View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by International Institute for Science, Technology and Education (IISTE): E-Journals

Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 2, No.6, 2011

Establishment of Public Key Infrastructure for Digital Signatures

M. Indrasena Reddy (Corresponding author)
R. G. M. College of Enginereing & Technology, Nandyal, A.P, India.
Tel:+91 8106448352 E-mail: mir555mittapalli@gmail.com

P.J. Bhat and Rajeev Chetwavani
ISRO, Satellite centre, Bangalore, India.
E-mail: pjbhat@isac.gov.in E-mail: rajivchetwani@gmail.com

M. Purushotham Reddy
CBIT College of Engineering, Proddatur
Email: purushotham.mps@gmail.com

Abstract

Open Security Socket Layer (SSL) is a cryptographic library that uses appropriate security systems such as encryption, digital signatures, digital certificates, public/private key pairs, non-repudiation and time-stamping to participate in the cryptography. A Public Key Infrastructure (PKI) comprises a system of certificates, certificate authorities, subjects, relying partners, registration authorities and key repositories that provide for safe and reliable communications. In this paper, open SSL has been implemented to provide an alternative to the Transmission Control Protocol (TCP). Open SSL is a real time protocol in which the parties negotiate interactively to authenticate each other and establish a session key, in contrast to a protocol such as email in which one party prepares a message encrypt and send, that can later be decrypted and authenticated by the intended recipient.

Keywords: Open SSL, Public Key Infrastructure, Digital signatures

1. Introduction

Transmission control protocol (TCP) can play an important role to deliver the data across the network. But TCP will not participate in the cryptography; it will have no way of noticing if malicious data is inserted into the packet stream, as the bogus data passes the (non-cryptographic) TCP checksum. TCP will acknowledge such data and send it. When the real data arrives, TCP will assume it is duplicate data and discard it, since it will have the same sequence numbers as the bogus data.

To address these, in this paper SSL provides an alternative to the standard TCP/IP socket API that has security implemented within it. The philosophy of SSL is that it is easier to deploy something if you don't have to change the operating system, so these protocols act “above TCP”. It requires applications to interface to SSL instead of TCP. The primary goal of the SSL protocol is to provide a private channel between communicating applications, which ensures privacy of data, authentication of the partners, and integrity.

Open SSL is a derived work from SSLeay. SSLeay was originally written by Eric A. Young and Tim J. Hudson beginning in 1995. In December 1998, development of SSLeay ceased, and the first version of Open SSL was released as 0.9.1c, using essentially two tools in one: a cryptography library and an SSL toolkit Pravir Chandra *et al* (2002) .The open SSL have the following advantages:

- Open SSL is a cryptographic library; it provides implementations of the industry's best-regarded algorithms, including encryption algorithms such as 3DES ("Triple DES"), AES and RSA, as well as message digest algorithms and message authentication codes.

- The SSL library provides an implementation of all versions of the SSL protocol, including TLSv1. The cryptography library provides the most popular algorithms for symmetric key and public key cryptography, hash algorithms, and message digests. It also provides a pseudorandom number generator, and support for manipulating common certificate formats and managing key material. There are also general-purpose helper libraries for buffer manipulation and manipulation of arbitrary precision numbers. Additionally, Open SSL supports most common cryptographic acceleration hardware (prior to Version 0.9.7, forthcoming as of this writing, hardware support is available only by downloading the separate "engine" release).
- Open SSL is the only free, full-featured SSL implementation currently available for use with the C and C++ programming languages. It works across every major platform, including all Unix Oss and all common versions of Microsoft Windows.

The Open SSL makes use the Commands of a configuration file. Although only a few commands currently make any use of a configuration file, other commands may be modified in the future to take advantage of them. Each command that currently uses the configuration file reads its base configuration information from a section that shares the name of the command. Other sections that are not named after a command may exist, and quite frequently, they do. Many keys' values are interpreted as the name of a section to use for finding more keys.

The following examples illustrate the usage of Open SSL commands.

Openssl genrsa -out priv.pem 1024

This generates 1024-bit RSA private key and writes the result into priv.pem.

Openssl req -new -config REQ_CONF_FILE -key priv.pem -out request.pem

This command is used to create, examine and manipulate certificates. It uses the REQ_CONF_FILE as a configuration file and specifies to use the private key present in the file priv.pem in the certificate by Pravir Chandra *et al* (2002).

Openssl ca -in request.pem -out cert.pem –passim pass: CA_PASSWORD Input File

The ca command is a basic certification authority that can be used to issue X.509 certificates and certificate revocation lists William Stallings (1999). Creates a certificate signed by CA and send it to the file containing the certificates. It uses the CA_PASSWORD as a password.

2. PKI background and standards developments

Public key cryptography was conceived by Diffie and Hellman (1976, 1977) and Rivest, Shamir in and Adleman designed the RSA Cryptosystems (1978), the first public key system. Each public key cryptosystem has its own technical features, however they all share the property that given an encryption key it is computationally infeasible to determine the decryption key and vice versa. Theoretically, no confidential information needs to be exchanged before secure communication is possible. Everyone has access to the recipient's public key and even though the communication is private, the message cannot be authenticated. This shows that public key cryptography on its own is not enough. If traditional paper based commerce are to be reproduced in the electronic environment, the following are required:

- Security policies to define the rules under which cryptographic systems should operate
- Products to generate store and manage certificates and their associated keys
- Procedures to dictate how keys and certificates are generated and distributed

A trusted and authenticated key distribution infrastructure is necessary to support the use of public keys in a public network such as the Internet. Recent efforts in standardization have seen developments on a number of fronts.

2.1 Evolution of PKI standards

The X.509 Recommendation provides a useful basis for defining data formats and procedures for the distribution of public keys via certificates that are digitally signed by CAs. X.509 does not however include a profile to specify the supporting requirements for many of the certificate's sub- fields, extensions or for

some data values.

The standards effort produced an outline for PKI of X.509 Version 3 certificates as well as Version 2 Certificate Revocation Lists. The Internet PKI profile went through eleven draft versions before becoming RFC 2459 Housley *et al* (1999). Other profiles have been developed for particular algorithms to make use of RFC 2459.

The development of the PKI management protocols has gone though a number of iterations. RFC 2510 Adams and Farrell (1999) was developed to specify a message protocol to be used between entities in a PKI. The need for an enrolment protocol and the preference to use PKCS#10 message format as the certificate request syntax lead to two parallel developments.

The Certificate Request Syntax was developed in the S/MIME WG which used PKCS#10 Yongge Wang (1993) as the certification request message format. Certificate Request Message Format RFC 2511 Myers *et al* (1999) draft was also developed but in the PKIX WG. It was to define a simple enrolment protocol that would work for the RFC 2510 Adams and Farrell (1999) enrolment protocols, but it did not use PKCS#10 as the certificate request message format. Then, RFC 2510 Adams and Farrell (1999) and Myers *et al* (1999) were developed to define an extended set of management messages that flow between the components of the Internet PKI. These, combined with CMS Myers et *al* (1999) allowed the use of an existing protocol (S/MIME) as a PKI management protocol, without requiring the development of an entirely new protocol such as CMP Adams and Farrell (1999). It also included PKCS#10 as the certificate request syntax.

Development of the operational protocols has been more straightforward. Two documents for LDAP have been developed — one for defining LDAPv3 as an access protocol to repositories Wahl *et al* (1997) and one for storing PKI information in an LDAP directory Boeyen *et al* (1999). Using FTP and HTTP to retrieve certificates and CRLs from PKI repositories is specified in RFC 2585 Housley and Hoffman (1998).

3. Public Key Infrastructure

Public Key Infrastructure (PKI) provides the means to establish trust by binding public keys and identities, thus giving reasonable assurance that we're communicating securely with who we think we are.

Using public key cryptography, we can be sure that only the encrypted data can be decrypted with the corresponding private key. If we combine this with the use of a message digest algorithm to compute a signature, we can be sure that the encrypted data has not been tampered with. What's missing is some means of ensuring that the party we're communicating with is actually who they say they are. In other words, trust has not been established. This is where PKI fits in.

In the real world, we often have no way of knowing firsthand who a public key belongs to, and that's a big problem. Unfortunately, there is no sure-fire way to know that we're communicating with who we think we are. The best we can do is extending our trust to a third party to certify that a public key belongs to the party that is claiming ownership of it.

Public Key Infrastructure is very successful in providing a robust and rigorous security measures to protect user data and credentials. It ensures trust in electronic transaction Carlisle Adams, Steve Lioyd, and Second Edition. Addison Wesley, s November (2002). . This Public Key Infrastructure provides a security services such as:

- **Authentication:** It is an ability to verify the identity of an entity.
- **Confidentiality:** Data is kept secret from those without the proper credentials, even if that data travels through an insecure medium. In practice, this means potential attackers might be able to see garbled data that is essentially "locked," but they should not be able to unlock that data without the proper information. In classic cryptography, the encryption (scrambling) algorithm was the secret. In modern cryptography, that isn't feasible. The algorithms are public, and cryptographic keys are used in the encryption and decryption processes.
- **Integrity:** The basic idea behind data integrity is that there should be a way for the recipient of a piece of data to determine whether any modifications are made over a period of time. For example, integrity

checks can be used to make sure that data sent over a wire isn't modified in transit. Plenty of well-known checksums exist that can detect and even correct simple errors. However, such checksums are poor at detecting skilled intentional modifications of the data. Several cryptographic checksums do not have these drawbacks if used properly.

- **Non-Repudiation:** Cryptography can enable receiver to prove that a message received from sender actually came from sender itself. Sender can essentially be held accountable when he sends receiver such a message, as he cannot deny (repudiate) that he sent it. In the real world, you have to assume that an attacker does not compromise particular cryptographic keys. The SSL protocol does not support non-repudiation, but it is easily added by using digital signatures.

The Components of Public Key Infrastructure are:

- Cryptographic Module
- Certifying Authority
- Registration Authority
- Certificate Repository

3.1 Cryptographic Module

Cryptography Module consists of 2 types :

- Symmetric Key Cryptography.
- Public Key Cryptography.

3.1.1 Symmetric Key Cryptography

Symmetric key algorithms encrypt and decrypt data using a single key. As shown in Figure 1 the key and the plaintext message are passed to the encryption algorithm, producing cipher text. The result can be sent across an insecure medium, allowing only a recipient who has the original key to decrypt the message, which is done by passing the cipher text and the key to a decryption algorithm. Obviously, the key must remain secret for this scheme to be effective.

The primary disadvantage of symmetric key algorithms is that the key must remain secret at all times. In particular, exchanging secret keys can be difficult, since you'll usually want to exchange keys on the same medium that you're trying to use encryption to protect. Sending the key in the clear before you use it leaves open the possibility of an attacker recording the key before you even begin to send data.

3.1.2 Public Key Cryptography

Public key cryptography suggests a solution to the key distribution problem that plagues symmetric cryptography. In the most popular form of public key cryptography, each party has two keys, one that must remain secret (the *private key*) and one that can be freely distributed (the *public key*). The two keys have a special mathematical relationship. For Alice to send a message to Bob using public key encryption (see Figure 2), Alice must first have Bob's public key. She then encrypts her message using Bob's public key, and delivers it. Once encrypted, only someone who has Bob's private key can successfully decrypt the message (hopefully, that's only Bob).

Public key encryption solves the problem of key distribution, assuming there is some way to find Bob's public key and ensure that the key really does belong to Bob. In practice, public keys are passed around with a bunch of supporting information called a *certificate*, and those certificates are validated by trusted third parties. Often, a trusted third party is an organization that does research (such as credit checks) on people who wish to have their certificates validated. SSL uses trusted third parties to help address the key distribution problem.

3.2 Certifying Authority

3.2.1Certificates

At the heart of PKI is something called a *certificate*. In simple terms, a certificate binds a public key with a *distinguished name*. A distinguished name is simply the name of the person or entity that owns the public

key to which it's bound. Perhaps a certificate can be best compared to a passport, which binds a picture with a name, thus solidifying a person's identity. A passport is issued by a trusted third party (the government) and contains information about the person to whom it has been issued (the subject) as well as information about the government that issued it (the issuer). Similarly, a certificate is also issued by a trusted third party, contains information about the subject, and contains information about the third party that issued it.

Not unlike a passport, which contains a watermark used to verify its authenticity, a certificate also contains safeguards intended to allow the authenticity of the certificate to be verified, and aid in the detection of forgery or tampering. Also similar to a passport, a certificate is valid only for a defined period. Once it has expired, a new certificate must be issued, and the old one should no longer be trusted.

A certificate is signed with the issuer's private key, and it contains almost all of the information necessary to verify its validity. It contains information about the subject, the issuer, and the period for which it is valid. The key component that is missing is the issuer's certificate. The issuer's certificate is the key component for verifying the validity of a certificate because it contains the issuer's public key, which is necessary for verifying the signature on the subject's certificate.

By signing a certificate with the issuer's private key, anyone that has the issuer's public key can verify its authenticity. The signature serves as a safeguard to prevent tampering. By signing the subject's certificate, the issuer asserts that it has verified the authenticity of the public key that the certificate contains and states that it may be trusted. As long as the issuer is trusted, the certificates that it issues can also be trusted.

It's important to note that the issuer's certificate or public key may be contained in an issued certificate. It's more important to note that this information cannot be trusted to authenticate the certificate. If it was trusted, the element of trust established from a third party is effectively eliminated. Anyone could create another key pair to use in signing a certificate and place that public key in the certificate.

Certificates are also created with a serial number embedded in them. The serial number is unique only to the issuer of the certificate. No two certificates issued by the same issuer should ever be assigned the same serial number. The certificate's serial number is often used to identify a certificate quickly.

3.2.2 Certification Authorities

A Certification Authority (CA) is an organization or company that issues certificates. By its very nature, a CA has a huge responsibility to ensure that the certificates it issues are legitimate. That is, the CA must ensure beyond all reasonable doubt that every certificate it issues contains a public key that was issued by the party that claims to have issued it. It must be able to produce acceptable proof for any certificate that it issues on demand. Otherwise, how can the CA itself be trusted?

There are two basic types of CAs. A private CA has the responsibility of issuing certificates only for members of its own organization, and is likewise trusted only by members of its own organization. A public CA, such as VeriSign or Thawte, has the responsibility of issuing certificates for any member of the public, and must be trusted by the public. The burden of proof varies depending on the type of CA that has issued a certificate and the type of certificate that is issued.

A CA must be trusted, and so for that trust to be extended, its certificate containing its public key must be widely distributed. For public CAs, their certificates are generally published so that anyone can obtain them. More commonly, the software that makes use of them, such as a web browser, is shipped containing them. Most often, the software allows certificates from other CAs to be added to its list of trusted certificates, thus facilitating the use of private CAs with off-the shelf software.

3.2.3 Some Certificate Authority Providers

Worldwide, the certificate authority business is fragmented, with national or regional providers dominating their home market. This is because many uses of digital certificates, such as for legally binding digital signatures, are linked to local law, regulations, and accreditation schemes for certificate authorities.

However, the market for SSL Certificates, a kind of certificate used for website security, is largely held by a small number of multinational companies. This market has significant barriers to entry since new providers must undergo annual security audits (such as WebTrust for Certification Authorities) to be included in the list of web browsers trusted authorities. More than 50 root certificates are trusted in the most popular web

browser versions. A 2009 market share report from Net Craft as of January of that year determined that VeriSign and its acquisitions (which include Thawte and GeoTrust) have a 47.5% share of the certificate authority market, followed by GoDaddy (23.4%), and Comodo (15.44%) www.certicom.com, (1999).

Licensed Certificate Authorities in India:

- Tata Consultancy Services
- Safescrypt– A subsidiary of Satyam Infoway
- National Informatics Center– Govt. of India
- Institute for Development & Research in Banking Technology (IDRBT)– A society of Reserve Bank of India.

3.3 Registration authority

A registration authority (RA) is an authority in a network that verifies user requests for a digital certificate and tells the certificate authority to issue it. RAs are part of a public key infrastructure (PKI), a networked system that enables companies and users to exchange information and money safely and securely. The digital certificate contains a public key that is used to encrypt and decrypt messages and digital Signatures.

3.4 Certificate repository

The **Certificate Repository** consists of the following 2 components.

3.4.1 Certificate Internal Database

- This database can only be updated by RA server and CA server.
- Use to keep track of the pending certificate request, issued or revoked certificate, private Certificate Revocation List (CRL), etc.
- Web user interface via the RA Server will be provided for users to query the status of their certificate request and any issued or revoked certificate.
- Various fields in certificate, such as serial no, expiry date, subject name, etc will be indexed. This will allow faster queries based on these standard attributes.

3.4.2 LDAP Directory Server

- Can only be updated by CA server.
- Public repository of Certificate Revocation List (CRL), user and CA certificate.
- Based on the RFC 2587 schema.
- Provide standard LDAP Wahl *et al* (1997) interface to native client for retrieving certificate, e.g. S/MIME, SSL client authentication.

4. Digital Signatures

A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery and tampering Rivets and Adelman (1978).

Digital signatures are often used to implement electronic signatures, a broader term that refers to any electronic data that carries the intent of a signature, but not all electronic signatures use digital signatures. In some countries, including the United States, and members of the European Union, electronic signatures have legal significance.

Digital signatures employ a type of asymmetric cryptography. For messages sent through an insecure channel, a properly implemented digital signature gives the receiver reason to believe the message was sent by the claimed sender. Digital signatures are equivalent to traditional hand written in many respects; properly implemented digital signatures are more difficult to forge than the handwritten type. Digital signature schemes in the sense used here are cryptographically based, and must be implemented properly to be effective. Digital signatures can also provide non-repudiation, meaning that the signer cannot

successfully claim they did not sign a message, while also claiming their private key remains secret; further, some non-repudiation schemes offer a time stamp for the digital signature, so that even if the private key is exposed, the signature is valid nonetheless. Digitally signed messages may be anything represent able as a bit string: examples include electronic mail, contracts, or a message sent via some other cryptographic protocol.

A digital signature scheme typically consists of three algorithms:

- An algorithm that selects a private key uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding public key.
- A signing algorithm which, given a message and a private key, produces a signature.
- A signature verifying algorithm which given a message, public key and a signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required. First, a signature generated from a fixed message and fixed private key should verify the authenticity of that message by using the corresponding public key. Secondly, it should be computationally infeasible to generate a valid signature for a party who does not possess the private key.

5. Problem Description

In this paper, the working principle of the open SSL has been discussed. The sender first generates a key pair by using open SSL commands, and then sender will request a certificate to the certificate authority. CA generate the certificate along with the necessary information such as a private key, public key, certificate and combined all these in a password protected file (P12). Select a file to be signed and signing the chosen document and save the signature in a separate file and encrypt the file by using a session key and also encrypt the session key with sender's public key. Sender can send the signature file and encrypted session key to the recipient across the network. Recipient receives the document and decrypt with his own private key. He finds the session key along with it. He decrypts the signature file and verify signed document and signature file .If the decrypted file and original document are same, the receiver assumes the message came from authenticate sender only.

The steps used in implementation of the open SSL are:

5.1 Setting up of the PKI

The configuration file, openssl.cnf which is referred for three openssl commands. Namely, ca, request and x.509. Make the necessary changes to openssl.cnf present on Linux system .Create two new folders named CA and PKI. Run he openssl command to generate the self signed root certificate. When this command is run it asks the user to enter a password for the root certificate. The CA's private key saves in the cakey.pem file. Using this private key and the configuration file, we create x.509 v3 CA root certificate named cacert.pem. This is a self signed root certificate.

5.2 Request a certificate

For requesting a certificate, run the shell script with a user name and password as command line arguments. This password is used to protect the p12 file. In this script check is made to ensure both the arguments have been entered. If the directory '/PKI' does not contain the certificate of the specified user then, a new one is created. This folder shall be used for storing the p12 and the certificate of that particular user. In the file /PKI/[user]/req. conf the necessary req and req distinguished information of the user is stored. The user's pfx password is converted to octal format and along with the user's name, is stored in a file priv. password at the location /PKI/[user].Using the openssl command *genrsa*, a 1024 bit private key is generated and subsequently stored in a file priv.pem at the location /PKI/[user].A request for a certificate contains the public key corresponding to user's private key. Using the openssl command *req*, the public key is extracted and later is stored in request.pem at the location /PKI/[user].Next, we use openssl *ca* command. The CA uses it's password to perform a digital signature on the user's request. The outcome of this operation is a Certificate of the user duly signed by the CA. Also, in our project we used PKCS#12 format as an enhanced security measure. The p12 file is password protected file which contains the sensitive data such as private key of the user. So we use the openssl *pkcs12* command to generate the [*username].p12* file. The previously, readable pem format certificate is finally encoded into the unreadable DER X509 standard. For

convenience, we store the p12 files and certificates separately in folders /PKI/P12-files and /PKI/Certificates respectively.

5.3 Signing a Document

In this section, we describe the procedure for performing digital signature on a document. A digital signature is performed on any document opens the file to be signed, reads it and then return the data contained. After taking the p12 file-name and pfx password from the user, a key store of the type PKCS12_KEYSTORE_TYPE is created and the p12 file-name and its pfx password is stored in the key store. The private key is read from the key store and a call is made to the sign Document. The Digital Signature Algorithm (DSA) is used for digitally signing the document

5.4 Encrypt he signed document and signature file

In this section, select a digitally signed document for encryption. We can use AES (Advanced Encryption Standard) algorithm to encrypt digitally signed document. Generate a random session key by using AES algorithm and encrypt the digitally signed document. Now session key also encrypted with the receiver's public key. Encrypted signed document is stored Encrypt.txt file at any location of the user's choice. Send the encrypted session key and signed document to the receivers id. AES is the best symmetric key algorithm for encryption when compare to other algorithms.

5.5 Decrypt encrypted session key and signed document

In this session, receiver will receive the encrypted signed file and encrypted session key and select a p12 file to enter the password. The private key of the user is obtained from the p12 file. By using his own private key to decrypt the session key .By using decrypted session key receiver will decrypt the encrypted signed document. Decrypted documents is stored any location of the user's choice. Here public key and private keys are generated by using RSA Algorithm. RSA is the best asymmetric key algorithm to sharing the private key's between the sender and receiver.

5.6 Verification of the Signature

In this section, the procedure for verifying the digital signature on a document and sender's id has been explained. In this procedure, first read the signed document, signature file and x.509 certificate of the sender to extract his public key. Using the signed document, public key and signature, verify the authenticity of the signature. In order to verify the validity of the sender's certificate, the issuer's certificate must be available. So the CA's certificate needs stored in a folder to which the verification process access to. For the successful verification of the document, digitally signed document, signature and sender's id are necessary. Both are same signature and certificate to be verified here.

5.7 Developing the GUI

In this section, The GUI must provide easy means for letting the user utilize the services available. The certificate request page uses the username and password to create the certificate and p12 file for requestor. The signing page can be used to browse for a document to be signed. The p12 file and its password is selected for signing the document

The verification file accepts the signed document, the signature file and the sender's id. The sender's id is mapped onto his certificate. Results are displayed about the validity of the digital signature and the validity of the certificate.

6. Conclusions

In this paper, SSL provide an alternative to the TCP socket API that has security implemented within it. SSL provide a private channel between communicating applications, which ensure privacy of data, authentication of the partners, and integrity. We have established means for establishing trust between two parties who are communicating data and credentials on a network. We accomplish this by setting up a private CA. A private CA exists for serving the sole purpose of acting as a trusted third party in a closed company network.

A document is digitally signed before it is sent to receiver. The digital signature detects tampering in transit and is non-repudiable. The CA reinforces the trust for the receiver. This is possible because the CA's

signature on the sender's certificate tells the receiver that the signature was not performed by any random user in the network who was capable of generating a key pair. The signing and verification services are located centrally at a server. This means that the private key needs transporting to the server for signing. To ensure that the private key is not compromised in transit, we use p12 file to securely transport the private key.

REFERENCES

Pravir Chandra, Matt Messier, John Viega. (2002)," Network security with Openssl".

Diffie, W. and Hellman, M. E. (1976), "New Directions inCryptography", *IEEE Transactions on Information Theory*, **22** ,644-654.

Rivest, R., Shamir, A., & Adleman, L. (1978), "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications *of the ACM*, **21**, 120-126.

Housley, R., Ford, W., Polk, W., & Solo, D. (1999), RFC 2459 "Internet X.509 Public Key Infrastructure Certificate and CRL Profile".

Adams, C., Farrell, S., (1999) RFC 2510, "Internet X.509Public Key Infrastructure Certificate Management Protocols",

Yongge Wang. (1993), PKCS#10, RSA, "The Public-Key Cryptography Standards ", RSA Data Security Inc.,

Myers, M., Adams, C., Solo, D., and Kemp, D.,(1999) RFC 2511 "Internet X.509 Certificate Request Message Format",

Myers, M., Liu, X., Fox, B., & Weinstein, J.(1999), "Certificate Management Messages over CMS".

Wahl, M., Howes, T., & Kille, S. (1997), RFC 2251"Lightweight Directory Access Protocol (v3)".

Boeyen, S., Howes, T., & Richard, P.(1999), RFC 2587 "Internet X.509 Public Key Infrastructure LDAPv2 Schema".

Housley, R., and Hoffman, P. (1998), RFC 2585 "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP".

Carlisle Adams, &Steve Lioyd. (2002), "Understanding PKI: Concepts, standards, and Deployment Considerations",2nd Ed. Addison Wesley, November 2002.

Kaufman., Charlie., Perlman., Radia., Speciner., &Mike. (1995), "Network Security Private communication in a Public world", Prentice Hall.

William Stallings. (1999), "Cryptography and Network Security: Principles and Practice", 2nd Edition, Prentice Hall.

Diffe,W., & Hellman, M.E. (1976), "New Directions in Cryptography", *IEEE Transactions on Information Theory*, **22**, 644-654.

Rivets, R., A. and Adelman, L. (1978), "A Method for Obtaining Digital signatures and Public Key Cryptosystems", *Communications of the ACM*, **21**, 120-126.

Wahl, M., Howe's, T., Kille, S. (1997), RFC 2251"Lightweight Directory Access Protocol (V3).

www.certicom.com, Public- Key Infrastructure- The VeriSign Difference; VeriSign whitepaper (1999).

RSA Data Security , " Understanding PKI" (1999).

Ray Hunt. (2002) ," PKI and Digital Certification Infrastructure".

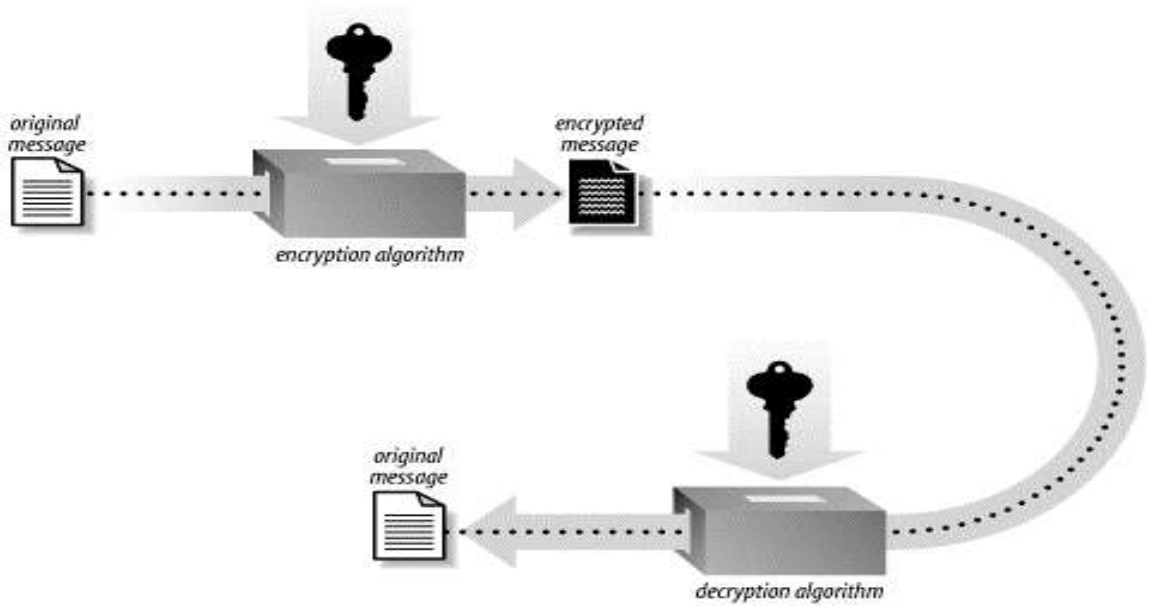

Figure 1. Symmetric Key Cryptography

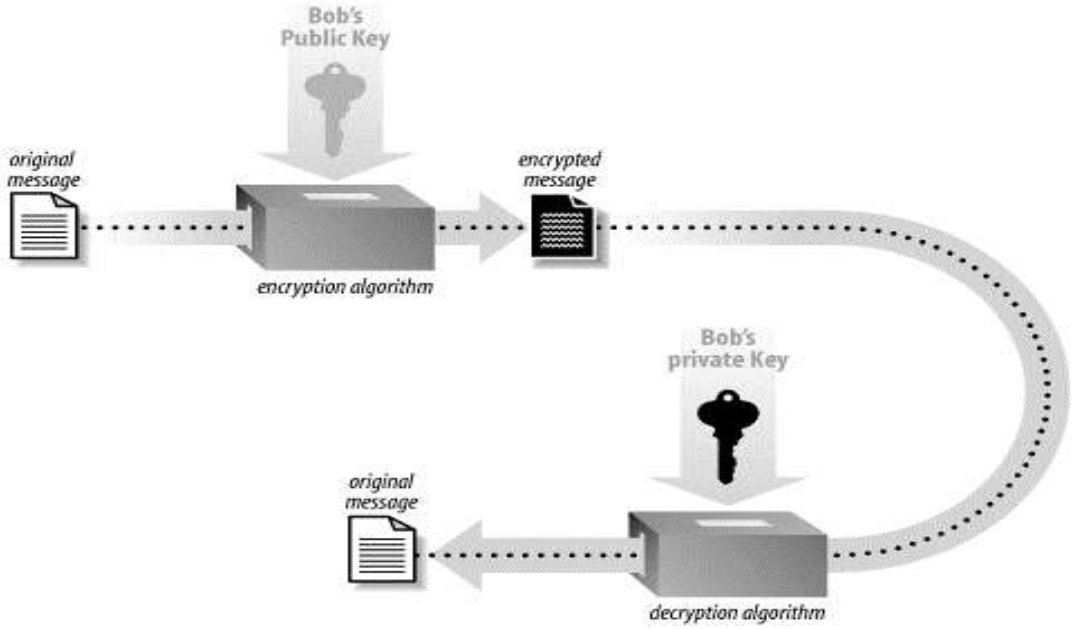

Figure 2. Public Key Cryptography.

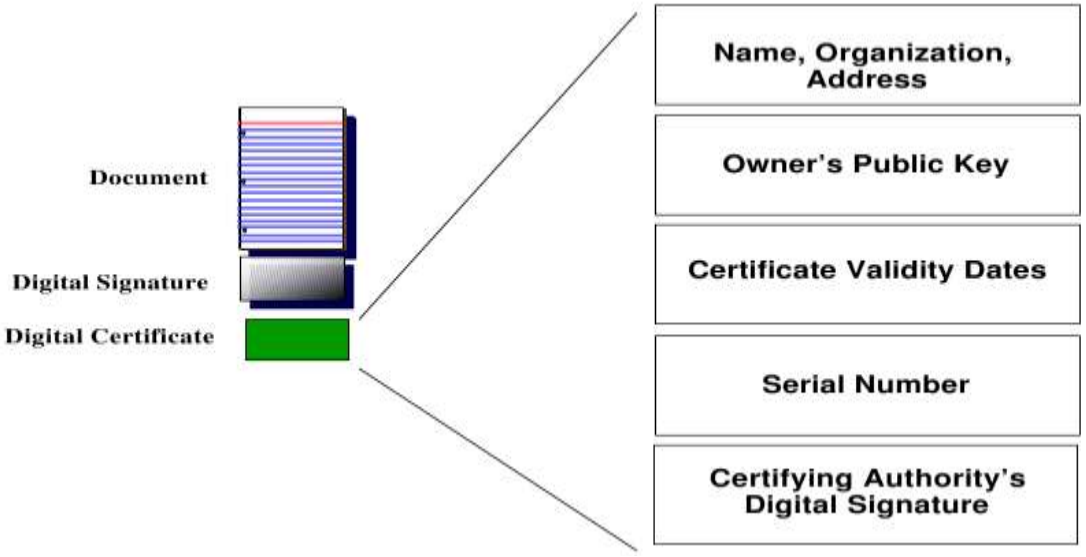

Figure 3. Structure of Digital Certificate.

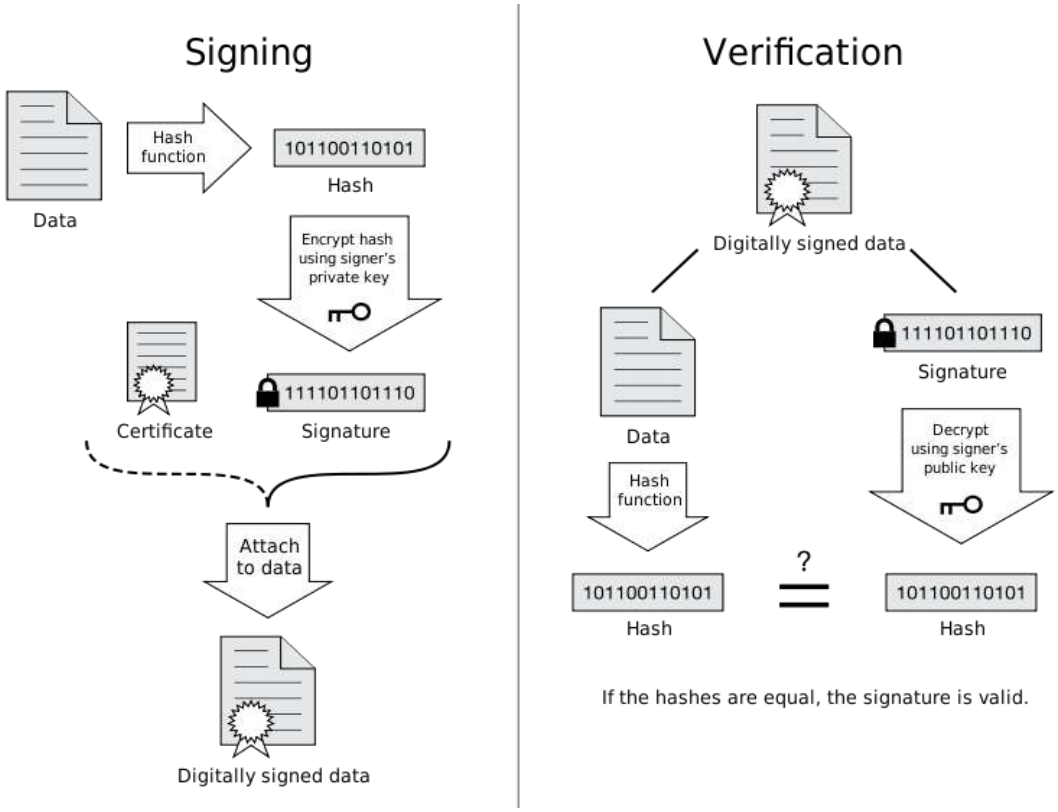

Figure 4. Diagram showing how a simple digital signature is applied and then verified

This academic article was published by The International Institute for Science, Technology and Education (IISTE). The IISTE is a pioneer in the Open Access Publishing service based in the U.S. and Europe. The aim of the institute is Accelerating Global Knowledge Sharing.

More information about the publisher can be found in the IISTE's homepage:
http://www.iiste.org

The IISTE is currently hosting more than 30 peer-reviewed academic journals and collaborating with academic institutions around the world. **Prospective authors of IISTE journals can find the submission instruction on the following page:** http://www.iiste.org/Journals/

The IISTE editorial team promises to the review and publish all the qualified submissions in a fast manner. All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Printed version of the journals is also available upon request of readers and authors.

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar