Scientific Report #2
- PKI – State of the Art and Future Directions -

Supervisor,
Prof. dr. Florian Mircea Boian

PhD candidate,
drd. Adam Mihai Gergely
Introduction and research motivation

This scientific research paper presents my progress achieved so far in researching the components needed for my PhD thesis.

The research has been achieved by studying online e-books, documentation, scientific articles from IEEE and ACM and other sources, by adapting the research to my PhD thesis title.

For the Report #2 presentation, this paper deals with the state of the art and future directions in the Public Key Infrastructure (PKI) area of the Information Security domain.

The preliminary title of my PhD thesis is "Protocoale de securitate în aplicații web, bazate pe comunicații scalabile" (in romanian language).

In English translation, the title is: "Security protocols in web applications based on scalable communications".

In order to create a PhD thesis with the above topic, I needed to research the following subjects:

- **Scalable Communications**: Computer Networking with IPv6 Layer #3 Protocol
- **Security Protocols**: Computer Security Protocols and how they work, with the following subcomponents:
  - PKI General Concepts and the X.509 Standard
  - Authentication
  - Symmetric-key Cryptography
  - Asymmetric-key Cryptography
  - Hash functions and checksumming
  - Digital Signatures and Digital Certificates
  - Digital Code Signing
  - Practical implementations of PKI
- **Web Applications**: Testing ground for the security protocols based on scalable communications. With some modifications, the security protocols based on scalable communications could be ported to desktop or mobile applications.

This paper presents my research and findings in the field of PKI. The content of this paper is based on my preliminary research document, which may accompany this paper, if needed.

This paper does not present all the details of PKI and may skip certain secondary or tertiary aspects, as they are not the focus of this paper. This paper only focuses on the basics of the current PKI, new findings or proposed features, and possible solutions to problems using the actual or proposed new concepts or modifications of the current PKI protocol.
1. State of the Art

In our days, the Internet has become an unsafe place for secure and private communications. As new threats appear, solutions against these threats are constantly being developed.

Besides the regular software threats, like viruses, worms, malware or spyware programs, there are also other kind of threats, like the network threats or the trust threats. Examples from these kind of threats include communication eavesdropping, user or service impersonation, secure systems hacking, certificate-bases threats, etc.

The Public Key Infrastructure represents a major area of the Computer Security field of study. It contains concepts, policies, hardware and software specific elements, and procedures for different types of actions.

It deals primarily with encryption, digital signing and digital certificates.

The primary reason for the PKI's existence is to offer secure and trustworthy means by which the communicating entities can exchange information, securely, without having to worry about information stealing, eavesdropping or trust misplacement.

It is primarily used for Internet online e-banking, secure transactions or secure shopping, classified information exchange, etc.

From a trust point of view, PKI offers a considerable degree of trust between the communicating partners by implementing a basic Authority/Server/Client schema.

From a cryptographic point of view, PKI offers a mathematically proven solution to establish a secure channel and encrypt communication based on a key pair (one private key and one public key which are mathematically related) which are issued to the communicating partners by a Certification Authority (CA).

The way the world does business is changing, and corporate security must change accordingly. [1]

For instance, e-mail now carries not only memos and notes, but also contracts and sensitive financial information. The Web is being used not only for publishing corporate brochures but for software distribution and e-commerce. Virtual private networks (VPNs) are extending corporate networks onto the Internet. Extranets turn the Internet into a selectively shared VPN. [1]

Secure e-mail, Web access, e-commerce, VPNs and extranets require strong security, which provides confidentiality, authentication, access control, data integrity, and accountability. Certificates and public key cryptography are emerging as the preferred enablers of strong security. Many large organizations will deploy public key cryptography and certificates throughout the company in the next few years. [1]

Public key cryptography requires a public key infrastructure (PKI), essential services for managing digital certificates and encryption keys for people, programs and systems. [1]

Also, the omnipresence of the Internet and e-commerce technologies present many opportunities, but also pose security and integrity issues. For e-commerce to flourish, businesses, customers, vendors, suppliers, regulatory agencies, and other stakeholders must be assured that trusted business relationships are maintained. [2]
1.1 The history of PKI

In the 1970s, significant discoveries were made in the field of encryption algorithms and key distribution which were kept classified until the 1990s.

In 1976, Diffie, Hellman, Rivest, Shamir and Adleman have publicly revealed both the secure key exchange and the asymmetric algorithms. This provided a new direction in cryptography and information security.

With the constant grow of the Internet and the growing demand for secure communications and peer authentication, and with the help of the already discovered algorithms, the first steps were taken to ensure the birth of PKI.

PKI is based on an advanced mathematic relationship called asymmetric cryptography discovered by two Stanford mathematicians, Whitfield Diffie and Martin Hellman, in 1976 (ref: http://purl.umn.edu/107353). These mathematicians were faced with two major problems in cryptography: key distribution and identification. In response to these issues, Diffie and Hellman identified a relationship between large prime numbers whereby data encrypted with one key could only be decrypted by its paired key. Due to the difficulty involved in establishing the relationship between these large prime numbers, it is unfeasible to reverse-engineer the relationship with a key of any significant size. This mathematic relationship was eventually called the Diffie-Hellman key exchange (ref: http://cr.yp.to/bib/1988/diffie.pdf, patent: http://www.docstoc.com/docs/31206972/Public-Key-Cryptographic-Apparatus-And-Method---Patent-4218582). [3]

Interestingly enough, another set of mathematicians by the names of Ron Rivest, Adi Shamir, and Leonard Adleman identified the same relationship in 1978 and published it as the RSA (based on their initials) algorithm (ref: http://people.csail.mit.edu/rivest/Rsapaper.pdf, patent: http://www.patents.com/us-4405829.html). Though Diffie and Hellman discovered the concept of asymmetric cryptosystems first, RSA monetized the concept more effectively ultimately resulting in RSA Corporation, now owned by EMC. [3]

PKI as we know it began with establishment of the X.509 certificate standard in 1993 with the establishment of RFC 1422 (ref: http://tools.ietf.org/html/rfc1422). This standard created the concepts of certification authorities, certificate revocation lists (CRL), and certificate trusts that provided the framework for more advanced PKI-based technologies in-use today. [3]

In 1991 Phil Zimmerman published PGP (standardized as OpenPGP), an alternative certificate infrastructure without certificate authorities. Instead OpenPGP allows entities and individuals to issue their own certificates. By mutual signing of certificates among different entities, OpenPGP forms a “web of trust” [2, 8] in which people assert the validity of each others certificates. The idea behind this system is that given enough signatures all signatures of a certificate can jointly assert the validity of the certificates, even though some individual signatures might potentially be fraudulent. [4]

Today X.509 is used standards such as S/MIME and SSL/TLS, making it the dominant certification infrastructure for secure HTTP communication and confidential mail communication. [4]
1.2 The current-state PKI

The classic PKI contains elements, procedures, policies and provides documentation for successful implementation of a PKI platform.

The key-concepts of PKI are:

- symmetric cryptography
- asymmetric cryptography
- digital signatures
- digital certificates
- a PKI platform consisting of:
  - a Certification Authority
  - a Validation Authority
  - a Server providing resources
  - a Client accessing resources

An illustration presents the point. If a merchant today has a physical presence at a store, that is, brick and mortar, and customers patronize them for goods and services, the merchant will typically request and receive payment for these directly from either the customers or their agent (e.g., their bank via the presentation of a monetary instrument such as a check), at the time that the goods and services were bargained for and/or provided. The process of exchanging goods and services for value is almost as universal as the rules by which those conversions take place. In many cases those rules are codified, in others they reflect accepted custom. [2]

Whether systematic or custom, the processes in use today provide for the establishment of a trusted business relationship in that the customer and merchant both authenticate one another to the extent that they are willing to undertake the transaction. If an easily recognized monetary instrument like cash is used for transactions, there may be very little authentication which must occur. If a credit card or check is used, then the authentication may include the establishment of the customer's identity to the merchant. In addition, the authentication may also allow for a measure of non-repudiation to be set so that the customer does not deny the transaction occurred. [2]

This traditional face-to-face transaction requires only minimal interaction and normally does not necessitate the use of other security and integrity mechanisms. [2]

However, for e-commerce on the Internet, additional security and integrity mechanisms become necessary. Merchants are typically not willing to ship goods or perform services until a payment has been accepted for them. In addition, authentication can allow for a measure of non-repudiation so the customer cannot deny the transaction occurred. Similarly, consumers need assurance that they are purchasing from a legitimate enterprise, rather than a hacker's site whose sole purpose is to collect credit card numbers. [2]

With the changes in today's business environments and the shift from the traditional face-to-face business models, mechanisms must be developed to ensure that trusted relationships are maintained and can flourish. [2]
The implementation of a PKI is intended to provide mechanisms to ensure trusted relationships are established and maintained. The specific security functions in which a PKI can provide foundation are confidentiality, integrity, non-repudiation, and authentication. [2]

1.3 The Web of Trust and PGP

The Web of Trust is a concept which is used in PGP compatible systems in order to establish the authenticity of a relationship between an owner and its corresponding public key.

The Web of Trust is a decentralized trust model, in opposition to PKI, which is a centralized model which relies on a certification authority.

*The Web of Trust works in this way:*

- each member of a web of trust has a private key (which belongs to the respective member) and a public key which is shared among the communicating partners;
- each communicating partner encrypts the information with the other party's public key, allowing only the holder of the private key (the other party's private key) to decrypt the information;
- In a web of trust, each member has the public keys of other members for encrypting data and has its own private key for decrypting the data addressed to himself/herself.
- After a member encrypts a piece of information with the other user's public key, it digitally signs it with its private key, which enables the other party to verify the authenticity of the information by using the initial user's public key.

*Differences between the Web of Trust and PKI:*

- using PKI, a X.509 certificate can be signed only by the Certification Authority, which, in turn can have its certificate signed by another CA or may posses a certificate chain;
- also, using PKI, there a some certificates which are called Root Certificates. These certificates form the basis of certification and are the “root” tree of other CAs. Each software client (web client, email client, etc.) must have Root Certificates installed, to prevent the user from manually registering a certificate. This ensures that the websites or services using the SSL/TLS protocol have a reliable basis of certification descending from the Root Certificates.

*Common problems with the web of trust:*

- the PGP system is not affected, in general, by the problems that affect a hierarchical system such as PKI. For example, if the communicating parties lose the
track of a private key, they can no longer decrypt messages sent to the owner of the private key and cannot digitally sign information. This raises the problem of creating a procedure for canceling a key, and must be done carefully, to let the other users know that a private key has been compromised and not let a possible attacker compromise secure communication.

- Also, the problem of expiring certificates has been resolved by using the “designated revokers” which appeared in the 1990s;

- Another problem is the very nature of the web of trust. Because it is a decentralized system, it doesn’t have a certification authority and it relies on its users for trust and confidence. For example, new users which automatically come with their certificates could be trusted more reluctantly by other users. In a web of trust, trust is usually earned, not quite given away.

- The concept of “small world phenomenon” resolves the problem of certificate chaining, partially, because of the small number of participants of a web of trust and the trusting architecture based on user relations.

PGP stands for Pretty Good Privacy, which uses the OpenPGP standard and is a program that provides encryption and decryption and authentication of information.

PGP is used for signing documents or other information, encrypting and decrypting all kind of information, like texts, files, directories, disk partitions and is used especially for securing e-mail communication.

The Mean Shortest Distance is used to measure the trust level of a PGP key in a set of other PGP keys in a web of trust and is a metric for analyzing PGP keys.

While working on a keyring analysis, Drew Streib wrote:

“There are a variety of metrics one could apply to this set, but I've chosen initially to measure the "mean shortest distance" (MSD) to each key. Since every key is reachable from every other in the strong set, it is possible to find out the shortest distance (number of hops) to any given key from any other key. Averaging these distances gives the MSD to that key from every other key in the strong set.

It is desirable to have as short as possible an MSD to your key, as that means that on average, people can reach your key quickly through signatures, and thus your key is relatively more trusted than a key with a higher MSD.

NOTE: This does not mean that you should universally trust keys with a low MSD. This is merely a relative measurement for statistical purposes.

The MSD has the property of being no more than 1 higher than your lowest signature. In the worst case, every key in the strong set could reach you by getting to that key, plus 1 hop to get to you. It also encourages the joining of keys that are separated by great distances in the graph, as it will make you a highway of sorts for shortest paths between keys in those groups. In the end, it encourages an overall tightening of the world graph, shortening distances between key owners. “ [5]
1.4 The X.509 Standard

X.509 is a standard developed by ITU-T for the practical implementation of PKI and PMI. It is used to represent the format of:

- public key certificates;
- certificate revocation lists;
- attribute certificates;
- certification path validation algorithm.

The first version of X.509 certificates appeared in 1988 and they were called v1 certificates. The X.509 from that time was tied to the X.500 standard.

X.509 brought a new model of PKI, based on a strict hierarchical system by using certification authorities (CAs) instead of relying on user-level trust like in a Web of Trust or PGP system.

In the CA hierarchical model, only a certification authority, which is trusted by all communicating parties, is allowed to issue certificates to all communicating parties. This model differs from a web of trust, where users share keys amongst themselves.

The second version of X.509 certificates appeared in 1993 and they were called v2 certificates. They were not used, except for experimental purposes.

The third version of X.509 certificates appeared in 1997 and they were called v3 certificates. They are the basis of today's PKIs and represent the "de facto" standard in the digital format of a X.509 v3 certificate. Also, these certificates can be used for peer-to peer or other PGP-like systems.

The X.500 system has been implemented rarely for state identity information, and the IETF's PKI, or the PKIX working group has refined the standard for Internet usage.

In our days, when we refer to digital certificates, actually we refer to IETF's PKIX type of certificate and for the CRL Profile we refer to to the X.509 v3 certificates.

The X.509 PKI basic sequence of certificate usage is as follows:

- A certification authority issues a digital certificate to a requesting entity based on a standard or extended validation of the requesting identity; It does this by signing a Certificate Signing Request (CSR) which contains the public-key of the requesting entity with the private key of the respective CA. Also, it links the public-key to a Distinguished Name (DN), like in X.500 and based on the validation performed earlier;

- The CAs public-key root certificate(s) are installed on many software clients (like web clients, email clients) through a process called CA Audit (in which a CA passes a software security audit to have its keys added to the certificate root store of a software client);
When a software client accesses a software server, that server presents its X.509 digital certificate to the client, and the client can verify the digital signature of the CA (which signed the CSR) by using that CAs public key root certificate (from the software client's certificate store);

If the signature checks out and other parameters are correct (like the DN and certificate validity period), secure communication may begin.

Also, an organization or company may use an internal PKI system. This means that the respective organization must create a "locally trusted" CA system, in which it can issue digital certificates to its software services, signed by its internal CA. For this to have the desired effects, the public-key root certificates of the local CA must be distributed by other means to all members or employees of the respective organization or company.

Also, X.509 includes a standard for revoking certificates by using the Certificate Revocation Lists (CRLs). IETF approves the usage of these kind of system by using the Online Certificate Status Protocol (OCSP). Many modern software clients support the implementation of OCSP.

From a security point of view, there are a few security concerns which were addressed by a number of papers written by security experts.

Certificate complexity:

This problem refers to the fact that the current state of PKI is not user friendly and requires a skilled person to implement and understand the basics behind the usage of PKI. Software vendors have tried to reduce the gap between the users and PKI by implementing “blind” trust, based on a certificate root store which the user automatically trusts, just because the root certificates are in the trusted certificate store. This means that the software vendor is directly responsible for which CA's root certificates it adds to the root store, impacting the end-user's trust, directly.

However, software vendors have included some security warnings and measures to protect the end users if problems with the certificate or certification chain are encountered.

Architectural problems:

- CRLs may become large-sized if a lot of certificates are revoked;
- Lack of history of revocation status in OCSP;
- Aggregation problems
- Federation problems
- Delegation problems

Certification authorities problems:

- In many cases, the certificate subject which request a certificate will look for the cheaper CA that sells certificates. In some cases, these CAs only perform a light check on the entity requesting a certificate. This could lead to lack of proper security. However, this problem is resolved by implementing Extended Validation SSL Certificates;
• Some CAs do not offer warranties or liabilities to the requesting entities if something goes wrong and is the CAs fault.

• Users use an undefined certification request protocol to obtain a certificate which is published in an unclear location in a nonexistent directory with no real means to revoke it [6];

Implementation problems:

Unfortunately, there are design flaws, or other how-tos, or implementation mechanisms which differ from CA to CA.

Some of these problems are mentioned below:

• Revocation check:
  Many CAs do not offer a revocation check system because of the way they operate or do not have a strict security policy.

• Key usage purpose:
  Many software clients do not properly check the key usage of certificates. For example, if a certificate is a Client Certificate, it should be treated as such! Or if it is a Server Certificate, it should be used only by server software, for example. Many software clients take the certificate “as granted” without actually paying interest to the key usage purpose of the certificate.

The X.509 standard has a few PKI standards developed for its usage, like PKCS7, TLS/SSL, PKCS12 or OCSP.

The PKIX (PKI X.509 working group) was developed by the IETF to create RFCs and standards for operating PKI over X.509.

The PKIX Working Group was established in the fall of 1995 with the goal of developing Internet standards to support X.509-based Public Key Infrastructures (PKIs). Initially PKIX pursued this goal by profiling X.509 standards developed by the CCITT (later the ITU-T). Later, PKIX initiated the development of standards that are not profiles of ITU-T work, but rather are independent initiatives designed to address X.509-based PKI needs in the Internet. Over time this latter category of work has become the major focus of PKIX work, i.e., most PKIX-generated RFCs are no longer profiles of ITU-T X.509 documents. [7]

PKIX has produced a number of standards track and informational RFCs. RFC 3280 (Certificate and CRL Profile), and RFC 3281 (Attribute Certificate Profile) are recent examples of standards track RFCs that profile ITU-T documents. RFC 2560 (Online Certificate Status Profile), RFC 3779 (IP Address and AS Number Extensions), and RFC 3161 (Time Stamp Authority) are examples of standards track RFCs that are IETF initiated. RFC 4055 (RSA) and RFC 3874 (SHA2) are examples of informational RFCs that describe how to use public key and hash algorithms in PKIs. [7]

PKIX will continue to track the evolution of ITU-T X.509 documents, and will maintain compatibility between these documents and IETF PKI standards, since the
profiling of X.509 standards for use in the Internet remains an important topic for the working group. [7]

PKIX does not endorse the use of specific cryptographic algorithms with its protocols. However, PKIX does publish standards track RFCs that describe how to identify algorithms and represent associated parameters in these protocols, and how to use these algorithms with these protocols. We anticipate efforts in this arena will continue to be required over time. [7]

PKIX will pursue new work items in the PKI arena if working group members express sufficient interest, and if approved by the cognizant Security Area director. For example, certificate validation under X.509 and PKIX standards calls for a relying party to use a trust anchor as the start of a certificate path. Neither X.509 nor extant PKIX standards define protocols for the management of trust anchors. Existing mechanisms for managing trust anchors, e.g., in browsers, are limited in functionality and non-standard. There is considerable interest in the PKI community to define a standard model for trust anchor management, and standard protocols to allow remote management. Thus a future work item for PKIX is the definition of such protocols and associated data models. [7]

The fourth version of X.509 certificates was released in 2001 and it contains the X.509 Infrastructure with the basis of OASIS SAML attribute assertions. [8]

There were other versions released, but they do not brought new major features, only bug fixes or other small features.

The next version, which is planned to be released in 2016/2017 will introduce a new trust model for open PKIs. [8]

The original X.509 PKI model assumed everyone would have a certificate from a CA, so that certificate subjects were also relying parties (Rps). It can be represented by the Three Cornered Model, in which every RP had a relationship with its trust anchor / root of trust. Also, cross certification ensured trust in other CAs when the RP and the certificate subject had different CAs. [8]

A Certification Authority (CA) is an entity that issues digital certificates and is responsible for managing these certificates in a way that ensures a trust relationship between all the communicating parties.

A CA has the following responsibilities:

- Analyzes the request of an entity who wants a digital certificate issued. If the requesting entity proves its ID (that it is the entity that it claims to be), by whatever means the CA requests from the entity, the CA can go to the next step. If the validation fails, the CA will deny signing a new certificate request for the respective entity.

- If the validation is correct, the CA receives a Certificate Signing Request (CSR) from the requesting entity, which contains the entity’s public key and a Distinguished Named (DN);
• If everything is OK, the CA signs this CSR with its Private Key, the result being a full X.509 digital certificate which can be used by the requesting entity.

• The CA also has to keep track of the certificates that expire or are being revoked and usually it is recommended for the CA to offer guidance or support on how to deploy or sub-manage these certificates.

• In extreme cases, some CAs offer compensation if their private keys are compromised or if anything else happens that may affect the confidentiality or may damage the secure status of a requesting entity’s cryptographic elements.

1.5 The Privilege Management Infrastructure (PMI)

The Privilege Management manages user authorizations by using the ITU-T X.509 attribute certificates (ACs). This was introduced in the 2001 version of X.509. In 2005, a delegation service was added to improve the PMI. In 2009, an interdomain authorization was also added to enhance the current version of PMI.

The PMI uses the attribute certificates to store the user privileges in the same manner as PKI uses the public key certificates to store public keys.

Also, PMIs have Sources of Authority (SoAs) and Attribute Authorities (AAs). They do not use Certification Authorities like PKI does.

However, there is a relationship between PKI and PMI, because the PMIs are based on PKIs, for example, when attribute certificates need to be digitally signed by the AA that issued them. At this point, when the signature of the AA needs to be verified, PKI is used.

A PMI consists of 5 types of components: [9]

• Attribute Authorities (Aas)
  - to issue and revoke ACs (also called Attribute Certificate Issuer)

• Attribute Certificate Users
  - to parse or process an AC

• Attribute Certificate Verifier
  - to check the validity of an AC and then make use of the result

• Clients
  - to request an action for which authorization checks are to be made

• Repositories
  - to store and make available certificates and Certificate Revocation Lists (CRLs)

[9]

The following diagram describes the elements of PMI:
There are two types of attribute certificate distribution as shown in the diagram, **push** and **pull**. [9]

In some environments it is suitable for a client to push an AC to a server. This means that no new connections between the client and server are required. It also means that no search burden is imposed on servers, which improves performance. [9]

In other cases, it is more suitable for a client simply to authenticate to the server and for the server to request or pull the client’s AC from an AC issuer or a repository. A major benefit of the pull model is that it can be implemented without changes to the client or to the client–server protocol. It is also more suitable for some inter-domain cases where the client’s rights should be assigned within the server’s domain, rather than within the client’s domain. [9]

An attribute certificate comprises a digitally signed SEQUENCE of: [10]

- the version number of this AC (v1 or v2)
- identification of the holder of this AC
- identification of the AA issuing this AC
- the identifier of the algorithm used to sign this AC
- the unique serial number of this AC
- the validity period of this AC
- the sequence of attributes being bound to the holder
- any optional extensions

[10]
Comparison table of PKI and PMI: [10]

<table>
<thead>
<tr>
<th>Concept</th>
<th>PKI entity</th>
<th>PMI entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate</td>
<td>Public Key Certificate</td>
<td>Attribute Certificate</td>
</tr>
<tr>
<td>Certificate issuer</td>
<td>Certification Authority</td>
<td>Attribute Authority</td>
</tr>
<tr>
<td>Certificate user</td>
<td>Subject</td>
<td>Holder</td>
</tr>
<tr>
<td>Certificate binding</td>
<td>Subject's Name to Public Key</td>
<td>Holder's Name to Privilege Attribute(s)</td>
</tr>
<tr>
<td>Revocation</td>
<td>Certificate Revocation List</td>
<td>Attribute Certificate Revocation List</td>
</tr>
<tr>
<td>Root of trust</td>
<td>Root CA or Trust Anchor</td>
<td>Source of Authority</td>
</tr>
<tr>
<td>Subordinate authority</td>
<td>Subordinate CA</td>
<td>Attribute Authority</td>
</tr>
</tbody>
</table>

X.509 AC Extensions:

Extensions to X.509 ACs are concerned with specifying some aspects of policy that govern the use and applicability of the ACs. The extensions can be split up into 5 categories: [10]

- basic extensions
- privilege revocation extensions
- an extension to support roles
- source of authority extensions, and
- delegation of authority extensions [10]

Although attribute certificates (ACs) were first defined in X.509(97), it was not until the 4th edition of X.509 (ISO 9594-8:2001) [1] that a full infrastructure for the use of attribute certificates was defined. This infrastructure is termed a Privilege Management Infrastructure (PMI), and it enables privileges to be allocated, delegated, revoked and withdrawn in an electronic way. A PMI is to authorisation what a Public Key Infrastructure (PKI) is to authentication. [10]

In order to understand the purpose of an X.509 PMI, it may help if we first consider how privileges are allocated and used today in a paper based privilege management system. A resource owner (e.g. the Financial Director of a company, or a General Manager), called the Source of Authority (SOA) in X.509, will sign a paper form to say that a particular person (the privilege holder) is to be allowed to use a particular resource in a particular way. For example, the Financial Director may say that a Head of Department can sign orders up to the value of ten thousand Euros, or the General Manager may sign a form authorising a user to have access to a particular restricted area of the organisation. [10]

Paper based systems may also support delegation of authority, whereby a privilege holder is allowed to delegate the use of resources to which he has been granted access, to one or more other people. In X.509, such a privilege holder is now termed an Attribute Authority (AA), since the holder has been authorised to assign privilege attributes to other people. [10]
1.6 Authentication [11]

Authentication represents the process of verifying by different means if the person or device that wants to access the resource is indeed authorized to access it or not. This is done by checking for the supplied credentials in a database, for verification purpose. If there is a result in the database that matches the claims of the person or device in matters of identity, than it is said that it has successfully authenticated.

Usually authentication is performed as a standard check before granting or denying access or before encryption begins, in order to be sure that the person or devices is truly who it says it is before engaging in other sensitive activities.

Authentication can be performed in multiple ways, but the most widely used methods are:

- Password/Passphrase authentication – authentication with what you know;

**Password/Passphrase authentication:**
This kind of authentication is the most basic and simple and the concept behind this has worked for hundreds of years.

The basic idea behind this is one secret code/password/passphrase that must be known by the authenticating entity in order to permit a pass. If the entity knows the code, it means that it can be trusted and access is granted. Plain and simple.

This authentication method is also know as “Authentication with what you know”.

The main advantage of this method is that you don't have to carry certificates or other things with you, and you don't have to worry about storing something somewhere. The password exists in your brain, that's all. If you know the password, you can simply type it whenever it is required.

The disadvantage of this method is that you can forget the password, in which case you must reset it and choose another one.

**Digital Certificate authentication:**
This kind of authentication is more advanced than the password authentication, because you must have an authentication token (like a key) to prove that you are who you say you are.

In this scenario, you must present your digital certificate (usually client certificate). A verification process begins to check the certificate. If the certificate contains the correct authentication data than you are granted access.

This authentication method is also known as “Authentication with what you posses”.

The main advantage of this method is that you don't have to remember anything, and only those entities who have a valid certificate can log in. Also, this method is immune to the "forgot the password" case.
The disadvantage of this method is that you always have to carry the certificate on a portable drive or security token, or you must store it somewhere, and that place must be properly secured from unauthorized access.

In a simple scenario where authentication takes place in order to access a secured resource, once the authentication is successful, the system offering the secured resource grants access to the authenticated entity after which the sequence may end.

1.7 Symmetric-key cryptography [11]

Next, this paper will focus on a more advanced scenario in which the authentication phase is required in order to encrypt data that must be transmitted on the network.

Symmetric encryption represents the process of deliberately altering (in a structured known way) the contents of an information in order to hide the real contents of that information. It is done controlled, in a special order, which allows later that information to be recovered, process known as decryption, by using the same key.

For this to work, it is necessary to run the same encryption/decryption algorithm with the same key and cipher on the plaintext, respectively on the ciphertext.

A typical example would be:

- Plain text: “mother”
- Encryption algorithm/cipher/key: “move 1 character up the alphabet”
- Resulting text would be: “npuifs”.

If somebody doesn't know the Encryption Algorithm/Cipher/Key and gets the text “npuifs”, that person will not have a clue about what that word means.

But if we specify the encryption method, that person could decrypt the ciphertext by applying the inverted algorithm: “move 1 character down the alphabet”. By applying this algorithm to the “npuifs” word, the result (decrypted cipher text) would revert back to the original plaintext: “mother”.

In almost all the cases this method is used when the communicating partners already posses the same shared secret key with which one encypts information and the other decrypts the information. This method assumes that the key was distributed among the communicating partners in a securely manner before the communication can actually take place.

Some examples of symmetric encryption are: the HTTPS and SSH protocols (after the PKI handshake), a Wireless WPA Encryption, a manual encryption of a resource.
1.8 Asymmetric-key cryptography [11]

The main difference between symmetric and asymmetric cryptography is that the asymmetric cryptography used two keys to encrypt and decrypt messages. It was implemented to overcome the problem of distributing the cipher to new communicating partners.

In symmetric cryptography, both of the communication partners had to possess the key (the method of obtaining the key was resolved by other means) in order to successfully encrypt or decrypt a message.

Asymmetric cryptography resolves this problem by implementing a pair of mathematically related keys: a private key and a public key:

- The Private Key belongs only to the owner and is used to decrypt messages;
- The Public Key can be distributed to anyone who desires to securely communicate with the owner of the Private Key. Anyone possessing the public key can encrypt messages, but only the holder of the Private Key can decrypt those messages.

An analogy would be two people who reside in the same town and want to communicate securely. They can meet face to face, one gives the other a key, and afterwards they can communicate securely over the Internet using symmetric cryptography. But what happens if they are 1000 miles apart? Well, it would be very hard for them to meet face to face to exchange the key, therefore in this case, asymmetric cryptography is used. One gives the other his or her public key, and the other one can encrypt information which only the first can decrypt using it's private key. Or, the one with the public key encrypts a common shared secret key and sends it to the owner of the private key, thus securely informing the other partner of a secret key, agreeing to use that shared secret key for symmetric encryption.

Mathematically, the Public Key can always be derived from the Private Key, but almost impossible the other way around.
The asymmetric cryptography is also known as Public-Key Infrastructure (P.K.I.). Almost every modern secure communication involves PKI. For example, the HTTPS and SSH protocols use PKI for the handshaking phase of the secure communication.

As a very brief truncated example: a web client accesses a HTTPS-enabled website. That website offers its server certificate to the client. The client uses the public-key from that Server Certificate to encrypt a random secret code that only the Server (which holds the private key) can decrypt. From that point on, based on that key and other things, the communicating parties compute a shared secret key with which they will symmetrically encrypt the communication.

In almost all the situations, the PKI is used to resolve the key distribution problem by allowing the establishment of a common secret symmetric key. Only after that phase, the effective encryption takes place by using that symmetric key.

A PKI (public key infrastructure) enables users of a basically unsecure public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority. The public key infrastructure provides for a digital certificate that can identify an individual or an organization and directory services that can store and, when necessary, revoke the certificates. Although the components of a PKI are generally understood, a number of different vendor approaches and services are emerging. Meanwhile, an Internet standard for PKI is being worked on. [12]

The public key infrastructure assumes the use of public key cryptography, which is the most common method on the Internet for authenticating a message sender or encrypting a message. Traditional cryptography has usually involved the creation and sharing of a secret key for the encryption and decryption of messages. This secret or private key system has the significant flaw that if the key is discovered or intercepted by someone else, messages can easily be decrypted. For this reason, public key cryptography and the public key infrastructure is the preferred approach on the Internet. (The private key system is sometimes known as symmetric cryptography and the public key system as asymmetric cryptography.) [12]
A public key infrastructure consists of:

- A certificate authority (CA) that issues and verifies digital certificate. A certificate includes the public key or information about the public key;
- A registration authority (RA) that acts as the verifier for the certificate authority before a digital certificate is issued to a requestor;
- One or more directories where the certificates (with their public keys) are held;
- A certificate management system. [12]

A few examples of applied PKI - Pretty Good Privacy:
For e-mail, the Pretty Good Privacy (PGP) product lets you encrypt a message to anyone who has a public key. You encrypt it with their public key and they then decrypt it with their private key. PGP users share a directory of public keys that is called a key ring. (If you are sending a message to someone that doesn't have access to the key ring, you can't send them an encrypted message.) As another option, PGP lets you "sign" your note with a digital signature using your private key. The recipient can then get your public key (if they get access to the key ring) and decrypt your signature to see whether it was really you who sent the message. [12]

1.9 Hash functions and checksumming [11]

Integrity Check represents the process of checking the information for evidence of tampering or accidental altering. It is done with the help of hash functions, which take the length of the information and “encrypts” (it is not a real encryption) it obtaining a fix length signature of that information. By comparison with different checks, if that signature remains exactly the same, it means that the information has not been tampered with or altered.

The cryptographic hash function is a deterministic procedure which takes a random number of data and returns a fixed-size string of data, which is known as the hash value, or checksum.

An ideal cryptographic hash function has four main properties:

- It's easy to compute the checksum for any given message;
- It's impossible to extract the original message from the checksum;
- After modifying the initial message, and reapplying the checksum, it's impossible to obtain the same checksum that the initial message had before the modification;
- It's impossible to find two different messages that, produce the same hash value;

<table>
<thead>
<tr>
<th>Brown fox</th>
<th>eb7a3af5da7c7bfb3c7e6803fde2b84a3486a030</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown fox</td>
<td>c76d75b4865aa633f10d2e974d6be3cfb6984d4e</td>
</tr>
<tr>
<td>Brown fox</td>
<td>cc83dc4b9931668caeb1152d325ea6ebe6d325b1</td>
</tr>
</tbody>
</table>

Fig. 4 – An example of a SHA1 hash function used on a string and its variations produce completely different checksums when something is altered in the original message.
Checksumming is useful for fixing a “stamp” on a piece of information, which acts as a verification token against tampering or accidental bit loss.

The TCP protocol at the Networking Transport Layer performs a small checksumming of the data segments.

Another example is to verify if a downloaded file is authentic. Usually when you download a larger or important file, the download website offers you the file’s MD5 or SHA1 checksum. After you download the file, you perform your own checksumming on that file. If your checksum matches the one on the website, it means that your file is authentic and did not lose a single bit in transit.

A recommendation for web-developers and database administrators is to never store plain passwords in the database. This is recommended for ethical reasons and for the security of that password. Administrators should try storing the checksum of the password and compare that checksum against the checksum of the entered password upon authentication. If the checksums match, it means that the original password had to be the correct one to checksum to the already stored-in correct checksum.

1.10 Digital signatures [11]

The main two practical applications of Authentication, PKI and checksumming are:

- Digital Signatures
- Digital Certificates

Digital Signatures are small pieces of data that contain authentication information which can be verified using a Public Key. Digital Signatures are used to prove that a message or content belongs to a person or an entity. A Digital Signature, if implemented properly is much more secure than a standard hand-written signature and harder to falsify. Of course, the Private Key used to generate the signature must not be compromised.

Digital Signatures also offer the so-called “non-repudiation” feature, which means that the holder of the Private Key used to generate that signature – practically the signature’s owner – cannot later deny that the signature does not belong to the owner of the signature, the owner being responsible for what is signing and the signing process itself. In the case of a fraud or other investigations, the owner cannot elude responsibility if the signature is verified and is confirmed to belong to the owner.

Digital Signatures are used mainly to sign e-mail, contracts, certificates or documents, or other messages that require a proof of identity. A Digital Signature contains usually three algorithms:

- A key-generating algorithm which generates a key pair;
- A signing algorithm, which produces a signature by processing a message with a private key;
- A verification algorithm, that can use a public-key, and the original message to check if the signature is confirmed or not;
Note that the signing is applied on the Hash Value of the message, not on the entire message. This is done for several reasons:

- **Efficiency**: The signature will be much shorter and it saves time;
- **Compatibility**: Messages are usually strings of bytes, a hash function can help convert the message into a stable input (BIN vs ASCII for example).
- **Integrity**: Assuming that we don't use hash functions, the signing would have to be executed on the entire document. Well, what happens if we must break the document into smaller pieces for transmission over a low bandwidth channel?

### 1.11 Digital certificates [11]

A digital certificate is an electronic document which has the role to prove the identity and authorization of an entity, person or electronic equipment, in order to execute one or more secure operations.

These documents are known also as Public-key certificates, or standard X509 type documents (which denotes the family of certificates). A digital certificate associates a digital signature to a public key with an identity. The certificate is used to verify if a public key belongs to an entity.

In a general PKI platform, the digital signature from within the digital certificate belongs to the Certification Authority (CA). In a WebTrust or a private trust area, the digital signature can be auto-generated. Regardless of the nature of the digital signature used, the signature proves that the signer trusts the fact that the public key belongs to the certificate owner (information extracted from within the certificate).
The general structure of a digital certificate is presented below:

- **Serial Number**: used to identify the number of the certificate;
- **Subject**: the identification data of the certificate owner;
- **Certificate algorithm**: the algorithm used for creating the certificate;
- **Issuer**: the entity that issued the certificate and signed for the accuracy of the data contained within the certificate;
- **Valid from**: self-explanatory, the date from which the certificate is valid;
- **Valid to**: self-explanatory, the date until the certificate is valid;
- **Public key**: the public key of the entity;
- **Checksum algorithm**: the algorithm used to perform a hash on the certificate;
- **Checksum**: the effective checksum of the certificate;

The most common form of these certificates is a DER encoded ASCII text file, which contains the certificate, encoded in Base64 form (usually .pem or .crt file extensions).

A client certificate is an X509 certificate used to authenticate clients to secure resources. They are also known as Personal Certificates. The client who wants to have a client certificate is recommended to own a special cryptographic device known as a Security Token. A security token is a special USB flash device, or a Smart Card which is specifically built to house encrypted information and cryptographic objects. It is not a flash general-purpose mass-storage drive! It's a special dedicated cryptographic device and it can be used with almost any modern operating system with special drivers.
1.12 Digital code signing

The digital code signing operation represents the digital signing of a piece of software (or information) by means of a digital signature, or digital certificate, to confirm that the software belongs to a particular author or developer or that the code is genuine or has not been altered.

It is generally being used for signing executable files or programs, scripts, object files, etc. to ensure that the software is genuine or “clean” of other “intrusions” or modifications.

The main features of digital code signing are:

• Verification of the author/developer (by means of a digital signature);
• Verification of the integrity of the software (by means of a checksum).

The mechanisms used are similar to that of PKI, which means that a private and a public key are used.

Usage of code signing:

• Authenticating .NET applications, ActiveX controls or Java applets or other scripts;
• In many modern operating systems, for authenticating system updates;
• Authentication on first run of an application, but rarely used with open-source products, because of direct source access possibility.

There are two main ways to authenticate trust information:

• by using a Certification Authority:
  This is the traditional method for verifying the authenticity of the code. It ensures that the source is trusted, but not the code itself. This follows the traditional way of PKI in that if a user trusts a CA, it will presumably trust the code that is signed by that respective CA.

• without the use of a Certification Authority:
  This a more ad-hoc way of verifying trust, by means of direct interaction with the developer. The software developer generates its self-signed key and distributes this key by means of another trustworthy channel. If the user trusts the developer, then presumably it will also trust the code that is signed by that respective developer.

Also, the time-stamping service can bypass the warning of trust for an expired certificate. This is done to “extend” the lifetime of a certificate beyond the point of validity.

A few examples of implementation:

• GNU/Linux Operating Systems' Update Managers
• Microsoft WHQL program for drivers
• IBM Lotus Notes
When customers buy software from a store, the source of that software is evident. Customers know who published the software, and also see whether the package has been unsealed. These factors enable them to make decisions about what software to buy and how much to rely on those products. [13]

Customers downloading applications signed with a Code Signing Certificate from your website can be confident that code comes from you and not been altered since its creation and signing. A Digital ID acts as virtual "shrink-wrap" for your software. If your code is tampered in any way, after it is signed, the digital signature will break and cautions customers that the code is not credible. [13]

Features and Benefits of Code Signing: [13]

- **Customer Confidence**
  They assure customers that you are trustworthy enough to do business with. [13]

- **Authenticity**
  After downloading, end users can be sure that the code they obtained really came from you, helping you preserve your business reputation and intellectual property. Digital ID's allow customers to identify the author of digitally signed code and contact them should an issue or query arise. [13]

- **Seamless Integration with Industry-Standard Technology**
  Most browsers will not accept action commands from downloaded code unless the code is signed by a certificate from a trusted Certificate Authority, such as Comodo. [13]

- **Ease of Use**
  Code signing certificates are easy to use in conjunction with the vendor software tools that developers use to create products, macros and objects. [13]

Most mass-market computing devices sold today come with pre-loaded software, but the software that comes with the device “out of the box” is never all that will be needed for the full life of the device. Whether for a personal computer or a mobile device, users will frequently run into situations where they need to download software or applications. In other cases, a user might not be planning to download software. Users are advised by an application on their device, or the site they are visiting, that in order to experience or use the offered they need to upgrade, patch or augment their current software. They are asked to make a spot decision: “Run” or “Don’t Run.” [14]

In these situations, “run/don’t run” asks the user whether or not to run the downloaded code. How does a user decide? How does a user or user agent (usually a "browser") know whether or not to trust the software? The answer is code signing. [14]

To assist in the trust-decision process, the software publisher can digitally sign their code. The digital signature answers the questions of authentication and integrity, that is:

- Who signed the code?
- Has the code been tampered with since it was signed?

Armed with this information, the user can now make the “run/don’t run” decision. [14]
Even though the digital signature does not answer whether you can “trust” the software not to harm your computer, unsigned code does not provide any evidence of origin or file integrity. The publisher is not identified and, therefore, cannot be held accountable. In addition, the code is subject to tampering. Digitally signed code, which is backed by a certificate issued by a CA acting as a trusted third party, is granted greater reliability than unsigned code, which should generally not be trusted. [14]

In order to sign the code, the publisher needs to generate a private-public key pair and submit the public key to a CA along with a request to issue a code signing certificate. The CA verifies the identity of the publisher and authenticates the publisher’s digitally signed certificate request. If this vetting and key-verification process is successful, the CA bundles the identity of the publisher with the public key and signs the bundle, creating the code signing certificate. [14]

Armed with the code signing certificate, the publisher is ready to sign the code. When the code is signed, several pieces of information are added to the original file holding the executable code. This bundled information is used by the recipient’s user agent to authenticate the publisher and check for code-tampering. The entire sequence for bundling the digitally signed code takes place as follows: [14]

- A hash of the code is produced
  - Public-key algorithms are inefficient for signing large objects, so the code is passed through a hashing algorithm, creating a fixed-length digest of the file
  - The hash is a cryptographically unique representation of the file
  - The hash is only reproducible using the unaltered file and the hashing algorithm that was used to create the hash
- The hash is signed using the publisher’s private key
  - The hash is passed through a signing algorithm using the publisher’s private key as an input
  - Information about the publisher and the CA is drawn from the code signing certificate and incorporated into the signature
- The original code, signature and code signing certificate are bundled together
  - The code signing certificate key is added to the bundle (as the public key is required to authenticate the code when it is verified). [14]

![Fig. 7 – Code Signing process [14]](image)
The signature is verified as follows:

- The original code is passed through the hashing algorithm to create a hash.
- The public key of the publisher is extracted from the bundle and applied to the signature information; applying the public key reveals the hash that was calculated when the file was signed.
- The two hashes are compared; if equal, then the code has not changed and the signature is considered valid.
- The code signing certificate is checked to make sure that it was signed by a trusted CA.
- The expiry date of the code signing certificate is checked.
- The code signing certificate is checked against the revocation lists to be sure that it is valid.
- If the file is considered valid, it is accepted by the user agent; if the file is not considered valid, the user agent displays a trust dialogue like the one above.  

![Signature verification process diagram](image)

**Fig. 8 – Signature verification process** [14]

<table>
<thead>
<tr>
<th><strong>Self-Signed Certificate</strong></th>
<th><strong>Certificates Issued from Publicly Trusted CA</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer provides their own identity, which is not published in the trust dialogue</td>
<td>CA performs identity verification, which is displayed in the trust dialogue</td>
</tr>
<tr>
<td>Issuer provides their own policy and quality</td>
<td>CA issues certificates in accordance with the industry policy and quality</td>
</tr>
<tr>
<td>Signatures will provide a trust warning indicating that there was an un-verified publisher and will display “Unknown Publisher”</td>
<td>Signatures will provide positive trust dialogue</td>
</tr>
<tr>
<td>Compromised certificates cannot be revoked and could harm the users of your software</td>
<td>Compromised certificates can be revoked and if time-stamping was used, code signed before revocation will remain trusted</td>
</tr>
</tbody>
</table>
1.13 Certification Authorities [15]

A Digital Certification Authority is an entity that issues digital certificates by signing the certificate requests requested by its clients.

In the standard PKI scheme, a CA represents a third party which is trusted by all the communicating parties and it certifies that at least one of the communicating party is genuine.

Usually, Certification Authorities are commercial in nature, which means they issue digital certificates for a fee, but there are other CAs that issue digital certificates for free. However, they're root certificates are, with a few exceptions, not recognized by the software clients. Also, organizations or companies may choose to implement a local CA for their needs, which issues digital certificates only for the respective entity.

Technically, the CA issues a digital certificate that contains the subject's public key and identification information. The private key used to obtain the respective public key is kept private in the possession of the subject. The CA does not need the subject's private key. The CA also binds the public key with the information regarding the subject's identity.

If a CA gets compromised, it means that all the security services of that respective CA is compromised, for every user or client of that particular CA.

For example:
An attacker named Eve convinces a CA to issue her a digital certificate which represents another person named Alice, with the scope of representing Eve as being Alice. Also, let's assume that Eve possesses Alice's private key. If Eve were to send an email signed with Alice's certificate, Bob would think that Alice sent that email. He would reply to that email by encrypting the information with Alice's public key from the certificate, Bob thinking that Alice will read the email, but he would never know that, in fact, Eve is behind this and she can decrypt the information by using her private key.

Another example, taken from real life, happened in 2001 when a third party convinced the VeriSign Certification Authority to issue 2 digital certificates for Microsoft, that third party claiming to represent Microsoft. The certificates were issued to "Microsoft Corporation" and could have been used to fool the clients into thinking that the rouge party was, in fact, tied to Microsoft. The fraud was quickly discovered and Microsoft and Verisign took action to limit the impact of this fraud.

From a security point of view, it is kind of difficult for a CA to verify the identification data submitted by a certificate requester. For this reason, a CA uses a combination of different techniques of authentication by using physical addresses, office addresses, payment infrastructures, databases or other proofs.

In larger PKI architectures, very often happens that a communicating party does not know or trust the other communicating party’s CA. That’s why the first communicating party must examine the Chain of Trust, to verify the direct trust relationship (starting with the Root CA all the way down to the last intermediate CA) or to analyze the cross-certification details.
The primary role of a Certification Authority is to issue digital certificates and to act as a responsible entity for digitally signing a trust relationship.

Also, a Certification Authority has other secondary duties, like rejecting certificates (for any number of reasons), revoking certificates, keeping a list of public certificates, managing a database of clients and issued certificates, verification of requesting entities, etc.

**Issuing certificates:**

This is the primary role of a CA. The procedure is described below:

- **The requesting entity's job:**
  
  Step 1: The generation of a private key;
  
  Step 2: The generation of a certificate signing request (CSR) which contains the public-key and some other identification information;
  
  Step 3: The submission of the CSR to the CA;

- **The Certification Authority's job:**
  
  Step 4: The CA checks the information and the authenticity of the requesting entity, and if the CA is a commercial one, the payment is requested;
  
  Step 5: If the information of the requester is not correct or there are other issues to be resolved, the certificate signing request is rejected and the submitter is notified;
  
  Step 6: If the information is correct, and the checks are all OK, the CSR gets signed by the CA's private key;
  
  Step 7: The signing of the CSR produces a final certificate (CRT) which is given to the requester, which can use the certificate and the private key to start communicating securely.

---

**Fig. 9 – The issuing of digital certificates by a CA [15]**
Revoking certificates:

There are cases when the requesting entity, now the certificate owner, stops its activity or does not need the certificate anymore. In this case, it must contact the Certification Authority and request its certificate to be revoked (or cancelled). The CA will revoke its certificate and will publish this certificate in its Certificate Revocation List (CRL). From that point on, any software client that has CRL checks implemented can be notified that the entity's certificate is no longer valid.

Also, a CA may revoke a certificate forcibly, in the case of fraud, for example. Examples of fraud include: false registration information, the usage of certificate for other purposes than the purposes for which it was created, information attacks or spamming which are linked to the certificate in question, etc.

Certificate expiration:

As mentioned earlier, a standard X.509 digital certificate has a validity period, denoted by the "Valid from:" and "Valid until:" fields. A digital certificate can be used correctly only in its validity period. If a certificate expires, it must be renewed. A certificate renewal must be made to the CA by the requesting entity. If the CA is a commercial one, it will first request the payment for renewing that respective certificate. After that, the CA takes the original CSR from its databases, and resigns it with its private key, resulting a new certificate with a new validity period, after which the certificate is given to the requesting entity.

Rejecting certificates:

The Certification Authority reserves the right to reject a certificate request before signing if it finds problems with the registration process, like incomplete or false information submitted by the requesting entity. The CA rejects the certificate signing request and informs the requesting entity about the reason for rejection. Depending on the problem severity, the requesting entity may, or may not apply for another certificate. If it can apply for another certificate, it must first correct the problems that prevented it from obtaining a certificate in the first attempt.

Browsers and devices trust a CA by accepting the Root Certificate into its root store – essentially a database of approved CAs that come pre-installed with the browser or device. Windows operates a root store, as does Apple, Mozilla (for its Firefox browser) and typically each mobile carrier also operates its own root store. [16]

CAs use these pre-installed Root Certificates to issue Intermediate Root Certificates and end entity Digital Certificates. The CA receives certificate requests, validates the applications, issues the certificates, and publishes the ongoing validity status of issued certificates so anyone relying on the certificate has a good idea that the certificate is still valid. [16]

CAs usually create a number of Intermediate CA Root Certificates to be used to issue end entity certificates, such as SSL Certificates. This is called a trust hierarchy. [16]
1.14 Practical implementations of PKI

The PKI system has gained very much popularity in the last 10 years, because of its easiness to use and straight-forward design.

Today, we use PKI in many areas of informatics, like:

- Basic certificate management system for Certification Authorities;
- Providing infrastructure support for digital certificates for web security (HTTPS);
- Providing infrastructure support for digital certificates for email security (IMAPS/POP3S/SMTPS);
- Providing infrastructure support for digital certificates for VPN implementation and security (OpenVPN), SSH Keys security, etc.
- Providing infrastructure support for digital certificates for general servers;
- Providing infrastructure support for digital certificates for client authentication;
- Providing infrastructure support for digital certificates for code signing;

A PKI does not serve a particular business function; rather, a PKI provides a foundation for other security services. The primary function of a PKI is to allow the distribution and use of public keys and certificates with security and integrity. A PKI is a foundation on which other applications and network security components are built. Systems that often require PKI-based security mechanisms include email, various chip card applications, value exchange with e-commerce (e.g., debit and credit cards), home banking, and electronic postal systems. [2]

A PKI has many uses and applications. A PKI enables the basic security services for such varied systems as:
- SSL, IPsec and HTTPS for communication and transactional security
- S/MIME and PGP for email security
- SET for value exchange
- Identrus for B2B

Some key benefits that PKI and its use of public key cryptography offers for e-commerce and other organizations are as follows:
- Reduces transactional processing expenses
- Reduces and compartmentalizes risk
- Enhances efficiency and performance of systems and networks
- Reduces the complexity of security systems with binary symmetrical methods. [2]

In addition, many other similar solutions rely on the fundamentals of PKI such as:
- Student IDs on college campuses
- Voting
- Anonymous value exchange
- Transit ticketing
- Identification (passports and drivers licenses)
- Notarization (contract, emails, etc.)
- Software distribution
- Symmetric key management

[2]
1.15 A Guide on how to create a small PKI using GNU/Linux & OpenSSL

This is a personal guide, created by me, for deploying a simple PKI system. This guide contains instructions on how to create the elements needed to form a small PKI system. The instructions in this guide can be run on any GNU/Linux/UNIX Systems that has the cryptographic library OpenSSL installed.

Depending on the operating system, a user attempting to put in practice these instructions, must install all the required components or libraries (OpenSSL or other dependencies). Refer to the operating system's documentation on how to installed these required packages.

Because the X.509 certificate extensions to work properly, a file named “myssl.cnf” or any other name must be created, which must contain the following directives:

```
[serverauth]
basicConstraints=CA:FALSE
nsCertType = server
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
nsComment = "SSL Server Certificate"
```

```
[clientauth]
basicConstraints=CA:FALSE
nsCertType = client
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
nsComment = "SSL Client Certificate"
```

The following instructions assume the existence of the “myssl.cnf” file, with the above contents, residing in the “/etc/ssl/” directory.

The steps required to create a small PKI system using:

- a self-signed Certification Authority – [CA]
- a subCA acting as a smaller Certification Authority, which is subordinate to the main Certification Authority – [subCA]
- a SSL Server Certificate – [Server]
- a SSL Client Certificate – [Client]

**Generating CA parameters:**

- Step A1: Create a Root CA Private Key:
  ```
  openssl genrsa -out RootCA.key 4096
  ```

- Step A2: Create a Self-Signed Root Certificate for our CA:
  ```
  openssl req -new -x509 -key RootCA.key -days 7305 -out RootCA.crt
  ```
Generating the subCA parameters:

• Step B1: Create a subRoot CA Private Key:
  \texttt{openssl genrsa -out subCA.key 4096}

• Step B2: Create a subRoot Certificate Signing Request (CSR):
  \texttt{openssl req -new -key subCA.key -out subCA.csr}

Generating the Server Certificate parameters:

• Step C1: Create a Server Private Key:
  \texttt{openssl genrsa -out server.key 3072}

• Step C2: Create a Server Certificate Signing Request (CSR):
  \texttt{openssl req -new -key server.key -out server.csr}

Generating the Client Certificate parameters:

• Step D1: Create a Client Private Key:
  \texttt{openssl genrsa -out client.key 2048}

• Step D2: Create a Client Certificate Signing Request (CSR):
  \texttt{openssl req -new -key client.key -out client.csr}

Signing the parameters with the (sub)CA to release the final certificates:

• Step X1: Signing the subCA CSR, resulting in the subCA Certificate (CRT):
  \texttt{openssl x509 -req -days 3652 -in subCA.csr -CA RootCA.crt -CAkey RootCA.key -set_serial 01 -extfile /etc/ssl/openssl.cnf -extensions v3_ca -out subCA.crt}

• Step X2: Signing the Server CSR with the subCA, resulting in the Server Certificate (CRT):
  \texttt{openssl x509 -req -days 1826 -in server.csr -CA subCA.crt -CAkey subCA.key -set_serial 01 -extfile /etc/ssl/myssl.conf -extensions serverauth -out server.crt}

• Step X3: Signing the Client CSR with the subCA, resulting in the Client Certificate (CRT):
  \texttt{openssl x509 -req -days 1826 -in client.csr -CA subCA.crt -CAkey subCA.key -set_serial 10 -extfile /etc/ssl/myssl.conf -extensions clientauth -out client.crt}

• Step X4: Generate a Portable Client Certificate (PKCS#12):
  \texttt{openssl pkcs12 -export -in client.crt -inkey client.key -out client.p12}
2. Future Directions

PKI is a very important component of the Information Security field of study. It is constantly growing and evolving, by acquiring new features, bug fixes, etc.

It is very important to understand the direction in which PKI evolves, and to assure that this direction is an efficient one. With every new standard release or RFC release, PKI has the possibility to evolve and improve itself.

However, a very important component is the marketing study. This marketing study assures that the real requirements of real users, based on real scenarios are taken into consideration when releasing a new version or a patch to update an existing version.

A summary of the state of the art of today's PKI, in the context of future directions:

- Technically, X.509 PKI works and is ubiquitous;
- The most common use of PKI is for SSL/TLS for secure communication with millions of web servers;
- Most RPs (users) do not have certificates or relationships with any CAs;
- There are over 600 commercial CAs in existence, from many different countries;
- How can an RP know if all of these are trustworthy, because reading their CPs/CPSs is not practical.
- How can an RP get damages if CA is untrustworthy or careless or is hacked etc, when it has no formal relationship with CA or taking into account cross border legal issues.

Many Certification Authorities are not trustworthy:

- In March 2011, Comodo’s root CA was hacked and issued 9 SSL certificates for 7 domains including Microsoft, Google, Skype, Yahoo and Mozilla
- In Sept 2011, Diginotar CA went out of business after hackers broke in and issued at least 531 fraudulent certificates for the Dutch Government!
- Malaysian Agricultural Research and Development Institute CA (DigiCert Sdn. Bhd.) had its keys stolen in 2011 which allowed a fake Adobe Flash Updater to be created which installed malware on users PCs turning them into spies. This CA's cert is now revoked by browsers.

The original trust model:

- The original X.509 PKI model assumed everyone would have a certificate from a CA, so that certificate subjects were also relying parties (RPs);
• The classic PKI is represented by the three cornered trust model;
• Every RP had a relationship with its trust anchor/root of trust;
• Cross certification ensured trust in other CAs when RP and subject had different CAs. 

2.1 The 3-cornered (closed) trust model

In a classic scenario, there is a CA, a Relying Party and a Certificate Subject.

The trust relationship between the Relying Party and the CA is a Direct Trust Relationship.

The trust relationship between the Certificate Subject and the CA is also a Direct Trust Relationship.

However, the trust relation between the Relying Party and the Certificate Subject is an Indirect Trust Relationship.
2.2 Cross-certification

Cross-certification represents the process of cross-certifying a certificate by a CA, that belongs to another CA (usually by grouping keypairs).

However, it rarely or almost never happens in practice. [8]

Also, there might be trust, legal and liability issues. [8]

For this to work efficiently, the certifying CA needs to trust the certified CA. [8]

The certifying CA takes on liabilities when the cross certified CA fails to act properly, or is attacked, or makes a mistake etc. That's why lawyers ensured cross certification was not commercially viable. [8]

![Fig. 11 – An example of cross certification of a public key with two private keys from other Transient-Key servers](image)

Also, a cross-certificate (X.509 standard certificate) is used to form a relationship of trust between two or multiple certification authorities.
2.3 Lots of Trust Anchor CAs

A trust anchor is an authoritative entity represented by a public key and associated data. The public key is used to verify digital signatures, and the associated data is used to constrain the types of information or actions for which the trust anchor is authoritative. [18]

Trust anchors are widely used to verify digital signatures and validate certification paths [RFC5280][X.509]. They are required when validating certification paths. Though widely used, there is no standard format for representing trust anchor information. This structure is intended to satisfy the format-related requirements expressed in Trust Anchor Management Requirements [TA-MGMT-REQS] and is expressed using ASN.1 [X.680]. [18]

It can provide a more compact alternative to X.509 certificates for exchanging trust anchor information and provides a means of associating additional or alternative constraints with certificates without breaking the signature on the certificate. [18]

In the above figure, there are some CA Certificates that are self-issued by trust anchors in a PKI environment.

Relying parties rely on these certificates, resulting the end-entity public-key certificates.

Fig. 12 – Lots of Trust Anchor CAs [8]
2.4 Compelled Certificate Creation attack

The general meaning of the phrase Compelled Certificate Creation attack is the fact that a government agencies or entity can compel a Certification Authority to issue a false digital certificate for a subject with the sole purpose of intercepting and decrypting secure communication.

Although we do not have direct evidence that this form of active surveillance is taking place in the wild, we show how products already on the market are geared and marketed towards this kind of use – suggesting such attacks may occur in the future, if they are not already occurring. [19]

Consider a hypothetical situation where an American executive is in France for a series of trade negotiations. After a day of meetings, she logs in to her corporate webmail account using her company-provided laptop and the hotel wireless network. Relying on the training she received from her company’s IT department, she makes certain to look for the SSL encryption lock icon in her web browser, and only after determining that the connection is secure does she enter her login credentials and then begin to upload materials to be shared with her colleagues. However, unknown to the executive, the French government has engaged in a sophisticated man-in-the-middle attack, and is able to covertly intercept the executive’s SSL encrypted connections. Agents from the state security apparatus leak details of her communications to the French company with whom she is negotiating, who use the information to gain an upperhand in the negotiations. While this scenario is fictitious, the vulnerability is not. [19]

The round-up summary:

• Government agency compels a national CA to issue a false SSL certificate to it in name of an Org or intermediate CA

• This certificate is then used by law enforcement to launch a MITM attack e.g. via a cyber café or hotel internet connection

• User’s browser sees a “genuine” trusted SSL certificate from the site and lock icon is displayed

• Whilst Agency decrypts data using its MITM certificate and re-encrypts it for the genuine web site

• Packet Forensics from Arizona produce a commercial box for this MITM attack  [8]

The security and confidentiality of millions of Internet transactions per day depend upon the Secure Socket Layer (SSL)/Transport Layer Security (TLS) protocol. At the core of this system are a number of Certificate Authorities (CAs), each of which is responsible for verifying the identity of the entities to whom they grant SSL certificates. It is because of the confidentiality and authenticity provided by the CA based public key infrastructure that users around the world can bank online, engage in electronic commerce and communicate with their friends and loved ones about the most sensitive of subjects without having to worry about malicious third parties intercepting and deciphering their communications. [19]
2.5 Relying-party management and alternatives

A relying party represents a server that provides access to a secure application software.

The following recommendations should be respected, to ensure security and efficiency:

- Browser manufacturers act as a proxy for all users in validating that a CA is trustworthy;
- They SHOULD only add root certificates of trustworthy CAs to their trust stores;
- They SHOULD check revocation information before validating a web sites certificates;
- They SHOULD check all policy information in certificates such as key usage, policy fields, name constraints etc. when validating certificates;
- They SHOULD remove untrustworthy root and subordinate CA certificates from their trust stores. Can still find MD5 root certs, used by APTs Flame, Stuxnet etc.;
- They SHOULD offer liabilities to users if they get it wrong and the user suffers a loss because of their neglect; [8]

The following alternatives have been proposed, to improve PKI and to address some already existing problems or issues:

- Introduce a trusted third party – trust broker – who acts on behalf of RPs in validating certificates;
- RP enters into a contractual relationship with TB, who will offer guarantees and compensation if it makes a wrong trust decision about a certificate;
- TB will read CPs and CPSs of CAs and determine how trustworthy they are, what their certificates can be used for, and what liabilities they offer;
- We have a four cornered trust model. [8]

A rationale and a model is presented in the following article:

2.6 The 4-cornered (open) trust model

The 4-cornered trust model introduces a new element: The Trust Broker.

Still, the new X.509 trust model cannot resolve all the problems, like:

- Still need standardized protocol(s) for communications between RP and TB;
- Support for TBs will need to be built into web browsers either via a plugin or direct manufacturer support;
- Needs a profitable business model to ensure that entrepreneurs will offer a TB service;
- All of the above is traditionally outside the scope of ITU-T X.509;
2.7 The Trust Broker Concept

A Trust Broker is an intermediary entity which validates trust, policies and certificates and acts on behalf of the Relying Party. The RP trusts that the Trust Broker will make a good choice of a trustworthy Certification Authority. However, that trust broker must be a responsible one and may offer liabilities if it makes a wrong decision.

A trust framework is a certification program that enables a party who accepts a digital identity credential (called the relying party) to trust the identity, security, and privacy policies of the party who issues the credential (called the identity service provider) and vice versa. The purpose of a trust framework is to reduce the residual risk to the trustor. [20]

RPs need to rely only on the trust broker and not on each and every CA issuing certificates to their holders. In this case, the X.509 trust model is fairer for the RPs. The trust broker evaluates objectively the CA and its certificates, and sends recommendations to RPs that helps them to make informed decisions about these certificates. The relation between the trust broker and the RPs must be regularized by explicit agreements. In such agreements, the trust broker recognizes its responsibility to the RPs about the provided recommendations and requires itself to respect and to protect the privacy of the RPs. On the other side, the trust broker must be independent from the CAs. Its relationship with CAs must also be regularized by explicit agreements, so that the trust broker can transfer the responsibility to a CA when a false recommendation is made resulting from incorrect information provided by the CA. The contractual agreements between the RPs and the trust brokers create trust communities. [21]

The role of trust broker could be provided by:

- Commercial organizations which make a business from giving recommendation about certificates;
- National governments which wish to facilitate e-commerce in their countries;
- An international body like the UN in order to facilitate international trade. [21]

Finally, to help RPs to make informed decisions about certificates, the trust broker must provide contextual recommendations. For example, recommendations about a certificate that authenticates an email server should be different from recommendations about a certificate that authenticates an e-commerce server. This is because the information sent by the RP to the certificate holder (login/passwd or credit card information) and the consequences of these transactions are different. In the first case, the critical information is the quality level of the certificate and the financial and juridical protection if the certificate is false. In the second case, this information should be supplemented with the maximum transaction amount that can be used in order to stay covered by the financial protection offered in the CP/CPS. [21]
2.8 Proposed changes in X.509

As time passes and the IT world evolves, so do the challenges in matters of security, authorization and confidentiality.

In an earlier chapter, a few suggestions were made regarding the future of PKI and X.509. We review them in more detail now.

Some proposed changes to X.509 for year 2016:

• Cleaning up of the text by removing errors and inconsistencies and replacing badly worded descriptions;

• Removing non-PKI and PMI material from X.509:
  ◦ Move the directory authentication specifications from X.509 to X.511;
  ◦ Move Password Policy specifications from X.509 to X.511;
  ◦ Move Password Policy schema definitions from X.509 to X.520;

• Cleanly separate PKI and PMI into different sections:
  ◦ In Aug 13 issued a defect report on text which said ACs and PKCs could appear in the same CRL;

• Removing unused and duplicate ASN.1 data structures like certificationPath, forwardCertificationPath and crossCertificate (pkiPath is used instead).

Also, there is other work related to improving or distributing PKI in other areas as well:

• PKI Profiles for:
  ◦ Smart Grids
  ◦ Wireless PKI (WPKI)
  ◦ Cloud Computing

• Cryptographic Message Syntax (CMS) to eliminate all obsolete ASN.1 features and make it usable with all ASN.1 standardized encoding rules;

• Procedures for establishing and maintaining a PKI for large PKI networks with machine to machine interactions;

• Certified Mail Transport and Certified Post Office Protocols like the electronic equivalent of registered post.
2.9 Smart Grids and PKI

The term “Smart grid” generally refers to a class of technology people are using to bring utility electricity delivery systems into the 21st century, using computer-based remote control and automation. These systems are made possible by two-way communication technology and computer processing that has been used for decades in other industries. They are beginning to be used on electricity networks, from the power plants and wind farms all the way to the consumers of electricity in homes and businesses. They offer many benefits to utilities and consumers – mostly seen in big improvements in energy efficiency on the electricity grid and in the energy users’ homes and offices. [22]

The Smart Grid is an update to the current power grid infrastructure, which is becoming outdated. Incorporating information systems into the traditional power grid allows the system to make smart decisions about the state of the power grid and dynamically alter the flow of electricity to meet changing needs and conditions. However, this new infrastructure presents cyber security risks that must be mitigated in such an essential system. Many information systems use a Public Key Infrastructure (PKI) to solve this problem, but the unique requirements of the electrical power grid don't fit well into an existing PKI solution. [23]

PKI would be a good solution to manage the security of a Smart Grid, however, it presents some problems:

- Vast numbers of devices connected to electricity grids – possibly more than existing Internet – so there could be PKI scalability issues;
- Means many more roots of trust. How will this be handled?
- Means millions of cheap devices holding their own private keys and roots of trust, may be built in foreign factories by unvetted workers, so probably easily compromised, thus revocation issues may occur;
- CRLs today are consistently larger than first envisaged, so cause bandwidth and latency problems. OCSP also has performance issues;
- What is the identity of the certificate subject? Manufacturer of device? Supplier of electricity? Owner of property? Occupant of property? Most not known when key is installed in device, so may need dynamic certification or other means of identifying subject such as an AC;
- Who is authorised to request re-certification, or revocation?
- How do we rekey a hardware device whose private key has been compromised?
- How do we protect the privacy of the data subject? [8]
2.10 Wireless PKI

The Wireless PKI (also known as WPKI) is an authentication method (2-factor scheme) which uses a laptop and a mobile telephone for setting up a secure operation.

With the advent of wireless communication and internet protocol, many technologies have been developed to provide mobile phone user with the wireless internet service. Security supporting wireless internet must be guaranteed at same level as the wired security. But PKI (Public Key Infrastructure) which is used for the security of e-commerce in wired internet is not suitable for the mobile phone because of the fundamental limitation of performance such as less memory and less powerful CPU. Therefore, we need to develop a wireless PKI (WPKI) that provides the similar security level as the wired PKI supporting mobile phone. [24]

As mobile user utilizes wireless internet through mobile phone, a variety of internet services supporting mobile phone have been also increasing. The wireless internet refers to accessing internet through wireless communication using mobile phone. For mobile users to successfully utilize data service and M-commerce through wireless internet, security must be guaranteed. Similar to the wired internet, for wireless internet to provide secure M-commerce service, following functions must be provided: confidentiality and integrity of data, entity authentication, and non-repudiation. Technologies that apply these security elements to mobile phones and wireless internet environment must be able to provide users with the same level of security as in the wired internet environment. [24]

Many security protocols on internet and most security applications for e-commerce are based on public key cryptography. PKI (Public Key Infrastructure) applies a public key cryptographic method to transmit user’s public key and user’s identity in a secure and reliable way. Users of public key cryptography transmit their public keys to others, and must safeguard private key corresponding to the public key. To guarantee security of M-commerce via wireless internet, public key infrastructure technology suitable for wireless environment must be required. In the wireless environment, we have two different elements from wired internet: mobile phone and wireless internet. [24]

Mobile phone has fundamental limitation of performance such as less memory and less powerful CPU. And wireless data network presents more constrained communication environment such as less bandwidth and has different protocol compared to wired internet protocol. Therefore, it is difficult to apply wired PKI technology for security than wired environment. At first, mobile phone must generate public key pair and compute digital signature using the key. And a public key certificate could be issued to a mobile user through wireless internet. The public key certificate provides a method to bind the public key and its owner. Using the certificate, the mobile user must authenticate itself and make secure channel for internet service such as M-commerce. [24]

For the WPKI to operate efficiently, there are some requirements. Also, in order to apply wireless PKI to mobile phone through wireless internet with the same level of security as that of the wired internet, the following requirements are necessary:

- Select optimal digital signature algorithm to be calculated in mobile phone;
- Minimize data size to be stored in mobile phone and to be transmitted through wireless bandwidth;
- Optimize CMP protocol to be processed in mobile phone and through wireless bandwidth;
- Optimize certificate validation scheme;

For digital signature, computation of public key pair generation, digital signature generation and verification in mobile phone are required. RSA based public key cryptographic algorithm has been selected for digital signature algorithm of PKI for a long time. But public key pair generation based on RSA algorithm in a mobile phone might be time consuming or be impossible due to the lack of memory and CPU performance. Therefore we need an alternative public key algorithm to make the key generation possible in the mobile phone. [24]

After a public key pair is generated, the mobile phone must perform computation for digital signature generation and verification, and time for digital signature operation must be acceptable to users. [24]

The article referenced as number #24 proposes the following WPKI model:

Fig. 14 – A Wireless PKI Model [24]
2.11 ISO/IEC JTC1/SC27

The ISO/IEC JTC1/SC27 is an ISO standard which describes IT Security Techniques.

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National Bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, government and non-governmental, in liaison with ISO and IEC, also take part in the work. [25]

In the field of information technology, ISO and IEC have established a Joint Technical Committee 1: ISO/IEC JTC 1.[25]

Scope of SC 27:

The development of standards for the protection of information and ICT. This includes generic methods, techniques and guidelines to address both security and privacy aspects, such as:

- Security requirements capture methodology;
- Management of information and ICT security; in particular information security management systems (ISMS), security processes, security controls and services;
- Cryptographic and other security mechanisms, including but not limited to mechanisms for protecting the accountability, availability, integrity and confidentiality of information;
- Security management support documentation including terminology, guidelines as well as procedures for the registration of security components;
- Security aspects of identity management, biometrics and privacy;
- Conformance assessment, accreditation and auditing requirements in the area of information security;
- Security evaluation criteria and methodology. [25]

SC 27 engages in active liaison and collaboration with appropriate bodies to ensure the proper development and application of SC 27 standards and technical reports in relevant areas. [25]

Regarding this standard, a New Study Group has been created: Framework for PKI Policy / Practices / Audit. [8]
TOR was created to gauge interest in the development of an internationally accepted and standardized approach to the management, operation, assessment, and certification of PKI Trust Service Providers at varying levels of assurance. This includes management, procedural, assurance and technical standards. [8]

ISO/IEC JTC1/SC27 is still at very early stage of fleshing out content. Focus seems to be on audit of trust service providers. [8]

ISO/IEC JTC1/SC27 subcommittees / working groups:

- ISO/IEC JTC 1/SC 27/SWG-M
  Special Working Group on Management
- ISO/IEC JTC 1/SC 27/SWG-T
  Transversal Items
- ISO/IEC JTC 1/SC 27/WG 1
  Information security management systems
- ISO/IEC JTC 1/SC 27/WG 2
  Cryptography and security mechanisms
- ISO/IEC JTC 1/SC 27/WG 3
  Security evaluation, testing and specification
- ISO/IEC JTC 1/SC 27/WG 4
  Security controls and services
- ISO/IEC JTC 1/SC 27/WG 5
  Identity management and privacy technologies [26]

ISO/IEC JTC1/SC27 statistics:

- Secretary: Mrs Krystyna Passia
- Chairperson: Mr Walter Fumy (until end 2016)
- Number of published ISO standards under the direct responsibility of ISO/IEC JTC1/SC27: 144
- Participating countries: 53
- Observing countries: 18 [26]
2.12 Proposals from IETF

In the PKI field of study, the Internet Engineering Task Force (IETF) has proposed a number of applicable changes to PKI and X.509.

*The PKIX Working Group:*

- Started in 1995 with the goal of developing Internet standards to support X.509 PKIs;
- Has published over 60 RFCs about X.509, including:
  - Protocols for certificate management, time stamping, online certificate status, use of LDAPv2 & FTP/HTTP, Data Validation and Certification Server, Delegated Path Validation and Delegated Path Discovery, Server based certificate validation, trust anchor management;
  - Profiles for PKCs and ACs, Qualified certificates
  - New certificate extensions: logotype, proxy certs, warranty, permanent ID, attributes supporting authentication in PPP and WLAN, IP addresses and AS identifiers, subject identification method, clearance attribute;
  - Others: Diffie-Hellman POP Algorithm, SHA 224, DNS Certification Authority Authorization (CAA) Resource Record.

*Certificate Transparency from Google:*

- Experimental RFC 6962, June 2013;
- Log servers hold Merkle hash trees (append only logs) of all issued certificates. Any CA can send a cert to a log server and get a signed time stamp in response;
- Monitor servers check on all log servers periodically and will flag any unauthorized or suspicious certificates;
- Auditors (typically running in browsers) can check that any certificate and time stamp they receive appears in the log. If not, the certificate of the SSL site is suspect and should not be trusted;
- Will stop MITM attacks, compelled certificate creation attacks, duplicate certs with stolen keys etc.;
- Sovereign Keys from Electronic Frontier Foundation is a similar idea, using “timeline servers” to hold public keys of web sites.
HTTP Strict Transport Security (HSTS):

- RFC 6797, Nov 2012, from Web Security Working Group;
- Allows web sites to say that they are only contactable via HTTPS;
- HTTP Response header contains the sites security policy;
- Browsers remember the policy and will strictly enforce it;
- This stops users “clicking through” browser security warnings of web sites that the browser does not trust. [8]

Public Key Pinning Extension for HTTP:

- Internet draft of Web Security WG;
- HTTP protocol extension allowing web sites to instruct browsers to remember ("pin") the hosts' public keys for a given period of time;
- During this time, browsers will require hosts to present a certificate chain including at least one Public Key that matches one of the pinned ones. [8]

Web PKI Operations (wpkops) working group:

- improve the consistency of Web security behavior;
- address the problems caused by hundreds of variations of Web PKI currently in use;
- describe how Web PKI "actually" works in browsers and servers in common use today by:
  - The trust model on which it is based;
  - The contents and processing of fields and extensions;
  - The processing of the various revocation schemes;
  - How the TLS stack deals with PKI, including varying interpretations and implementation errors, as well as state changes visible to the user.
  - The state changes that are visible to and/or controlled by the user (to help predict the decisions that will be made the users and so determine the effectiveness of the Web PKI);
  - Identification of when Web PKI mechanisms are reused by other applications and implications of that reuse.
- The working group will not:
  - describe how the Web PKI should work;
  - examine the certification practices of certificate issuers;
  - investigate applications (such as client authentication, document signing, code signing, and secure email). [8]
2.13 HTTP Strict Transport Security (HSTS)

HSTS represents a web security policy mechanism for protecting HTTPS websites against downgrade attacks and cookie hijacking and is defined in IETF RFC 6797. By using HSTS, web servers can notify the web browsers that they only accept HTTPS connections.

The HSTS policy is sent in the HTTP response header, with a field named “Strict-Transport-Security”, and it specifies a time period in which HTTPS can be used by the accessing client.

This specification defines a mechanism enabling web sites to declare themselves accessible only via secure connections and/or for users to be able to direct their user agent(s) to interact with given sites only over secure connections. This overall policy is referred to as HTTP Strict Transport Security (HSTS). The policy is declared by web sites via the Strict-Transport-Security HTTP response header field and/or by other means, such as user agent configuration, for example. [27]

An HSTS Policy directs UAs to communicate with a Known HSTS Host only over secure transport and specifies policy retention time duration. HSTS Policy explicitly overrides the UA processing of URI references, user input (e.g., via the "location bar"), or other information that, in the absence of HSTS Policy, might otherwise cause UAs to communicate insecurely with the Known HSTS Host. [27]

An HSTS Policy may contain an optional directive – includeSubDomains – specifying that this HSTS Policy also applies to any hosts whose domain names are subdomains of the Known HSTS Host's domain name. [27]

*The use cases of HSTS:*

The high-level use case is a combination of:

- Web browser user wishes to interact with various web sites (some arbitrary, some known) in a secure fashion;
- Web site deployer wishes to offer their site in an explicitly secure fashion for their own, as well as their users', benefit; [27]

*HSTS Policy effects:*

The effects of the HSTS Policy, as applied by a conformant UA in interactions with a web resource host wielding such policy (known as an HSTS Host), are summarized as follows:

- UAs transform insecure URI references to an HSTS Host into secure URI references before dereferencing them;
- The UA terminates any secure transport connection attempts upon any and all secure transport errors or warnings. [27]
2.14 Public Key Pinning Extension for HTTP

The Public Key Pinning Extension for HTTP is defined in IETF RFC 7469.

It defines a new HTTP header that allows web host operators to instruct user agents to remember ("pin") the hosts' cryptographic identities over a period of time. During that time, user agents (UAs) will require that the host presents a certificate chain including at least one Subject Public Key Info structure whose fingerprint matches one of the pinned fingerprints for that host. By effectively reducing the number of trusted authorities who can authenticate the domain during the lifetime of the pin, pinning may reduce the incidence of man-in-the-middle attacks due to compromised Certification Authorities. [28]

To ensure the authenticity of a server's public key used in TLS sessions, this public key is wrapped into a X.509 certificate which is usually signed by a certificate authority (CA). Web clients such as browsers trust a lot of these CAs, which can all create certificates for arbitrary domain names. If an attacker is able to compromise a single CA, he can perform MITM attacks on various TLS connections. HPKP can circumvent this threat for the HTTPS protocol by telling the client which public key belongs to a certain web server. [29]

HPKP is a Trust on First Use (TOFU) technique. The first time a web server tells a client via a special HTTP header which public keys belong to it, the client stores this information for a given period of time. When the client visits the server again, it expects a certificate containing a public key whose fingerprint is already known via HPKP. If the server delivers an unknown public key, the client should present a warning to the user. [29]

In order to enable HPKP, the following HTTP header message needs to be sent:

"Public-Key-Pins: pin-sha256="base64=="; max-age=expireTime [; includeSubdomains][; report-uri="reportURI"]" [29]

The Server Processing Model:

This section describes the processing model that Pinned Hosts implement. The model has 2 parts: (1) the processing rules for HTTP request messages received over a secure transport (e.g., authenticated, non-anonymous TLS); and (2) the processing rules for HTTP request messages received over non-secure transports, such as TCP. [28]
2.15 Web PKI Operations (wpkops)

The charter for the working group is presented below:

The Web Public Key Infrastructure (PKI) is the set of systems, policies, and procedures used to protect the confidentiality, integrity, and authenticity of communications between Web browsers and Web content servers. The Web PKI is used in conjunction with security protocols such as TLS/SSL and OCSP. [30]

More specifically, the Web PKI (as considered here) consists of the fields included in the certificates issued to Web content and application providers by Certification Authorities (CAs), the certificate status services provided by the Authorities to Web browsers and their users, and the TLS/SSL protocol stacks embedded in web servers and browsers. [30]

The Web PKI Operations (wpkops) working group will work to improve the consistency of Web security behavior. It will address the problems caused by the many hundreds of variations of the Web PKI currently in use:

- For end-users (i.e. relying parties), there is no clear view of whether certificate "problems" remain once they see an indication of a "good" connection. For instance, in some browsers, a "good" indication is displayed when a "revoked" response has been received and "accepted" by the user, whereas other browsers refuse to display the contents under these circumstances.

- Many certificate holders are unsure which browser versions will reject their certificate if certain certificate profiles are not met, such as a subject public key that does not satisfy a minimum key size, or a certificate policies extension that does not contain a particular standard policy identifier.

- Certificate issuers (i.e., CAs) find it difficult to predict whether a certificate chain with certain characteristics will be accepted. For instance, some browsers include a nonce in their OCSP requests and expect one in the corresponding responses, not all servers include a nonce in their replies, and this means some certificate chains will validate while others won’t. [30]

The working group's goal is to describe how the Web PKI "actually" works in the set of browsers and servers that are in common use today. To that end, the working group will document current and historic browser and server behavior. For each this will include:

- The trust model on which it is based;
- The contents and processing of fields and extensions;
- The processing of the various revocation schemes;
- How the TLS stack deals with PKI, including varying interpretations and implementation errors, as well as state changes visible to the user.
- The state changes that are visible to and/or controlled by the user (to help predict the decisions that will be made the users and so determine the effectiveness of the Web PKI).
- Identification of when Web PKI mechanisms are reused by other applications and
implications of that reuse. [30]

Where appropriate, specific products and specific versions of those products will be identified, but recording the design details of the user interfaces of specific products is not necessary. [30]

Only server-authentication behavior encountered in more than 0.1 percent of connections made by desktop and mobile browsers is to be considered. While it is not intended to apply the threshold with any precision, it will be used to justify the inclusion or exclusion of a technique. [30]

A number of activities are outside the immediate scope of this working group, but might be considered in future re-chartering activity or included in the work of other working groups:

- The working group will not work to describe how the Web PKI should work.
- The working group will not examine the certification practices of certificate issuers.
- The working group will not investigate applications (such as client authentication, document signing, code signing, and email) that often use the same trust anchors and certificate processing mechanisms as those used for Web server authentication. [30]

Given the urgency of the required developments and the scale of the task, it is agreed that adherence to the published milestones will take precedence over completeness of the results, without sacrificing technical correctness. [30]
Conclusions

This paper presents the research I have done so far in the fields of PKI and related subjects.

All the topics presented in this paper were read, understood and, in some cases, put in practice.

The research done so far has been very helpful in establishing a direction on which I shall continue researching in the field of the PKI and related subjects.

Finally, I want to research the actual security mechanisms and protocols of PKI, to be able to blend them with IPv6 and the Information Security field of study in a nice comprehensive PhD thesis.
References

[1] - RSA Data Security - Understanding Public Key Infrastructure (PKI)
[2] - Joel Weise - SunPSSM Global Security Practice - Sun BluePrints OnLine - Public Key Infrastructure Overview
[4] - Sebastian Wiesner - Simple PKI - Seminar Innovative Internet Technologies and Mobile Communications, Chair for Network Architectures and Services, Fakultät für Informatik, Technische Universität München
[5] - Drew Streib - Explanation of this Keyringing Analysis
[6] - Peter Gutmann - Everything you Never Wanted to Know about PKI but were Forced to Find Out
[8] - David Chadwick - PKI : state of the art and future trends
[9] - Symeon (Simos) Xenitellis - The Open-source PKI Book - A guide to PKIs and Open-source Implementations - OpenCA Team
[10] - David Chadwick - The X.509 Privilege Management Infrastructure - Information Systems Institute, University of Salford, Salford M5 4WT, England
[12] - Margaret Rouse - URL: http://searchsecurity.techtarget.com/definition/PKI
[19] - Christopher Soghoian, Sid Stamm - Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL
[20] - David Chadwick - Themes in current PKI's trust framework and its future - University of Kent
[23] - Baumeister, T. - Adapting PKI for the smart grid
[24] - Yong Lee, Jeail Lee, JooSeok Song - Design and implementation of wireless PKI technology suitable for mobile phone in mobile-commerce - Korea Information Security Agency
[27] - IETF RFC #6797 - URL: https://tools.ietf.org/html/rfc6797#section-5.2