel free to skip this section — it is not crucial to learning the aforementioned concepts. If you have at least some Linux familiarity, however, it might help to see these algorithms in action.

The standard Linux terminal typically comes with at least two of the Hashing Algorithms mentioned above: MD5 and SHA1. You can use the **`echo`** command along with **`md5sum`** or **`sha1sum`** to run either algorithm on a string of text:

```
$ echo "Practical Networking .net" | md5sum
018aa3ff55842e546c661b7027aed5d7 -
```

If you typed the exact same command into any Linux terminal, the resulting digest would be the exact same. In fact, if you fed the string **`Practical Networking .net`** into any MD5 algorithm, you would see the exact same digest *(remember, the **`echo`** command also appends a new line character to the string)*. If we were to change something small, the resulting digest should be completely different. For example, we can capitalize the ***n*** in *`.net`* and take a look at how the digest changes:

```
$ echo "Practical Networking .Net" | md5sum
6b9298494fb90a1a57efddaee60fbfc1 -
```

We could also run a much larger sample of text through the MD5 algorithm. We can echo the same string 10,000 times and calculate the **`md5sum`**, and notice the resulting digest is still the same length (but the digest is different, of course):

```
$ for i in {1..10000}; do echo "Practical Networking.net"; done | md5sum
938a98abd28c5da2dee6aa07bbc25134 -
```

Try these same examples using **`sha1sum`**. If you don't have easy access to a Linux shell, you can use a free online Linux terminal.

Message Integrity

In the world of secured communications, Message **Integrity** describes the concept of ensuring that **data has not been modified in transit**. This is typically accomplished with the use of a Hashing algorithm. We learned earlier what a Hashing Algorithm does. Now we can take a look at how they are actually used to provide Message Integrity.

The basic premise is a sender wishes to send a message to a receiver, and wishes for the integrity of their message to be guaranteed. The sender will calculate a hash on the message, and include the digest with the message.

On the other side, the receiver will independently calculate the hash on *just* the message, and compare the resulting digest with the digest which was sent with the message. If they are the same, then the message must have been the same as when it was originally sent.

Pretty straight forward. Except for one major problem. Can you guess what it is?

If someone intercepted the message, changed it, and recalculated the digest before sending it along its way, the receiver's hash calculation would also match the modified message. Preventing the receiver from knowing the message was modified in transit!

So how is this issue averted? By adding a Secret Key known only by the Sender and Receiver to the message before calculating the digest. In this context, the Secret Key can be any series of characters or numbers which are only known by the two parties in the conversation.

Before sending the message, the Sender combines the Message with a Secret key, and calculates the hash. The resulting digest and the message are then sent across the wire (*without the Secret!*).

The Receiver, also having the same Secret Key, receives the message, adds the Secret Key, and then re-calculates the hash. If the resulting digest matches the one sent with the message, then the Receiver knows two things:

1. The message was definitely not altered in transit.
2. The message was definitely sent by someone who had the Secret Key — ideally only the intended sender.

This animation reflects this process:

When using a Secret Key in conjunction with a message to attain Message Integrity, the resulting digest is known as the **Message Authentication Code**, or **MAC**. There are many different methods for creating a MAC, each combining the secret key with the message in different ways. The most prevalent MAC in use today, and the one worth calling out specifically, is known as an **HMAC**, or **Hash-based Message Authentication Code**.

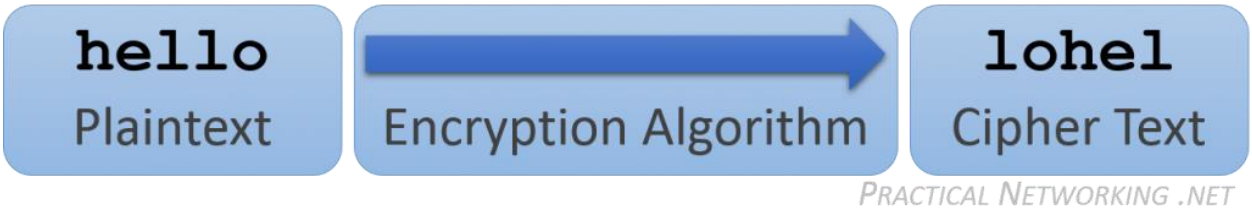
Of course, this doesn't answer the question of "How did the Sender and Receiver establish mutual secret keys?" This is known as the Key Exchange problem, which comes up a few times in cryptography. However, the answer lies outside the scope of the concept of Integrity, and will be discussed in another article in this series.

Confidentiality

Confidentiality is the concept of hiding or scrambling your data so that only the intended recipient has access. This is typically accomplished by some means of **Encryption**.

Data before it has been encrypted is referred to as **Plain text**, or **Clear text**. After the data has been encrypted, it is referred to as **Cipher text**. The Cipher text should be completely unrecognizable, revealing no patterns or hints as to what the original Plain text was. Only the intended receiver(s) should have the ability to **Decrypt** the Cipher text and extract the original Plain text.

The process by which the the Plain text is converted to Cipher text is known as the **Encryption Algorithm**.



In this basic encryption example, all that is needed to reverse the encryption and decrypt the Cipher text is insight into what happened in the Encryption Algorithm. You've probably picked up by now, that to take `hello` and scramble it to `lohel`, all I did was shift the letters forward twice. To undo this, you just need to shift the letters back twice.

There are, however, a few issues with this type of basic encryption:

- **It does not scale**. For each new person you wish to securely exchange data with, you would need to devise a new encryption algorithm. You wouldn't want the communication you had between you and your bank to be secured the same way as it was between you and your employer. How many different algorithms could you come up with before you were forced to reuse them?
- **Once the algorithm is discovered, the security is comprised for *all* time.** Everything that was secured with the compromised algorithm in the past is now fully decryptable. And everything that you might ever continue to secure with that algorithm in the future is now fully decryptable.

- **In the end, all you've done is obfuscate the data.** It may be enough to prevent a passerby from accidentally reading your Clear text, but it won't be enough to thwart a truly determined hacker.

As a result of these weaknesses, modern confidentiality makes use of what is sometimes referred to as **Cryptographic Encryption**. Which is **combining a publicly known encryption algorithm along with a secret key**.

The math behind the algorithm is publicly disclosed, which gives it the benefit of having been vetted by many mathematicians and cryptographers before any particular algorithm is accepted for common use.

The secret key can be a randomly generated set of characters — which makes it easy to produce. It is not difficult to use a different key for each entity you wish to speak securely with, even if the algorithm for each of these parties is the exact same. It is also not difficult to periodically regenerate the secret key, so even if a particular key becomes compromised, only a subset of your communication can be decoded.

There are two types of Cryptographic Encryption: **Symmetric Encryption** and **Asymmetric Encryption**. The main difference between the two types of encryption can be summarized as follows:

- Symmetric encryption – Encrypt and Decrypt using the *same* key.
- Asymmetric encryption – Encrypt and Decrypt using two *different* keys.

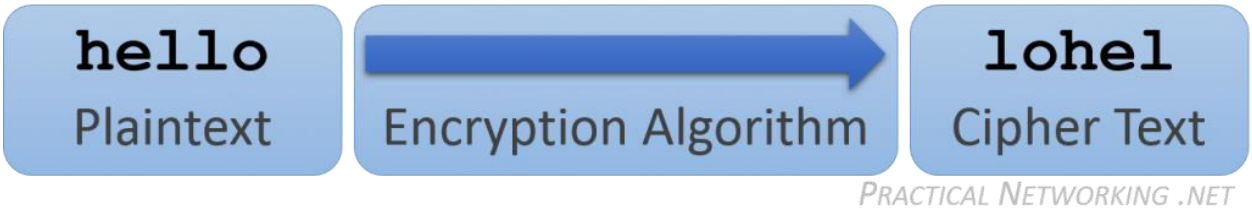
We will look at both of these and how they are used to provide Confidentiality in more detail in next articles in this series.

Confidentiality

Confidentiality is the concept of hiding or scrambling your data so that only the intended recipient has access. This is typically accomplished by some means of **Encryption**.

Data before it has been encrypted is referred to as **Plain text**, or **Clear text**. After the data has been encrypted, it is referred to as **Cipher text**. The Cipher text should be completely unrecognizable, revealing no patterns or hints as to what the original Plain text was. Only the intended receiver(s) should have the ability to **Decrypt** the Cipher text and extract the original Plain text.

The process by which the the Plain text is converted to Cipher text is known as the **Encryption Algorithm**.



In this basic encryption example, all that is needed to reverse the encryption and decrypt the Cipher text is insight into what happened in the Encryption Algorithm. You've probably picked up by now, that to take `hello` and scramble it to `lohel`, all I did was shift the letters forward twice. To undo this, you just need to shift the letters back twice.

There are, however, a few issues with this type of basic encryption:

- **It does not scale**. For each new person you wish to securely exchange data with, you would need to devise a new encryption algorithm. You wouldn't want the communication you had between you and your bank to be secured the same way as it was between you and your employer. How many different algorithms could you come up with before you were forced to reuse them?
- **Once the algorithm is discovered, the security is comprised for *all* time.** Everything that was secured with the compromised algorithm in the past is now fully decryptable. And everything that you might ever continue to secure with that algorithm in the future is now fully decryptable.

- **In the end, all you've done is obfuscate the data.** It may be enough to prevent a passerby from accidentally reading your Clear text, but it won't be enough to thwart a truly determined hacker.

As a result of these weaknesses, modern confidentiality makes use of what is sometimes referred to as **Cryptographic Encryption**. Which is **combining a publicly known encryption algorithm along with a secret key**.

The math behind the algorithm is publicly disclosed, which gives it the benefit of having been vetted by many mathematicians and cryptographers before any particular algorithm is accepted for common use.

The secret key can be a randomly generated set of characters — which makes it easy to produce. It is not difficult to use a different key for each entity you wish to speak securely with, even if the algorithm for each of these parties is the exact same. It is also not difficult to periodically regenerate the secret key, so even if a particular key becomes compromised, only a subset of your communication can be decoded.

There are two types of Cryptographic Encryption: **Symmetric Encryption** and **Asymmetric Encryption**. The main difference between the two types of encryption can be summarized as follows:

- Symmetric encryption – Encrypt and Decrypt using the *same* key.
- Asymmetric encryption – Encrypt and Decrypt using two *different* keys.

We will look at both of these and how they are used to provide Confidentiality in more detail in next articles in this series.

Asymmetric Encryption

Earlier, we learned that Symmetric encryption is an encryption scheme that uses the *same* key to encrypt and decrypt. Conversely, **Asymmetric encryption**, uses *different* keys to encrypt and decrypt. Lets take a look at a simple example.

For the sake of simplicity, let us pretend for this example that there are only the lower case letters `a - z` available. No capitals, no numbers, no symbols. This would give us a total of 26 possible characters.

In the example above, we are taking the plain text of **`hello`**, and encrypting it with an Asymmetric encryption key of `5`. This results in our cipher text, **`mjqqt`**.

Instead of simply reversing the encryption, as you would for a Symmetric encryption, let us instead continue rotating the letters forward `21` more times:

Notice if we continue to move forward, with a Decryption key of `21`, we end up back where we started at the original plain text of **`hello`**.

Now obviously, in this simplistic example, we could have simply rotated backwards with the Encryption key of 5. But in a real Asymmetric encryption algorithm, attempting to re-use the Encryption key (either forwards or backwards) would simply scramble the text further.

That said, there is something significant worth pointing out that we can learn from the Asymmetric encryption example above.

We used an Encryption key of 5, and were able to decrypt successfully with a Decryption key of 21. *BUT*, we could also have used an Encryption key of 21, and a successfully decrypted with a Decryption key of 5. The **Asymmetric keys are mathematically linked.** What one key encrypts, only the other can decrypt — and *vice versa*.

Can you determine another set of keys that would work as an Asymmetric key pair for the simple example above?

A Tale of Two Keys

So what can we do with an Asymmetric Key Pair?

We discussed earlier the existence of the two keys — that they are mathematically linked, and that whatever data is encrypted by one of the keys can only be decrypted by the other key.

 One of these keys is stored securely, and *never* shared with anyone else. This key is from then on referred to as the **Private Key**. In the rest of this series, this key will always be checkered, and point to the left.

 The other key is made available to the world. This key then becomes your **Public Key**. In the rest of the series, this key will always be solid colored, and pointing to the right. The "key pair" will be identified by the matching color.

Every participant in Asymmetric encryption has their own, unique key pair. Each of these keys can be used in different ways in order to attain different security features. These will be outlined in a dedicated article in this series.

Comparison with Symmetric Encryption

Understandably, the math involved in Asymmetric encryption is slightly more complex than what might be required with Symmetric encryption. As a result, the CPU cost of Asymmetric encryption tends to be higher than its symmetric counterpart.

Moreover, a side effect of the math is also that the resulting cipher text often ends up being larger than the original plain text. If you only intend to Asymmetrically encrypt a small amount of text, then this is negligible. But if you are looking to encrypt bulk data transfer, this makes Asymmetric encryption not ideal.

That said, the primary (and most significant) benefit to using Asymmetric encryption is **the Private Key *never* needs to be shared**. As opposed to Symmetric encryption, where the *same* Secret Key must exist on both sides of the conversation.

As a result, Asymmetric encryption is regarded as more secure than its symmetric counterpart. There is no risk of compromise while the key is being transferred (since it never needs to be transferred at all). There is no risk of compromise from the other party's potential lack of security (since the other party never has your Private key).

Using Asymmetric Keys

We've established how Asymmetric encryption makes use of two mathematically linked keys: One referred to as the Public Key, and the other referred to as the Private Key. We've also established that what one key encrypts, only the other can decrypt.

These two attributes allow us to perform two separate operations with a Key Pair.

Asymmetric Encryption

Below is an illustration of Bob (on the right in red) looking to send an encrypted message to Alice (on the left in purple).

Since Bob and Alice are two different entities, they each have their own set of Public and Private Keys. Their public keys are on the inside, available to each other. While their private keys are on the outside, hidden and out of reach.

When Bob has a message he wishes to securely send to Alice, he will use **Alice's Public Key to Encrypt** the message. Bob will then send the encrypted message to Alice. **Alice will then use her Private Key to Decrypt** the message and extract the original message.

Since Bob encrypted the message with Alice's **Public** key, he knows that the only possible key that could extract the message is Alice's **Private** key. And since Alice never shared her key with anyone, Bob knows that only Alice was able to read the message.

Thus, the concept of confidentiality can be provided with an Asymmetric key pair.

Asymmetric Message Signing

But confidentiality isn't the only thing you can do with a Public and Private Key. Remember, *either* key can be used for encryption. This fact can be used to give us one additional feature from an asymmetric key pair.

Let us imagine that now Alice wants to send a message to Bob. This time, however, Alice does not care about the confidentiality of her message. Which is to say, she doesn't care if anyone can read it. But she is very concerned that Bob knows beyond a shadow of a doubt that it was definitely Alice that sent the message.

Alice can use **her own Private Key to encrypt the message**. Which makes it so the only key in the world that can decrypt her message is her Public key — *which she knows Bob (and anyone else) has access to*.

The message is sent to Bob, who then uses **Alice's Public Key to decrypt the message**. If Bob was able to successfully extract a message, and not a scrambled series of bits, then he can be assured that the message must have been originally encrypted by Alice's Private Key. And since Alice never shared her Private Key with anyone, Bob can be assured that Alice indeed sent the message.

This process is known as **Message Signing**. It is a creative use of the fact that the keys are mathematically linked, and that what one key encrypts, only the other can decrypt.

Real World Usage

Now that we have illustrated the basic premise. We can take it a step further and really look at how these concepts are actually used in modern cryptography.

Real World Encryption

Earlier, we discussed that Asymmetric encryption is slower and has properties which make it not ideal for bulk encryption. We should instead find a way to use Symmetric encryption, since it is better suited for bulk data encryption. But with Symmetric encryption, we have to deal with the Key Exchange issue.

The solution is to use what is sometimes referred to as **Hybrid encryption**, which **combines the strengths of both Symmetric and Asymmetric encryption**, while avoiding all their weaknesses.

Let's describe how that works by continuing to use Alice and Bob from above as an example.

Bob starts by randomly generating a Symmetric Secret Key. Then, instead of Bob using Alice's public key to encrypt the *message* directly, Bob uses Alice's Public Key to encrypt the Symmetric Secret Key. This encrypted symmetric key is sent across the wire to Alice.

Alice can then use her Private Key to extract the Secret Key that Bob sent. At this point, both parties now have an identical Secret Key that they can use to *Symmetrically* encrypt as much data as they please, in both directions.

In this way, Bob and Alice use Asymmetric Keys to securely exchange a Symmetric Key, which is then used for Symmetric encryption. They are getting the security of Asymmetric encryption, with the speed and efficiency of Symmetric encryption — the best of both worlds.

Real World Signatures

Similarly, the Message Signing process is more than simply using the Private Key to encrypt the message. Again, the limitations of Asymmetric encryption would end up imposing a limitation on what sort of data can be signed.

Can you guess what method is employed to reduce the message of variable length to a constant, more manageable representational value?

You guessed it… a Hashing algorithm. Lets talk through it using Bob and Alice.

Alice wants to sign a message to Bob. She runs her message through a Hashing Algorithm, and then encrypts the resulting digest with her own Private Key. The encrypted digest then gets sent to Bob, along with the original message.

Bob then uses Alice's public key to decrypt the digest he received, then he independently calculates the hash of the original message. Bob then compares the (now decrypted) digest which was sent, and the digest which he calculated.

If they are the same, then Bob knows that Alice indeed must have sent the original message.

Moreover, Bob also knows that the message has not changed since Alice calculated the original digest — the signature had the bonus effect of also ensuring the Integrity of the original message!

Math is Hard

Most people can wrap their mind around Symmetric encryption fairly easily. Take a starting value, perform some mathematical operation, and you end up with cipher text. To convert it back, you simply perform the operation in reverse.

But Asymmetric encryption is slightly more complicated. Without prior exposure to Asymmetric encryption, its difficult to imagine a mathematical operation that you can perform on a starting value that is *impossible* to reverse. Even if you know the Public Key and the Algorithm used.

To that end, we've added an article as an appendix to the Cryptography series which explores the math behind a widely used Asymmetric algorithm in use today.

If math causes your eyes to glaze over, feel free to skip it, so long as you understand the basic concepts described throughout this series. But if you are slightly curious about how an Asymmetric algorithm works, head on over to the post on the RSA algorithm.

Authentication

In Cryptography, the concept of **Authentication** serves to **provide proof that the other side of a communication is indeed who they claim to be**, and who you intend for them to be.

There are multiple ways to verify the opposing party's Authentication. We will look at three of the most common:

- Username and Password
- Pre-Shared-Key
- Digital Certificates

Username and Password

Using a username and a password to identify who you are to a server is extremely common. Each user on a website can create a unique username, as well as a password tied specifically to that user. If the user is able to reproduce the password, then we can be assured that they are indeed the user they claim to be.

This is how most bank websites and email clients identify who you are.

The password itself should not be sent across the wire. That would easily lead to potential compromise. Instead, the password is run through some sort of hashing algorithm, and the resulting digest is then sent across the wire.

On the receiving end, it would be poor security practice to store the user's passwords directly. Instead, all that is stored is a hash of the password. Then, the digest sent with the user can be compared to the digest in the server's password database to see if they match. If they match, then the user must have had the expected password.

This is why most online websites that use a username and password are unable to recover lost passwords — all they can do is reset them. The password itself is never stored, only the digest of the password — which, as you recall, is impossible to reverse engineer and 'decrypt'.

Within this Authentication scheme, there are three different types of passwords that can exist:

- Something you *know*
- Something you *have*
- Something you *are*

Something you *know*

This is is the common password. You memorize a series of letters, special characters, words, and/or numbers, and you prove you *know* them when asked for the password.

This scheme is susceptible to users using weak passwords, storing them insecurely, or reusing them for different websites.

Something you *have*

This requires you to reproduce a physical object that only you can have in order to validate you are who you say you are. An ATM card or a employee badge are examples.

Today, many websites will send a random code to your phone via SMS when you are trying to log in, forcing you to have possession of your phone to log in. In such a case, even though you are inputting the random code to prove you are who you say you are, the code's purpose is simply to validate that you *have* your phone.

This is also the same concept behind the various authentication tokens in use today. You carry it around, and when you need to identify yourself, you input the code on the token (which changes every 30-60 seconds). If you can put in the code the server is expecting, then you must have *had* the token.

Often, the code you input is further prefixed or suffixed with a password known only to you. This would then create a system that validates who you are with both something you *know* (the password), and something you *have* (the token).

Something you *are*

Lastly, the various types of bio-metric identification fall under the category of something you *are*. Systems like fingerprint scanners or retina scanners or hand-print scanners all identify you based upon an attribute that is physically tied to who you *are*. Only you can have your hand. Only you can have your eyes. And so on.

Facial recognition, or even voice recognition, also falls under this category.

A password can be eavesdropped or shoulder-surfed. A token or mobile phone can be stolen. But bio-metrics can not be compromised without seriously maiming or killing the user being impersonated.

Two-factor Authentication

Many websites or security services offer or require what is known as **Two-factor Authentication.** This means the user is being identified from a password scheme from *two different* categories above.

For instance, the example above of a random code sent to your phone via SMS and your regular username and password is a perfect example of Two-factor authentication. To successfully authenticate against such a system you would need to both *know* your password, as well as *have* your mobile phone.

If a website simply required you to enter two passwords, or a password and a pin, this would *not* qualify as two-factor authentication because both methods of authentication fall under *something you know*.

Pre-Shared-Keys

The concept of Pre-Shared-Key authentication is to share a secret key or passphrase between two communicating nodes, then see if the two nodes can show proof of having said key.

This Pre-Shared-Key (PSK) should be shared out-of-band. For example, if you mean to use PSKs to prove someone's identity on the other side of the Internet, you should not use the Internet to share the key. You might use the phone, or a fax machine, or carrier pigeons.

Later on, if either node can show they have the correct PSK, it proves that the party on the other side of the wire is indeed the same entity which you initially exchanged the PSK with.

Obviously, simply sending the PSK across the wire to prove you have it would be a huge security flaw. We also can't simply run the PSK through a hashing function and send the resulting digest, because an eavesdropper could then capture the digest and spoof our identity in the future by reusing the same digest.

Instead, you would want to combine the PSK with values that are tied to that particular authentication session, so that the hash of the PSK is only good for *that one* session.

For example, in the case of IPsec, both parties generate and publicly exchange a random number. The PSK is then hashed together with both random numbers, and the resulting digest is shared. If both parties can generate the same digest, then they must have had the correct PSK. Furthermore, if an eavesdropper captures the verification digest and tries to reuse it to spoof the identify of one of the parties in a future session, they will be unable to because the future session will have different random numbers.

There are two primary differences between Pre-Shared-Key authentication and Username and Password authentication. The first: the PSK must be initially shared out-of-band. It can not be initially established using the medium upon which you want a secure connection. The second: the PSK is shared among two individuals, whereas a username and password is always unique to each user (or ought to be, at least).

Digital Certificates

The final authentication scheme we are going to discuss is that of Digital Certificates. This is the primary method of identification in use on the Internet. The protocols securing your browsing session when visiting a webpage of HTTPS make heavy use of Digital Certificates (SSL/TLS).

A digital certificate works similar to a driver's license. It contains the identity of a particular individual or website, and it is issued by a governing entity (the state you live in).

Inside a digital certificate is a very important piece of information: a Public Key of an Asymmetric Key Pair. This key is used to verify that the entity who presents the certificate is the true owner of the certificate. Much like your picture or signature on your driver's license.

Recall that with an Asymmetric Key pair, one key is kept private and never shared with anyone, and the other key is made public. Anyone can get a hold of anyone else's digital certificate (and therefore, public key), but theoretically,

only one person can exist with the correlating private key. Before accepting a digital certificate as proof of someone's identity, that someone must provide evidence that they are in possession of the matching private key.

There are two ways this can be verified. We'll take a look at an example of Alice presenting a Digital Certificate to Bob, and how she can provide evidence that she is in possession of the private key.

1. If Alice presents Bob with her Certificate, Bob can generate a random value and encrypt it with Alice's Public Key. Alice should be the only person with the correlating Private Key, and therefore, Alice should be the only person that can extract the random value. If she can then prove to Bob that she extracted the correct value, then Bob can be assured that Alice is indeed the true owner of the certificate.
2. Alice can encrypt a value known to both parties with her Private Key, and send the resulting Cipher Text to Bob. If Bob can decrypt it with Alice's Public Key, it proves Alice must have had the correlating Private Key.

These two methods are the basis for how authentication works with digital signatures.

Anti-Replay

The Problem

Before we can describe the solution, we must first adequately describe the problem Anti-Replay is trying to solve.

Imagine your local bank branch office. Imagine someone going to that branch location, and depositing $100 in cash into their account. At some point following the transaction, that branch location will send some packets to the bank headquarters which essentially state to increase their account balance by $100.

Those packets will cross the branch office's Internet Service Provider (ISP) network on their route to the Bank Headquarters. If a malicious user (or even the depositor themselves) happens to work at the ISP, it would not be overly difficult to capture the packets going from the Branch to the Headquarters that day, and by time reference, narrow down the particular packet(s) that make up the deposit's transaction.

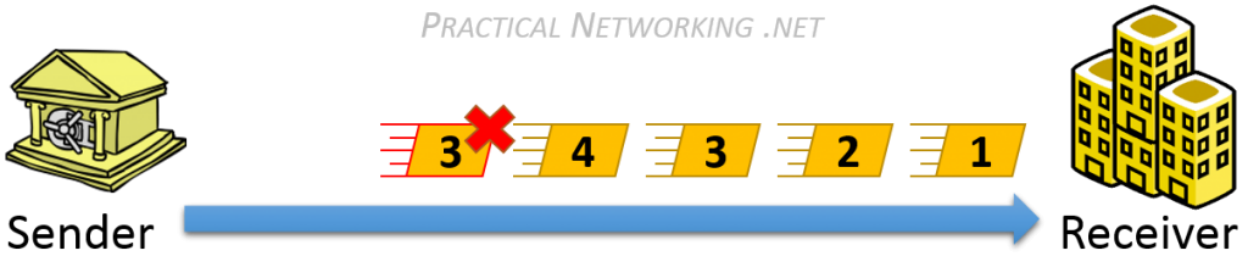
If the packets were protected by Confidentiality, the malicious user would be prevented from reading into the data packet itself. If the packets were protected by an Integrity scheme, the malicious user would be unable to increase the deposit amount to $1000 or $10,000. But, despite those two protections, there would be nothing that would prevent the malicious user from simply duplicating those packets and putting them back on the wire in order to continually re-deposit $100 over and over again.

…nothing if not for Anti-Replay protection, that is.

The Solution

Anti-Replay protection exists specifically to thwart the scenario described above. Anti-Replay injects what is known as a **Sequence Number** into the data packet. This number typically starts at 1, and increases with every packet sent, uniquely identifying one packet from the one prior.

On the receiving end, the last Sequence Number received is tracked. If a packet with a repeated sequence number is ever received, it can be discarded and presumed to be a Replayed packet. Take this example:



Notice, the packet with Sequence Number #3 is a replayed packet. The receiver can easily detect this because they have already received a packet with Sequence Number #3, and was expecting #5 next.

The sequence number, along with the data, will be protected by some sort of Message Integrity scheme to prevent a malicious user from tampering with the numbers in order to send a replayed packet. The sequence number should be included in the Hashing algorithm, along with the Message and the Secret Key. In this way, any illicit modification of any of these fields will be detected.

Sequence Number and Finite Fields

It is important to keep in mind that **the Sequence Number is a finite field**. Which is to say, it does not go on forever.

It has a pre-defined ranged based upon the number of bits allocated in the Data Packet. For example, if the data packet only allows a 16 bit field for the Sequence Number, you would have a total range of 1 – 65536.

It is important to consider this maximum so that it does not impose a limit on the number of packets that can be sent. Or worse, create a looped sequence number vulnerability when the sequence number rolls back to zero.

For example, using the banking transaction above, let us assume the packets only used a 16 bit Sequence Number field, and the captured packets included Sequence Numbers 10,000-10,099. The malicious user could simply wait for the Sequence number to loop past 65,536 and restart at 0, then count out 9,999 packets, and inject the replayed 10,000-10,099. Since the replayed packets would have arrived at the right time, they would have been accepted by the receiver.

This is addressed by forcing a rotation of the Secret Keys used in the Encryption and Integrity algorithms when the Sequence Number resets back to zero. This aids to ensure the continued, indefinite avoidance of replayed packets.

If an attacker attempts to replay a packet before the Sequence Number resets, then the repeated Sequence Number itself would identify the replayed packet. If an attacker attempts to replay a packet after the Sequence Number resets, then the *old packet* secured with the *old keys* will identify the packet was replayed.

Rivest Shaman Adleman

The RSA algorithm is the most widely used Asymmetric Encryption algorithm deployed to date.

The acronym is derived from the last names of the three mathematicians who created it in 1977: Ron **R**ivest, Adi **S**hamir, Leonard **A**dleman.

In order to understand the algorithm, there are a few terms we have to define:

- **Prime** – A number is said to be Prime if it is only divisible by 1 and itself. Such as: 2, 3, 5, 7, 11, 13, etc.
- **Factor** – A factor is a number you can multiple to get another number. For example, the factors of 12 are 1, 2, 3, 4, 6, and 12.
- **Semi-Prime** – A number is Semi Prime if its only factors are prime (excluding 1 and itself). For example:
  12 is *not* semi-prime — one of its factors is 6, which is not prime.
  21 *is* semi-prime — the factors of 21 are 1, **3**, **7**, 21. If we exclude 1 and 21, we are left with 3 and 7, both of which are Prime.
  (Hint: Anytime you multiply two Prime numbers, the result is always Semi Prime)
- **Modulos** – This is a fancy way of simply asking for a remainder. If presented with the problem `12 MOD 5`, we simply are asking for the remainder when dividing 12 by 5, which results in 2.

With that out of the way, we can get into the algorithm itself.

RSA Key Generation

The heart of Asymmetric Encryption lies in finding two mathematically linked values which can serve as our Public and Private keys. As such, the bulk of the work lies in the generation of such keys.

To acquire such keys, there are five steps:

1. **Select two Prime Numbers: P and Q**

This really is as easy as it sounds. Select two prime numbers to begin the key generation. For the purpose of our example, we will use the numbers `7` and `19`, and we will refer to them as `P` and `Q`.

2. **Calculate the Product: (P*Q)**

We then simply multiply our two prime numbers together to calculate the product:

```
7 * 19 = 133
```

We will refer to this number as **N**. Bonus question: given the terminology we reviewed above, what kind of number is N?

3. **Calculate the Totient of N: (P-1)*(Q-1)**

There is a lot of clever math that goes into both defining and acquiring a Totient. Most of which will be beyond the intended scope of this article. So for now, we will simply accept that the formula to attain the Totient on a Semi Prime number is to calculate the product of one subtracted from each of its two prime factors. Or more simply stated, to calculate the Totient of a Semi-Prime number, calculate P-1 times Q-1.

Applied to our example, we would calculate:

```
(7-1)*(19-1) = 6 * 18 = 108
```

We will refer to this as **T** moving forward.

4. **Select a Public Key**

The Public Key is a value which must match three requirements:

- It must be Prime
- It must be less than the Totient
- It must NOT be a factor of the Totient

Let us see if we can get by with the number 3: 3 is indeed Prime, 3 is indeed less than 108, but regrettably 3 is a factor of 108, so we can not use it. Can you find another number that would work? Here is a hint, there are multiple values that would satisfy all three requirements.

For the sake of our example, we will select **29** as our **Public Key**, and we will refer to it as **E** going forward.

5. **Select a Private Key**

Finally, with what we have calculated so far, we can select our Private Key (which we will call **D**).The Private Key only has to match one requirement: The Product of the Public Key and the Private Key when divided by the Totient, must result in a remainder of 1. Or, to put it simply, the following formula must be true:

```
(D*E) MOD T = 1
```

There are a few values that would work for the Private Key as well. But again, for the sake of our example, we will select **41**. To test it against our formula, we could calculate:

```
(41*29) MOD 108
```

We can use a calculator to validate the result is indeed 1. Which means **41** will work as our Private Key.

And there you have it, we walked through each of these five steps and ended up with the following values:

Now we simply pick a value to be used as our Plaintext message, and we can see if Asymmetric encryption really works the way they say it does.

For our example, we will go ahead and use **99** as our Plaintext message.

(the math gets pretty large at this point, if you are attempting to follow along, I suggest to use the Linux Bash Calculator utility)

Message Encryption

Using the keys we generated in the example above, we run through the Encryption process. Recall, that with Asymmetric Encryption, we are encrypting with the Public Key, and decrypting with the Private Key.

The formula to Encrypt with RSA keys is: `Cipher Text = M^E MOD N`

If we plug that into a calculator, we get:

```
99^29 MOD 133 = 92
```

The result of 92 is our Cipher Text. This is the value that would get sent across the wire, which only the owner of the correlating Private Key would be able to decrypt and extract the original message. Our key pair was 29 (public) and 41 (private). So lets see if we really can extract the original message, using our Private Key:

The formula to Decrypt with RSA keys is: `Original Message = M^D MOD N`

If we plug that into a calculator, we get:

`92^41 MOD 133 = 99`

As an experiment, go ahead and try plugging in the Public Key (29) into the Decryption formula and see if that gets you anything useful. You'll notice that, as was stated before, it is impossible to use the same key to both encrypt and decrypt.

Message Signing

But remember, that isn't all we can do with a key pair. We can also use same key pair in the opposite order in order to verify a message's signature.

To do this, we will use the same formulas as above, except this time we will revere the use of the Public and Private Key. We're going to encrypt with the Private Key and see if we can decrypt with the Public Key.

We'll use the same formula to encrypt, except this time we will use the Private Key:

`Signature = M^D MOD N`

If we plug that into a calculator, we get:

`99^41 MOD 133 = 36`

The result of 36 is the Signature of the message. If we can use the correlating public key to decrypt this and extract the original message, then we know that only whoever had the original Private Key could have generated a signature of 36.

Again, the same Decryption formula, except this time we will use the Public Key:

Original **M**essage = **M^E MOD N**

If we plug that into a calculator, we get:

36^29 MOD 133 = **99**

Diffie-Hellman

NOVEMBER 4, 2015 by ed harmoush 6 comments

This article is a part of a series on Cryptography. Use the navigation boxes to view the rest of the articles.

Cryptography

- Hashing Algorithm
- Message Integrity
- Confidentiality
- Symmetric Encryption
- Asymmetric Encryption
- Using Asymmetric Keys
- Authentication
- Anti-Replay
- RSA Example
- **Diffie-Hellman**

How can two people in a crowded room derive a secret that only the pair know, without revealing the secret to anyone else that might be listening?

That is exactly the scenario the Diffie-Hellman Key Exchange exists to solve.

The **Diffie-Hellman Key Exchange is a means for two parties to jointly establish a shared secret over an unsecure channel**, without having any prior knowledge of each other.

They never actually exchange the secret, just some values that both combine which let them attain the same resulting value.

Conceptually, the best way to visualize the Diffie-Hellman Key Exchange is with the ubiquitous paint color mixing demonstration. It is worth quickly reviewing it if you are unfamiliar with it.

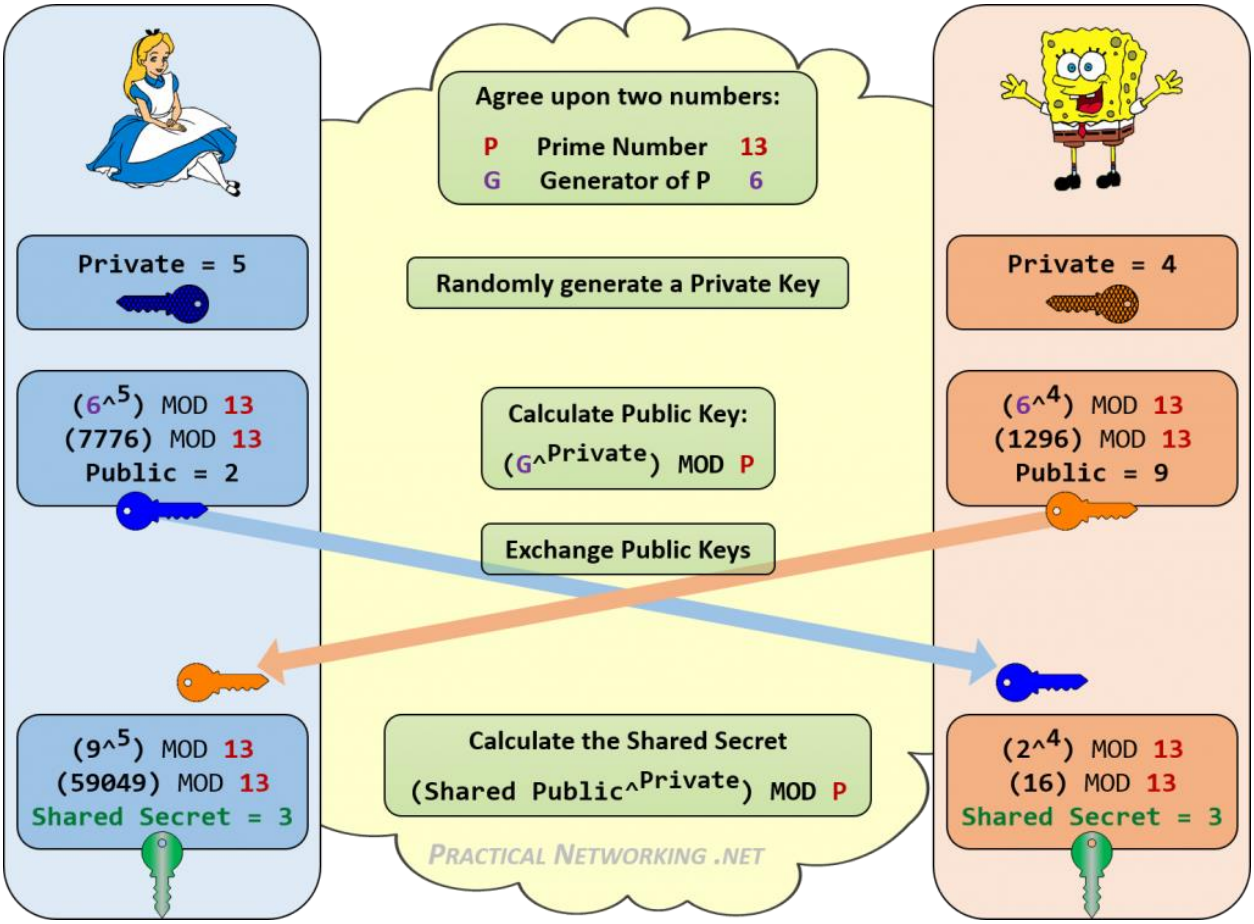
However, in this article we want to go a step further and actually show you the math in the Diffie-Hellman Key Exchange.

DH Math

Before you get into the math of Diffie-Hellman, you will want to have a basic understanding of what a **Prime** number is, and what the **Modulus** operation is

(aka, remainder division). Both of these terms have been defined in another article.

Below is an infographic outlining all the steps of the Diffie-Hellman exchange between Alice and Bob.



Notice how both Alice and Bob were able to attain the same Shared Secret of 3. Anyone listening in on their DH Key exchange would only know the Public Values, and the starting P and G values. There is no consistent way to combine those numbers (13, 6, 2, 9) to attain 3.

DH Numbers

In our example, we used a Prime number of 13. Since this Prime number is also is used as the Modulus for each calculation, the entire key space for the

resulting Shared Secret can only ever be 0-12. The bigger this number, the more difficult a time an attacker will have in brute forcing your shared secret.

Obviously, we were using very small numbers above to help keep the math relatively simple. True DH exchanges are doing math on numbers which are vastly larger. There are three typical sizes to the numbers in Diffie-Hellman:

DH Group 1	768 bits
DH Group 2	1024 bits
DH Group 5	1536 bits

The bit-size is a reference to the Prime number. This directly equates to the entire key space of the resulting Shared Secret. To give you an idea of just how large this key space is:

In order to fully write out a 768 bit number, you would need 232 decimal digits.
In order to fully write out a 1024 bit number, you would need 309 decimal digits.
In order to fully write out a 1536 bit number, you would need 463 decimal digits.

Using the Shared Secret

Once the Shared Secret has been attained, it typically becomes used in the calculation to establish a joint Symmetric Encryption key and/or a joint HMAC Key – also known as Session Keys.

But it is important to point out that the Shared Secret itself should not directly be used as the Secret Key. If it were, all you can be assured of is that throughout the secure conversation you are still speaking to the same party that was on the other side of the Diffie-Hellman exchange.

However, you still have no confirmation or assurance as to who the other party is. Just that no one else can all of a sudden pretend to be them in the middle of your secure conversation.

The generation of the actual Session Keys should include the DH Shared Secret, along with some other value that would only be known to the intended other party, like something from the Authentication scheme you chose.

Sources:

https://www.practicalnetworking.net/series/cryptography/cryptography/

https://www.practicalnetworking.net/series/cryptography/hashing-algorithm/

https://www.practicalnetworking.net/series/cryptography/message-integrity/

https://www.practicalnetworking.net/series/cryptography/confidentiality/

https://www.practicalnetworking.net/series/cryptography/symmetric-encryption/

https://www.practicalnetworking.net/series/cryptography/asymmetric-encryption/

https://www.practicalnetworking.net/series/cryptography/using-asymmetric-keys/

https://www.practicalnetworking.net/series/cryptography/authentication/

https://www.practicalnetworking.net/series/cryptography/anti-replay/

https://www.practicalnetworking.net/series/cryptography/rsa-example/

https://www.practicalnetworking.net/series/cryptography/diffie-hellman/

CRYPTOLOGY

Godwin S. Monserate

MSIT, CCNA/CCAI

Objectives

- Explain the difference between cryptology, cryptography and cryptanalysis
- Understand the basic principles of cryptography
- Understand the operating principles of the most popular tools in the area of cryptography
- List and explain the major protocols used for secure communications

History and Timeline of Cryptography

- 1,900 BCE - Monumental Hieroglyphs of the Old Kingdom of Egypt
- 1,500 BCE - Mesopotamian Secrets of Pottery
- 800 - Al-Kindi, "The Philosopher of the Arabs“
- 1467 - Leon Battista Alberti, "The Father of Western Cryptography“
- 1586 - The Babington Plot
- 1853 - Vigenère’s autokey cipher and the weaker Vigenère cipher
- 1917 - The Vernam Cipher
- 1923 - The Enigma Rotor Machine
- 1940 - Edgar Allan Poe Cracks the Code!
- 1942 - WW2 Japanese Navy Cryptography
- 1943 - The Colossus Computer
- 1953 - The VIC Cipher
- 1975 - The Data Encryption Standard
- 1976 - Diffie-Hellman key exchange
- 1991 - Phil Zimmermann, PGP (Pretty Good Privacy)
- 2001 - Advanced Encryption Standard
- November 2, 2007 -- NIST hash function competition announced.
- 2009 - Bitcoin network was launched.
- 2010 - The master key for High-bandwidth Digital Content Protection (HDCP) and the private signing key for the Sony PlayStation 3 game console are recovered and published using separate cryptoanalytic attacks. PGP Corp. is acquired by Symantec.
- 2012 - NIST selects the Keccak algorithm as the winner of its SHA-3 hash function competition.

History and Timeline of Cryptography

- 2013 - Edward Snowden discloses a vast trove of classified documents from NSA. See Global surveillance disclosures (2013–present)
- 2013 - Dual_EC_DRBG is discovered to have a NSA backdoor.
- 2013 - NSA publishes Simon and Speck lightweight block ciphers.
- 2014 - The Password Hashing Competition accepts 24 entries.
- 2015 - Year by which NIST suggests that 80-bit keys be phased out.

- References
- http://www.cypher.com.au/crypto_history.htm
- https://www.timetoast.com/timelines/the-history-of-cryptography
- https://en.wikipedia.org/wiki/Timeline_of_cryptography#2000_and_beyond

Terminology

- Cryptography: process of making and using codes to secure transmission of information
- Encryption: converting original message into a form unreadable by unauthorized individuals
- Cryptanalysis: process of obtaining original message from encrypted message without knowing algorithms
- Cryptology: science of encryption; combines cryptography and cryptanalysis

Abash/Atbash Cipher

- Abash Cipher is one of the easiest methods for cryptography and crypto-analysis.
- It was first used for the Jewish language but it can be used for the other languages.
- The way of cryptography is to make the last letter of the language to the first letter.
- The method of cryptography in English:

  Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ

  Cipher: ZYXWVUTSRQPONMLKJIHGFEDCBA
- Example: Plain Text: money

  Cipher Text: nlmvb

Answer this Cipher… using Transposition Cipher Encryption

u	y	h	f		t	i	i	o	n
a	t	t		t		h	k	h	e
e	b		u	c	t	l	s	j	n
a	r	o	o	t	i	n	f	m	
r	s	c	s	a	n	e	a	u	
s	d	u	n	e	c	r	a		i
s		i	y		s	m	t	i	p
h			e	e	n	t	l	t	r
y	c	o		p	t	g	y	r	r
	h		p				a	y	

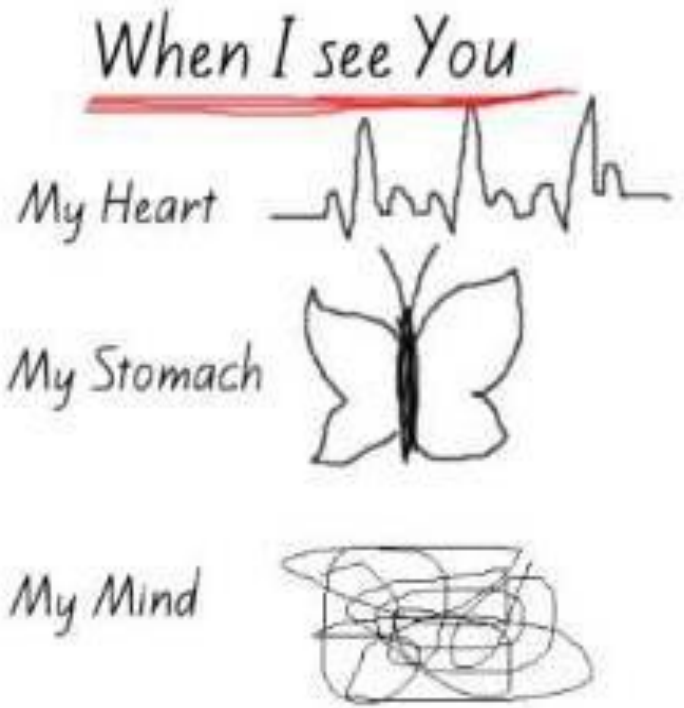
Cryptology Model

BOB

ALICE

Cryptology Model

BOB

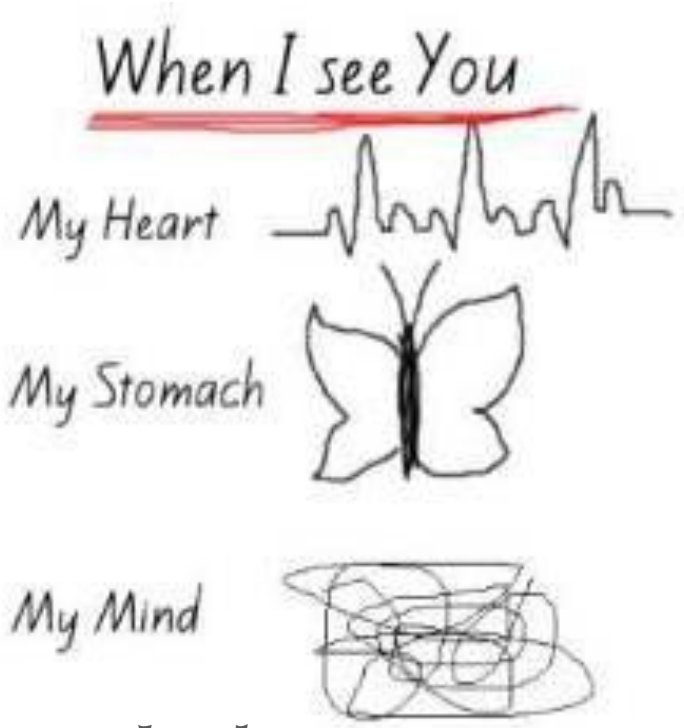


I think I am in love with you

ALICE

Cryptology Model

BOB



I think I am in love with you

ALICE

Cryptology Model

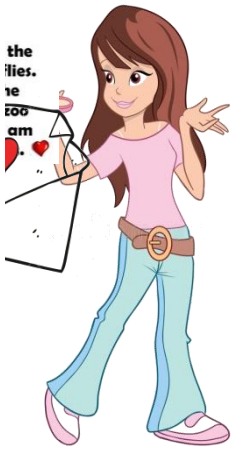
BOB

ALICE

Cryptology Model

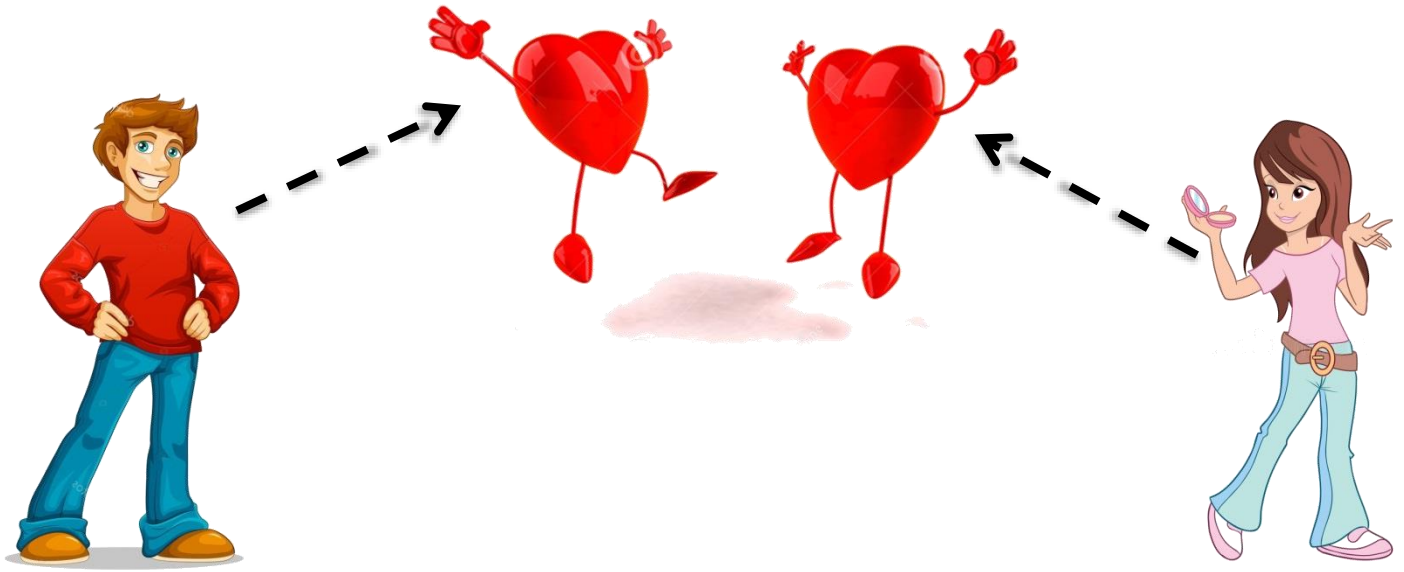
BOB

Forget the butterflies. I feel the whole zoo when I am with you.



ALICE

Cryptology Model



Cryptology Model - Adversary

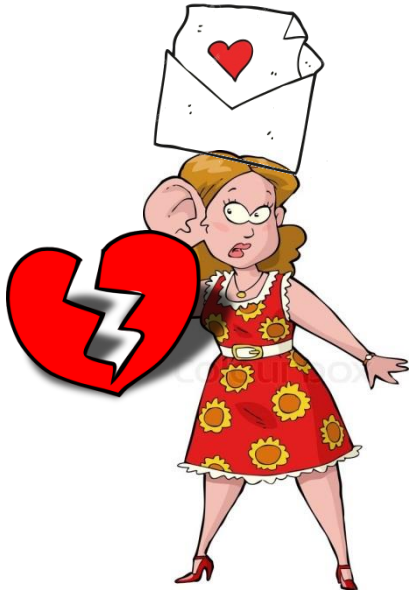
BOB

ALICE

EVE (Eavesdropper)

Cryptology Model

BOB



EVE (Eavesdropper)

ALICE

Cryptology Model

BOB

EVE (Eavesdropper)

ALICE

Cryptology Model

BOB

ALICE

EVE (Eavesdropper)

Cryptology Model – Applying Encryption

ENCRYPTED MESSAGE

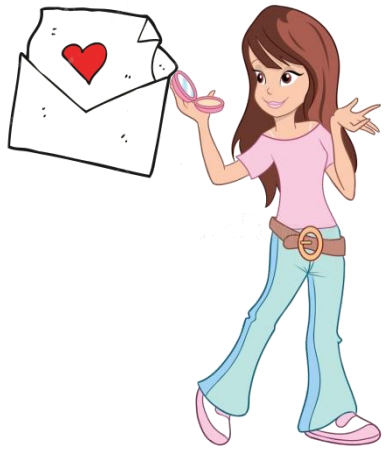
BOB

ALICE

Cryptology Model – Applying Encryption

ENCRYPTED MESSAGE

BOB



ALICE

Cryptology Model – Applying Encryption

ENCRYPTED MESSAGE

BOB

ALICE

EVE (Eavesdropper)

Cryptology Model – Applying Encryption

BOB

ENCRYPTED MESSAGE

EVE (Eavesdropper)

ALICE

Cryptology Model – Applying Encryption

BOB

ENCRYPTED MESSAGE

EVE (Eavesdropper)

ALICE

Cryptology Model – Applying Encryption

DECRYPT THE MESSAGE

BOB

ALICE

EVE (Eavesdropper)

Cryptology Model – Applying Encryption

ENCRYPTED MESSAGE

BOB

ALICE

EVE (Eavesdropper)

Cryptology Model – Applying Encryption

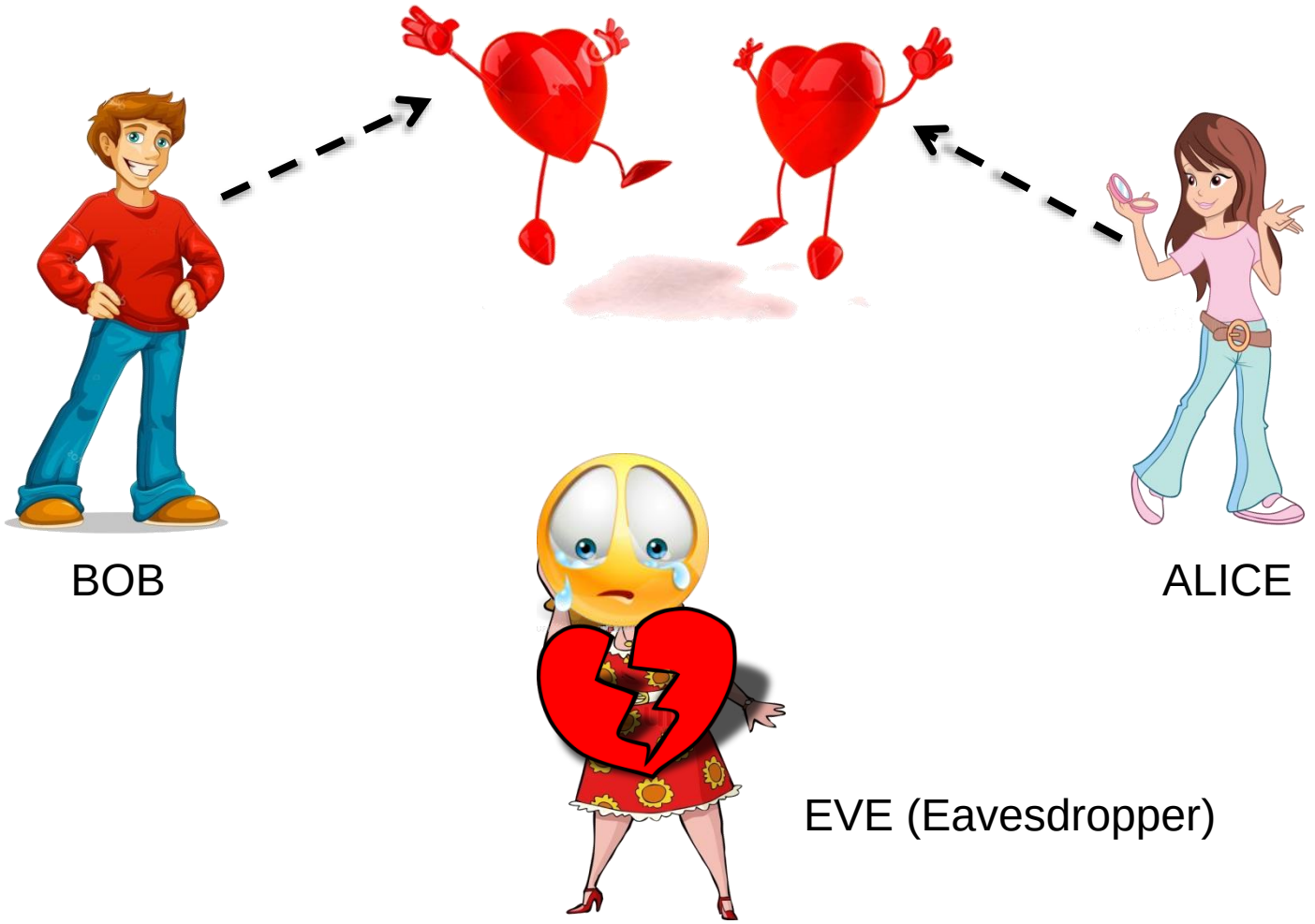
DECRYPT
MESSAGE

BOB

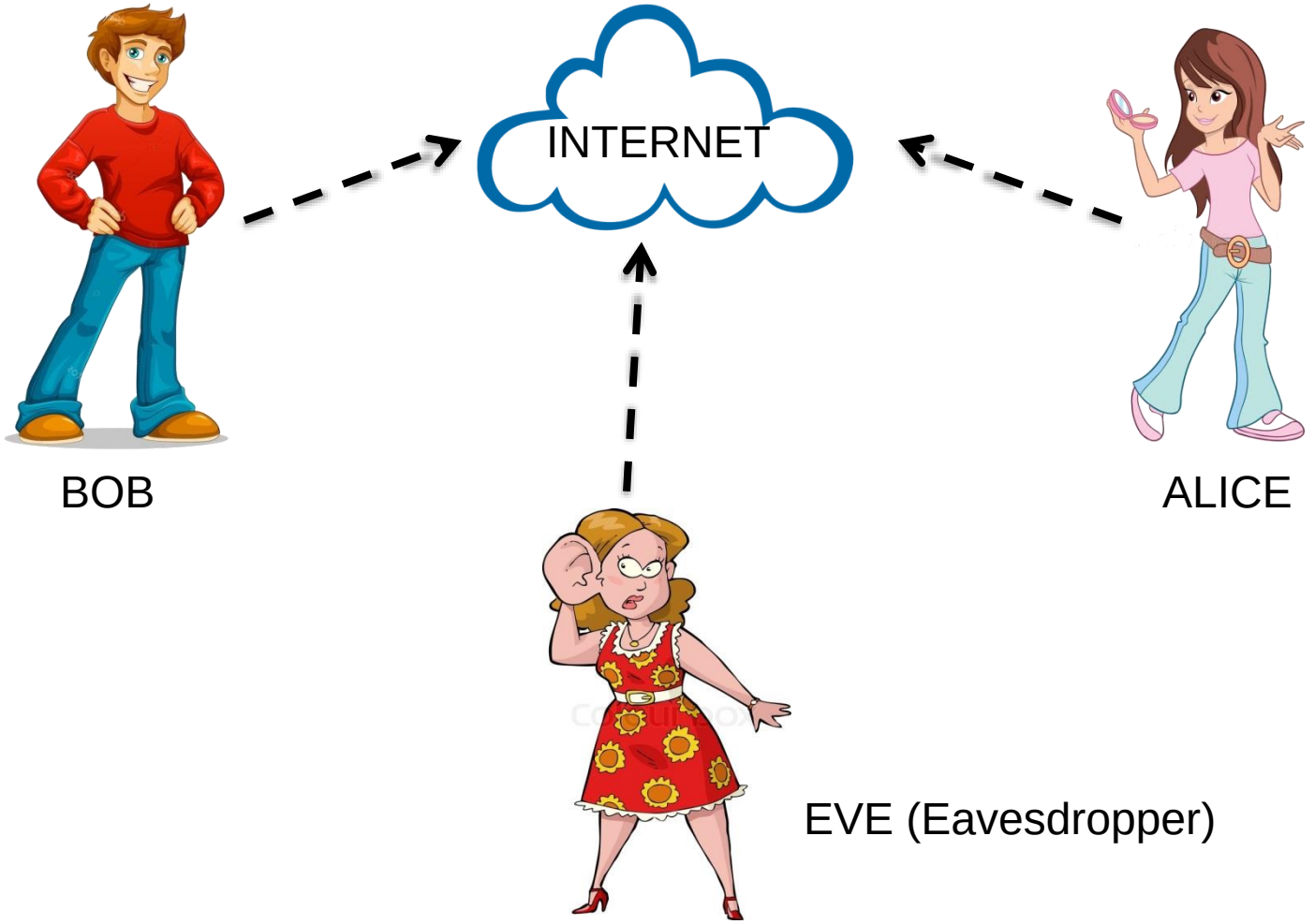
ALICE

EVE (Eavesdropper)

Cryptology Model – Applying Encryption



Cryptology in the Internet



Is there any need for Encryption?

ALICE

Hackers

BOB

Cyberterrorist

Eavesdroppers

Principles of Cryptography

- With emergence of technology, need for encryption in information technology environment greatly increased
- All popular Web browsers use built-in encryption features for secure e-commerce applications

Terminologies

- **Plaintext** – message or information that can be directly read by humans or a machine, ordinary readable text before being encrypted.
- **Ciphertext** is also known as encrypted or encoded information
- **Key** is a variable value that is applied using an algorithm to a string or block of unencrypted text to produce encrypted text, or to decrypt encrypted text.
- **Encryption** is the conversion of electronic data into another form, called ciphertext, which cannot be easily understood by anyone except authorized parties.
- **Decryption** is the process of taking encoded or encrypted text or other data and converting it back into text that you or the computer can read and understand.

Elements of Cryptosystems

- Cryptosystems typically made up of algorithms, data handling techniques, and procedures
- Substitution cipher: substitute one value for another

m: Plaintext
c: Ciphertext

k_e: Encryptionn Key
E: Encryption Program

k_d: Decryption Key
D: Decryption Program

Cryptosystem

m: Plaintext
c: Ciphertext

k_e: Encryption Key
E: Encryption Program

k_d: Decryption Key
D: Decryption Program

Cryptography And Encryption-Based Solutions

- The notation used to describe the encryption process differs depending on the source.
- The first uses the letters m to represent the original message, C to represent the ending ciphertext, and E to represent the encryption process: E(M) = C.
- This formula represents the application of encryption to a message to create ciphertext. D represents the decryption or deciphering process, thus D[E(m)]=m.
- K is used to represent the key, thus E(m, K) = C, or encrypting the message with the key results in the ciphertext.

Different Types of Ciphers

- **Monoalphabetic Ciphers**
- A monoalphabetic cipher mixes up the characters of the alphabet and uses that same arrangement for the entire message.
- The simple case would be to advance each letter some number of spaces, for example moving 10 letters down the alphabet, **A → L**
- There are only 25 possibilities to check, so this type of cipher is trivial to solve

Caesar Cipher

- The **Caesar Cipher**, also known as a *shift cipher,* is a monoalphabetic cipher and one of the oldest and simplest forms of encrypting a message.
- Probably successful for Caesar because his foes were not able to read Roman and were mostly illiterate.
- It is a type of substitution cipher where each letter in the original message (which in cryptography is called the plaintext) is replaced with a letter corresponding to a certain number of letters shifted up or down in the alphabet.

Monoalphabetic Ciphers Tool

How to use Caesar Cipher Encryption

- For example, here's the Caesar Cipher encryption of a full message, using a right shift of 3.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

Plaintext:
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ciphertext:
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Caesar Cipher Mathematical form

- The Caesar Cipher can be expressed in a more mathematical form as follows:

$$E_n(x) = (x + n) \ mod \ 26$$

- Decryption of the encrypted text (called the **ciphertext**) would be carried out similarly, subtracting the shift amount.

$$D_n(x) = (x - n) \ mod \ 26$$

Different Types of Ciphers

- **Polyalphabetic Ciphers**
- keep the same cipher alphabet throughout a message
- thwart the efforts of a cryptanalyst from using letter frequency analysis to break the code.
- each letter is now substituted by several different letters within the message.
- It was invented in 1467 along with a cipher disk to make it more user friendly.
- This disk is called a Vigenère cipher disk, even though it was invented by Leon Battista Alberti 56 years before Blaise de Vigenère was born.
- It is made up of two disks with the normal alphabet around the circumference of one and a mixed up alphabet around the smaller disk.

Polyalphabetic Ciphers Tool

Vigenère cipher disk

Jefferson Wheel Cypher

Vigenere Encryption and Decryption

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Encrypting message

1. Plaintext: **I LOVE YOU ALICE**
2. Create a keyword is **CEBUCITY**. Then, the keyword must be repeated
3. Remove all spaces and punctuation, and dividing the result into 5-letter blocks. As a result, the above plaintext and keyword become the following:

ILOVE YOUAL ICE
CEBUC ITYCE BUC

How to use Vigenere Cipher Encryption

1. To encrypt, pick a letter in the plaintext and its corresponding letter in the keyword
2. use the keyword letter and the plaintext letter as the row index and column index, respectively, and the entry at the row-column intersection is the letter in the ciphertext.
3. For example, plaintext is I, keyword letter is C. Row of I and column of C, intersect at K which is the encrypted result.

Col	i	l	o	v	e	y	o	u	a	l	i	c	e
Row	c	e	b	u	c	i	t	y	c	e	b	u	c
encryption	K	P	P	Q	G	H	G	T	C	P	J	E	M

Vigenere Encryption and Decryption

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenere Cipher Encryption Output

Message: I love you alice

Cipertext: k ppsg ghs cpjwg

Different Types of Ciphers

- **Transpositions and Grills**
- The beginnings of cryptography can be dated to the use of transpositions of hieroglyphs in the tomb of Khnumhotep II in 1900 BC.
- The transpositions weren't necessarily made to keep the text secret, but to add dignity to the words and make it a form of riddle to keep the text interesting.
- A transposition is simply moving the letters around in a prescribed fashion so the resulting cipher text is mixed up and unintelligible.
- A grill is usually a thin paper or metal sheet with a grid where some of the letter positions or syllables or words are cut out. The grill is a type of cipher machine used to make the transposition cipher easier to use.

Transposition Ciphers Tool

How to use Transposition Cipher Encryption

1. Create a plaintext example:

   Meet at three pm today at the usual location

2. Arrange the plaintext according to 6 columns character and create a keyword eg. ZEBRAS

Z	E	B	R	A	S
m	e	e	t	a	t
t	h	r	e	e	p
m	t	o	d	a	y
a	t	t	h	e	u
s	u	a	l	l	o
c	a	t	i	o	n

How to use Transposition Cipher Encryption

1. Arrange the plaintext according to 6 columns character using the keyword ZEBRAS
2. Look for Alphabetic sequence of the word ZEBRA e.i. A, B, E, R, S, Z

aeaelo

Z	E	B	R	A	S
m	e	e	t	a	t
t	h	r	e	e	p
m	t	o	d	a	y
a	t	t	h	e	u
s	u	a	l	l	o
c	a	t	i	o	n

How to use Transposition Cipher Encryption

1. Arrange the plaintext according to 6 columns character and create a keyword eg. ZEBRAS
2. Look for Alphabetic sequence of the word ZEBRA e.i. A, B, E, R, S Z

aeaelo erotat

Z	E	B	R	A	S
m	e	e	t	a	t
t	h	r	e	e	p
m	t	o	d	a	y
a	t	t	h	e	u
s	u	a	l	l	o
c	a	t	i	o	n

How to use Transposition Cipher Encryption

1. Arrange the plaintext according to 6 columns character and create a keyword eg. ZEBRAS
2. Look for Alphabetic sequence of the word ZEBRA e.i. A, B, E, R, S Z

aeaelo erotat

ehttua

Z	E	B	R	A	S
m	e	e	t	a	t
t	h	r	e	e	p
m	t	o	d	a	y
a	t	t	h	e	u
s	u	a	l	l	o
c	a	t	i	o	n

How to use Transposition Cipher Encryption

1. Arrange the plaintext according to 6 columns character and create a keyword eg. ZEBRAS
2. Look for Alphabetic sequence of the word ZEBRA e.i. A, B, E, R, S Z

aeaelo erotat
ehttua tedhli

Z	E	B	R	A	S
m	e	e	t	a	t
t	h	r	e	e	p
m	t	o	d	a	y
a	t	t	h	e	u
s	u	a	l	l	o
c	a	t	i	o	n

How to use Transposition Cipher Encryption

1. Arrange the plaintext according to 6 columns character and create a keyword eg. ZEBRAS
2. Look for Alphabetic sequence of the word ZEBRA e.i. A, B, E, R, S Z

aeaelo erotat
ehttua tedhli
tpyuon

Z	E	B	R	A	S
m	e	e	t	a	t
t	h	r	e	e	p
m	t	o	d	a	y
a	t	t	h	e	u
s	u	a	l	l	o
c	a	t	i	o	n

How to use Transposition Cipher Encryption

1. Arrange the plaintext according to 6 columns character and create a keyword eg. ZEBRAS
2. Look for Alphabetic sequence of the word ZEBRA e.i. A, B, E, R, S Z

aeaelo erotat
ehttua tedhli
tpyuon mtmasc

Z	E	B	R	A	S
m	e	e	t	a	t
t	h	r	e	e	p
m	t	o	d	a	y
a	t	t	h	e	u
s	u	a	l	l	o
c	a	t	i	o	n

How to use Transposition Cipher Encryption

1. Look for Alphabetic sequence of the word ZEBRA e.i. A, B, E, R, S Z
2. The result will be…

Z	E	B	R	A	S
m	e	e	t	a	t
t	h	r	e	e	p
m	t	o	d	a	y
a	t	t	h	e	u
s	u	a	l	l	o
c	a	t	i	o	n

Meet at three pm today at the usual location

aeaelo erotat ehttua tedhli tpyuon mtmasc

Symmetrical Stream Cipher

- **Vernam** cipher: developed at AT&T; uses set of characters once per encryption process
- Digital bit-wise XOR
- The Vernam Cipher is based on the principle that each plaintext character from a message is 'mixed' with one character from a key stream.
- If a truly random key stream is used, the result will be a truly 'random' ciphertext which bears no relation to the original plaintext. In that case the cipher is similar to the unbreakable One-Time Pad (OTP).

Vernam Cipher Security

- The above procedure is 100% safe if, and only if, the following conditions are all met:
  1. There are only two copies of the key (OTP),
  2. Both sides of the communications link have the same key (OTP),
  3. The key (OTP) is used only once,
  4. The key (OTP) is destroyed immediately after use,
  5. The key (OTP) contains truly random characters,
  6. The equipment is hack proof or uncompromised,
  7. The key (OTP) was not compromised during transport.

How to use Vernam Cipher Encryption

1. Create a plaintext and group the characters 6 columns, find the value of the text in the alphabet and convert it to binary:

Meet at three pm today at the usual location

Plaintext								Plaintext Value in the Alphabet					
meetat	M	e	e	t	a	t		13	5	5	20	1	20
threep	t	h	r	e	e	p		20	8	18	5	5	16
mtoday	m	t	o	d	a	y		13	20	15	4	1	25
attheus	a	t	t	h	e	u		1	20	20	8	5	21
suallo	s	u	a	l	l	o		19	21	1	12	12	15
cation	c	a	t	i	o	n		3	1	20	9	15	14

How to use Vernam Cipher Encryption

Binary Value of the plaintext

Plaintext Value in Binary Digits					
00001101	00000101	00000101	00010100	00000001	00010100
00010100	00001000	00010010	00000101	00000101	00010000
00001101	00010100	00001111	00000100	00000001	00011001
00000001	00010100	00010100	00001000	00000101	00010101
00010011	00010101	00000001	00001100	00001100	00001111
00000011	00000001	00010100	00001001	00001111	00001110

How to use Vernam Cipher Encryption

Decimal Value of the OTP

Decimal Value of the One Time Pad					
14	25	24	20	7	19
8	6	22	19	14	14
14	14	12	2	2	13
18	3	26	5	24	15
25	9	11	3	18	26
7	6	19	19	25	21

How to use Vernam Cipher Encryption

- Create a key (One Time Pad)

One Time Pad - Key					
00001110	00011001	00011000	00010100	00000111	00010011
00001000	00000110	00010110	00010011	00001110	00001110
00001110	00001110	00001100	00000010	00000010	00001101
00010010	00000011	00011010	00000101	00011000	00001111
00011001	00001001	00001011	00000011	00010010	00011010
00000111	00000110	00010011	00010011	00011001	00010101

Binary XOR Operation

The inputs to a binary **XOR** operation can only be **0** or **1** and the result can only be **0** or **1**

The binary **XOR** operation (also known as the binary **XOR** function) will always produce a **1** output if either of its inputs is **1** and will produce a **0** output if both of its inputs are **0** or **1**.

If we call the inputs **A** and **B** and the output **C** we can show the **XOR** function as:

A		B		C
0	XOR	0	->	0
0	XOR	1	->	1
1	XOR	0	->	1
1	XOR	1	->	0

How to use Vernam Cipher Encryption

- Compute for the Cyphertext using XOR

Result of XOR (Plaintext, OTP) value in Binary digits					
00000011	00011100	00011101	00000000	00000110	00000111
00011100	00001110	00000100	00010110	00001011	00011110
00000011	00011010	00000011	00000110	00000011	00010100
00010011	00010111	00000100	00001101	00011101	00011010
00001010	00011100	00001010	00001111	00011110	00010101
00000100	00000111	00000111	00011010	00010110	00011011

- 00001101
- 00001110
- 00000011 = 3
- 00000101
- 00011001
- 00011100

How to use Vernam Cipher Encryption

- Convert the binary value into decimal, use the result and look for the value in the alphabet table

Decimal Value of the XOR(Plaintext, OTP)					
3	28	29	0	6	7
28	14	4	22	11	30
3	26	3	6	3	20
19	23	4	13	29	26
10	28	10	15	30	21
4	7	7	26	22	27

How to use Vernam Cipher Encryption

- Convert the binary value into decimal, use the result and look for the value in the alphabet table

Alphabet Value of the XOR Result					
C	B	C	Z	F	G
B	N	D	V	K	D
C	Z	C	F	C	T
S	W	D	M	C	Z
J	B	J	O	D	U
D	G	G	Z	V	A

How to use Vernam Cipher Encryption

- The result of converting the decimal value to alphabetic characters

- Plaintext
  Meet at three pm today at the usual location

- Ciphertext
  CBCZ FG BNDVK DC ZCFCT SW DMC ZJBJO DUDGGZVA

Cryptographic Algorithms

- Often grouped into two broad categories, symmetric and asymmetric; today's popular cryptosystems use hybrid combination of symmetric and asymmetric algorithms
- Symmetric and asymmetric algorithms distinguished by types of keys used for encryption and decryption operations

Cryptographic Algorithms (continued)

- Symmetric encryption: uses same “secret key” to encipher and decipher message
  - Encryption methods can be extremely efficient, requiring minimal processing
  - Both sender and receiver must possess encryption key
  - If either copy of key is compromised, an intermediate can decrypt and read messages
- The problem is that both the sender and the receiver must own the encryption key.
  - If either copy of the key is compromised, an intermediate can decrypt and read the messages.
  - One of the challenges of symmetric key encryption is getting a copy of the key to the receiver, a process that must be conducted out-of-band to avoid interception.

Cryptographic Algorithms (continued)

- There are a number of popular symmetric encryption cryptosystems.
- Data Encryption Standard (DES): one of most popular symmetric encryption cryptosystems. developed in 1977 by IBM and based on the Data Encryption Algorithm (DEA)
- On May 15, 1973, during the reign of Richard Nixon, the National Bureau of Standards (NBS) published a notice in the Federal Register soliciting proposals for cryptographic algorithms to protect data during transmission and storage.
- Adopted by NIST in 1976 as federal standard for encrypting non-classified information

Cryptographic Algorithms (continued)

- DEA uses a 64-bit block size and a 56-bit key. The algorithm begins by adding parity bits to the key (resulting in 64 bits) and then applies the key in 16 rounds of XOR, substitution, and transposition operations.
- With a 56 bit key, the algorithm has 256 possible keys to choose from (over 72 quadrillion).
- DES was cracked in 1997 when Rivest-Shamir-Aldeman (RSA) put a bounty on the algorithm.

DES Encryption

- This is the encrypted form of
- **M** = 0123456789ABCDEF
- **C** = 85E813540F0AB405

How DES Work?

1. **Step 1: Create 16 subkeys, each of which is 48-bits long.** Create a key.

   K = 133457799BBCDFF1

2. Convert the key into Binary (ASCII ➔ Hex ➔ Binary)

   00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

3. Create blocks of 64 bits

8 Bits									8 Bits									8 Bits									8 Bits							
0	1	2	3	4	5	6	7		8	9	10	11	12	13	14	15		16	17	18	19	20	21	22	23		24	25	26	27	28	29	30	31
0	0	0	1	0	0	1	1		0	0	1	1	0	1	0	0		0	1	0	1	0	1	1	1		0	1	1	1	1	0	0	1

8 Bits									8 Bits									8 Bits									8 Bits							
32	33	34	35	36	37	38	39		40	41	42	43	44	45	46	47		48	49	50	51	52	53	54	55		56	57	58	59	60	61	62	63
1	0	0	1	1	0	1	1		1	0	1	1	1	1	0	0		1	1	0	1	1	1	1	1		1	1	1	1	0	0	0	1

How DES Work?

a. Change the order using the permutation table – these is a randomly generated numbers from 0 - 55

PC – 1 (56 bits)						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

How DES Work?

b. Result after changing the order of the original text using the random permutation table

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
57	49	41	33	25	17	9	1	58	50	42	34	26	18	10	2	59	51	43	35	27	19	11	3	60	52	44	36	63
1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1

29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
55	47	39	31	23	15	7	62	54	46	38	30	22	14	6	61	53	45	37	29	21	13	5	28	20	12	4
1	0	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	1	0	0	0

How DES Work?

c. Next, split this key into left and right halves, L0 and R0, where each half has 28 bits.

LO	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1
RO	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1

How DES Work?

d. With L0 and R0 Refined, we now to create sixteen blocks Ln and Rn, 1<=n<=16. Each pair of blocks Ln and Rn is formed from the previous pair Ln-1 and Rn-1, respectively, for n = 1, 2, ..., 16

Iteration Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

How DES Work?

Example: From original pair, L_0 and R_0 we obtain:

L_0 = 11110000110011001010101011111
R_0 = 01010101011001100111100011111

L_0 = 11110000110011001010101011111
R_0 = 01010101011001100111100011111

How DES Work?

Example: From original pair, L_0 and R_0 we obtain:

L_0 = 11110000110011001010101011111

R_0 = 01010101011001100111100011111

How DES Work?

Example: From original pair, L_0 and R_0 we obtain:

L_1 = 11100001100110010101010111111
R_1 = 10101010110011001111100011110

How DES Work?

Example: Output from the shifted pair L_0 and R_0 we obtain:

L_1 = 111000011001100101010101111 1

R_1 = 101010110011001111100011110

How DES Work?

Example: From shifted pair, L_1 and R_1 we obtain L_2 and R_2

L_1 = 1110000110011001010101011111

R_1 = 1010101011001100111100011110

How DES Work?

Example: From original pair, L_0 and R_0 we obtain:

L_2 = 11000011001100101010101111111
R_2 = 01010101100110011110001111101

How DES Work?

Example: From original pair, L_0 and R_0 we obtain:

L_2 = 11000011001100101010101111111
R_2 = 01010101100110011110000111101

How DES Work?

Example: From original pair, L_0 and R_0 we obtain L_1 and R_1 and L_1 and R_1 using 1 shift

L_0 = 11110000110011001010101011111

R_0 = 01010101011001100111100011111

L_1 = 11100001100110010101010111111

R_1 = 10101010110011001111000111110

L_2 = 11000011001100101010101111111

R_2 = 01010101100110011110001111101

How DES Work?

Basing on the shifting table, L3 and R3 are obtained from L2 and R2, respectively, by **two left shifts**, and L16 and R16 are obtained from L15 and R15, respectively, by one left shift.

In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,..., 28, 1.

How DES Work?

Example: From shifted pair, L_2 and R_2 we obtain:

L_2 = 11000011001100101010101111111
R_2 = 01010101100110011110000111101

How DES Work?

Example: From shifted pair, L_2 and R_2 we obtain:

L_2 = 11100001100110010101010111111
R_2 = 01010101100110011110000111101

How DES Work?

Example: From shifted pair, L_2 and R_2 we obtain:

L_3 = 00001100110010101010111111111
R_3 = 01010110011001111000111101101

How DES Work?

Example: From shifted pair, L_2 and R_2 we obtain:

L_3 = 00001100110010101010111111111
R_3 = 01010110011001111100011110101

How DES Work?

Example: From shifted pair, L_3 and R_3 we obtain:

L_3 = 00001100110010101010111111111
R_3 = 01010110011001111000111101 01

How DES Work?

Example: From shifted pair, L_3 and R_3 we obtain:

L_4 = 00110011001010101011111111100
R_4 = 01011001100111100011110101010

How DES Work?

This will go on until it satisfies the switch table

Iteration Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

How DES Work?

L_5 = 11001100101010101111111110000
R_5 = 01100110011110001111010101010

L_6 = 00110010101010111111111000011
R_6 = 10011001111000111101010101010

L_7 = 11001010101011111111100001100
R_7 = 01100111100011110101010101010

L_8 = 00101010101111111110000110011
R_8 = 10011110001111010101011001100

How DES Work?

L_9 and L_9 Goes back to 1 - shift

L_9 = 01010101011111111000011100110
R_9 = 00111100011110101010101110011

L_{10} and L_{10} goes back again to 2 - shifts

L_{10} = 01010101111111100001100111001
R_{10} = 11110001111010101010111001100

How DES Work?

L_{12} = 01011111111000011001100110010101
R_{12} = 00011110101010101100110011001111

L_{13} = 01111111100001100110011001010101
R_{13} = 01111010101010110011001111100

L_{14} = 11111110000110011001010101010101
R_{14} = 11101010101011001100111100001

L_{15} = 11111000011001100101010101010111
R_{15} = 10101010101100110011111000111

How DES Work?

L_{16} and L_{16} goes back again to 1 - shift

L_{16} = 11110000110011001010101011111
R_{16} = 01010101011001100111100011111

How DES Work?

1. Create a Key of 48 bits by performing DES Algorithm
   a. Change the order using the permutation table – PC 2, a randomly generated numbers from 0 - 47

- We now form the keys K_n, for 1<=n<=16, by applying the following permutation table to each of the concatenated pairs L_nR_n. Each pair has 56 bits, but **PC-2** only uses 48 of these.

How DES Work?

Create a Key of 48 bits

PC – 2 (48 bit)					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

The first bit of $\boldsymbol{K_n}$ is the 14th bit of $\mathbf{L_n}\boldsymbol{R_n}$, the second bit the 17th, and so on, ending with the 48th bit of $\boldsymbol{K_n}$ being the 32th bit of $\mathbf{L_n}\boldsymbol{R_n}$.

How DES Work?

- Use 48 bit permutation for L_1 and R_1

L_1 = 11100001100110010101011111
R_1 = 10101010110011001111100011110

L1R1 is equals to…
1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

Result for 48-bit permutation K1

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

How? refer to the table for conversion

Permutation Table using 48-bit

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4	26	8	16	7	27	20	13	2	41
0	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0	1	1	1	0	1	1	1	1	1

25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
52	31	37	47	55	30	40	51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32
1	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	1	0

Thus the result…

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

How DES Work?

- Use 48 bit permutation for L_1 and R_1

L2R2 is equals to…
110000110011001010101011111101010101100110011110001111 01

L3R3 is equals to…
000011001100101010101111111101010110011001111000111 0101

Result for 48-bit permutation K2 and K3

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101
K_3 = 010101 011111 110010 001010 010000 101100 111110 011001

Permutation Table using 48-bit

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4	26	8	16	7	27	20	13	2	41
0	1	1	1	1	0	0	1	1	0	1	0	1	1	1	0	1	1	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1	1	1	1	1	1	1	0	0	1	0	0	0	1	0	1	0	0

25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
52	31	37	47	55	30	40	51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32
1	0	1	1	1	1	1	1	1	0	0	1	0	0	1	1	1	1	0	0	1	0	1
1	0	0	0	1	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1	0	0	1

Thus the result…

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101

K_3 = 010101 011111 110010 001010 010000 101100 111110 011001

Permutation Table using 48-bit

K_4 = 011100 101010 110111 010110 110110 110011 010100 011101
K_5 = 011111 001110 110000 000111 111010 110101 001110 101000
K_6 = 011000 111010 010100 111110 010100 000111 101100 101111
K_7 = 111011 001000 010010 110111 111101 100001 100010 111100
K_8 = 111101 111000 101000 111010 110000 010011 101111 111011
K_9 = 111000 001101 101111 101011 111011 011110 011110 000001
K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111
K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110
K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001
K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001
K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010
K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010
K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

Summary of Step 1

1. Created 16 subkeys, each of which is 48-bits long.
   a. We created a KEY
   b. Convert the key into binary of 64 bit long
   c. We permutated or changed the order of the 64 bit binary digit using the **PC-1 table**
   d. We shifted the binary according to the shift table
   e. We permutated the shifted key on a 48 bit long key **PC-2 table.**

Step 2: Encode each 64-bit block of data.

1. Create a 64 bits of the message data **M**
   - **M** = 0123456789ABCDEF
   - **M** = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
2. *Use an initial permutation* (**IP)** of the 64 bits of the message data. (use IP table)
3. Perform permutation similar to 56 bit and 48 bit permutation.

64-bit long message in binary

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1

34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
0	0	0	1	0	0	1	1	0	1	0	1	0	1	1	1

50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
1	0	0	1	1	0	1	1	1	1	0	1	1	1	1

Step 2: Encode each 64-bit block of data.

Initial Permutation (IP)							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Permutated Message in IP 64-bit table

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4	62
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
54	46	38	30	22	14	6	64	56	48	40	32	24	16	8	57
1	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	41	33	25	17	9	1	59	51	43	35	27	19	11	3	61
1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	1

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
53	45	37	29	21	13	5	63	55	47	39	31	23	15	7
1	1	1	0	0	0	0	1	0	1	0	1	0	1	0

Permutated Message in IP 64-bit table

- **M** = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
- **IP** = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Step 2: Continuation

- Next divide the permuted block **IP** into a left half L_0 of 32 bits, and a right half R_0 of 32 bits.

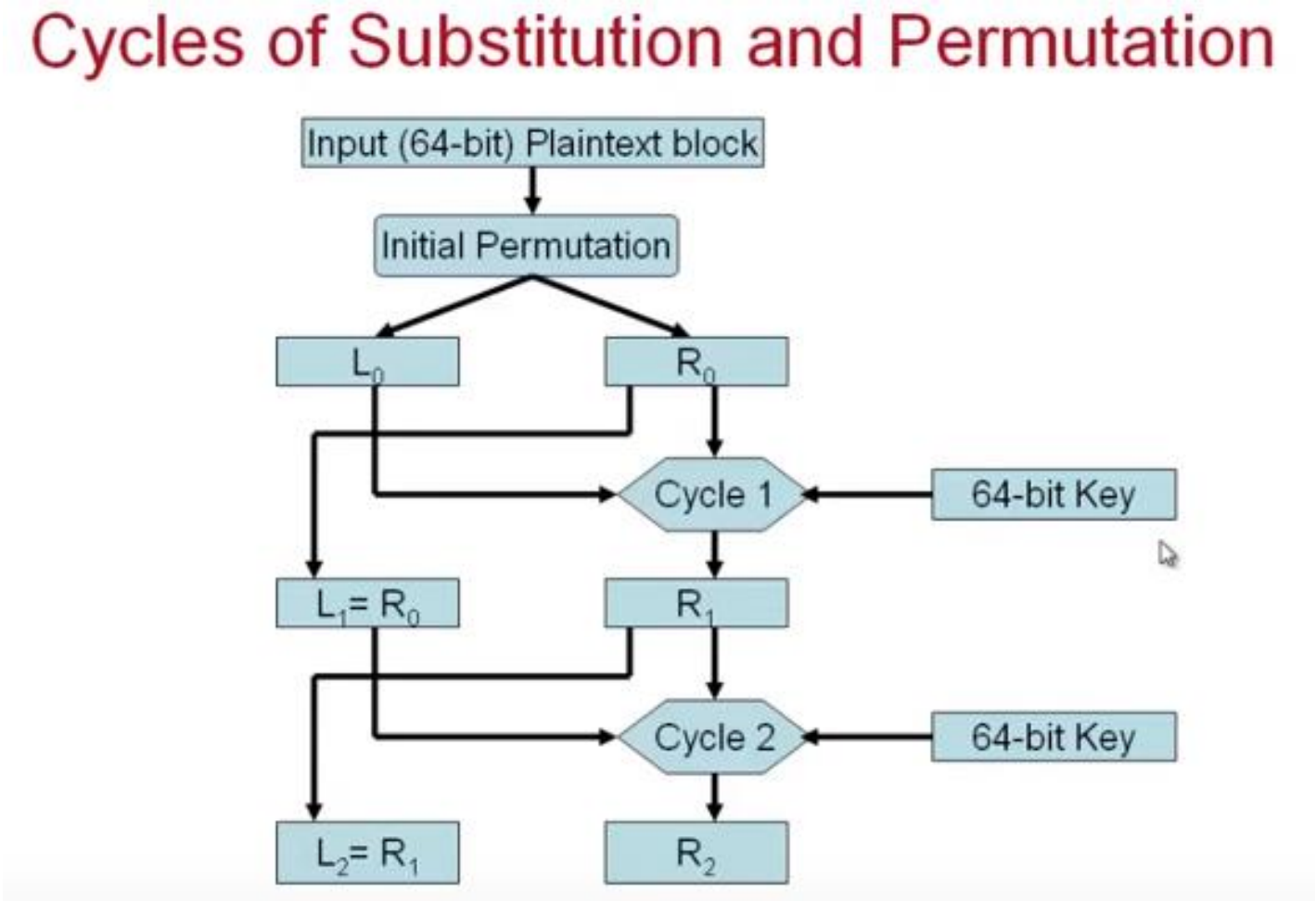
- L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
- R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

Step 2: Continuation

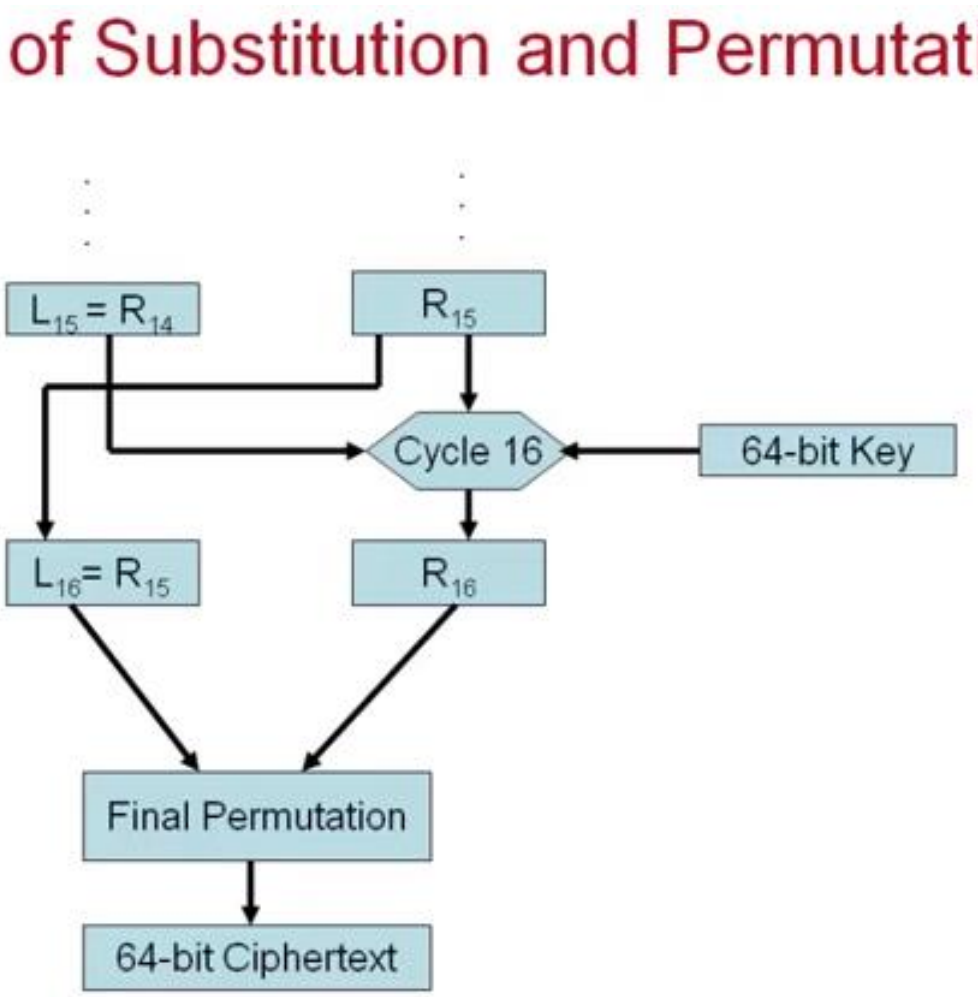
- We now proceed through 16 iterations, for 1<=***n***<=16, using a function ***f*** which operates on two blocks--a data block of 32 bits and a key K_n of 48 bits--to produce a block of 32 bits.
- **Let + denote XOR addition, (bit-by-bit addition modulo 2)**. Then for **n** going from 1 to 16 we calculate

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1}, K_n)$$

Step 2: Continuation



Step 2: Continuation



Final Output

- This is the encrypted form of
- **M** = 0123456789ABCDEF
- **C** = 85E813540F0AB405
- Rivest-Shamir-Aldeman (Cracked DES)

Cryptographic Algorithms (continued)

- Asymmetric Encryption (public key encryption)
  - Uses two different but related keys; either key can encrypt or decrypt message
  - If Key A encrypts message, only Key B can decrypt
  - Highest value when one key serves as private key and the other serves as public key

Examples of Asymmetric Encryption

- Examples of **asymmetric encryption or public key encryption** are **DSA, RSA and PGP**.
- **RSA** is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called *public key cryptography*, because one of them can be given to everyone.
- The other key must be kept private. It is based on the fact that finding the factors of an integer is hard (the factoring problem).

Examples of Asymmetric Encryption

- RSA stands for Ron **Rivest**, Adi **Shamir** and Leonard **Adleman**, who first publicly described it in 1978.
- A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key.
- The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message.
- Using the acoustic cryptanalysis, carried out by Daniel Genkin, Adi Shamir (who co-invented RSA), and Eran Tromer, uses what's known as a *side channel attack*.

How does RSA Work?

- **Step 1 –** Choose two prime numbers, Prime1 and Prime2

- Prime1 and Prime2 should be very large prime numbers, at minimum 100 digits long but as larger is more secure and less efficient.
- Prime 1 and Prime2 should not be the same prime number

```
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97;
```

- p = prime 1; q = prime 2; n = p x q

How does RSA Work?

- **Step 1 –** Choose two prime numbers, Prime1 and Prime2

```
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97;
```

p = 2 ; q = 7 ; n = p x q

n = 2 x 7

n = 14

(this number will become the modulo or remainder of the key)

How does RSA Work?

- **Step 2 –** Get the value of *n* (Product of 2 prime numbers

p = 2 ; q = 7 ; *n* = p x q

n = 2 x 7

***n* = 14**

(this number will become the modulo or remainder of the key)

How does RSA Work?

Step 3 – Find the Totient of ProductOfPrimes

- Totient - The **totient** function , also called Euler's **totient** function, is defined as the number of positive integers that are relatively prime to (i.e., do not contain any factor in common with) , where 1 is counted as being relatively prime to all numbers.
- Represented with the symbol Φ (Phi)
- Totient = ΦN

How does RSA Work?

Step 3 – Find the Totient of ProductOfPrimes

- Get the Totient = φ(14)
- Φ – Phi
- Look for the numbers that has a common factor of 1 – 14, and 2 and 7
- Cross out the numbers

1
2
3
4
5
6
7
8
9
10
11
12
13
14

How does RSA Work?

Step 3 – Find the Totient of ProductOfPrimes

- Cross out the numbers that has a common factor of 1 – 14, and 2 and 7, except 1.
- Remove all even numbers because it has factors of 14, and similar factors of 2 and 7

1

3

5

7

9

11

13

How does RSA Work?

Step 3 – Find the Totient of ProductOfPrimes

- Cross out the numbers that has a common factor of 1 – 14, and 2 and 7
- Remove 7
- That leaves 1, 3, 5, 9, 11, 13 (6 remaining numbers)
- They are called co-prime with 14, since they share no common factors with 14.
- **Totient = ΦN = 6**

1

3

5

9

11

13

How does RSA Work?

Step 3 – Find the Totient of ProductOfPrimes

- Totient = φ(ProductOfPrimes)
- Φ – Phi
- Totient = Φ(14)
- Totient = (Prime1 -1) * (Prime2 – 1)
- Totient = (2-1) * (7– 1)
- Totient = (1) * (6)
- **Totient = ΦN = 6**

How does RSA Work?

Step 4 – Choose a number for *e (Encryption key)*

e – stands for Encryption key and should obey the following rules:

- Should be 1 < e < ΦN
- Should be co-prime with N and ΦN

- 1 < e < ΦN = { 2, 3, 4, 5}
- co-prime with N and ΦN = **5**
- {2,3 4} are factors of 14 and 6 while 5 is not

e = 5 ; encryption key or the lock key$_e$(5,14)

How does RSA Work?

Step 5 – Choose a number for *d (decryption)*

d – stands for Decryption and should obey the following rule

$d = de(\mathrm{mod}\ \Phi N) = 1$

$d = 5d(\mathrm{mod}\ (6)) = 1$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5	4	3	2	1	0	5	4	3	2	1	0	5	4	3	2	1	0	5

d = 11 ; $key_d(11,14)$

Encrypting and Decrypting RSA

1. Create a message:

   Meet at three pm

2. Convert the message into its numeric value – using the alphabet table, the following numeric value is identified

m	e	e	t	a	t	t	h	r	e	e	p	m
13	5	5	20	1	20	20	8	18	5	5	16	13

3. Compute for the ciphertext using the formula;

$$e = 5, \text{mod} = 14; c = m^e \pmod{(14)}$$

$$c = 13^5 \bmod 14 ;$$

$$c = 371293 \bmod 14$$

$$c = 13$$

Encrypting and Decrypting RSA

4. Look for the value of the ***c*** in the alphabetic table:

c = 13, c = 3 … c = 13

Iterate until the entire message is converted into a cipher

ciper	13	3	3	6	1	6	6	8	2	3	3	4	13
cipertext	m	c	c	f	a	f	f	h	b	c	c	d	m

Message

Meet at three pm

Cyphertext

Mccfaffhbccdm

Encrypting and Decrypting RSA

1. To decrypt, reverse the process of encryption :

   mccfaffhbccdm

2. Convert the message into its numeric value – using the alphabet table, the following numeric value is identified

Cipertext	m	c	c	f	a	f	f	h	b	c	c	d	m
value	13	3	3	6	1	6	6	8	2	3	3	4	13

3. Compute for the message using the formula;

$$d = 11, \text{mod} = 14; m = c^d \ (\text{mod}\ (14)$$

$$c = 13^{11} \ \text{mod}\ 14 ;$$

$$m = 1792160394037 \ \text{mod}\ 14$$

$$m = 13$$

Encrypting and Decrypting RSA

4. Look for the value of the *m* in the alphabetic table:

m = 13, m = 5 … m = 13

Iterate until the entire message is converted into a cipher

m	13	5	5	20	1	20	20	8	18	5	5	16	13
Ciphertext	m	e	e	t	a	t	t	h	r	e	e	p	m

Cyphertext

Mccfaffhbccdm

Message

Meet at three pm

Encryption Key Power

Number of Bits in Key	Odds of Cracking: 1 in	Estimated Time to Crack*
8	256	.000032 seconds
16	65,536	.008192 seconds
24	16,777,216	2.097 seconds
32	4,294,967,296	8 minutes 56.87 seconds
56	72,057,594,037,927,900	285 years 32 weeks 1 day
64	18,446,744,073,709,600,000	8,090,677,225 years
128	3.40282E+38	5,257,322,061,209,440,000,000 years
256	1.15792E+77	2,753,114,795,116,330,000,000,000,000, 000,000,000,000,000,000,000 years
512	1.3408E+154	608,756,305,260,875,000,000,000,000,000, 000,000,000,000,000,000,000,000,000,000, 000,000,000,000,000,000,000,000,000,000, 000,000,000 years

[NOTE] *Estimated Time to Crack is based on a general-purpose personal computer performing eight million guesses per second.

Hybrid Cryptography Systems

- Except with digital certificates, pure asymmetric key encryption not widely used
- Asymmetric encryption more often used with symmetric key encryption, creating hybrid system
- Diffie-Hellman Key Exchange method: most common hybrid system; provided foundation for subsequent developments in public key encryption

Cryptography Tools

- Public Key Infrastructure (PKI)**:** integrated system of software, encryption methodologies, protocols, legal agreements, and third-party services enabling users to communicate securely
- PKI systems based on public key cryptosystems; include digital certificates and certificate authorities (CAs)

Cryptography Tools (continued)

- PKI protects information assets in several ways:
  - Authentication
  - Integrity
  - Privacy
  - Authorization
  - Nonrepudiation

PKI protects information assets in several ways:

- **Authentication.** Digital certificates in a PKI system permit parties to validate the identity of other of the parties in an Internet transaction.
- **Integrity.** A digital certificate demonstrates that the content signed by the certificate has not been altered while being moved from server to client.

PKI protects information assets in several ways:

- **Privacy.** Digital certificates keep information from being intercepted during transmission over the Internet.
- **Authorization.** Digital certificates issued in a PKI environment can replace user IDs and passwords, enhance security, and reduce some of the overhead required for authorization processes and controlling access privileges.

PKI protects information assets in several ways:

- **Nonrepudiation.** Digital certificates can validate actions, making it less likely that customers or partners can later repudiate a digitally signed transaction.

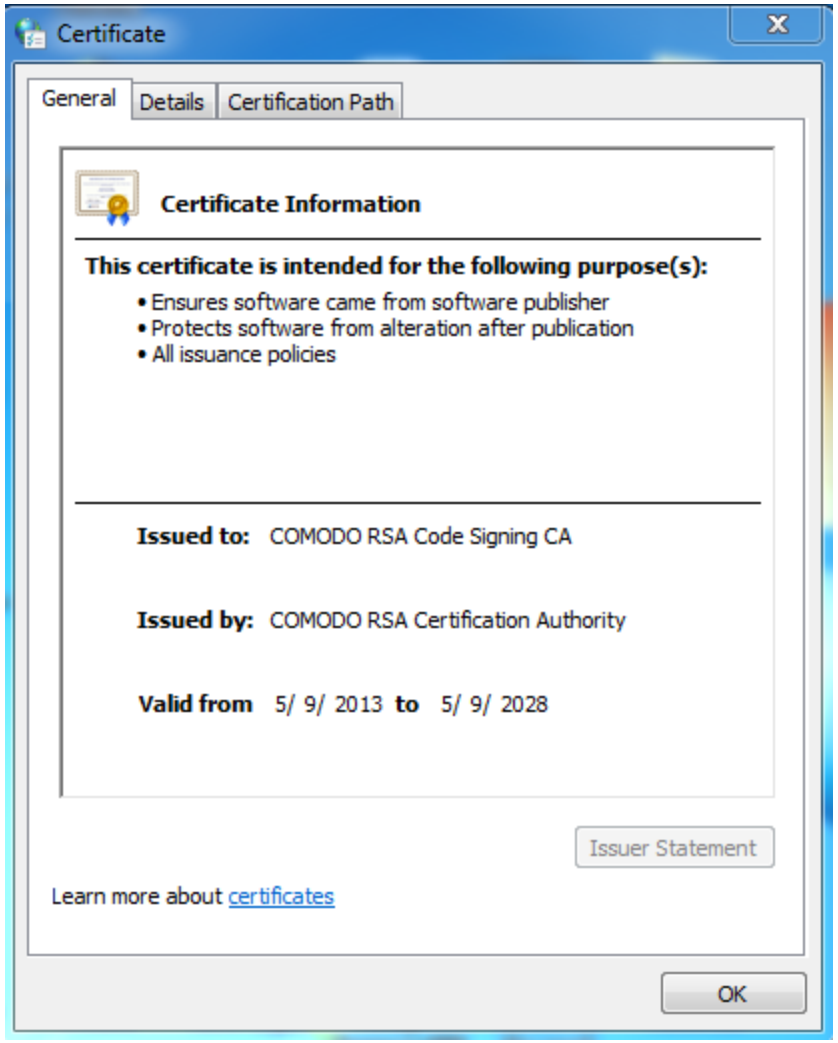
Digital Signatures

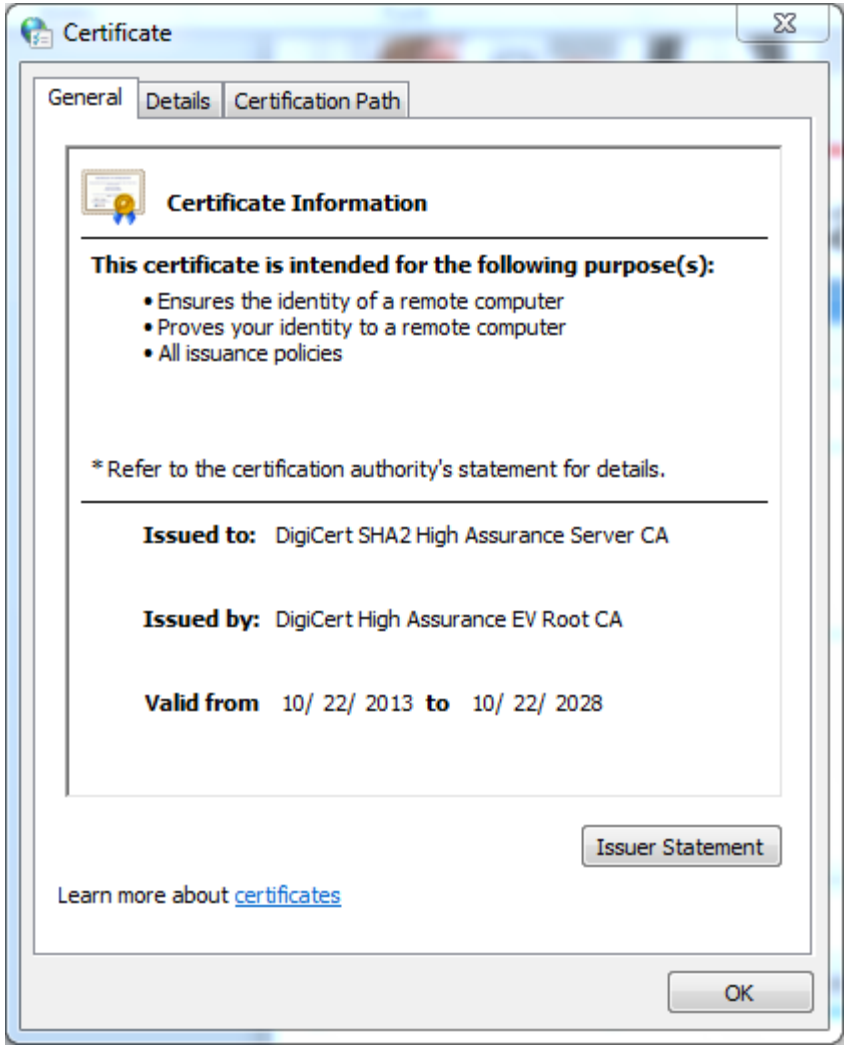
- Encrypted messages that can be mathematically proven to be authentic
- Created in response to rising need to verify information transferred using electronic systems
- Asymmetric encryption processes used to create digital signatures

Digital Certificates

- Electronic document containing key value and identifying information about entity that controls key
- Digital signature attached to certificate’s container file to certify file is from entity it claims to be from

Digital Certificates and Signatures





Steganography

- Process of hiding information; in use for a long time
- Most popular modern version hides information within files appearing to contain digital pictures or other images
- Some applications hide messages in .bmp, .wav, .mp3, and .au files, as well as in unused space on CDs and DVDs

Protocols for Secure Communications

- Secure Socket Layer (SSL) protocol: uses public key encryption to secure channel over public Internet
- Secure Hypertext Transfer Protocol (S-HTTP): extended version of Hypertext Transfer Protocol; provides for encryption of individual messages between client and server across Internet
- S-HTTP is the application of SSL over HTTP; allows encryption of information passing between computers through protected and secure virtual connection
- HTTPS and S-HTTP

Protocols for Secure Communications (continued)

- Securing E-mail with S/MIME, PEM, and PGP
  - Secure Multipurpose Internet Mail Extensions (S/MIME): builds on Multipurpose Internet Mail Extensions (MIME) encoding format by adding encryption and authentication
  - Privacy Enhanced Mail (PEM): proposed as standard to function with public key cryptosystems; uses 3DES symmetric key encryption
  - Pretty Good Privacy (PGP): uses IDEA Cipher for message encoding

Protocols for Secure Communications (continued)

- Securing Web transactions with SET, SSL, and S-HTTP
  - Secure Electronic Transactions (SET): developed by MasterCard and VISA in 1997 to provide protection from electronic payment fraud
  - Uses DES to encrypt credit card information transfers
  - Provides security for both Internet-based credit card transactions and credit card swipe systems in retail stores

Protocols for Secure Communications (continued)

- Securing TCP/IP with IPSec
  - Internet Protocol Security (IPSec): open source protocol to secure communications across any IP-based network
  - IPSec designed to protect data integrity, user confidentiality, and authenticity at IP packet level

Protocols for Secure Communications (continued)

- IPSec combines several different cryptosystems: Diffie-Hellman; public key cryptography; bulk encryption algorithms; digital certificates
- In IPSec, IP layer security obtained by use of application header (AH) protocol or encapsulating security payload (ESP) protocol

Protocols for Secure Communications (continued

- Freeware and low-cost commercial PGP versions are available for many platforms
- PGP security solution provides six services: authentication by digital signatures; message encryption; compression; e-mail compatibility; segmentation; key management

Hashing

Engr. Juliet S. Mendez

MBA, CCNA

Hashing

- Any function that can be used to map data of arbitrary size of data of fixed size.
- Used in checksums, check digits, fingerprints, lossy compression, randomization functions, error-correcting codes and ciphers
- File Verification
- Password Storage
- Database Searching

Popular Cryptographic Hash Functions

- Message Digest 5(MD5) Algorithm
- Secure Hashing Algorithm (SHA)
  - SHA- 1
  - SHA- 256

MD5 Algorithm

- 128 bit hash value
- Suffer from extensive vulnerabilities
- Can still be used as a checksum

SHA -1

- 160 bit hash value
- 40 digits long
- Microsoft, Google, Apple and Mozilla : Stop accepting SHA-1 SSL certificates by 2017
- Used in distributed revision control systems(GIT, Mercurial & Monotone)

SHA-2

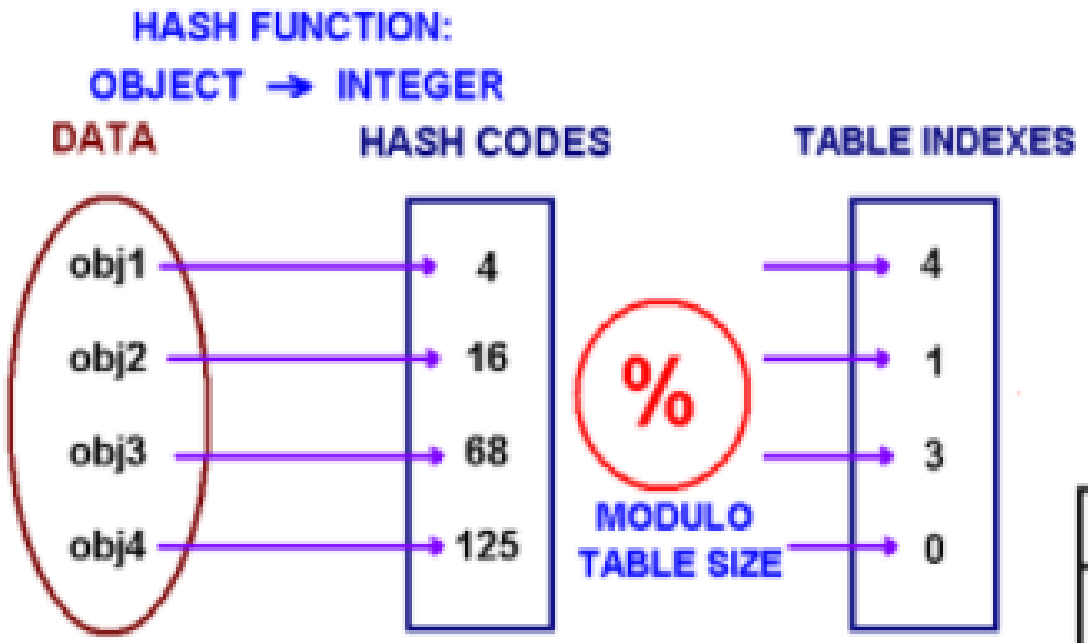
- Consist of six hash functions with hash values
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512

Published in	Year	Attack method	Attack	Variant	Rounds	Complexity
New Collision Attacks Against Up To 24-step SHA-2[32]	2008	Deterministic	Collision	SHA-256	24/64	$2^{28.5}$
				SHA-512	24/80	$2^{32.5}$
Preimages for step-reduced SHA-2[33]	2009	Meet-in-the-middle	Preimage	SHA-256	42/64	$2^{251.7}$
					43/64	$2^{254.9}$
				SHA-512	42/80	$2^{502.3}$
					46/80	$2^{511.5}$
Advanced meet-in-the-middle preimage attacks[34]	2010	Meet-in-the-middle	Preimage	SHA-256	42/64	$2^{248.4}$
				SHA-512	42/80	$2^{494.6}$
Higher-Order Differential Attack on Reduced SHA-256[2]	2011	Differential	Pseudo-collision	SHA-256	46/64	2^{178}
					33/64	2^{46}
Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family[1]	2011	Biclique	Preimage	SHA-256	45/64	$2^{255.5}$
				SHA-512	50/80	$2^{511.5}$
			Pseudo-preimage	SHA-256	52/64	2^{255}
				SHA-512	57/80	2^{511}
Improving Local Collisions: New Attacks on Reduced SHA-256[35]	2013	Differential	Collision	SHA-256	31/64	$2^{65.5}$
			Pseudo-collision	SHA-256	38/64	2^{37}
Branching Heuristics in Differential Collision Search with Applications to SHA-512[36]	2014	Heuristic differential	Pseudo-collision	SHA-512	38/80	$2^{40.5}$
Analysis of SHA-512/224 and SHA-512/256[37]	2016	Differential	Collision	SHA-256	28/64	practical
				SHA-512	27/80	practical
			Pseudo-collision	SHA-512	39/80	practical

SHA- 3

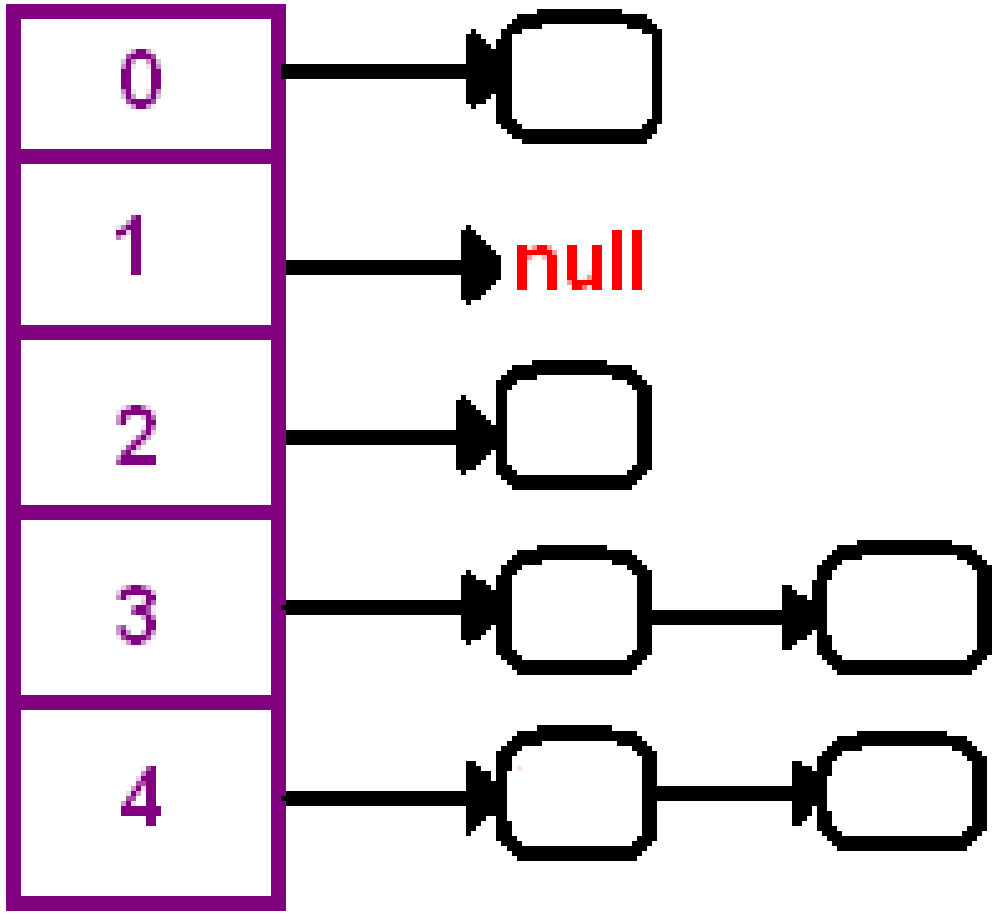
- released on August 5, 2015 by NIST
- Is a subset of Keccak
- Data is absorbed into sponge, result is squeezed out.

Storing Hash Function



HASH TABLE

OBJ4	OBJ2		OBJ3	OBJ1
0	1	2	3	4


0
1
null
2
3
4

JAVA Sample Code

```
17      public static void main(String[] args) {
18          // TODO code application logic here
            Integer obj1 = new Integer(2009);
            String obj2 = new String("ABC");
21          System.out.println("hashCode for an integer is " + obj1.hashCode());
22          System.out.println("hashCode for a string is " + obj2.hashCode());
23
24      }
```

Formula

$$s.charAt(0) * 31^{n-1} + s.charAt(1) * 31^{n-2} + s.charAt(2) * 31^{n-3} + \ldots + s.charAt(n-1)$$

where *s* = string value in ASCII and *n*= length

Sample Computation:

ABC

ABC = ‘A’ * 31^2 + ‘B’ * 31 + ‘C’

ABC = 65 * 31^2 + 66 * 31 + 67

ABC = 64578

Attack on Crytosystems

Attacks on Cryptosystems

- Attempts to gain unauthorized access to secure communications have typically used brute force attacks (ciphertext attacks)

- Attacker may alternatively conduct known-plaintext attack or selected-plaintext attach schemes

Man-in-the-Middle Attack

- Designed to intercept transmission of public key or insert known key structure in place of requested public key
- From victims' perspective, encrypted communication appears to be occurring normally, but in fact attacker receives each encrypted message, decodes, encrypts, and sends to originally intended recipient
- Establishment of public keys with digital signatures can prevent traditional man-in-the-middle attack

Correlation Attacks

- Collection of brute-force methods that attempt to deduce statistical relationships between structure of unknown key and ciphertext
- Differential and linear cryptanalysis have been used to mount successful attacks
- Only defense is selection of strong cryptosystems, thorough key management, and strict adherence to best practices of cryptography in frequency of changing keys

Dictionary Attacks

- Attacker encrypts every word in a dictionary using same cryptosystem used by target
- Dictionary attacks can be successful when the ciphertext consists of relatively few characters (e.g., usernames, passwords)

Timing Attacks

- Attacker eavesdrops during victim’s session; uses statistical analysis of user’s typing patterns and inter-keystroke timings to discern sensitive session information
- Can be used to gain information about encryption key and possibly cryptosystem in use
- Once encryption successfully broken, attacker may launch a replay attack (an attempt to resubmit recording of deciphered authentication to gain entry into secure source

Defending From Attacks

- No matter how sophisticated encryption and cryptosystems have become, if key is discovered, message can be determined
- Key management is not so much management of technology but rather management of people

Video Clip – Cyber Security

- Video Clip showing simple definition about cybersecurity, how does it work and why do we need it.
- Cyber codes

Thank you

Have a nice day!!!

Answer to the Ciphertext

WELCOME TO CEBU
CARAGA STATE UNIVERSITY
CABADBARAN CAMPUS

Summary

- Cryptography and encryption provide sophisticated approach to security
  - Many security-related tools use embedded encryption technologies
  - Encryption converts a message into a form that is unreadable by the unauthorized
- Many tools are available and can be classified as symmetric or asymmetric, each having advantages and special capabilities
- Strength of encryption tool dependent on key size but even more dependent on following good management practices
- Cryptography is used to secure most aspects of Internet and Web uses that require it, drawing on extensive set of protocols and tools designed for that purpose
- Cryptosystems are subject to attack in many ways

Digital Certificate - Implementing SSL for your website

INFORMATION ASSURANCE AND SECURITY 1

GODWIN S. MONSERATE

What is a Digital Certificate?

- A **Digital Certificate** is an electronic "password" that allows a person, organization to exchange data securely over the Internet using the public key infrastructure (PKI).
- **Digital Certificate** is also known as a public key **certificate** or identity **certificate**.

What is a Digital Certificate?

- A digital certificate (DC) is a digital file that certifies the identity of an individual or that certifies the identity of an individual or institution, or even a router seeking institution, or even a router seeking access to computer- based information.
- It access to computer- based information. It is issued by a Certification Authority, and serves the same purpose as a driver's and serves the same purpose as a driver's license or a passport

What is a Digital Certificate?

- The Certification Authority (CA) signs the certificate with their own private key. An SSL/Digital Certificate typically contains the following information:
  - Owner's public key
  - Owner's name
  - Expiration date of the public key
  - Name of the issuer (the Certifying Authority that issued the Digital Certificate)
  - Serial number of the Digital Certificate
  - Digital signature of the issuer

Certification Authorities

- CA's are the digital world's equivalent to passport offices.
- They issue digital equivalent to passport offices.
- They issue digital certificates and validate holders' and validate holders' identity and authority.
- They embed an individual or institution's public key along with other identifying information into each digital certificate and then cryptographically sign it as a tamper-proof seal verifying the integrity of the data within it and validating its use.

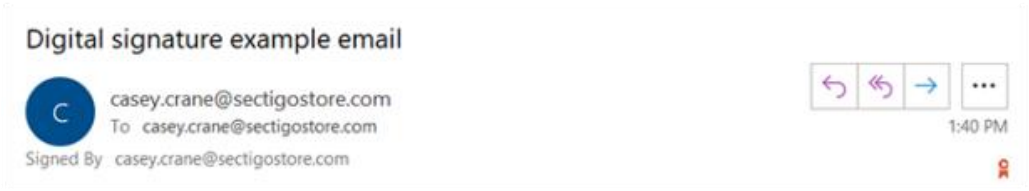
What does it do?

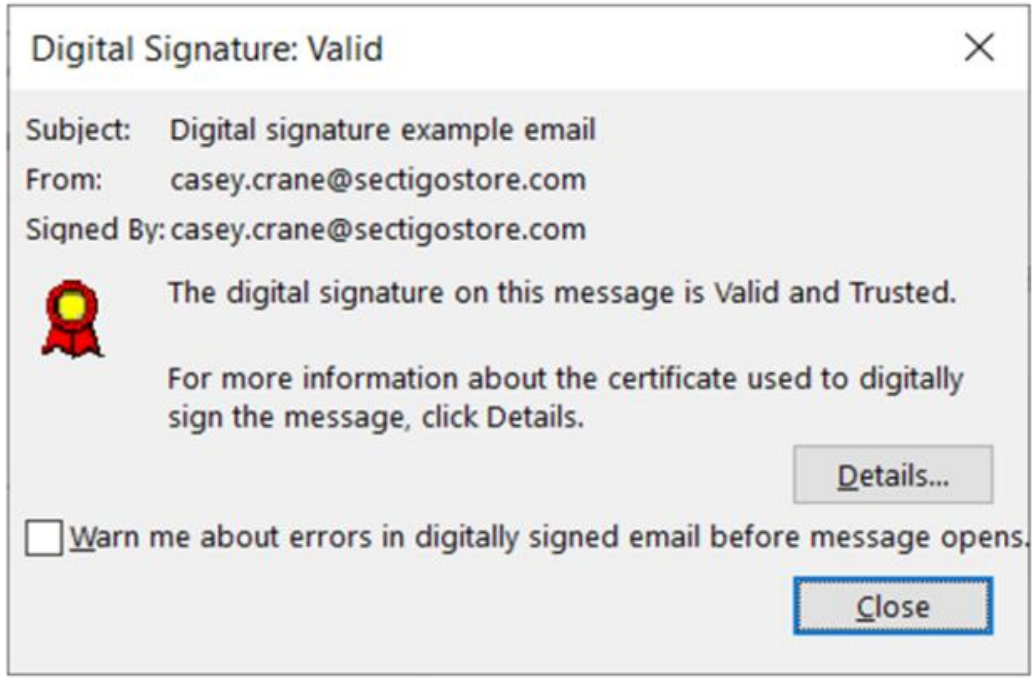
- Digital Certificates can be used for a variety of electronic transactions including e-mail, electronic commerce, groupware and electronic funds transfers.
- If you are running an online e-commerce website, an electronic banking website or any other electronic services website then customers may abandon your website due to concerns about privacy and security.
- You will hence need to provide secure access to your website visitors via **https** protocol.
- To do this you will need to setup your website on a dedicated IP address and install a valid digital certificate on your hosting server.

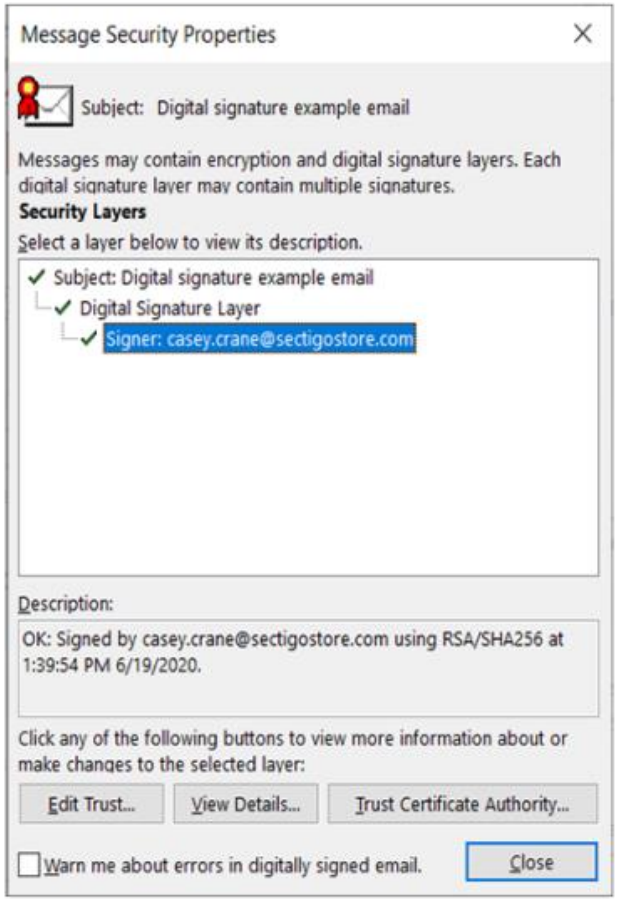
What does it do?

- Digital Certificates, bind an identity to a pair of electronic keys that can be used to encrypt and sign digital information.
- A Digital Certificate makes it possible to verify someone's claim that they have the right to use a given key, helping to prevent people from using phony keys to impersonate other users.
- Used in conjunction with encryption, Digital Certificates provide a more complete security solution, assuring the identity of all parties involved in a transaction.
- A digital certificate also is known as public key certificate allows exchanging data securely over the internet using public key infrastructure

Sample Digital Certificate







Types of Digital Certificates

- There are 4 main types of Digital Certificates
  1. Server Certificates or /TLS/SSL Certificate
  2. Personal Certificates
  3. Organizational Certificates or Client Certificate
  4. Developer's Certificates or Code Signing Certificate

Types of Digital Certificates

- Server Certificates
  - Allows visitors to exchange personal information such as credit card numbers, free from the threat of interception or tampering.
  - Server Certificates are a must for building and designing e-commerce sites as confidential information is shared between clients, customers and vendors.
  - TLS/SSL (Transport Layer Security/Secure Socket Layer) Certificates are installed on the server. The purpose of these certificates is to ensure that all communication between the client and the server is private and encrypted.
  - The server could be a web server, app server, mail server, LDAP server, or any other type of server that requires authentication to send or receive encrypted information. The address of a website with a TLS/SSL certificate will start with “https://” instead of “http://”, where the “s” stands for “secure.”

Types of Digital Certificates

- Personal Certificates
  - Personal Certificates allow one to authenticate a visitor's identity and restrict access to specified content to particular visitors.
  - Personal Certificates are perfect for business to business communications such as offering suppliers and partners controlled access to special web sites for updating product availability, shipping dates and inventory management.

Types of Digital Certificates

- Organization Certificates
  - are used by corporate entities to identify employees for secure e-mail and web-based transaction.
  - Client Certificates or Digital IDs are used to identify one user to another, a user to a machine, or a machine to another machine.
  - One common example is emails, where the sender digitally signs the communication, and the recipient verifies the signature.
  - Client certificates authenticate the sender and the recipient.
  - Client certificates also take the form of two-factor authentication when the user needs to access a protected database or arrives at the gateway to a payment portal, where they'll be expected to enter their passwords and be subjected to further verification.

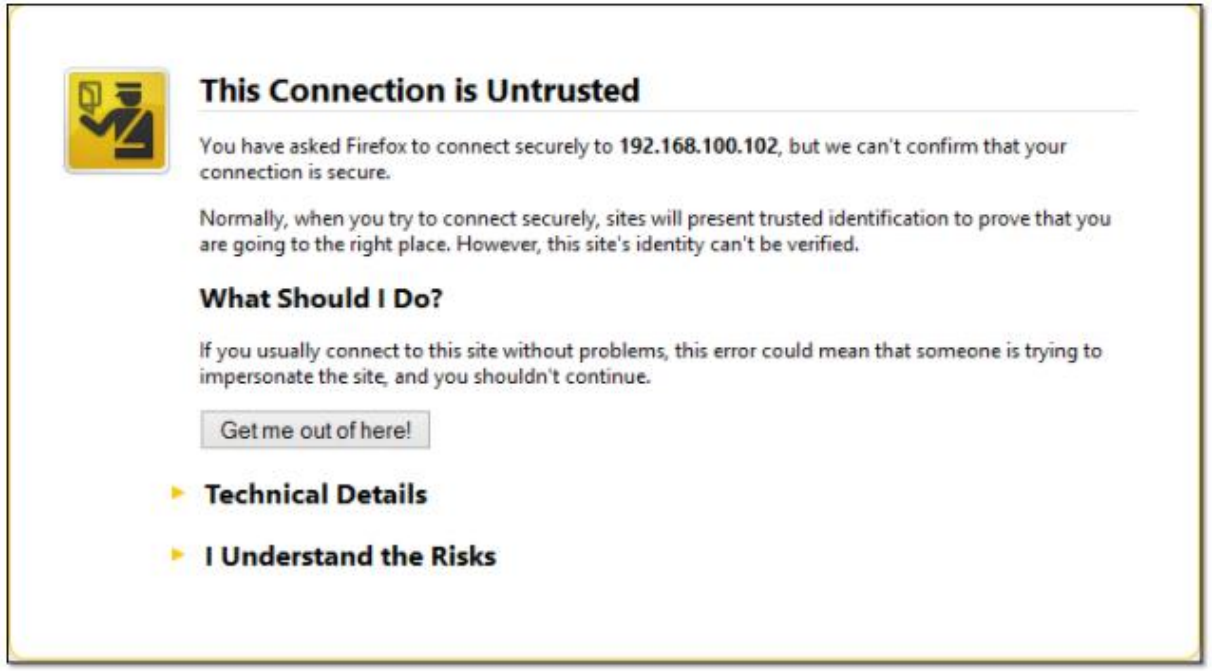
Types of Digital Certificates

- Developer Certificates
  - Prove authorship and retain integrity of distributed software programs e.g. installing a software on a computer system in most instances requires what is called a “serial key”
  - Used to sign software or files that are downloaded over the internet. They’re signed by the developer/publisher of the software.
  - Their purpose is to guarantee that the software or file is genuine and comes from the publisher it claims to belong. They’re especially useful for publishers who distribute their software for download through third-party sites. Code signing certificates also act as a proof that the file hasn’t been tampered with since download.

Signed vs Self-signed certificates

- In theory, certificate authorities are supposed to exercise due diligence before signing digital certificates submitted to them through Certificate Signing Request or CSRs.
- They need to verify first whether the information placed on the digital certificates are in fact true. This is important because their attestation would later on serve as the sole basis that certain websites who are able to present certs signed by them can really be trusted.
- It would be safe to assume that signed certificates are more reliable and trustworthy than self-signed certificates. In fact, when a user attempts to connect to your site and your site only has a self-signed certificate, the user's browser will display something like this:

Signed vs Self-signed certificates



- Self-signed certificates are relatively safe to use internally, i.e., within your organization, where you have more control over the servers that operate in the network.
- So, for instance, you can use it to add security to a web file transfer that takes place behind your corporate firewall.

Implementing SSL in Website

What is an SSL?

- SSL stands for **Secure Sockets Layer**, a now-deprecated cryptographic protocol that has kept only its name in common use. All SSL certificates are technically TLS certificates, TLS being the successor of the SSL technology.
- TLS or **Transport Layer Security** is a more advanced and secure protocol, that’s been now the standard encryption technology for more than a decade.

What is an SSL Certificate

- **SSL Certificate** is a digitally signed Certificate which is built of complex hashing functions and algorithm that keeps user's information encrypted during the data transmission form Client to Server and server to client.
- So, here both term SSL Certificate and Digital Certificate comes up different meaning and definition.

What is an SSL Certificate

- Also known as **SSL Certificate** is a digitally signed certification by an established authority to confirm the identity of your website/business and uses encryption to send/receive data between your website and its visitors.
- SSL certificate authenticates the connection between web server and browsers by encrypting communication between the website and its users.
- It is issued for a domain by a trusted authority referred as Certificate Authority (CA). Example CAs are Comodo and Thawte.

What does it do?

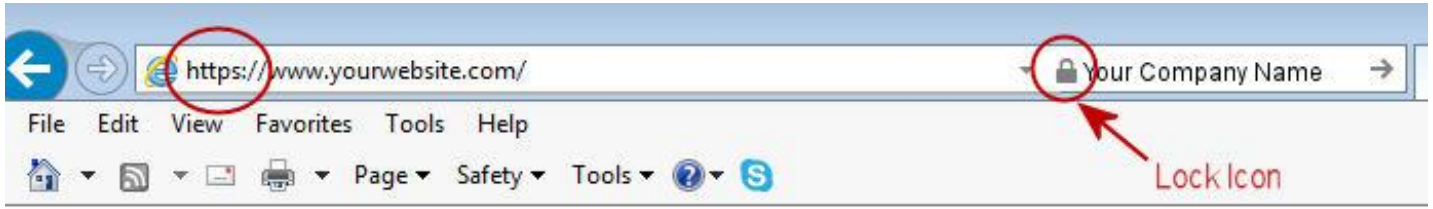
- An SSL certificate allows you to establish your credentials when doing business or other transactions on the Web.
- It is generally used when a website wants to accept sensitive information like passwords, credit card details and other sensitive information.
- The SSL Certificate protects your customer's personal data including passwords, credit cards and identity information. Thus, getting an SSL certificate for your website is the easiest way to increase your customer's confidence in your online business.

When do you require an SSL/Digital Certificate?

- An SSL Certificate does 2 things:
  a. Encrypt the information sent from your website visitor's browser to your website
  b. Authenticate your website's identity.
- By doing these 2 things, an SSL Certificate protects your customers and in turn increases their trust in your online business.
- This is especially important if your website requires users to login using passwords or enter sensitive information such as credit card details.
- Many customers actively look for the SSL lock icon before handing over sensitive data.

How do you know you have a digital certificate?

- If you come across a website whose URL begins with **https://**, you can view the website's SSL Certificate by clicking on the lock icon in the address bar of your browser.



Types of SSL Certificates

- Certifying authorities provide SSL certificates in a few variety of branded names, each serving a specific purpose.
- For example, Comodo sells a basic SSL certificate in the name of *Positive SSL* while Thawte sells an equivalent certificate named as *SSL123 Certificate*.
- Likewise, a wildcard SSL certificate is named as *Positive SSL Wildcard* by Comodo and *Wildcard Server Certificate* by Thawte.

Types of SSL Certificates

- Broadly there are two types of SSL certificates:
  1. **Basic SSL certificate:** allows you to secure one sub-domain. For example, if your e-commerce website is store.yourwebsitename.com, a basic SSL will secure only this sub domain and hence people will be able to access your website as https://store.yourwebsitename.com. If you want to also secure www.yourwebsitename.com, you may need to purchase a second separate certificate for this second sub-domain. A basic SSL certificate is quite well suited for small websites and blogs.
  2. **Wildcard SSL certificate:** allows you to secure your primary domain name as well as all its sub-domains. Thus one certificate will secure both www.yourwebsitename.com and store.yourwebsitename.com, and any other sub domains such as support.yourwebsitename.com, webmail.yourwebsitename.com, etc. A Wildcard SSL is best suited for large e-commerce websites.

How to get an SSL Certificate for your website?

- To be issued an SSL Certificate, you need to purchase one from a web service provider and then go through a process that entails the following:
- **Step 1: Purchasing SSL**
  - As a first step you place an order for an ssl certificate with the web service provider. While placing order, you will need to specify the exact domain name for which you require the ssl certificate.
  - For example, if you need to secure *store.yourwebsitename.com*, you should specify *store.yourwebsitename.com* while placing order and not *www.yourwebsitename.com*.
  - Once your order has been executed by the service provider, you will be provided with a control panel from where you can apply for your certificate.

How to get an SSL Certificate for your website?

- **Step 2: Private Key and CSR Generation**
  - Prior to applying/enrolling for a Certificate with the CA, you must generate a minimum of 2048-bit Private Key and CSR pair from your hosting server.
  - Digital IDs make use of a technology called *Public Key Cryptography*, which uses Public and Private Key files.
  - The *Public Key*, also known as a *Certificate Signature Request (CSR)*, is the key that will be sent to the CA.
  - The Public Key is generated on your server and validates the computer-specific information about your web server and Organization when you request a Certificate from a CA.

How to get an SSL Certificate for your website?

- **Step 2: Private Key and CSR Generation**
  - The *Private Key* will remain on your hosting server and should never be released into the public. Even the Certifying Authority will *not* have access to your Private Key.
  - It is generated locally on your server and is *never* transmitted to the CA or any browser visiting your website. The integrity of your Digital ID depends on your Private Key being controlled exclusively by you.
  - A CSR *cannot* be generated without generating a Private Key file. Similarly the Private Key file *cannot* be generated without generating a CSR file.
  - In certain web server software platforms like Microsoft IIS, both are generated simultaneously through the Wizard on the web server.

How to get an SSL Certificate for your website?

- ## Step 2: Private Key and CSR Generation
  - Most hosting service providers provide you with a hosting management control panel which has an *SSL/TLS Manager* interface using which you can generate your CSR - private key pair.
  - You will be required to enter certain relevant details about your organization while generating the CSR.
  - On completion of this process, your hosting server will generate an encoded file, viz. your CSR. This CSR can now be used to submit your SSL Certificate application to the Certificate Authority.

How to get an SSL Certificate for your website?

- **Step 3: Enrollment**
  - After you have generated a minimum of 2048-bit Private Key and CSR pair from your web hosting server, the next step is to submit your Enrollment information to the CA for the CA to verify your information and issue the Digital certificate to you.
  - The enrollment is done from the interface that the web service provider will provide to you after you have purchased the SSL certificate.

How to get an SSL Certificate for your website?

- **Step 3: Enrollment**
  - Enrollment essentially requires you to submit a form wherein you provide relevant details about your organization such as Organization name, Contact details, Admin email address, Approver Email Address, etc.
  - The contact details that you provide here must match with the ones available in your domain's whois lookup.
  - Also, you must ensure that prior to enrollment, your domain is not privacy protected and that it's whois information is publicly visible.
  - Subsequently, after the certificate is issued to you, you may re-enable your domain's privacy protection.

How to get an SSL Certificate for your website?

- ## Step 4: Verification Process & Certificate Issue
  - After you have submitted the enrollment form, the Certifying Authority will now carry out a verification of your organization and the information you have submitted. If required, they may call you at your specified phone number for additional verification of your business.
  - This process is much faster and usually automatic when you apply for a basic ssl certificate. Subsequently, after the CA is satisfied with the verification, you will receive an email from the CA to approve the issue of ssl certificate.
  - After you have done the approval, you will receive an email from the CA informing you that your certificate has been issued. The email will also contain information on how you can retrieve the issued certificate.

How to get an SSL Certificate for your website?

- Image shows how your issued ssl certificate will look:

How to get an SSL Certificate for your website?

- **Step 5: Certificate Installation**
- This is the final step wherein you need to install the issued certificate on your hosting server.
- Additionally, you will also need to install the SSL Certificate of the Certificate Authority (known as the CA bundle).
- The CA bundle contains root and intermediate certificates of the CA and is available for download from the website of the CA.
- Depending upon the web server where you intend to install your SSL Certificate, you need to refer to the appropriate instructions provided by your hosting service provider.
- Once successfully installed, your website will become accessible via **https://....**

Difference between Digital Certificate and SSL Certificate

- **Digital Certificate:**
  - A Digital Certificate is a digital "password" which permits an individual, organization to exchange information securely across the Web utilizing the public key infrastructure (PKI).
  - Digital Certificate can be referred to as a public key certification or identity certification.
- **SSL Certificate:**
  - SSL Certificates are small data files which bind a cryptographic key to a company's particulars. Once installed on an internet server, then it activates the padlock along with the https protocol also enables safe connections from a web server to a browser.
  - Normally, SSL is used to secure credit card transactions, information transport and logins, and much more lately has become the standard when procuring browsing of social networking websites.

Digital Signatures and Certificates

Encryption – Process of converting electronic data into another form, called cipher text, which cannot be easily understood by anyone except the authorized parties. This assures data security.

Decryption– Process of translating code to data.

- Message is encrypted at the sender's side using various encryption algorithms and decrypted at the receiver's end with the help of the decryption algorithms.
- When some message is to be kept secure like username, password, etc., encryption and decryption techniques are used to assure data security.

Types of Encryption

1. **Symmetric Encryption**– Data is encrypted using a key and the decryption is also done using the same key.
2. **Asymmetric Encryption**-Asymmetric Cryptography is also known as public key cryptography. It uses public and private keys to encrypt and decrypt data. One key in the pair which can be shared with everyone is called the public key. The other key in the pair which is kept secret and is only known by the owner is called the private key. Either of the keys can be used to encrypt a message; the opposite key from the one used to encrypt the message is used for decryption.

Public key– Key which is known to everyone. Ex-public key of A is 7, this information is known to everyone.
Private key– Key which is only known to the person who's private key it is.
Authentication-Authentication is any process by which a system verifies the identity of a user who wishes to access it.
Non- repudiation– Non-repudiation means to ensure that a transferred message has been sent and received by the parties claiming to have sent and received the message. Non-repudiation is a way to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.
Integrity– to ensure that the message was not altered during the transmission.
Message digest -The representation of text in the form of a single string of digits, created using a formula called a one way hash function. Encrypting a message digest with a private key creates a digital signature which is an electronic means of authentication..

Digital Signature

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document.

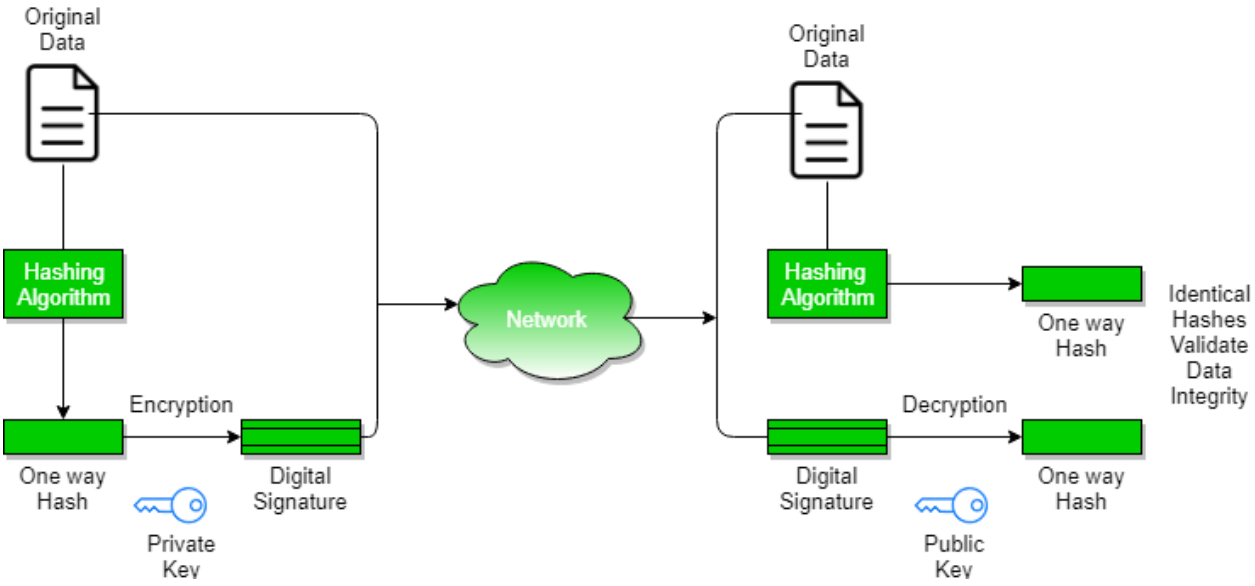
1. **Key Generation Algorithms** : Digital signature are electronic signatures, which assures that the message was sent by a particular sender. While performing digital transactions authenticity and integrity should be assured, otherwise the data can be altered or someone can also act as if he was the sender and expect a reply.
2. **Signing Algorithms**: To create a digital signature, signing algorithms like email programs create a one-way hash of the electronic data which is to be signed. The signing algorithm then encrypts the hash value using the private key (signature key). This encrypted hash along with other information like the hashing algorithm is the digital signature. This digital signature is appended with the data and sent to the verifier. The reason for encrypting the hash instead of the entire message or document is that a hash function converts any arbitrary input into a much shorter fixed length value. This saves time as now instead of signing a long message a shorter hash value has to be signed and moreover hashing is much faster than signing.
3. **Signature Verification Algorithms** : Verifier receives Digital Signature along with the data. It then uses Verification algorithm to process on the digital signature and the public key (verification key) and generates some value. It also applies the same hash function on the received data and generates a hash value. Then the hash value and the output of the verification algorithm are compared. If they both are equal, then the digital signature is valid else it is invalid.

The steps followed in creating digital signature are :

1. Message digest is computed by applying hash function on the message and then message digest is encrypted using private key of sender to form the digital signature. (digital signature = encryption (private key of sender, message digest) and message digest = message digest algorithm(message)).
2. Digital signature is then transmitted with the message.(message + digital signature is transmitted)
3. Receiver decrypts the digital signature using the public key of sender.(This assures authenticity, as only sender has his private key so only sender can encrypt using his private key which can thus be decrypted by sender's public key).
4. The receiver now has the message digest.
5. The receiver can compute the message digest from the message (actual message is sent with the digital signature).

The message digest computed by receiver and the message digest (got by decryption on digital signature) need to be same for ensuring integrity.

Message digest is computed using one-way hash function, i.e. a hash function in which computation of hash value of a message is easy but computation of the message from hash value of the message is very difficult.



Digital Certificate

Digital certificate is issued by a trusted third party which proves sender's identity to the receiver and receiver's identity to the sender.

A digital certificate is a certificate issued by a Certificate Authority (CA) to verify the identity of the certificate holder. The CA issues an encrypted digital certificate containing the applicant's public key and a variety of other identification information. Digital certificate is used to attach public key with a particular individual or an entity.

Digital certificate contains:

1. Name of certificate holder.
2. Serial number which is used to uniquely identify a certificate, the individual or the entity identified by the certificate
3. Expiration dates.
4. Copy of certificate holder's public key. (used for decrypting messages and digital signatures)
5. Digital Signature of the certificate issuing authority.

Digital certificate is also sent with the digital signature and the message.

Digital certificate vs digital signature :
Digital signature is used to verify authenticity, integrity, non-repudiation ,i.e. it is assuring that the message is sent by the known user and not modified, while digital certificate is used to verify the identity of the user, maybe sender or receiver. Thus, digital signature and certificate are different kind of things but both are used for security. Most websites use digital certificate to enhance trust of their users.

Feature	**Digital Signature**	**Digital Certificate**
Basics / Definition	Digital signature is like a fingerprint or an attachment to a digital document that ensures its authenticity and integrity.	Digital certificate is a file that ensures holder's identity and provides security.
Process / Steps	Hashed value of original message is encrypted with sender's secret key to generate the digital signature.	It is generated by CA (Certifying Authority) that involves four steps: Key Generation, Registration, Verification, Creation.
Security Services	**Authenticity** of Sender, **integrity** of the document and **non-repudiation**.	It provides security and **authenticity** of certificate holder.
Standard	It follows Digital Signature Standard (DSS).	It follows X.509 Standard Format

Source:

https://www.geeksforgeeks.org/digital-signatures-certificates/

Introduction to Digital Certificates

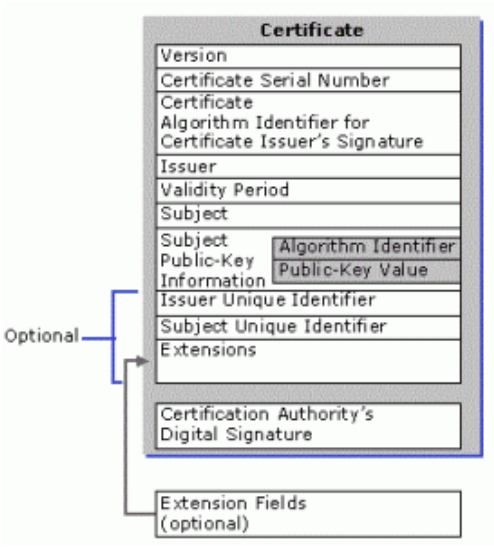
What is a digital certificate?

A digital signature or ID is more commonly known as a digital certificate. To digitally sign an Office document, you must have a current (not expired) digital certificate. Digital certificates are typically issued by a certificate authority (CA), which is a trusted third-party entity that issues digital certificates for use by other parties. There are many commercial third-party certificate authorities from which you can either purchase a digital certificate or obtain a free digital certificate. Many institutions, governments, and corporations can also issue their own certificates.

A digital certificate is necessary for a digital signature because it provides the public key that can be used to validate the private key that is associated with a digital signature. Digital certificates make it possible for digital signatures to be used as a way to authenticate digital information.

Digital certificates function similarly to identification cards such as passports and drivers' licenses. Digital certificates are issued by recognized (government) authorities. When someone requests a certificate, the authority verifies the identity of the requester, certifies that the requester meets all requirements to receive the certificate, and then issues it. When a digital certificate is presented to others, they can verify the identity of its owner because the certificate provides the following security benefits:

- It contains personal information to help identify and trace the owner.
- It contains the information that is required to identify and contact the issuing authority.
- It is designed to be tamper-resistant and difficult to counterfeit.
- It is issued by an authority that can revoke the identification card at any time (for example, if the card is misused or stolen).
- It can be checked for revocation by contacting the issuing authority.



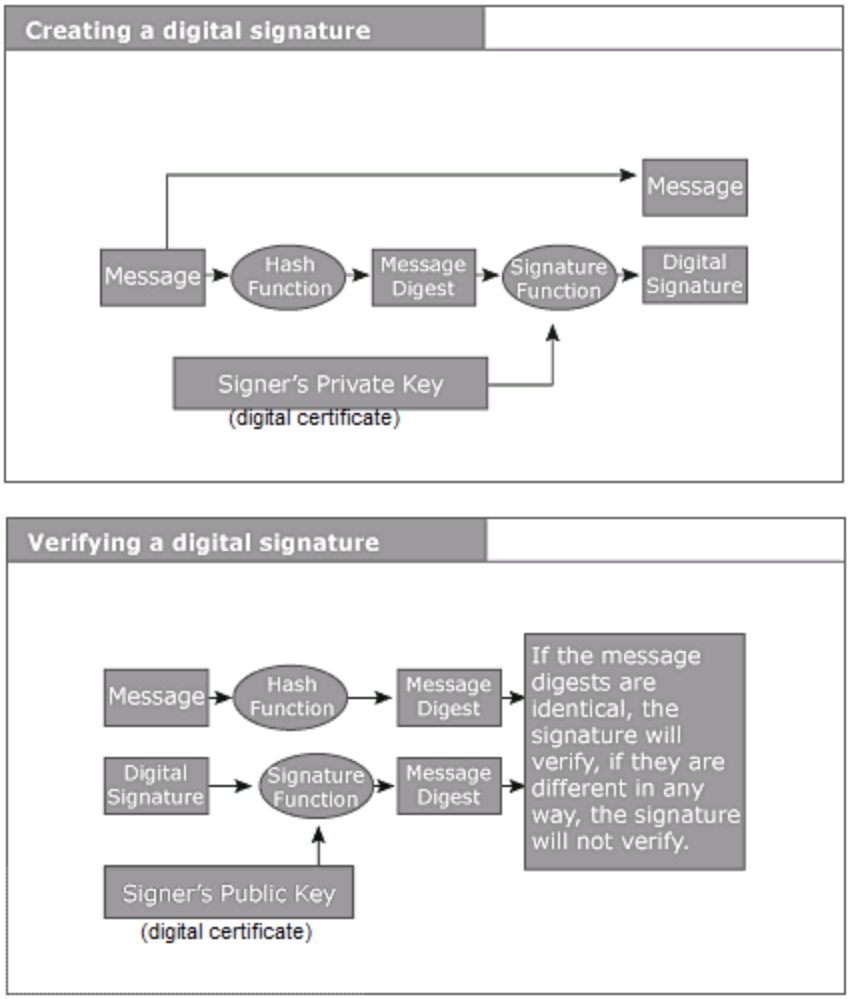
A digital certificate is necessary for a digital signature because it provides the public key that can be used to validate the private key that is associated with a digital signature. Digital certificates make it possible for digital signatures to be used as a way to authenticate digital information.

Example of a Digital Certificate

The use of a digital certificate to sign documents

When the signer uses a certificate to digitally sign a document, other people (known as relying parties) can trust the digital signature because they trust the CA has done their part to ensure the signer matches their digital identity.

So, technically speaking the difference between a digital signature and digital certificate is that a certificate binds a digital signature to an entity, whereas a digital signature is to ensure that a data/information remain secure from the point it was issued. In other words: digital certificates are used to verify the trustworthiness of a person (sender), while digital signatures are used to verify the trustworthiness of the data being sent.



Creating a digital signature using digital certificates (private key)

The difference between a digital signature and digital certificate

Digital business with digital *trustA* digital signature and a digital certificate, while both security measures, are different in the ways they are implemented and the background why they are implemented for. The technology industry loves to use acronyms and words that seem to either overlap with other similar words, or that are a slight variation on a word, but with widely different meanings. To understand more, these are the comparison Between Digital Signature vs Digital Certificate (Infographics)

Digital Signature	Digital Certificate
	
It verifies the identity of the document.	It verifies the identity of the ownership of an online medium.

Digital Signature	Digital Certificate
	
It is issued to a specific individual by an authorized agency.	It is issued after the background check of the applicant by the Certificate Authority(CA).

Digital Signature	Digital Certificate
	
It ensures that the signer cannot non-repudiate the signed document.	It ensures that two parties who are exchanging the information are secured.

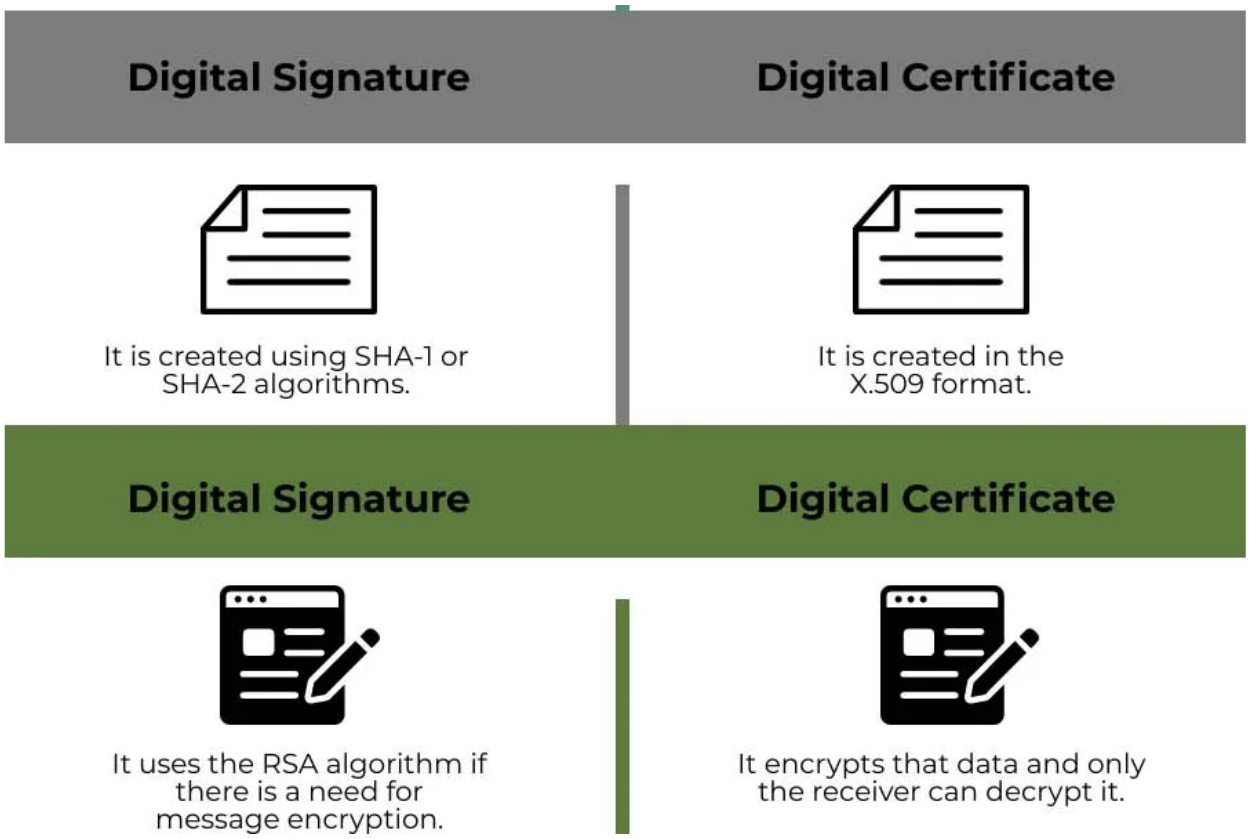
Digital Signature	Digital Certificate
	
It works on DSS (Digital Signature Standard).	It works on the principles of public-key cryptography standards.

Digital Signature	Digital Certificate
	
The digital signature uses a mathematical function (Hashing function).	It contains personal information to help in identifying the trace of the owner.

Digital Signature	Digital Certificate
	
It is widely used for avoiding forging the document.	It is used in an online transaction for the trustworthiness of the data and the sender.

Digital Signature	Digital Certificate
	
It is an attachment to a document that can be viewed as a signature.	It is a medium to prove the holder's identity for a particular transaction.

Digital Signature	Digital Certificate
	
It ensures the sender and the receiver have the same document containing the same data.	It builds the trust between the user and the business (Certificate holder).



Key Difference Between Digital Signature vs Digital Certificate

Let us discuss some of the major key differences between Digital Signature and Digital Certificate:

- The digital signature is used to identify the owner of the document whereas the digital certificate is a document that identifies the identity of the organization.
- The digital signature is signed created by the signer's private key and verified by the public key of a signer whereas digital certificate is issued by a third party and an end-user can check its validity and authenticity.
- The digital signature uses a mathematical function (Hashing function) wherein a Digital certificate contains personal information to help in identifying the trace of the owner.
- A digital signature is created using DSS (Digital Signature Standard) whereas digital certificate works on the principles of public-key cryptography standards.
- A digital signature uses the RSA algorithm when there is a need for message encryption whereas digital certificate is proof that the data transmission will be on the secured layer and in an encrypted way.
- Digital signatures are used to validate the sent data whereas digital certificates are used to validate the identity of the sender.
- With a digital certificate, an end user may have a relationship with the sender whereas in the digital certificate the end user trusts the third party and does not have a relationship with the business owner or the entity.

Digital Signature vs Digital Certificate Comparison Table

Let’s discuss the top comparison between Digital Signature vs Digital Certificate:

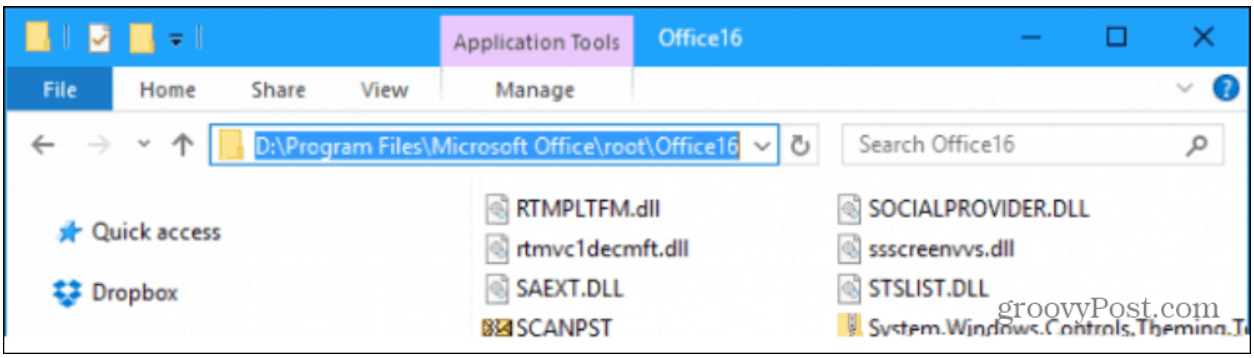
Digital Signature	Digital Certificate
It verifies the identity of the document.	It verifies the identity of the ownership of an online medium.
It is issued to a specific individual by an authorized agency.	It is issued after the background check of the applicant by the Certificate Authority(CA).
It ensures that the signer cannot non-repudiate the signed document.	It ensures that two parties who are exchanging the information are secured.
It works on DSS (Digital Signature Standard)	It works on the principles of public-key cryptography standards.
The digital signature uses a mathematical function (Hashing function).	It contains personal information to help in identifying the trace of the owner.
It is widely used for avoiding forging the document.	It is used in an online transaction for the trustworthiness of the data and the sender.
It is an attachment to a document that can be viewed as a signature.	It is a medium to prove the holder’s identity for a particular transaction.
It ensures the sender and the receiver have the same document containing the same data.	It builds the trust between the user and the business (Certificate holder).
It is created using SHA-1 or SHA-2 algorithms.	It is created in the X.509 format.
It uses the RSA algorithm if there is a need for message encryption.	It encrypts that data and only the receiver can decrypt it.

How to Create a Self-Signed Digital Certificate in Microsoft Office 2016

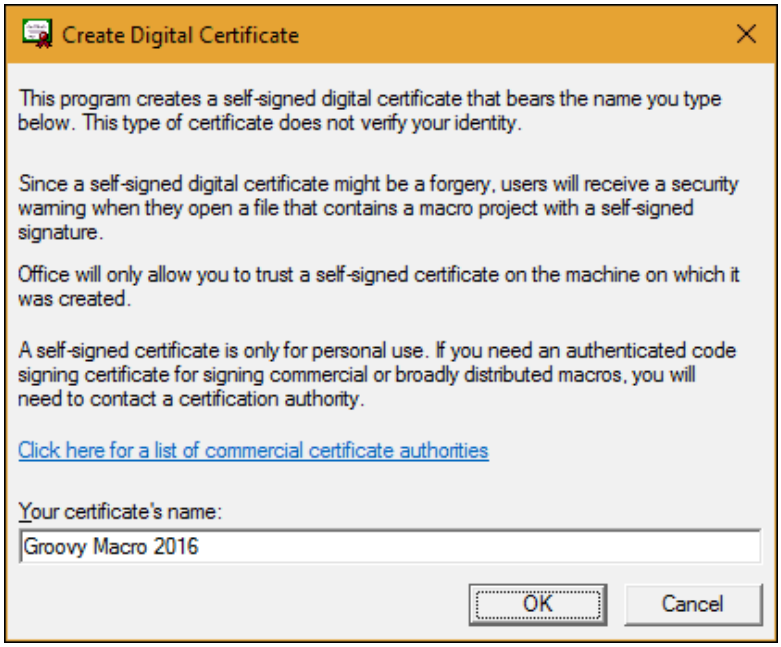
One of the most compelling parts of the Microsoft Office productivity suite for power users is automating functionality using Visual Basic for Application code. Applications such as Word, Excel, and Outlook can be used to create Macros. Macros are small bits of programming code used for performing repetitive tasks. In versions of Office before 2007, VBA support was notorious for being exploited. Since then, Microsoft has enhanced the security within the suite, limiting the impact of rogue code causing potential damage.

Setup Self-Signed Digital Certificate in Office 2016 Applications

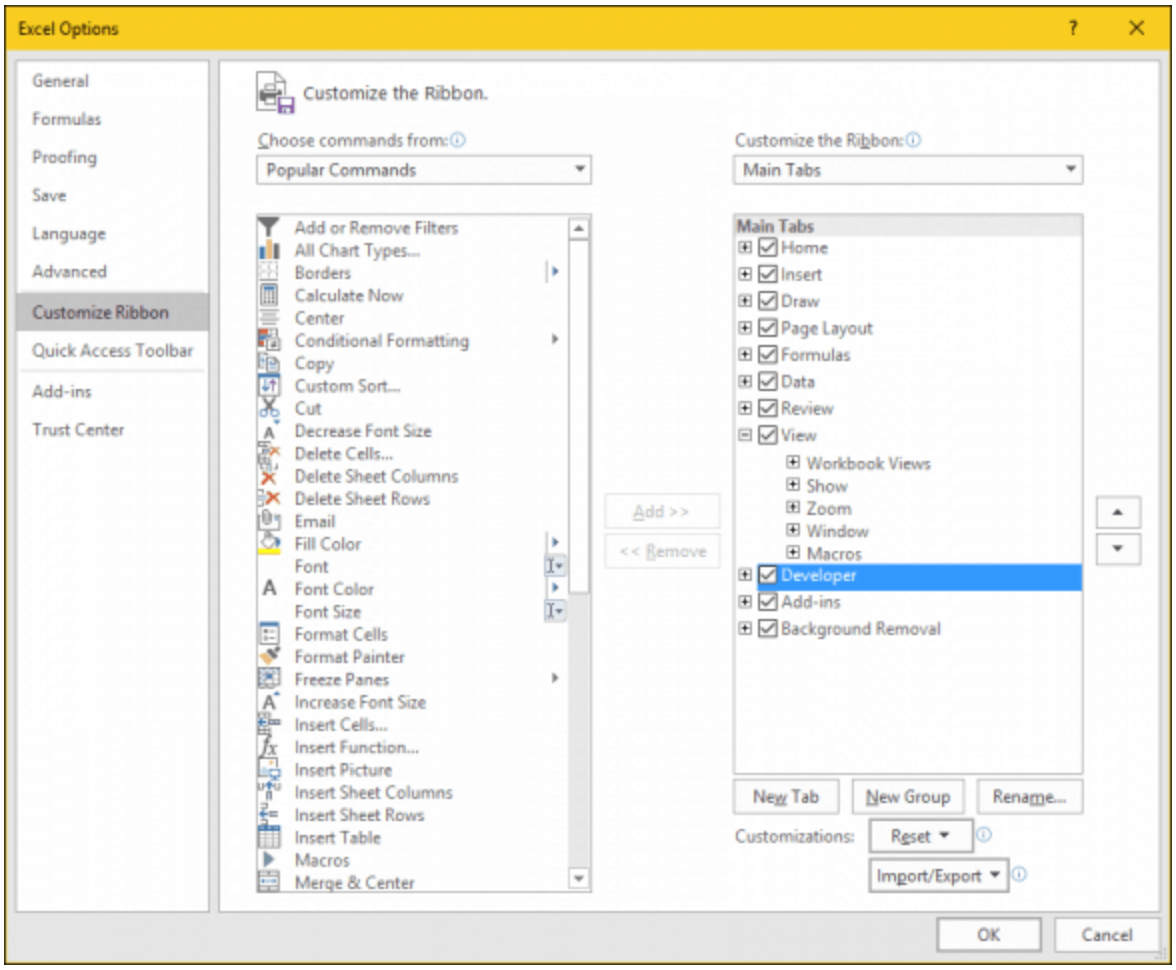
The Digital Certificate for VBA Projects can now be found within **Program Files > Microsoft Office > root > Office16**.



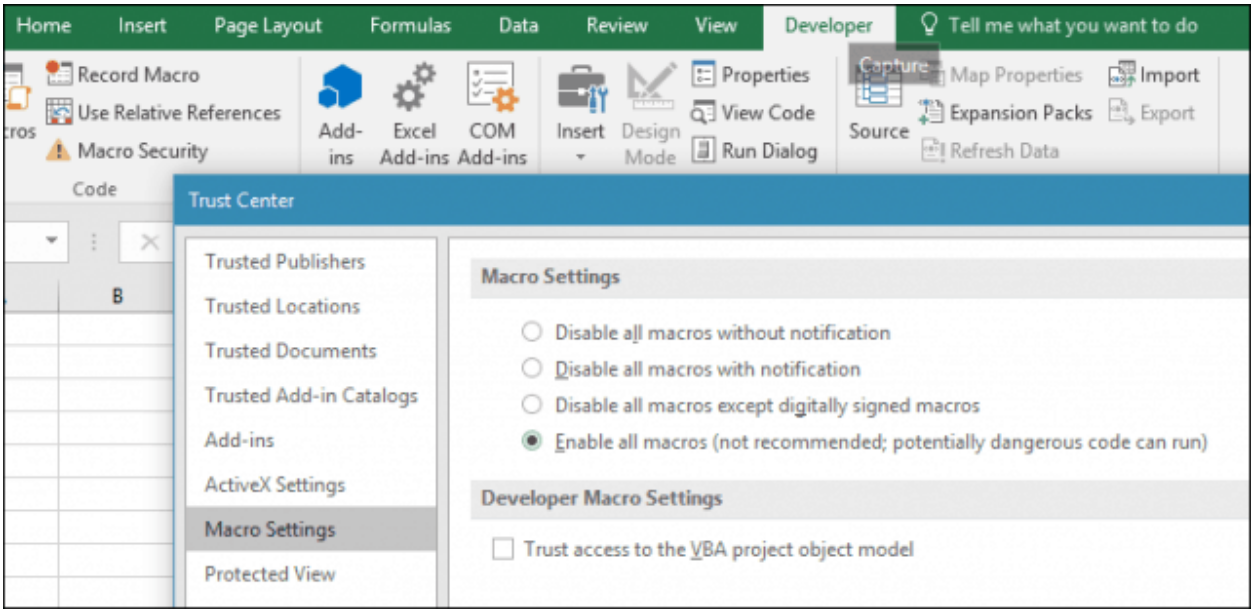
Double click the SELFCERT file, enter a name for your Digital Certificate, then click OK.



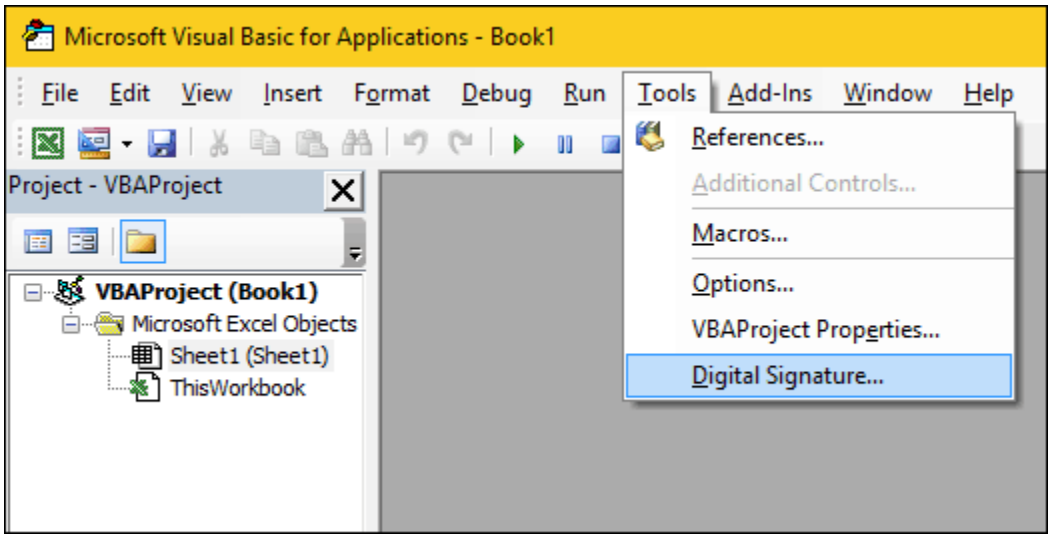
Launch any of the Office applications you would like to use the digital certificate in. For this article, I am going to use Excel. The first thing you will need to do is enable the *Developer* tab. Click File > Options > Customize Ribbon > check the box *Developer* then click OK.



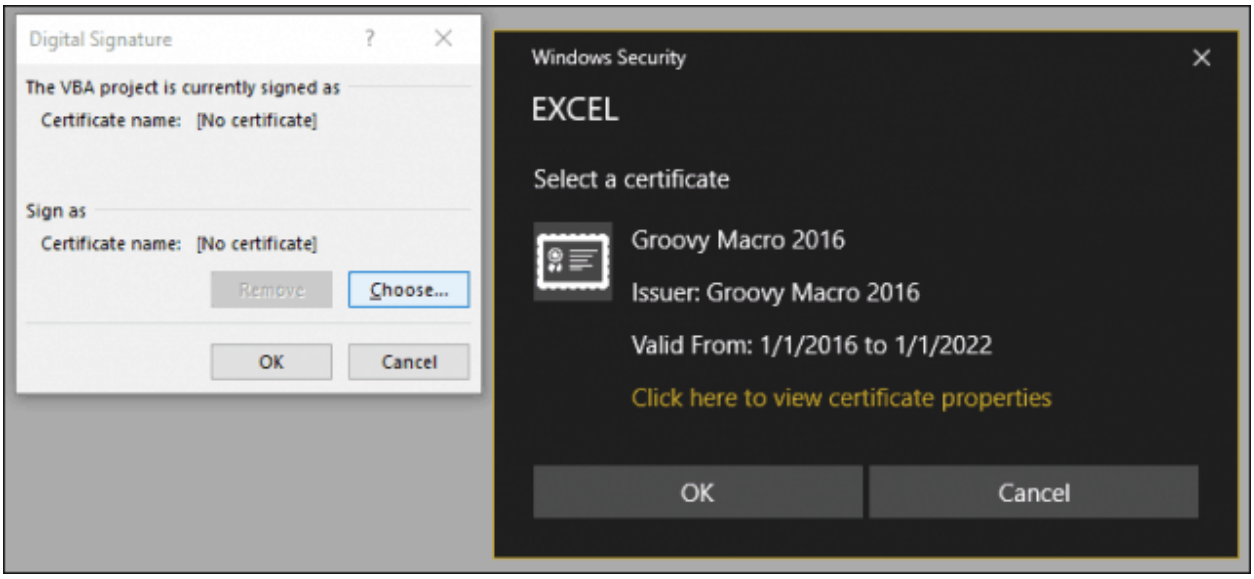
Select the Developer tab, then click the *Macro Security* button within the *Code* group, select the *Enable all Macros* radio box, then click *OK*.



Within the *Code* group, click *Visual Basic*. The Visual Basic for Applications component will be launched. Next, click Tools, then click Digital Signature.

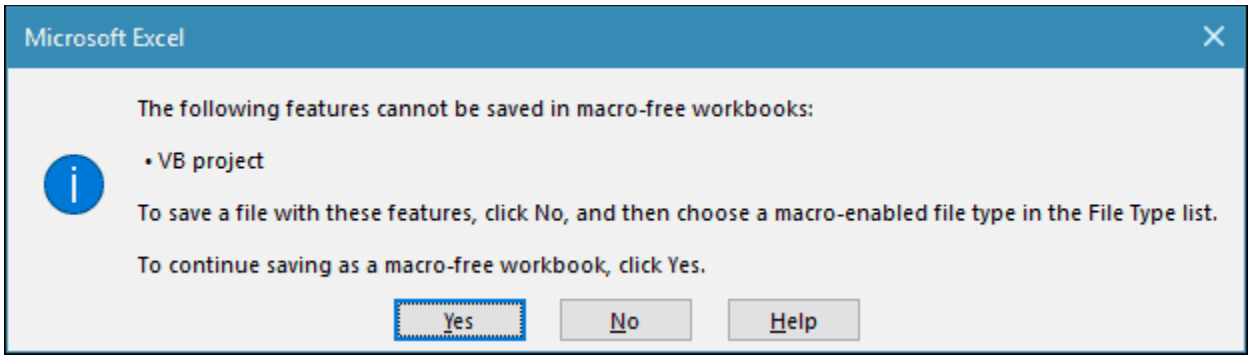


Click Choose, the recently created digital certificate will be presented. Click OK, then proceed to save your project.

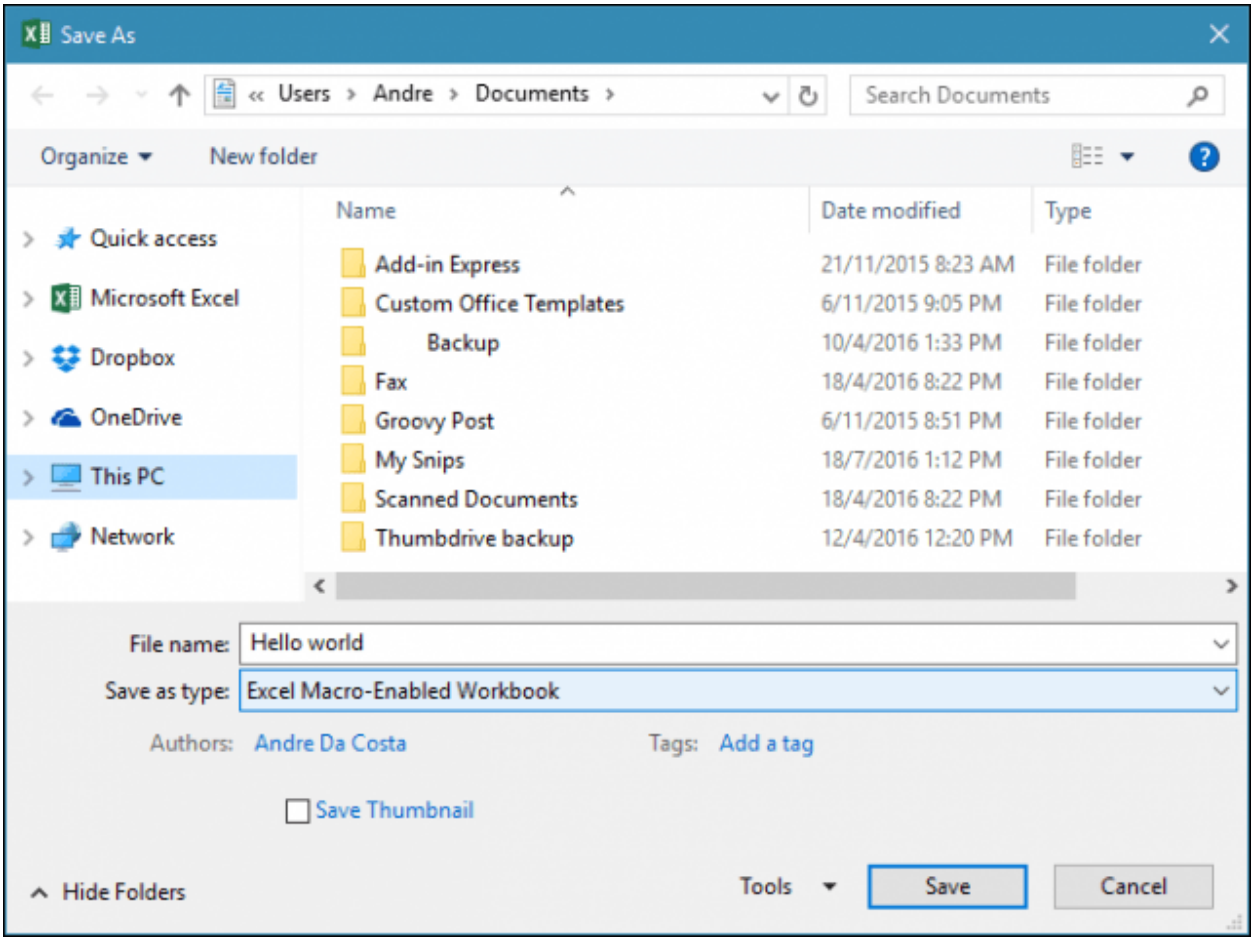


Ensuring your Macros Work

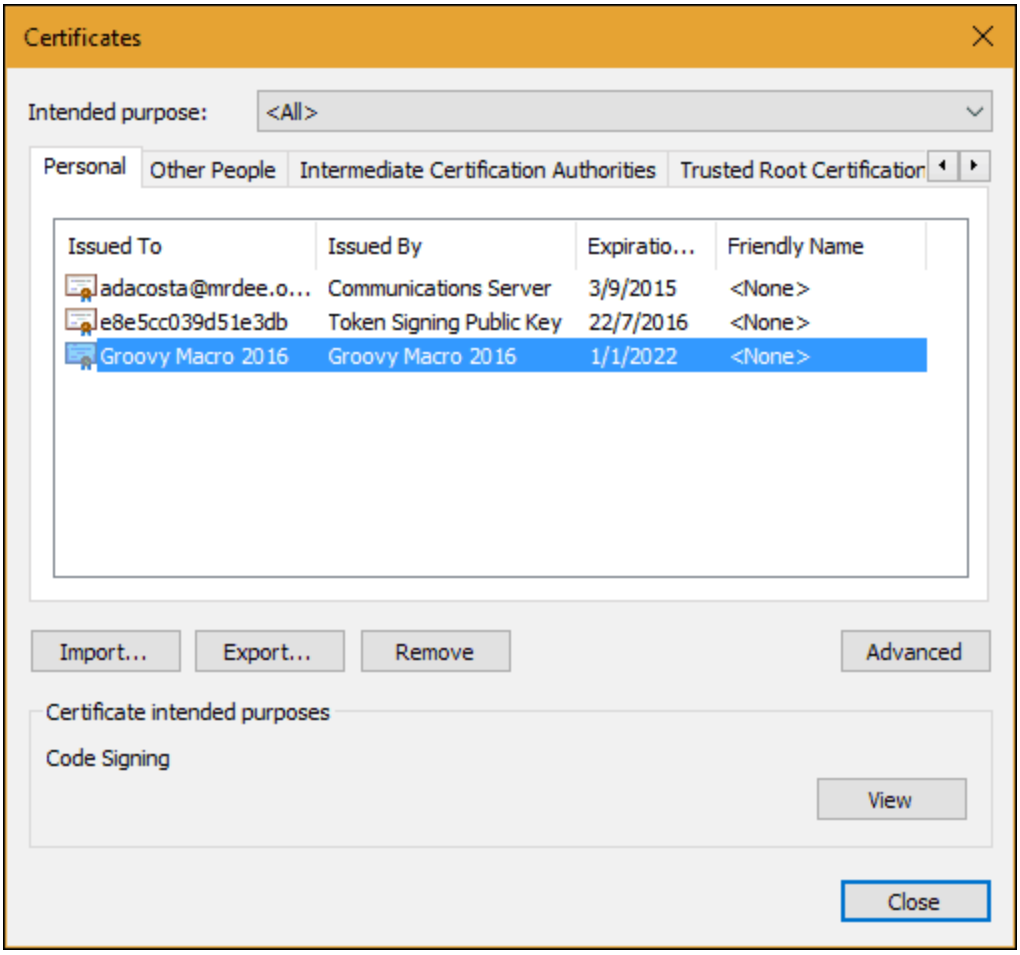
I noted earlier; Microsoft has made security changes to how Macros work in Office applications over the years. Saving your Macro's is not allowed in a standard workbook or document.



Instead, users must correctly choose Macro-Enabled as the file type when saving.



Users can manage their signed certificate by using launching Internet Options. First, click Start, then **type:** *internet options*, hit Enter on your keyboard, select the *Content* tab, then click *Manage Certificates*. Here you have the choice of deleting or exporting your certificate for use on another computer.


Certificates
Intended purpose:
<All>
Personal
Other People
Intermediate Certification Authorities
Trusted Root Certificatior
Issued To
Issued By
Expiratio...
Friendly Name
adacosta@mrdee.o...
Communications Server
3/9/2015
<None>
e8e5cc039d51e3db
Token Signing Public Key
22/7/2016
<None>
Groovy Macro 2016
Groovy Macro 2016
1/1/2022
<None>
Import...
Export...
Remove
Advanced
Certificate intended purposes
Code Signing
View
Close

Sources:

How to Create a Self-Signed Digital Certificate in Microsoft Office 2016 (groovypost.com)

What is Digital Certificate? | A Technology Overview from Comodo

PowerPoint Presentation (globalsign.com)

What is a Digital Signature? (techtarget.com)

What Is a Digital Signature (and How Does it Work) | Signaturely

The difference between a digital signature and digital certificate » AET Europe