Trust Model in PGP and X.509 Standard PKI

Eyas Al-Hajery
V. 1.2b

1. Introduction

In real life, you can authenticate a person using an ID issued by a "trusted" entity. This ID can be, for example, a passport issued by a passport office or a driving license issued by a department of motor vehicle. However, how can you authenticate people in "cyber life" ?. *Digital signature*, developed using public-key cryptography, is a means for communicating entities in cyberspace to authenticate themselves to each other. **Public Key Infrastructure (PKI)** provides a framework to generate and publish public keys securely and efficiently. Furthermore, PKI should achieve and manage trust relationship among parties wishing to communicate securely. Different PKI trust models have been proposed [1,2]. In this paper, we present the trust model introduced in X.509 standard Public Key Infrastructure (referred as PKI) and the trust model in Pretty Good Privacy public-key cryptographic system (referred as PGP).

2. PKI, PGP and Cryptography

In this section, we provide a general overview of the cryptographic techniques that are essential in developing PKI.

Cryptographic algorithm is a set of mathematical functions used to convert understandable text "plain text" into obscure text "cipher", and vice versa. The process of converting plaintext into cipher text is called *encryption*, while *decryption* is the process of converting cipher text back to its corresponding plaintext [3,4]. A *key* is an essential parameter of the cryptographic algorithm. Different keys produce different ciphers for the same plain text and using the same cryptographic algorithm. Therefore, algorithms are made public while keys should be kept private to insure the secrecy of the encrypted information .

In **Private Key (Symmetric) Cryptography** , a single key is used to encrypt and decrypt the text. In order for Alice to send private information to Bob, both should share the same key that is used to encrypt and decrypt transmitted messages. On the other hand, in **Public Key (Asymmetric) Cryptography** , two different keys are used for encryption and decryption. A message encrypted using one key will only be decrypted using the other key. Both keys are mathematically related, yet not driven from one another. This solves key distribution problem of symmetric cryptography; every entity wants to engage in private communication with other entities over public network must first generate a *key-pair*. One called *private*, which should be confidential, and the other key, called *public*, which should be published. If Alice wants to send a private message to Bob, she should use Bob's public key to encrypt this message. Only Bob can read the message by decrypting it using his own private key. The main disadvantage of public key cryptography, though, is that it is much slower to process than private key cryptography. Therefore, in practice, both techniques have been used. Messages are usually encrypted using secret shared keys (symmetric cryptography), then only the secret key is encrypted using an asymmetric cryptographic system. Since the secret key is usually shorter than the actual message, this technique will result in a major reduction in processing time if only asymmetric cryptography is used to encrypt/decrypt needed messages.

2.1 Digital Signature

It is the most significant application of public key cryptography. The message is first *hashed* using a hashing algorithm, which computes a fixed -length string called *message digest*. Two different messages will produc e two different digests. Furthermore, hashing algorithms are one -way. In other words, the actual message cannot be reproduced using its digest. If the message changes, the corresponding digest will also change. This verifies the integrity of the message. *Digital signature* uses both asymmetric cryptography and hashing algorithm. The digest of the message is encrypted with the sender's private key. The recipient decrypts the digest using the sender's public key, computes the digest of the message using the hash algorithm and compares the two results. If they match, this verifies the integrity of the message and authenticates the sender to the recipient. Figure 1 below illustrates the mechanism of generating and verifying digital signature.

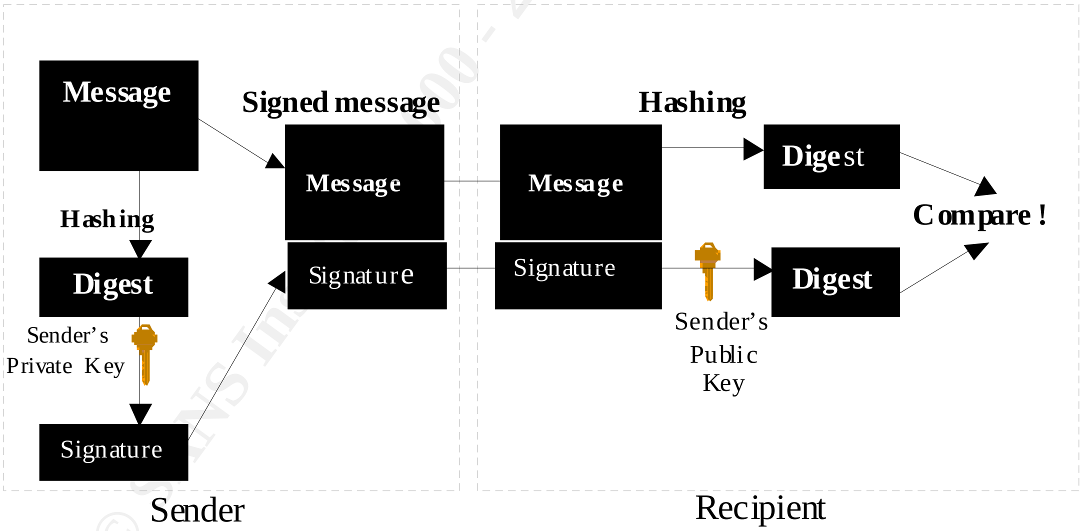

Figure 1: Generation and verification of digital signatures.

2.2 Digital Certificate

When Alice wants to send a message to Bob, she uses the "published" Bob's public key to send him her message. Alice assumes that this public key belongs to Bob and not anyone else. But is it possible that someone else publishes a phony key claiming that he is Bob ?. This is a form of what is known as *man-in-the-middle* attack, in which an attacker will impersonate a legitimate user by po sting a phony key that claims to be for the legitimate user. Encrypted data intended to that legitimate recipient is intercepted and opened by the wrong guy !. In a large public network such as Internet, with hundreds of million users, it is impractical to assume that two users wishing to communicate securely should physically exchange their public keys to overcome a possible man -in-the-middle attack. The solution that has been widely applied is to use a trusted third party that *certifies* the public keys of individuals who need to communicate securely over a public network [3].

A **digital certificate** is a credential that assures the identity of someone, which could be an individual, a server, or an organization [3,5]. This certificate is vouched by the trusted third party. It will be the responsibility of this trusted party to verifying the identity of the certificate holder before vouching it. The mostly used standard format for digital certificates is X.509. Fields of the X.509 certificate include: the public key of the certificate holder, a serial number, identification information about the certificate holder, a validity period, name of the certificate issuer. The certificate is signed by the private key of the issuer.

3. Trust Model in PKI

The main objective of Public Key Infrastructre (PKI) is to provide a framework for the generation, management, and distribution of digital certificates [4,5,6]. PKI combines *digital certificates* and *Certifying Authorities* (CAs),. The role of CA is to be a trusted third party that issues digital certificate. Individuals and organizations apply to CAs for digital certificates. CA will therefore, verify the identity of these individuals or organizations, get their public keys, and issue certificates signed by the CA's private key . A set of pre -defined rules (called Certificate Policy) are normally established to indicate whether a particular person/entity is entitled to get a certificate. When an individual gets a signed certificate from a CA, he/she uses it to communicate over a public network. Other parties may trust the CA that issues and signs this certificate. This trust has different components [7]:

- The CA system (hardware/software) used to issue and maintain certificate is secure.
- The CA's keys are secured and have not been compromised.
- The process of verifying the identity of the certificate applicant is robust.
- The process of maintaining the certificates and making sure they are still valid is also robust.
- And most importantly, those who run the CA are trustworthy .

We may assume that there is a single CA that is trusted worldwide. All certificates should be generated by this central, Certifying Authority that every body trusts. However, this approach has some disadvantageous [1]:

- If this CA is compromised, all certificates worldwide will be nullified.
- This CA exhibits a single point of failure. It could be also a congestion point with the increasing number of certificates.

- The assigned single entity that runs this central CA should be trusted by all organizations and individuals in all countries of the whole world. This might not be achieved in practice.

For these reasons, it is impractical to have a single CA that acts as the only authority worldwide. A single CA is suitable, though for a small establishment. For the cases where a single CA is not practical, multiple CAs are maintained within the PKI structure. These CAs are arranged in a hierarchy, as shown in Figure 2, to have a distributed certificate issuance system [3,4]. A root CA, being the most trustworthy CA in the hierarchy signs other CAs below it in the hierarcy (called subordinate CAs), which can further sign other CAs in the next level, or users. This creates what is called a *Certification Chain*. An individual signed by one of the subordinates CAs must present the certificates of all CA along its certificate chain.

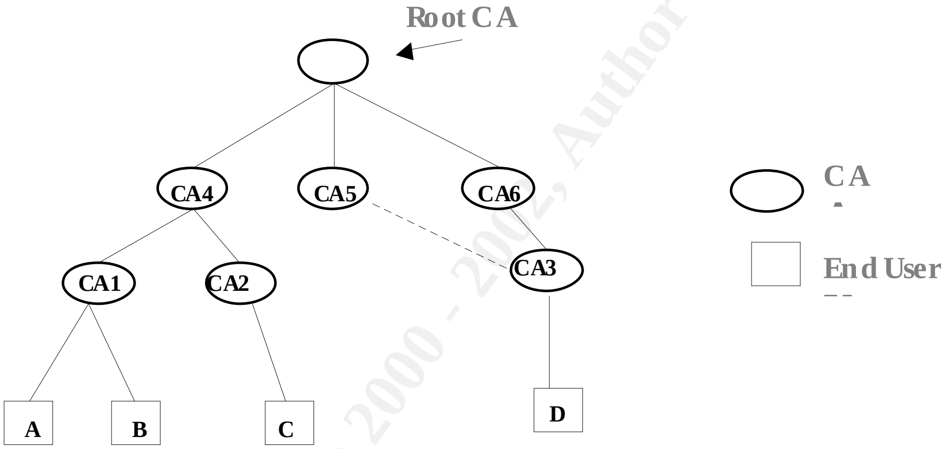

Figure 2: PKI Hierarchy

In order to achieve trust between two parties, each should verify all certificates along the chain of certificates supplied by the other party, until each of them reaches the certificate of a CA that both trust. For example, if user A wants to communicate with user C, both will verify the certificate chain until CA4 which both should trust. However, if user B wants to communicate with user D, the verification will end up in the root CA [8].

Trust can also be achieved if cross certification between two different CAs is established. Cross certification enables two CAs to trust the certificates issues by each other without following the normal path of the hierarchy. This is illustrated in Figure 2 using a dashed line.

In practice, there are several companies worldwide that issue digital certificates through their PKI system, they also help other corporates, organizations, and even countries to build their own PKIs. VeriSign, Entrust, and Baltimore are among the mostly known companies that provide these services. Some countries also implement (or plan to implement) a framework for a country -wide PKI such as the government of Canada [9].

Web browsers such as Netscape Navigator and Microsoft IE include certificates of many CAs that belongs to different companies such as VeriSign, enTrust, Deutsche Telecom. These certificates are considered trusted by the user of the browser to vouch for the site certificates that the user wishes to securely communicate with. For example, if you have a digital certificate from Entrust CA pre -installed on your browser, then the certificate of any web site signed by Enturst CA 's certificate will be accepted by your browser. This default assumed trust by the browser is somehow arguable by many security experts who review these pre -installed CA's certificates and remove those they considered not trustworthy. Furthermore, there is always a possibility that a trusted known CA will wrongfully sign a certificate to a non -trusted entity. A good example of this is what happened early this year when VeriSign issued two digital certificates to a person who fraudulently claimed to be a Microsoft employee [10].

4. Trust Model in PGP

Pretty Good Privacy (PGP) is a free public -key cryptographic system created by Phil Zimmermann in 1991 [3,4,5]. It uses widely recognized and reviewed encryption algorithms (RSA, IDEA) to encrypt/decrypt and/or sign messages. PGP is widely used for exchanging secure e -mail over Internet.

Trust in PGP is achieved using the *web of trust* model. The underlying idea of this model, is that you accept the public key of a PGP user if it has been signed by one or more other trustworthy PGP users. In other words, you are relying on trusted PGP users to introduce others. Each PGP user maintains a list of public keys, called a keyring. Keyrings can be exchanged between users. When a key is inserted, the user assigns the owner of the key to be:

1. **Complete trust**: fully trusted to certify others public keys.
2. **Marginal trust**: marginally trusted to certify others public keys.
3. **Not trusted** to certify others.

If you insert a new key to your keyring, it is considered valid if it has been signed by at least one completely trusted key or two marginally trusted keys.

Figure 3 depicts how PGP trust works in action. The solid lines between each two persons indicate that they met physically, each is considered fully trusted to the other, and they exchange their public keys. Now, since Alice trusts Zaid, she includes the public key of Badr to her keyring because it is signed by Zaid. Bob also decides to get the keyring of Alice which includes both Zaid's and Badr's public keys. The flaw of this trust model appears when Bob decides to communicate with Badr. Bob relys on Zaid to get Badr's key although Bob and Zaid never met ! [4]. For every trustworthy friend of you, you are assuming that he will never certify someone whose is not trustworthy. This is a simple assumption that cannot be fully fulfilled in practice.

Therefore, the trust model of PGP is simple and is not appropriate to use it beyond secure personal communication [11].

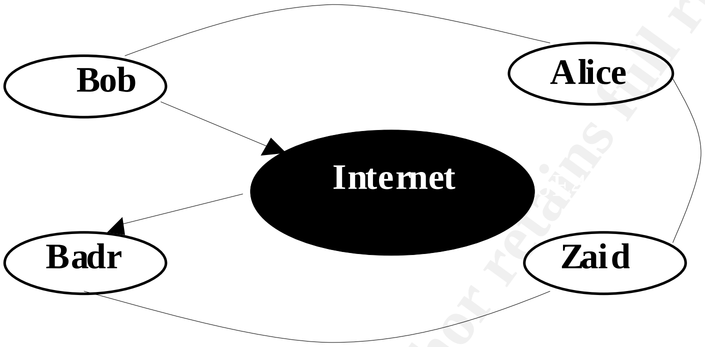

Figure 3: PGP trust in action

5. Conclusion

In this paper, we have presented how trust can be achieved in two different PKI models. These models are X.509 standard PKI and PGP. The PKI trust model is based on Certificate Authorities that generate and manage certificates, while the trust model of PGP depends on the trust level that individuals can put in people whom they know to vouch others certificates. Even if you trust someone, you may not know his/her standard in vouching others certificates. Although PGP trust model is simple, it cannot, however, be used for critical applications where strong authentication is essential. The PKI trust model, on the other hand, is more complex. However, it can provide stronger authentication, and hence it is more suitable for critical applications.

6. References

1. R. Perlman, "An Overview of PKI Trust Models," IEEE Network Magazine, Nov/Dec 1999.

2. Y Yang *et al.*,"Token of Trust: Different Certificates for Different Trust Models". URL: http://www.cs.adfa.oz.au/~yany97/NZ99.html .

3. An Introduction to Cryptography, the International PGP Home Page, URL:http://www.pgpi.org/doc/guide/6.5/en/intro/

4. M. Branchaud, A Survey of Public Key Infrastructures, Master of Science Thesis, Department of Computer Science, McGill University, Canada, March 1997.

5. R. Richardson,"Public Key Infrastructure," June 2000. URL: http://www.PlanetIT.com/docs/PIT20000629S0017 .

6. R. Farrow, "Public Key Infrastucture," Network Magazine, June 1999. URL: http://www.networkmagazine.com/article/NMG20000517S0061 .

7. K. Seifried,"Certificate Authority, Web of Trust, or ..?", SecurityPortal. URL:http://www.securityportal.com/closet/closet20001227.html .

8. Y. Yang *et al.*, "A Trusted W3 Model: Transitivity of Trust in a Hetrogenenous Web Environment." URL: http://ausweb.scu.edu.au/aw99/papers/yang/paper.html

9. Policy for Public Key Infrastructure Management in the Government of Canada. URL: http://www.tbs-sct.gc.ca/pubs_pol/ciopubs/PKI/pki_e.html

10. Microsoft Security Bulletin. URL : http://www.microsoft.com/technet/security/bulletin/MS01-017.asp

11. P. Zimmermann, "Why do you need PGP ?". URL : http://www.pgpi.org/doc/whypgp/en .

12. L. Stein, *Web Security*, Addison Wesley, 1998.