## Abstract

The intention of this thesis project is to evaluate the security and capabilities offered by a PKI enabled network between a Smart House and different vendors, such as electric Utility Company, Security Company, Fire brigade and the owner of a Smart House. Then I will show how a third party signed certificates can be used for such networks. It includes the use of public key cryptography, digital signatures, certificates, and certificate authorities (Trusted Third Party). How to apply PKI security services in different ways with different requirements is the main issue of this work. One of the end entities will be the Smart House and it will behave as the client. While the other end entity will always behave as the server or web server, owned by one of the other vendors.

The transactions that take place between the Smart House and the Fire brigade or the SecuritAs can be sent across the Internet as clear text. Allowing the transaction to remain in the clear text will avoid the overhead of the symmetric cryptography, but it is vulnerable to eavesdropping. However by verifying the Fire brigade's digital signature, we can be sure that the message is coming from the Fire brigade and it is not modified in transit.

It is important to notice that the communications between the Smart House and the Energy provider are encrypted so that attackers cannot gather the information as it flows across the Internet. It also shows how we could take the advantages of the symmetric and asymmetric cryptography together, and apply them to the two communicating end entities. Using of their benefits will allow us have the ideal solutions of PKI's security.

The SSL and IPsec Protocols are compared in terms of; Accessibility, ease of use, complexity, ownership, performance and scalability. SSL is frequently used to secure web sessions at application level, while the IPsec secures communications between two nodes.

In the WPKI scenario, the Smart House owner must use attribute certificate in order to get special privileges and authorization.

## Preface

This postgraduate thesis is the final part of the Master of Science degree in Information and Communication Technology (ICT) at the Agder University College. This assignment is a closure of the education that leads to the title Master of Science.

The work has been carried out in the period between the beginning of May and the end of September 2003, and has taken place at Agder University College.

I would like to use this opportunity to thank my supervisor Høgskolelektor Magne Arild Haglund, and the director of study Stein Bergsmark at Agder University College, for their great help and continuous advice, during the process of this thesis project. I would like also to thank Catherine Krath expedisjonsleder for her psychological help when I was in need.

Last, but not least I want to thank my wife Khadan for her encouragement and understanding throughout my work’s period. Numerous people have also given me additional ideas, comments and motivation. I am grateful for all help and support, but do not find the space to thank everybody by name.

Grimstad,
September 2003

Ahmed Guleid

# Table of Contents

## List of Figures

*All copyright of Ahmed Guleid*

# 1 Introduction

## *1.1 Background*

Information systems are mission critical in today’s organizations. The explosion of computing power, computer internetworking, and innovative technologies have revolutionized communication. Face to face negotiations that relied on words and expressions have evolved to sophisticated protocols that rely on circuits and bits. Unlike traditional forms of communication, e-communication has introduced major security vulnerabilities. From the National Security Agency’s urgency to defend secrets, to medical professionals protecting patient’s health records, to the financial sector scrambling to secure multi billion dollar transactions, organizations and governments are becoming more proactive with security .

There are several areas of Internet security that have emerged as foundational, including firewalls, intrusion detection, and secret key cryptology. This thesis project will focus on cryptography and the role of public key infrastructure in the protection of information systems. Public Key Infrastructures (PKI), provide the most effective method of assuring communication authentication, integrity, and none repudiation on the Internet.

Public key cryptography is suitable for distributed and dynamic environments, with a medium or big number of communicating parties sending data through insecure channels. In fact, it provides a secure communication method for recipients not previously known each other [16].

The Norwegian government has decided to support the Public Key Infrastructure (PKI). Therefore Norway is ready to take back its neighbours advance in this field.

The government has decided that the PKI will be one of three ICT priority areas in the near future. To implement the government’s strategy for ICT in the public sector, can Norway fight back compared with the lost to other countries, because of the lack of focusing. The strategy is so clear and points out that the importance to induct electronic signature in Norway is important, reasonable and nice. Qualifications and exchanging of data are the two other fields, which are part of the priority strategy, are also natural and important.

The encouragement of the Norwegian government to implement Public Key Infrastructure in the public sector, shows that the importance of this thesis project, which is handling of that subject Public Key Infrastructure (PKI) [15].

## *1.2 Thesis definition*

This thesis project will focus on how to apply public key infrastructure (PKI) services to two end to end users communicating over the Internet. One of the end users will be the Smart House as (client side) and the other end user will be one of the vendors such as utility electric company LoS AS, as the (server side). The Norwegian main Post office (Posten) will be used as the Trusted Third Party (CA). These security services have different objectives and requirements. The final thesis definition is therefore formulated as follows:
All those Servers and Messages which have the same colour will have the same security types and requirements see Figure 1-2. The following services are the fundamental security services of the PKI that I am going to apply on the different end users: *Authentication, Integrity, Confidentiality, None repudiation.*

**How to apply these (PKI) seurity services to the Smart house and the other vendors and are shown here:**

**1.** Messages that flows between Smart House and LoS AS, Agder Energi must use all the PKI security services.

- ***LoS AS or Agder Energi*** *and* ***Smart House*** *must mutually authenticate each other, because the transactions that flow between them are very important.*
- *Messages between* ***LoS AS or Agder Energi*** *and* ***Smart House*** *must not be modified during transit.*
- *Messages between* ***LoS AS or Agder Energi*** *and* ***Smart House*** *must be protected from eavesdropping.*
- *If* ***LoS AS or Agder Energi*** *sends message to the* ***Smart House*** *there must be no way of denying that* ***LoS AS or Agder Energi*** *has sent a message to the* ***Smart House****, claiming at a later date that the information was never sent.*

**Figure 1- 1. This modified WPKI mobile figure is from Telecom**

2. Messages that flows between Smart House and Fire brigade or Securitas uses some of the PKI security services.

    - *The **Fire brigade or Securitas** and the **Smart House** may mutually authenticate each other, But the **Smart House** must authenticate the **Fire brigade*** and ***Securitas***
    - *Messages between the **Fire brigade or Securitas** and the **Smart House** must not be modified during transit.*
    - If th***e Fire brigade or Securitas** has sent a message to the **Smart House**, there must be no way of denying that the **Fire brigade or Securitas** has sent a message to the **Smart House,** claiming at a later date that the information was never sent.*

3. Messages that flow between the Smart House and the house owner, uses some of the PKI security services.
    - *The **Smart House*** and the ***house owner** must mutually authenticate each other, because the transactions that flow between them are very important*
    - *Messages between* the ***house owner** and the **Smart House** must not be modified during transit.*
    - *If the **house owner** sends a message to the **Smart House** there must be no way of denying that the **house owner** has sent a message to the **Smart House**, claiming at a later date that the information was never sent.*
    - *Here the owner must be authorised to do what he/she wants with his own home computer. The owner must use attribute certificate (AC) instead of public key certificate.*

The title of the thesis was formulated as follows:

*"The Use of PKI in a Smart House Solution."*

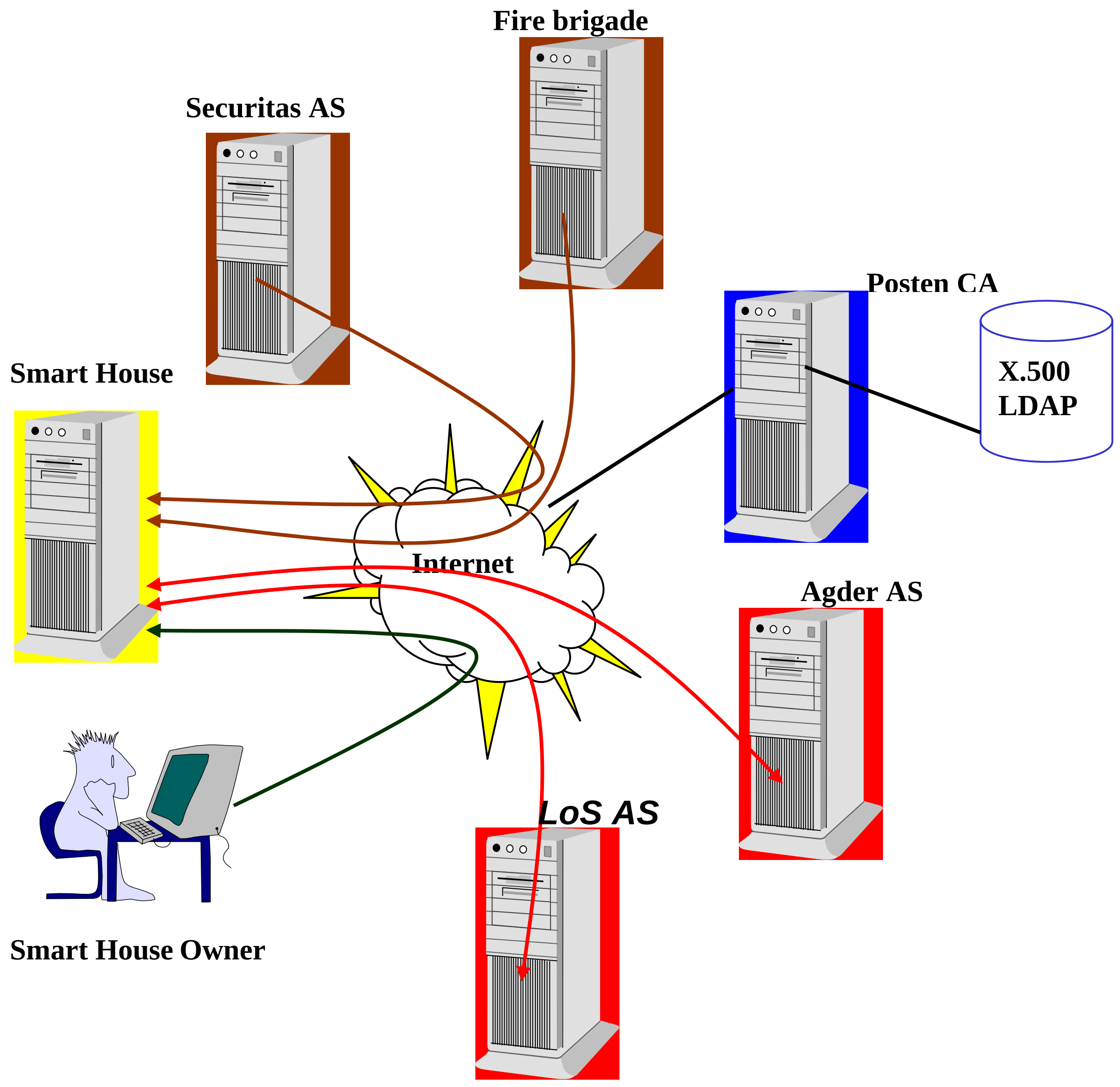


**Figure 1-2 Simplified overview of this thesis project**

## *1.3 Limitations*

This thesis project was originally meant to discuss and evaluate the possibility of using secure transactions with PKI through GPRS networks (Wireless network), and the Internet. Wireless networks use Wireless Public Key Infrastructure (WPKI), which is a big area of discussion and was under development in the last couple years. Because of the difficulty to understand both PKI and WPKI at the same time and because of due to the change of the title of this work, will this thesis project discuss only Public Key Infrastructure (PKI), and I will show a brief scenario of mobile transaction using WPKI.

PKI security services, in my work are limited to cover only the communication that flows between the Smart House and the external participants. This thesis project does not handle the security issues of the Smart House's internal network.

## 1.4 A brief overview of PKI

Public key cryptography is widely recognized as being a fundamental technology on which several basic security services can be built, such as authentication, integrity, non-repudiation, and confidentiality. Crucial to the operation of a global public key crypto system on the Internet is a practical and reliable method for publishing the public keys, called a Public Key Infrastructure or PKI.

Public key cryptography is suitable for distributed and dynamic environments, with a medium or big number of communicating parties sending data through insecure channels. In fact, it provides a secure communication method for recipients not previously known each other. In order to get this kind of secure communications, a common element of trust is necessary. Public key certificates used in these communications must be certified (signed) by trusted entities (certification authorities), in order to assure the identity of the parties involved.

A PKI is an absolute solution to all of the following security four categories [4]:

- ***Authentication (Identification)***

  Verifying that someone actually is the one he or she claims to be. Documents like passports, driving license are commonly used to verify identities in the real world. In two communicating parts through the Internet, it must be possible to verify identities remotely.

- ***Integrity***

  An integrity is the process that makes sure the data was not modified during transit and the information is complete and accurate valid.

- ***Confidentiality***

  Means that information must only be available to those entitled to see it, and must be protected from those who are not.

- ***Non-repudiation***

  Where a party of communication can't deny that the communication ever occurred

## 1.5 Work and methodology

In order to establish a framework for discussion, I first define the essentials of cryptology and high light on the most significant components of a PKI. Different scenarios of PKI security services will be discovered, and the two main end to end communication protocols (SSL and IPsec). Then the discussion in chapter 7 begins with these transaction protocol's advantages and disadvantages, followed by the discussion of the different scenarios, combining the benefits of the symmetric and asymmetric cryptography, and their PKI security services. By using the advantages of these two cryptographies I will show an ideal solution of PKI's security services. Then a communication without encryption (without confidentiality), but with authentication, integrity and non-repudiation will be shown.
All the analyses and scenarios are based on the messages and their protection between possibly different vendors (servers) and the Smart House (client).The exception here is that the Smart House will behave as a server in the Wireless scenario.

## *1.6 Report outline*

This thesis project is mainly a theoretical evaluation and it covers a wide area of Public key Infrastructure. The reader gets familiar with the digital certificates, and how to manage it. I will show how we could use PKI to secure transactions between two communicating end to end systems. In chapter two, we will go through the basic concept of information security and cryptography. In chapter three and four the PKI basics and PKI components are introduced in more detail. In chapter five I will describe the two main transaction protocols, which are relevant to my work. As a result of my work, I introduce in chapter six a set of scenarios showing possible ways of applying PKI security services, between two communicating parties. I assume that the Smart House will behave as (Client side) and other vendors such as Utility electric Company LoS as a (Server side), and the Norwegian main post office (Posten) as a trusted third party (CA). In the Wireless case, the Smart House will behave as a server and the owner’s laptop will be the client. In chapter seven I will summarize the results and discuss the different scenarios and the transaction protocols that I used in this work. In chapter eight will be the conclusion of this work.

# 2 Attacks against PKI security services and Cryptography

## *2.1 Introduction*

In this chapter we will look at the possible attacks against PKI security services, and the Cryptography that gives us the hope to protect our data against these attacks. I will introduce the two main crypto types, the public cryptography and the symmetric cryptography, which are relevant to this work, digital signature and hashes are also introduced.

## *2.2 Attacks against PKI security services*

An attacker (adversary) is an entity that uses unauthorized means to threaten communication. Passive and active attacks are the two main categories. A passive attacker threatens privacy by listening and analyzing for example network traffic in order to be able to conceive pieces of confidential information. An active attacker uses active means to disturb, alter, and destroy or to otherwise produce harm against authentication and integrity of information systems. Some of the most common threats to information systems are [3]:

- System penetration where unauthorized person gains access to a system. An attacker may use for example brute force technique to guess passwords or exploit a security weakness in the system.
- Authorization violation where authorized person misuses the system. For example a user account is used to gain administrator rights.
- Planting where an intruder leaves behind a planted capability to perpetrate future attacks. For example a piece of software that appears to be legitimate (Trojan) leaves a back door to the system for the attacker.
- Communication monitoring (eavesdropping) that can be used for example to listen passwords and other pieces of confidential information. Even if the communication is encrypted a clever attacker may be able to deduce important hints about the data merely by for example collecting statistical information.
- Communication tampering where an attacker modifies or disturbs communication for example by changing data records. Sometimes an attacker may cheat others by claiming to be some other legitimate party of communication and persuading hand over confidential information.
- Denial of Service where an attacker for example floods a server system with bogus requests so that not even authorized users can get service.
- Repudiation where a party of communication denies that the communication ever occurred.

## *2.3 Cryptography Background*

Historically, if two entities needed to communicate privately they used a shared secret or password. This meant that they initially established a shared secret over a secure channel. Not was only this time consuming and non-scalable, but it also made it impossible for two strangers to communicate securely.
Cryptography was the gateway that allowed two different parties to exchange sensitive information through a secure channel using encryption. Encryption uses mathematical

algorithms to encrypt or scramble the contents of a message. Only with the proper complementary tools could the receiver decrypt or unscramble the message back to readable form.  Thus, by using encryption and decryption to send and receive messages assures confidentiality [12].

## *2.4 Types of Cryptography*

Cryptography is fundamentally based on the use of keys that are used to encrypt and decrypt data. The word cryptography means hidden or secret writing. There are two types of cryptography:

- Secret key or Symmetric and
- Public key or asymmetric.

### 2.4.1 Symmetric Key Cryptography

In symmetric cryptography, both the sender and receiver use the same key. Let's assume that LoS AS wants to send the Smart House a message.  Then LoS AS encrypts the plain text using encryption algorithm and the key KeySL obtaining the cipher text. Smart House uses the decryption algorithm and the same key KeySL to recover the plain text from the cipher text (Figure2-1).

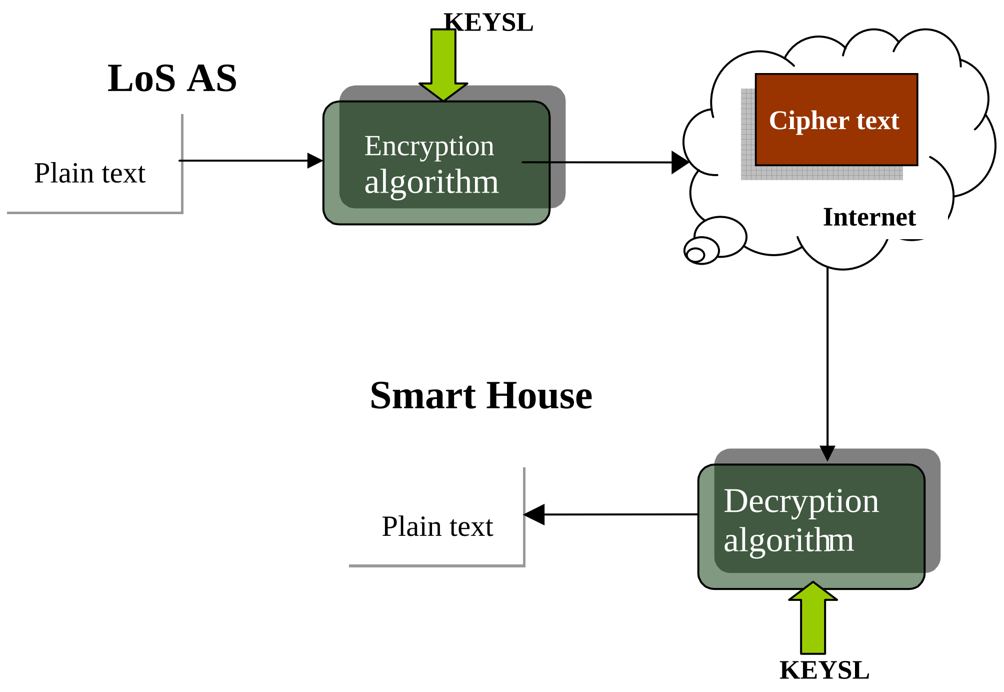


**Figure 2. 1. Symmetric encryption and decryption**

As a result, symmetric keys are sometimes called shared secret key systems. Clearly, this key must be kept secret among the communicating parties; otherwise the communication can be intercepted and decrypted by others. Until the mid 1970's, symmetric cryptography was the only form of cryptography available, so the same secret had to be known by all individuals participating in any application that provided a security service. Although this form of cryptography was computationally efficient, it suffered from the fact that it could not support certain security services, and it presented a difficult key management problem since the secret keys had to be distributed securely to the communicating parties. However, this all changed when Whitfield Diffie and Martin Hellman introduced the notion of public key cryptography with the publication of their “New Directions in Cryptography” paper [DH] in 1976. This represented a significant breakthrough in cryptography because it enabled services that could not previously have been entertained as well as making traditional security services more expedient [21].

### 2.4.2 Public Key Cryptography

Public key cryptography is based on the use of key pairs. When using a key pair only one of the keys, referred to as the private key must be kept secret and (usually) under the control of the owner. The other key, referred to as the public key, can be available freely for use by any person who wishes to participate in security services with the person holding the private key. This is possible because the keys in the pair are mathematically related but it remains computationally infeasible to derive the private key from knowledge of the public key. Any individual can send the holder of a private key a message encrypted using the corresponding public key and only the holder of the private key can read the secure message. Take as an example that the LoS AS wants to send an encrypted text to the Smart House. LoS AS encrypts the plain text with Smart House’s Public Key, and the Smart House decrypts the plain text with its own Private Key (Figure 2-2).

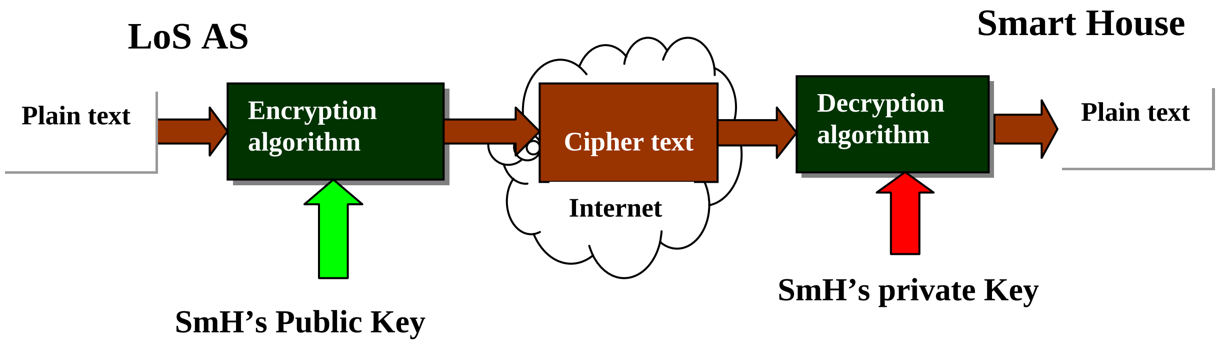


**Figure 2. 2. Encryption and decryption with SmH’s Public Key/Private Keys**

Similarly, the Smart House can establish the integrity and origin of the data the Smart House sends to LoS AS by digitally signing the data using its private key. Anyone who receives that data can use the associated public key to validate that it came from the holder of the private key and verify the integrity of the data has been maintained. The Smart House has signed the plain text with its Private Key, and everyone who gets its Public Key can decrypt it (Figure 2-3).

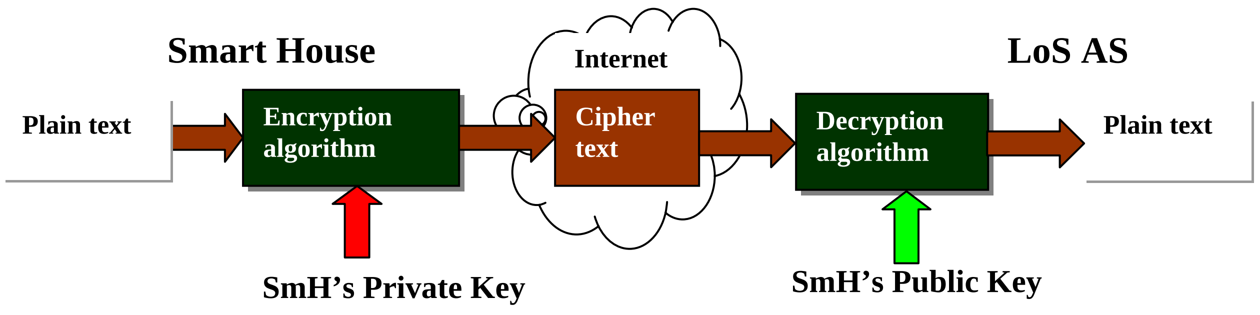


**Figure 2. 3. The principal of signing with Private Key**

## *2.5 Key Length and Encryption Strength*

Encryption strength is often described in terms of the size of the keys used to perform the encryption: in general, longer keys provide stronger encryption. The strength of encryption is related to the difficulty of discovering the key, which in turn depends on both the cipher used and the length of the key. For example, the difficulty of discovering the key for the RSA cipher most commonly used for public key encryption depends on the difficulty of factoring large numbers. Key length is measured in bits. For example, 128-bit keys for use with the RC4 symmetric key cipher supported by SSL provide significantly better cryptographic protection than 40-bit keys for use with the same cipher. Roughly speaking, 128-bit RC4 encryption is 3 x $10^{26}$ times stronger than 40-bit RC4 encryption. Different ciphers may require different key lengths to achieve the same level of encryption strength. The RSA cipher used for public key encryption, for example, can use only a subset of all possible values for a key of a given length, due to the nature of the mathematical problem on which it is based. Other ciphers, such as those used for symmetric key encryption, can use all possible values for a key of a given length, rather than a subset of those values. Thus a 128-bit key for use with a symmetric key encryption cipher would provide stronger encryption than a 128-bit key for use with the RSA public key encryption cipher.
This difference explains why the RSA public key encryption cipher must use a 512-bit key (or longer) to be considered cryptographically strong, whereas symmetric key ciphers can achieve approximately the same level of strength with a 64-bit key. Even this level of strength may be vulnerable to attacks in the near future. The U.S. Government restricts export of cryptographic software, including most software that permits use of symmetric encryption keys longer than 40 bits [19].

## *2.6 Hashes*

A hash algorithm will take a large chunk of data and compress it into a fingerprint or digest of the original data. I think we should take an example to explain hashes.
If I were to give you the number 483820, and asked you to divide that number by 4, you would get the result of 120955. In a way, 120955 is a fingerprint for the equation (483820 divided by 4). No matter how many times you divide 483820 by 4, you always get 120955. If you changed either number in the equation by any amount, the division would not produce

120955. Alternatively if I handed the number 120955, but did not tell you any further information, you would be unlikely to tell me what the original equation was, since there are an infinite number of divisions the could have produced the same result.

In many ways, these features are the same as those associated with hash algorithms. With a hash, you take a large block of data and compute an equation across the data. The output of the hash is a value that is smaller than the original data. If you change even one bit of the original data, the output hash value will be different. Also, as with the division example, there are many different sets of data that could compute the same hash value [4].

## 2.7 Digital signatures

The digital signature provides means for ensuring integrity and non-repudiation of electronic messages. A digital signature is a number dependent on the private key known only to the signer, and additionally, on the content of the message being signed. If the message is changed the signature calculated again would not be the same as the original. In theory it may be possible form two messages that produce the same signature but it is highly improbable that the other message makes any sense to the receiving party [7]. Signature must be verifiable: if a dispute arises as to whether a party signed a document, an unbiased third party should be able to resolve the matter equitably, without requiring access to the signer's secret private key. In public key systems the signature can be verified by using the public key corresponding to the private key that was used to sign the message.

An encrypted version of the message is sent attached to a copy of the plaintext message. The receiver must decrypt the signature with the sender’s public key. If the plaintext and the decrypted signature are the same, the message is intact and originated from the sender. The problem here is that, the signature doubles the size of the message. Signing long messages will be a waste of recourses. Instead of signing the whole message, the hash of the message must be signed with message originator’s private key (Figure 2.4).
The Smart House signs the digest of the message with the Smart House’s private key, and then sends the message (plaint ext) with the signed digest to the LoS AS.

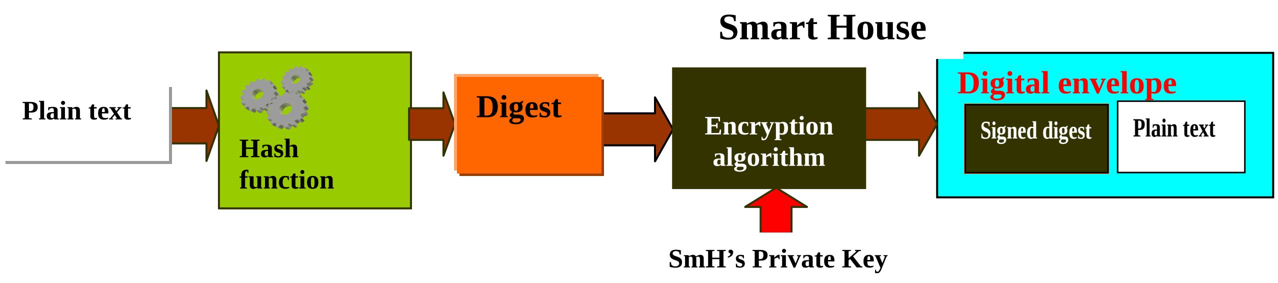


**Figure 2. 4. Protecting data with digital signature**

LoS AS validates the message by checking the signed hash value (digest). LoS AS decrypts the digital signature with Smart House's public key, yielding the hash value (digest). Then the same hash function is applied to the plain text received. The calculated hash value (new digest) must match, the one protected by the digital signature (Figure 2.5). Since Smart House is the only node which has the signed private key, it makes guarantee that the Smart House wrote the message. If one bit of the message changes (plaintext), the value resulting from the hash algorithm will be different. This capability of hashes to detect the smallest change in the message (plain text) is what makes them useful for verifying message integrity.

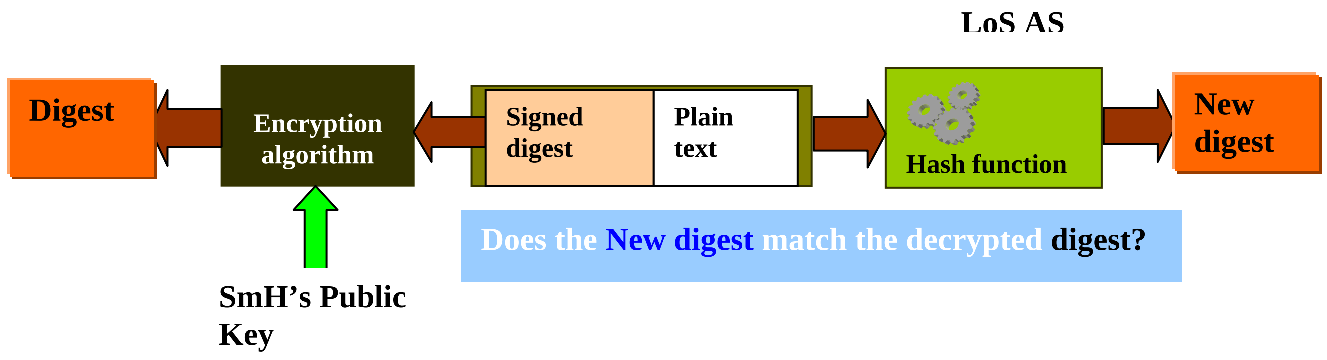


**Figure 2. 5. Validating a message's digital signature**

# 3 PKI Components

## *3.1 Introduction*

This chapter introduces in detail, the main Public Key Infrastructure components, their interaction with each other and the functionality of each of them.

## *3.2 Certification Authorities*

Certificate Authorities are entities that validate identities and issue certificates. They can be either independent third parties or organizations running their own certificate issuing server software. The methods used to validate an identity vary depending on the policies of a given CA just as the methods to validate other forms of identification vary depending on who is issuing the ID and the purpose for which it will be used. In general, before issuing a certificate, the CA must use its published verification procedures for that type of certificate to ensure that an entity requesting a certificate is in fact who it claims to be [1].

## *3.3 Registration Authority*

Interactions between entities identified by certificates (sometimes called end entities) and CAs are an essential part of certificate management. These interactions include operations such as registration for certification, certificate retrieval, certificate renewal, certificate revocation, and key backup and recovery. In general, a CA must be able to authenticate the identities of end entities before responding to the requests. In addition, some requests need to be approved by authorized administrators or managers before being services.

It is clear that, the way used by different CAs to verify an identity before issuing a certificate can vary widely, depending on the organization and the purpose for which the certificate will be used. To provide maximum operational flexibility, interactions with end entities can be separated from the other functions of a CA and handled by a separate service called a Registration Authority (RA). An RA acts as a front end to a CA by receiving end entity requests, authenticating them, and forwarding them to the CA. After receiving a response from the CA, the RA notifies the end entity of the results. RAs can be helpful in scaling a PKI across different departments, geographical areas, or other operational units with varying policies and authentication requirements [1].

## *3.4 Certificate Server*

A certificate authority (CA) server issues, manages, and revokes certificates. The CA's certificate is well known and trusted by all the participating end entities. The CA can delegate its authority to a subordinate authority by issuing a CA certificate, creating a certificate hierarchy. This is done for administration (e.g., different issuance policies) and performance. The ordered sequence of certificates from the last branch to the root is called a certificate chain. Each certificate contains the name of that certificate's issuer, and this is the subject name of the next certificate in the chain. A self-signed certificate means that the signer's public key corresponds to its private key [2].

## *3.5 Certificate Repository*

A certificate repository provides a single point of administration for corporation and user personal attribute information. Entries might include network resources such as file servers, printers, URLs, or people. User information such as E-mail address, telephone, privilege information, and certificates are accessible from a multitude of clients in a controlled fashion.

Directory clients are able to locate entries and attributes about those entries using a directory access protocol.

Lightweight Directory Access Protocol (LDAP) was originally designed to make it possible for applications running on a wide array of platforms to access X.500 directories. LDAP is defined by RFCs 1777 and 1778 as an on the wire bit protocol (similar to HTTP) that runs over TCP/IP. It creates a standard way for applications to request and manage directory information. The directory entries are arranged in a hierarchical tree like structure that reflects political, geographic, and/or corporation boundaries [17].

## *3.6 Key Recovery Server*

A key recovery server allows end entities to backup and recover encryption keys. This is useful for secure recovery of encrypted files or E-mail. This server could also provide key escrow functions. This would allow for session based traffic to be read by government agencies and law enforcement. The key recovery server holds the "keys to the enterprise." Accordingly, it requires the highest level of security protection of all the PKI components. The key recovery server security requirements are summarized in Figure 3.1.

| **A Key Recovery Server's Primary Concerns** |
|---|
| • Enable Strong Authentication for Administrator and Users |
| • Allow Message Privacy and Integrity |
| • Secure Storage of Keys |

**Figure 3. 1. A Key recovery Server's primary concerns**

Cryptographic keys must be stored encrypted and then split. A single master key that unlocks all other keys is not enough. Only a portion of the key should be stored, the end entity or a trusted party must hold the remaining portion. If the key recovery server is comprised, it should not expose all of the organization's keys [21].

## *3.7 Certificate Policy and Certificate Policy Statement*

The Certificate Policy (CP) is high level document that describes a security policy for issuing certificates and maintaining certificate status information. This security policy describes the operation of the CA, as well as the user's responsibilities for the requesting, using, and handling of certificates and keys. The CP asserts that this security policy shall be implemented from certificate generation until its expiration for revocation. It does not specify how the policy shall be implemented. For example, a CP might state: "All subscribers shall be authenticated in person by an RA before a certificate is issued." The CP excludes these details all operational details, since they may evolve over time. The CP would not identify the physical location of the CA or the products used in the CA. By excluding these details, the CP becomes a very stable and high level document. It is reasonable to assume that a CP could be used for 10 years or more.

The scope of a CP may be the operations of a single CA and its supporting components. This is generally the case when a single CA serves an enterprise or a CA participates in a mesh PKI. Since the CA issues the subscriber certificates, and serves as the trust point, a CP covering the operations of the CA is meaningful. Multiple CAs may operate under a single CP. This will often be the case when multiple CAs are maintained by a single enterprise (for example, company or government agency) and jointly support a single community of users through a mesh PKI.

Alternatively, the scope of a CP may be the operations of a hierarchical PKI for this policy. Since the CA that issues the subscriber certificates is different from the trust point, describing the policy of a single CA is insufficient to determine the level of security provided. In this case, the CP must address the operations of the root CA and all the CAs that issue certificates under this policy. Different hierarchical PKIs could share a single CP as well. For example, different health care organizations could implement hierarchical PKIs that implemented the same industry standard policy. This is not a common occurrence today, but could be in the future.

Different people will use the CP for different reasons. For example, the CP will be used to guide the development of the CPS for each CA that operates under its provisions. CAs from other enterprise PKIs will review the CP before cross certification. Application owners will review a CP to determine whether these certificates are appropriate for their application.

The CPS is a highly detailed document that describes how a CA implements a specific CP. The CPS identifies the CP and specifies the mechanisms and procedures that are used to achieve the security policy. The CPS asserts that the specified products will be used in combination with the specified procedures. The CPS might state: “users will receive their certificates and smartcards from the RA after presenting the following credentials in person: Current driver’s license, (2) work identification card, (3) blood sample, and (4) hair sample.” A CPS includes sufficient operational details to demonstrate that the CP can be satisfied by this combination of mechanisms and procedures.

Each CPS applies to a single CA. the CPS may be considered the overall operations manual for the CA. Specific portions of the CPS may be extracted to form the CA Operator’s Guide, RA Manual, PKI Users Guide, or other role specific documentation. Auditors will use the CPS to supplement the CP during their review of CA operations. This may occur periodically as a matter of policy or could be performed whenever cross certification occurs. The combination of a CP and the results of an accredit [9].

# 4 Digital Certificates

## *4.1 Introduction*

A digital certificate is an electronic counterpart for passport. It is an assurance of an identity of the subject and issued by a trusted third party. Oversimplifying, in Public Key Infrastructures, certificates consist of the subject's public key and a digital signature of a trusted third party, who is responsible for reasonably strict checking of the subject's real identity. The trusted third party (TTP) is often called Certification Authority (CA). A reliable signature assures the integrity of the certificate.

For verification of the signature included in the certificate a public key of the CA is needed. Fortunately, the CA's public key can be delivered also in a certificate. The only problem is that how to make sure we have an untouched CA certificate. The answer to the problem is a special kind of certificate called root certificate. Root certificates are guaranteed to be trusted and usually they must be distributed via off line route. For example, in browsers, root certificates are included already in the software package and the user cannot alter them.

In addition to further introduced X.509 and WTLS certificates there are other commonly used certificate types as well. For example, Pretty Good Privacy (PGP) certificates, which are well suited for messaging applications, like email. However these certificate types are not used to offer seamless, protocol level security. PGP certificates are distributed from a friend to another or collecting certificates into public on line repositories forming networks of trusted parties. The “Net of Trust” formed by PGP communities cannot really offer a common. Framework for generic security services because the PGP is too specialized in one application, email. The same applies to other certificate based systems as well. They are too specialized, proprietary or otherwise not widely used. That is why they are not interesting in the context of this work.

## *4.2 Employment of Certificates*

Certificates, public key certificates especially, provide a fundamental building block for secured electronic communication. Underlying cryptographic algorithms can easily be changed to meet respective needs of parties. The encryption algorithm or the digest algorithm used in any particular certification type is not fixed and often the certificate itself can contain information of the used algorithms.

### 4.2.1 Entity Authentication

Authentication is the process of confirming an identity. In the context of network interactions, authentication involves the confident identification of one party by another party. Authentication over networks can take many forms. Certificates are one way of supporting authentication. Network interactions typically take place between a client, such as browser software running on a personal computer, and a server**,** such as the software and hardware used to host a Web site. Client authentication refers to the confident identification of a client by a server (that is, identification of the person assumed to be using the client software). Server authentication refers to the confident identification of a server by a client (that is, identification of the organization assumed to be responsible for the server at a particular network address) [1].

Client and server authentication are not the only forms of authentication that certificates support. For example, the digital signature on an email message, combined with the certificate that identifies the sender, provide strong evidence that the person identified by that certificate did indeed send that message. Similarly, a digital signature on an HTML form, combined with a certificate that identifies the signer, can provide evidence, after the fact, that the person identified by that certificate did agree to the contents of the form. In addition to authentication, the digital signature in both cases ensures a degree of non-repudiation that is, a digital signature makes it difficult for the signer to claim later not to have sent the email or the form.

### 4.2.2 Authorization

Public key certificates bind a public key and an identity, and include additional data fields necessary to clarify this binding, but are not intended for certifying additional information. Sometimes additional information for authorization is needed to be included into certificates directly. These kinds of certificates are called attribute certificates. Attribute certificates are similar to public key certificates but spherically intended to allow specification of information (attributes) other than public keys, such that it may also be conveyed in a trusted manner.

Attribute certificates provide a way to represent authorization. For example a system administrator grants user a certificate that allows him to access specific database information but does not allow him to make modifications to this database. Traditionally, the authorization is handled by maintaining access rights in the system where the users are required to authenticate themselves. When the user logs into the system his access rights are updated according to the access list. These access right lists may be a burden for the system administrators. In attribute certificates the authorization is carried in the certificate itself [4].

Every system has at least implicitly defined policy dictating what is allowed and to whom. These policies can be collected together and grant users attribute certificates that for example define their level of rights to the system. This level of rights (authorization) may consist of, for example, the following levels: basic, exclusive and administrator. For database systems these levels could mean, respectively, read only access for certain tables, access to additional tables and full control. When a basic level user needs access to exclusive level tables, the old certificate is discarded and a new with updated level granted.

### 4.2.3 Data Integrity

Integrity provides assurance that the data has not been modified. This is accomplished by calculating a hash function (e.g., SHA-1) over the content of the message, then encrypting the hash value (message digest) with the sender’s private key. The hash function’s properties are such that any changes to the content will produce a different digest. In addition, it is sufficiently difficult to produce the same hash value from more than one content source. A digital signature is the result of the encrypted portion of the digest. Messages with digital signatures are not reusable, but they are subject to replay attacks [6].

### 4.2.4 Non-repudiation

In cryptography, the term non-repudiation means a service for providing a proof of data integrity and origin so that none of the parties of communication can deny it occurred. In other words, neither parties of the communication can repudiate being in contact with each other, nor can they falsify the data sent during communication. The system has to support third party verification at any given time during or after data exchange. It is then necessary to collect evidences during the communication for later verification. Evidences include information about the parties, their authorization and the data that is exchanged.

There is, however, no need to disclose the data itself to third party. That would not be in line with communication security. Third parties must be trusted, but not to such extent that the secret information should be disclosed to them. To collect evidences Meta data of the communication is enough; signatures, timestamps and other pieces of information [4].

### 4.2.5 Secure Transactions

Secure transaction is actually a combination of all the parameters in characteristics above. Certificates are considered as a key part of securing transactions. There are implementations of seamless security protocol build into user level applications. Probably the most famous of them is SSL (Secure Socket Layer) developed by Netscape Corporation, and TLS which is based on SSL and developed by IETF. There is WTLS protocol for wireless devices, but I will not introduce it in this thesis project.

## 4.3 Certificate Life Cycle Issues

Many standards for managing and delivering certificates have been proposed. The most common of them, X.509 PKI, was originally presented by RSA. Later the standard was adopted and developed by International Telecommunication Union (ITU) [21].

### 4.3.1 Certificate Issuance

Before a certification authority can issue a certificate for a subscriber, the subscriber needs to register the certification authority, typically by completing and submitting a certificate application. Registration involves the establishment of a relationship between the subscriber and CA, and the lodging of certain subscriber information with the CA. When the CA is assured of the identity of the applicant, the certificate can be generated and delivered to the subject.

### 4.3.2 Certificate Update

Normally certificates have a limited validity time, which can vary from few years to few minutes. After the expiration time the certificate must be updated. The way in which this is accomplished depends upon practices of the CA. In case of compromised key, either CA's or subjects, the certification must also be updated.

### 4.3.3 Revocation

At the time of a suspected key compromise or other reasons during the validity period of a certificate the CA by itself can issue a revocation or another authorized party can request revocation from CA. If your private key is disclosed or needs to be changed for any reason, you face a difficult problem. The certificate for the corresponding public key will still look valid even though you may wish otherwise. There is no way to reach out to every computer that may have a copy of your certificate. There’s no way to keep track of such things. While various mechanisms have been designed to deal with this problem, few are effective and the mechanisms have not been widely deployed.

Key revocation is less a problem in secret key systems. Key distribution is so critical you always know who holds what keys, if you need to revoke a particular key; you inform the systems on either end of the connection. The task is more difficult if you have shared the same secret key between several sites. You must change the key for all of them. In any case we can keep the problem under control even though the procedures may be burdensome.

The typical solution in today's public key systems is the certificate revocation list. This is a list of certificates that are no longer valid. CAs frequently publishes certificate revocation lists (CRL), from which the certificate verifier can check if the certificate is revoked [7].

## *4.4 Public Key Certificate Types*

The most widely recognized standard public key certificate format for communication protocol level security is that defined in the X.509 standard. Another certificate format, WTLS certificate, is based on X.509 but designed for wireless communication. In this section those two certificate formats are introduced. There are also plenty of other certificate types, but they are not really so meaningful in this thesis [21].

### 4.4.1 X.509 Certificate

Certificates core is the use of public key identity certificate with each user of a system. The certificates in X.509 are identity based and an ITU-T recommendation. The recommendation specifies the certificate format as well as the role of the CA. The CAs can create hierarchical trust models. All CA's except the root CAs are certified by other CAs. Also cross certification is possible. This enables creating trust relationships between different CA certification trees. The X.509 is based on a X.500 directory. According to the paradigm there is a global directory, where there is an entry for each individual. The certificates are stored in the X.500 directory along with other data about the individuals. According to the original approach, the names of the individuals would be globally unique. In real world deployments of the X.509 the directories normally cover only one organization or the users of some application.

X.509 is de facto standard format for the certificates in Internet. In Figure 4.1, a simplified X.509 public key certificate is illustrated [20]. The X.509 certificate consists of the fields shown in Figure 4.1.
*Version* field describes the version of the encoded certificate. When extensions are used, as expected in this profile, use X.509 version 3 (value is 2). If no extensions are present, but a Unique Identifier is present, use version 2 (value is 1). If only basic fields are present, use version 1 (the value is omitted from the certificate as the default value).

The entity that created the certificate is responsible for assigning it a *serial number* to distinguish it from other certificates it issues. This information is used in numerous ways; for example, when a certificate is revoked its serial number is placed in a Certificate Revocation List (CRL).

*Signature Algorithm ID* identifies the algorithm used by the CA to sign the certificate.

*Issuer Name* follows the X.500 format name of the entity that signed the certificate. This is normally a CA. Using this certificate implies trusting the entity that signed this certificate. Note that in some cases, such as root or top-level CA certificates, the issuer signs its own certificate. This is called a self signed certificate.

Each certificate is valid only for a limited amount of time.
The Validity Period (Valid Not Before, Valid Not After) is described by a start date and time and an end date and time, and can be as short as a few seconds or almost as long as a century. The validity period chosen depends on a number of factors, such as the strength of the private key used to sign the certificate or the amount one is willing to pay for a certificate. This is the expected period that entities can rely on the public value, if the associated private key has not been compromised.

*Subject Name* is a name of the entity whose public key the certificate identifies. This name uses the X.500 standard, so it is intended to be unique across Internet. This is the

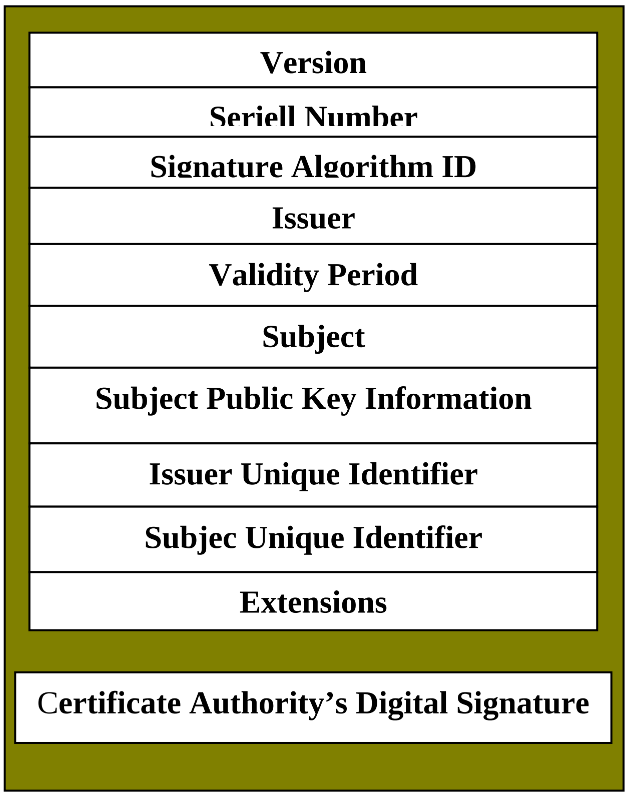
Distinguished Name (DN) of the entity, for example, CN=Ahmed Guleid, OU=IKT, O=Hia, C=NO (These refer to the subject's Common Name, Organizational Unit, Organization, and Country.)

*Subject Public Key Information* is the public key of the entity being named, together with an algorithm identifier which specifies which public key crypto system this key belongs to and any associated key parameters.

Fields *Issuer unique identifier* and *Subject unique identifier* may only appear if the version is 2 or 3. The subject and issuer unique identifiers are present in the certificate to handle the possibility of reuse of subject and/or issuer names over time.

X5.09 certificates in real life usually contain some *Extensions* that are only additions in version 3. Some of the extensions are proprietary serving some specific function.

*Signature* of the above fields using the algorithm identified in Signature Algorithm ID field.



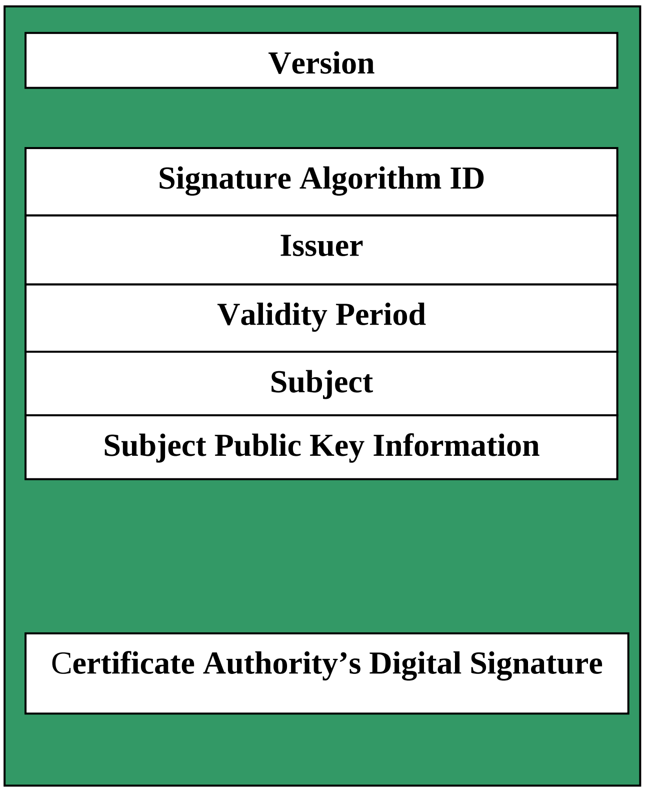

Version
Signature Algorithm ID
Issuer
Validity Period
Subject
Subject Public Key Information
Certificate Authority's Digital Signature

**Figure 4. 1. X.509 certificates**

**Figure 4. 2. WTLS certificates**

## 4.4.2 WTLS Certificate

The WTLS certificate is specified by WAP Forum. Compared to X.509 it is smaller and thus, optimized for wireless communication. The WTLS certificate consists of the fields shown in Figure 4.2.

*Version* of the certificate for the current specification is always 1.

*Signature* Algorithm used to sign the certificate may be any of the supported in WTLS specification.

*Issuer* of the certificate defines who signed the certificate. Certificates are usually signed by Certification Authorities.

*Validity Period (Valid Not Before and Valid Not After)* defines the beginning and the end of the validity period of the certificate.

*Subject* is the owner of the key, associated with the public key being certified.

*Public Key Information* consists of *Public key type* that is the algorithm of the public key and *Parameter Specified* that define parameter relevant to the public key.

*Signature* of the above fields using the algorithm identified in Signature Algorithm ID field [21].

## *4.5 Certificate Chain*

The recursive paradigm of obtaining and verifying certificates leads us to a general model, called the certificate chain. Figure 4.3 illustrates the situation. The most top certificate, the root CA certificate, must be self signed. Under the Root certificates there are other CA certificates that are signed by the root. The user certificates are at the end of the branches. The verification hierarchy under one root CA certification system is called certification tree. Several certifications trees can be cross certified by other root CAs so that they form a forest of hierarchies. In real life there might be number of cross certified root CA's used even inside of one organization.

In commercial products there can be dozens of root certificates preinstalled from different commercial certification authorities.

These root certificates are usually not the ones from the top of the tree, but some certificates under the root. Figure 4.3, illustrates a simplified representation of an ideal case where there is only one trusted root CA certificate in the system and the direction of the certification always goes from top to bottom. If the subject of certificate A needs to verify that the certificate C's public key can be trusted, the verification path will go through six nodes. It is mandatory that A have access to all certificates along the path.

In addition to root CA certificates also non-root CA certificates can be cross Certified as illustrated in Figure 4.3 where CA certificate “Local CA 1.2” is cross certified by “Sub CA 3”. If the B's subject needs to verify C's public key the certification path without the “shortcut” would have to go through seven nodes but with the cross certification only five nodes are involved. From the software point of view, verifying any arbitrary certificate using the chain looks easy. However, in reality, it is the most difficult part of the whole security system based on certificates. Obtaining and verifying missing nodes between the root and leafs are not a trivial task [8].

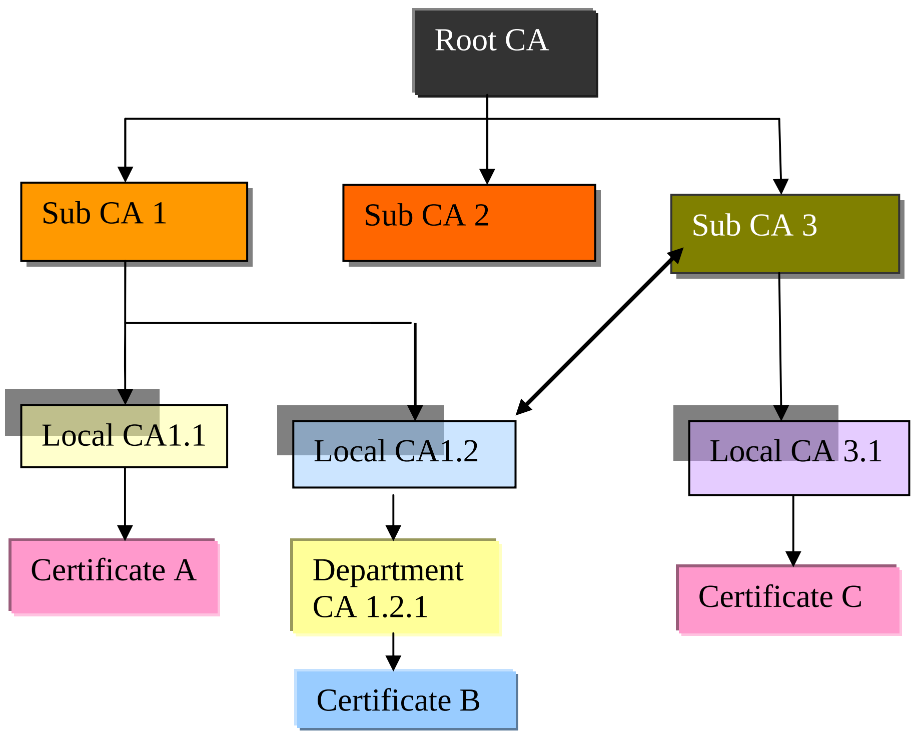


**Figure 4. 3. Certificate chain**

## *4.6 User Authentication*

User authentication is the process by which the identity of a user is verified. As you will see, PKI can be used as part of the authentication process for users, and can also be used to authenticate nonhuman entities such as routers or other network components.

The process of user authentication has traditionally taken many forms, but the one most familiar is the use of an identifying name or user ID, and a password or PIN. The security of a user authentication scheme is generally related to the number of pieces of proof, or factors that are offered during the process of establishing an identity [7].

### 4.6.1 Factors Used for Authentication

Password based authentication is considered a single factor scheme, as the only piece of information required is the user's demonstrated knowledge of a password. The problems with passwords are well understood. These include poor selection of the password, improper construction using too few characters, lack of change control, cost of resetting passwords, and attacks based on social engineering or shoulder surfing. If you solve all of the policy related issues and produce really strong, difficult to crack passwords, the problem becomes that with so many strong (difficult) passwords to remember.

The ease of attacking password schemes is heightened by the use of a single piece of information for the authentication process. Addition of an extra factor or proof leads to **two factor** authentication schemes.

Generally, a two factor authentication scheme requires that the user prove *possession* of some item such as the use of a token of some kind, in addition to *knowing* something like a password. In most two factor authentication systems; you must both have something and know something.

The additional proof generally requires demonstration that the user has possession of the token when the authentication process is being run. SecurID tokens produced by RSA Security use the time at which the authentication process is run combined with a shared secret held in the token and on the authentication server to verify a user's identity.

Other examples include challenge response tokens, in which a one time challenge is sent to the token. A symmetric key within the token is used to encrypt the challenge, enabling the challenger to ascertain that the token was present when the user authenticated. Use of a charge card in an automated teller machine is another example. The card must first be inserted into the ATM to prove that the user has it in his possession before a *personal identification number* (PIN) is entered to show that the user knows the password.

An alternate second factor might be the use of a biometric of some kind. Biometric authentication schemes rely on proving "what you are." In these schemes, some unique identifying feature of the person being identified is used. This might include a user's thumbprint, retinal pattern, voiceprint, or possibly the way in which she signs her signature.

A biometric system may sometimes be combined with a token to provide three factors of authentication. This might include high security environments where a smart card must be used in addition to thumbprint or voice recognition.

### 4.6.2 Authenticating with PKI

PKI can be used to provide authentication to verify the identity of a client when using a protocol like SSL. Some people hold the view that PKI, when used in this fashion, enables the identity of a user to be verified. The use of public/private keys and certificates is considered by some to be equivalent to a two factor authentication scheme.

Despite all of the effort taken to tie public/private keys to an identity, questions still remain to be answered. Are you sure the person using the private key is the same person the CA originally certified? Is the correct individual in the driver's seat making use of those keys and certificates underlying your security services? The answer to these questions is tied to the security of the key store and what authentication mechanism is used to get access to the keys.

Let's look at the way in which current implementations of PKI based authentication typically works. In the case of a browser or other application registering for a certificate, the first time a key is generated, a key store is created. The browser user is prompted to supply a password used to construct the encrypting key to protect the key store. In many applications, there is no policy based control over the type of password selected. Some applications enable a user to completely ignore the password request and create a key store with no password. This is perfectly reasonable choice for ease of use purposes, but what does it mean for authentication using PKI?

In the case where there is weak or nonexistent password protection on the key store, any user with access to the browser has access to the private keys and certificate. If the certificate is used as part of a Web based authentication scheme, the whole process continues to work the same way, but you cannot be sure of the identity of the user driving the browser [2]. In this scenario, how much trust can you place in the expensive identities you have created using PKI?

The fundamental issue here is reliably establishing the identity of the user who is accessing or using a private key when a cryptographic operation is performed. When performing an operation such as generating a digital signature, how can you be confident of the identity of the user accessing the private key? Requirements on the use of digital signatures in some security domains specify that users must validate themselves when the private key is used to generate the signature.

So what is the relationship between authentication of a user and use of PKI in an authentication process such as generating digital signatures? The identity of a user must still be proven when a key store is accessed to perform a cryptographic operation. In this case, two factor schemes like time base tokens or use of a smart card and PIN enable a high level of confidence to be established in the identity of the user.

# 5 Transaction PKI Based protocols relevant to this work

The main and frequently used transaction protocols like SSL and IPsec will be introduced in this chapter. SSL protocol is developed by Netscape to provide a secure communication channel for data in transit. IPsec is a set of extensions to the IP protocol family, which takes care of the security issues at the network level [7].

## *5.1 The Secure Socket Layer Protocol and TLS*

SSL provides security for application to application communications, most commonly communication between a web browser and web server. After email the web is the most widely used Internet service.

Netscape originally published Secure Sockets Layer (SSL) specification. Netscape released to the IETF, and then the IETF made several improvements and the result is the TLS specification. The goal of both, TLS and SSL is to support the PKI security services and provides authentication, integrity, and confidentiality between communication applications [10]. We will concentrate here the original SSL protocol and how it works.

The SSL protocol consists of the following protocols:

1. ***The SSL record protocol***
   This functions as a layer beneath all SSL messages and indicates the encryption integrity protection being applied to the data.
2. ***The SSL handshake protocol***
   This protocol authenticates the server and client, negotiates an encryption algorithm, and establishes cryptographic keys before any application data is transferred.

The SSL record protocol defines the format used to transmit data. The SSL handshake protocol involves using the SSL record protocol to exchange a series of messages between an SSL enabled server and an SSL enabled client when they first establish an SSL connection. This exchange of messages is designed to facilitate the following actions:

- *Authenticate the server to the client.*
- *Allow the client and server to select the cryptographic algorithms, or ciphers, that they both support.*
- *Optionally authenticate the client to the server.*
- *Use public key encryption techniques to generate shared secrets.*
- *Establish an encrypted SSL connection.*

The Transmission Control Protocol/Internet Protocol (TCP/IP) governs the transport and routing of data over the Internet. Other protocols, such as the HyperText Transport Protocol (HTTP), Lightweight Directory Access Protocol (LDAP), or Internet Messaging Access Protocol (IMAP), run "on top of" TCP/IP in the sense that they all use TCP/IP to support typical application tasks such as displaying web pages or running email servers. Figure 5.1, SSL runs above TCP/IP and below high level application protocols

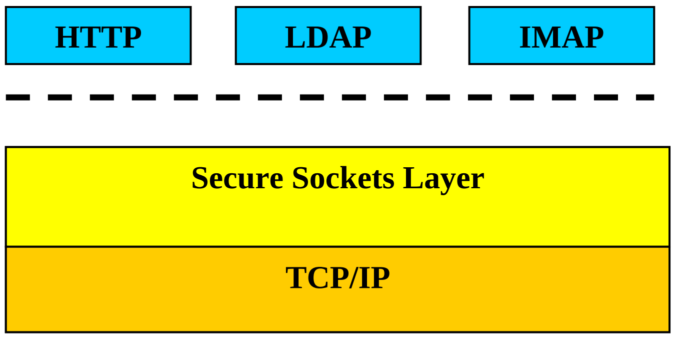


**Figure 5. 1. Secure Sockets Layer Protocol**

The SSL protocol runs above TCP/IP and below higher level protocols such as HTTP or IMAP. It uses TCP/IP on behalf of the higher level protocols, and in the process allows an SSL enabled server to authenticate itself to an SSL enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection. These capabilities address fundamental concerns about communication over the Internet and other TCP/IP networks:

- SSL server authentication allows a user to confirm a server's identity. SSL enabled client software can use standard techniques of public key cryptography to check that a server's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the client's list of trusted CAs. This confirmation might be important if the user, for example, is sending a credit card number over the network and wants to check the receiving server's identity.

- SSL client authentication allows a server to confirm a user's identity. Using the same techniques as those used for server authentication, SSL enabled server software can check that a client's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the server's list of trusted CAs. This confirmation might be important if the server, for example, is a bank sending confidential financial information to a customer and wants to check the recipient's identity.

- An encrypted SSL connection requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality. Confidentiality is important for both parties to any private transaction. In addition, all data sent over an encrypted SSL connection is protected with a mechanism for detecting tampering that is, for automatically determining whether the data has been altered in transit

The SSL 2.0 and SSL 3.0 protocols support overlapping sets of cipher suites [20]. Administrators can enable or disable any of the supported cipher suites for both clients and servers. When a particular client and server exchange information during the SSL handshake, they identify the strongest enabled cipher suites they have in common and use those for the SSL session. Decisions about which cipher suites a particular organization decides to enable depend on trade offs among the sensitivity of the data involved, the speed of the cipher, and the applicability of export rules. Some organizations may want to disable the weaker ciphers to prevent SSL connections with weaker encryption. However, due to U.S. government restrictions on products that support anything stronger than 40 bit encryption, disabling support for all 40-bit ciphers effectively restricts access to network browsers that are available only in the United States (unless the server involved has a special Global Server ID that permits the international client to "step up" to stronger encryption).

To serve the largest possible range of users, its administrators may wish to enable as broad a range of SSL cipher suites as possible. That way, when a U.S. located client or server is dealing with another U.S. located server or client, respectively, it will negotiate the use of the strongest ciphers available. And when a U.S. located client or server is dealing with an international server or client, it will negotiate the use of those ciphers that are permitted under U.S. export regulations. However, since 40 bit ciphers can be broken relatively quickly, administrators who are concerned about eavesdropping and whose user communities can legally use stronger ciphers should disable the 40 bit ciphers.

### 5.1.1 The Secure Socket Layer Handshake protocol

The SSL protocol uses a combination of public key and symmetric key encryption. Symmetric key encryption is much faster than public key encryption, but public key encryption provides better authentication techniques. An SSL session always begins with an exchange of messages called the SSL handshake [7]. The handshake allows the server to authenticate itself to the client using public key techniques, then allows the client and the server to cooperate in the creation of symmetric keys used for rapid encryption, decryption, and tamper detection during the session that follows. Optionally, the handshake also allows the client to authenticate itself to the server. The steps involved can be summarized as follows see Figure 5.2:

1. The client sends the server the client's SSL version number, cipher settings, randomly generated data, and other information the server needs to communicate with the client using SSL.
2. The server sends the client the server's SSL version number, cipher settings, randomly generated data, and other information the client needs to communicate with the server over SSL. The server also sends its own certificate and, if the client is requesting a server resource that requires client authentication, requests the client's certificate. If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client goes on to next step.
3. The client uses some of the information sent by the server to authenticate the server. If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client goes to next step.
4. Using all data generated in the handshake so far, the client (with the cooperation of the server, depending on the cipher being used) creates the premaster secret for the session, encrypts it with the server's public key (obtained from the server's certificate), and sends the encrypted premaster secret to the server.
5. If the server has requested client authentication (an optional step in the handshake), the client also signs another piece of data that is unique to this handshake and known by both the client and server. In this case the client sends both the signed data and the client's own certificate to the server along with the encrypted premaster secret.
6. If the server has requested client authentication, the server attempts to authenticate the client. If the client cannot be authenticated, the session is terminated. If the client can be successfully authenticated, the server uses its private key to decrypt the premaster

secret, then performs a series of steps (which the client also performs, starting from the same premaster secret) to generate the master secret.

7. Both the client and the server use the master secret to generate the session keys, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity that is, to detect any changes in the data between the time it was sent and the time it is received over the SSL connection.
8. The client sends a message to the server informing it that future messages from the client will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the client portion of the handshake is finished.
9. The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the server portion of the handshake is finished.
10. The SSL handshake is now complete, and the SSL session has begun. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

Before continuing with the session, almost all servers can be configured to check that the client's certificate is present in the user's entry in an LDAP directory. This configuration option provides one way of ensuring that the client's certificate has not been revoked. It's important to note that both client and server authentication involves encrypting some piece of data with one key of a public private key pair and decrypting it with the other key:

In the case of server authentication, the client encrypts the premaster secret with the server's public key. Only the corresponding private key can correctly decrypt the secret, so the client has some assurance that the identity associated with the public key is in fact the server with which the client is connected. Otherwise, the server cannot decrypt the premaster secret and cannot generate the symmetric keys required for the session, and the session will be terminated.

In the case of client authentication, the client encrypts some random data with the client's private key that is, it creates a digital signature. The public key in the client's certificate can correctly validate the digital signature only if the corresponding private key was used. Otherwise, the server cannot validate the digital signature and the session is terminated.

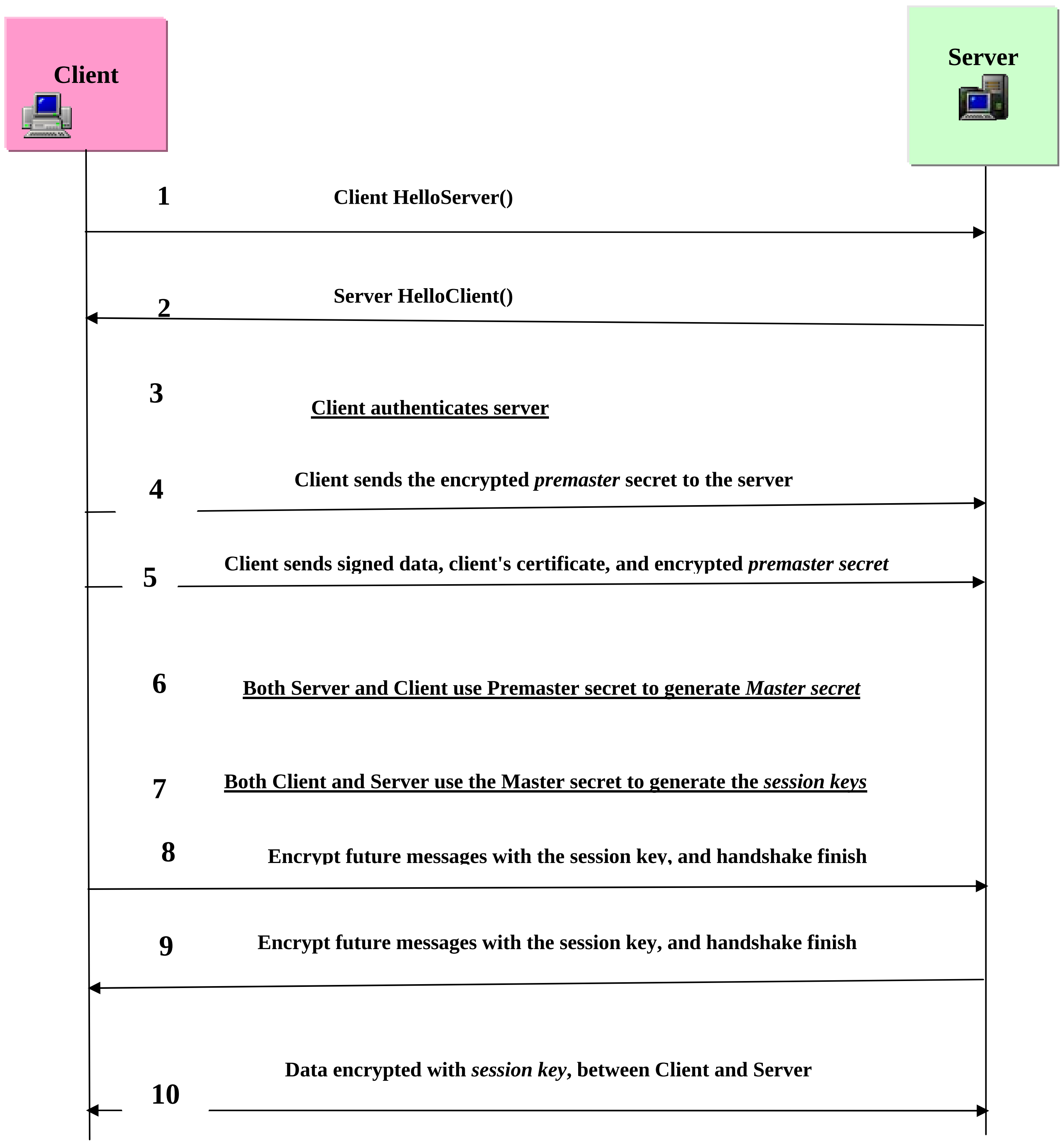

Client
Server
1
Client HelloServer()
2
Server HelloClient()
3
Client authenticates server
4
Client sends the encrypted premaster secret to the server
5
Client sends signed data, client's certificate, and encrypted premaster secret
6
Both Server and Client use Premaster secret to generate Master secret
7
Both Client and Server use the Master secret to generate the session keys
8
Encrypt future messages with the session key, and handshake finish
9
Encrypt future messages with the session key, and handshake finish
10
Data encrypted with session key, between Client and Server

**Figure 5. 2. How handshake protocol works**

## *5.2 IPsec Protocol*

IPsec defines a secure framework and a set of security services for network level communications, parts of which employ PKI.
IPsec is an extention to the existing IP networking protocol to product IP packets from snooping or modification. By operating at the network level, IPsec protections do not interfere with existing application software or protocols, and packets protected by IPsec can be handled by existing routers and routing hosts. IPsec can be used with both IPv4 and IPv6 environments [8].

IPsec is designed to provide privacy, forgery detection, or both for IP packets. IPsec defines two optional packet headers, one for each type of protection. The headers contain both a numerical value called the Security Parameter Index (SPI). Whenever a host processes an IPsec header it uses the SPI to identify the crypto keys and procedures to use with it. A packet may contain one or both headers, depending on which security services are needed. In practice most packets use both. It operates in one of two modes: tunnel mode or transport mode. In tunnel mode, the entire IP packet is encrypted and becomes the data portion of a new, larger IP packet that has a new IP header and an IPsec header added (see Figure 5.3).

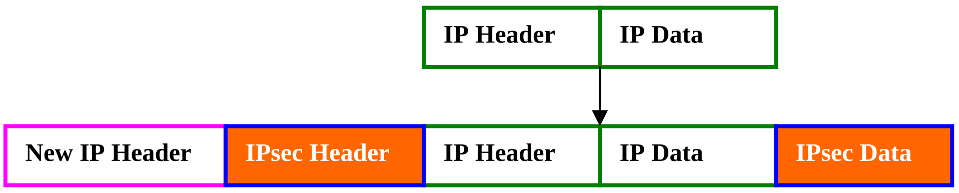


**Figure 5. 3. IPsec tunnel mode IP packet**

If the IPsec Encapsulating Security Payload (ESP) service is used, the packet will also have an IPsec data trailer. In transport mode, the IPsec header is inserted directly into the IP packet (see Figure 5.4).

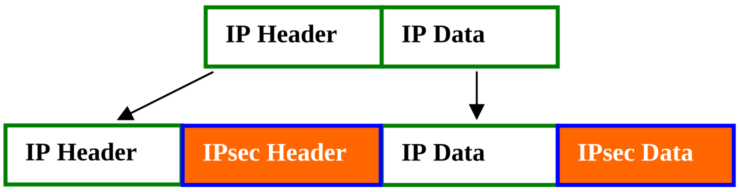


**Figure 5. 4. IPsec transport mode IP packet**

Tunnel mode is primarily used by gateways and proxies (see Figure 5.5).

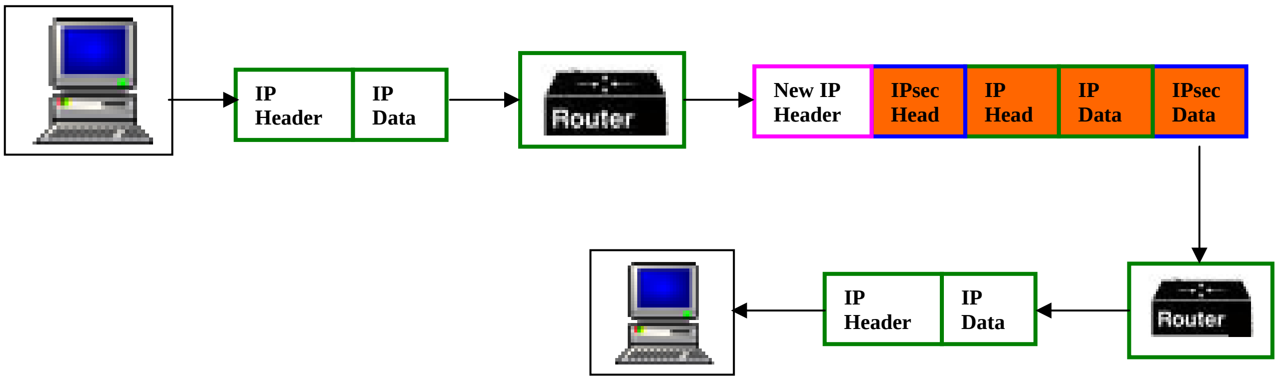


**Figure 5. 5. IPsec tunnel mode**

The intermediate systems implement the IPsec services; the end points do not know about the IPsec. In transport mode the end points must both implement the IPsec services, the intermediate do not perform any IPsec processing on the packet (see Figure 5.6).

IPsec provides strong security and great flexibility. As a result it is fairly complex to understand

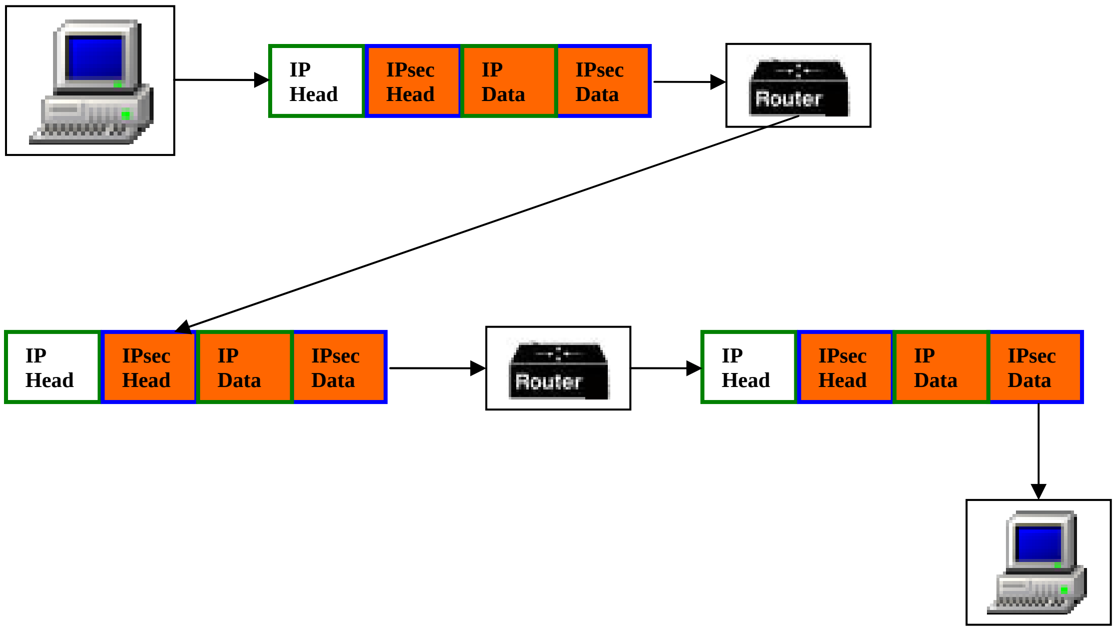


**Figure 5. 6. IPsec transport mode**

The IPsec Authentication header provides integrity, data origin authentication, and anti replay services. The AH integrity uses *Integrity Check Value* (ICV) that is computed over the entire packet except for the header field values that may change during the transmission (For example time to live). The ICV can be hash value, a keyed message authentication code (such as HMAC), or digital signature. The ICV algorithm is specified in the IPsec SA. In general a simple or keyed hash is used for point to point communications. Data origin authentication is done through verification of a keyed HMAC computed with a shared secret key or a digital signature. Replay prevention is based on monotonically increased sequence number. The sequence number is not allowed to “wrap”; thus when a counter reaches its maximum value, it cannot cycle back to zero. IPsec mandates that a new SA must be created if a counter reaches its limit. The new SA will have a fresh counter and a new encryption key [1].

The Encapsulating Security Payload (ESP) protocol provides several features that are similar to AH, including integrity via an ICV, data origin authentication with keyed MACs, and replay protection through sequence numbers. ESP packets are formatted slightly differently from AH packets with ESP, the ICV is added to the end of the IPsec packet, whereas AH places the ICV in the IP header. The biggest difference between the two, though, is ESP’s confidentiality service: ESP provides confidentiality through encryption. The encryption algorithm is negotiated in the SA.
.

# 6 How to apply PKI to the Smart House and the other vendors

This chapter summarizes the results of my work, through the introduction of a different PKI security services. The three first scenarios (scenario 1, 2 and 3), will be referred as the main PKI scenarios throughout the rest of this work. The main PKI scenarios that I will introduce use both symmetric and asymmetric cryptography to provide any security services (PAIN MODEL) users are likely to need.

## *6.1 Scenario 1: Messages using Symmetric cryptography*

Assume the Fire brigade wishes to send message to the Smart House using symmetric key. The Smart House must have the same symmetric key to decrypt the message. Refer to Figure 6.1.

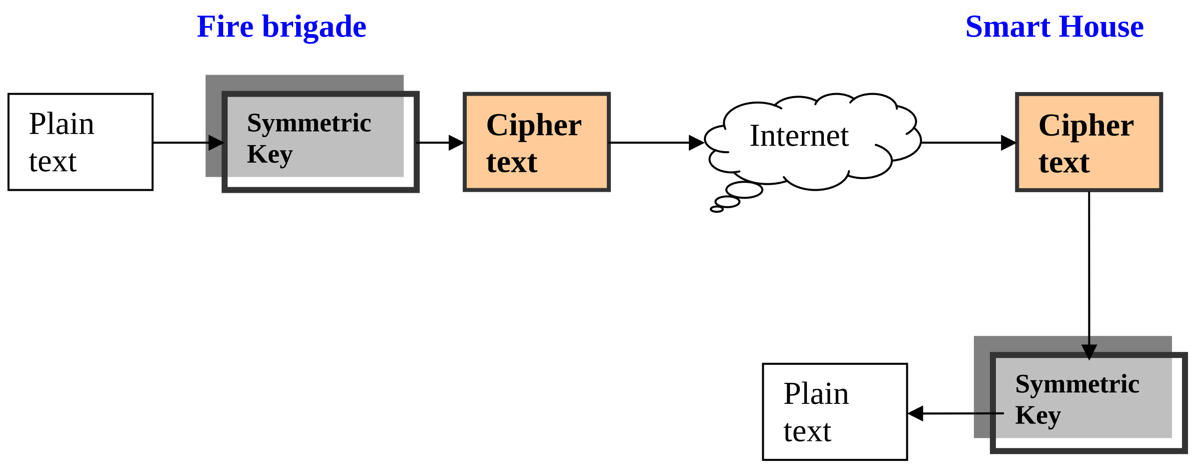


**Figure 6. 1. Encrypting and decrypting with the same key.**

The Fire brigade and Smart House agree on a symmetric algorithm in advance. The Fire brigade will then create a random number of the correct length to use as the symmetric key. Using that key, the symmetric algorithm will encrypt the clear text message and produce the encrypted cipher text message. Fire brigade will then send the Smart House the cipher text through the Internet, and even if the hacker intercepts it, the hacker will not have the symmetric key to decrypt the data. When the cipher text message is delivered to Smart House, the same symmetric key is used to decrypt the cipher text message and recover the original clear text message. Symmetric encryption has been around for quite some time, and the algorithms, which have survived the test of time, are quite secure. Symmetric algorithms are also quite fast, so the encryption of large amounts of data can proceed at a rapid pace without significant impact to the processor load. As an additional benefit, the cipher text produced from a symmetric encryption is compact, usually about the same size as the original clear text message.

The problem with symmetric cryptography is that, how could we deliver the symmetric key to the other party? Look at Figure 6.1. The hacker must not get a copy of the key used to encrypt the message. But the Fire brigade must get a copy of the key to the Smart House so that the Smart House can perform the decryption. If the Fire brigade sends the Smart House the key over the Internet, the hacker will get a copy. Key delivering is the main problem for symmetric algorithm. Once you use symmetric key, it should be discarded and a new random key generated. This is because it is inadvisable for a symmetric key to be used repeatedly. Each time the symmetric key is reused; more data is generated, which can be used to attack the security of the symmetric key. In the next scenario we will explore that whether the asymmetric crypto will help about our symmetric key exchange problem.

## *6.2 Scenario 2: Messages between the Fire brigade and the Smart House*

In this scenario, we take a look at what PKI security services can offer to the messages that takes place between the Fire brigade and the Smart House. In the second part of this scenario, we will discuss an example that fulfils the requirements needed for the messages between the Fire brigade and the Smart House. The Fire brigade sends these messages to the Smart House across the Internet as clear text messages. It can be seen or read by the hacker, but we have to know if the hacker tries to modify or substitutes it. In the first part of this scenario will mostly be the introduction of asymmetric cryptographic and how we could use the combination of both symmetric and asymmetric cryptography to secure the communication between two communicating end entities.

In the previous scenario we have seen that, the symmetric algorithms are fast, secure, compact, and the encryption of a large amount of data can proceed at rapid pace without significant problem with the processor load. However we had key exchange problems with the symmetric keys.

The public key cryptography will help solve our key exchange problem. Let us see how this would work to help the Smart House with its problem (key exchange). See Figure 6.2.

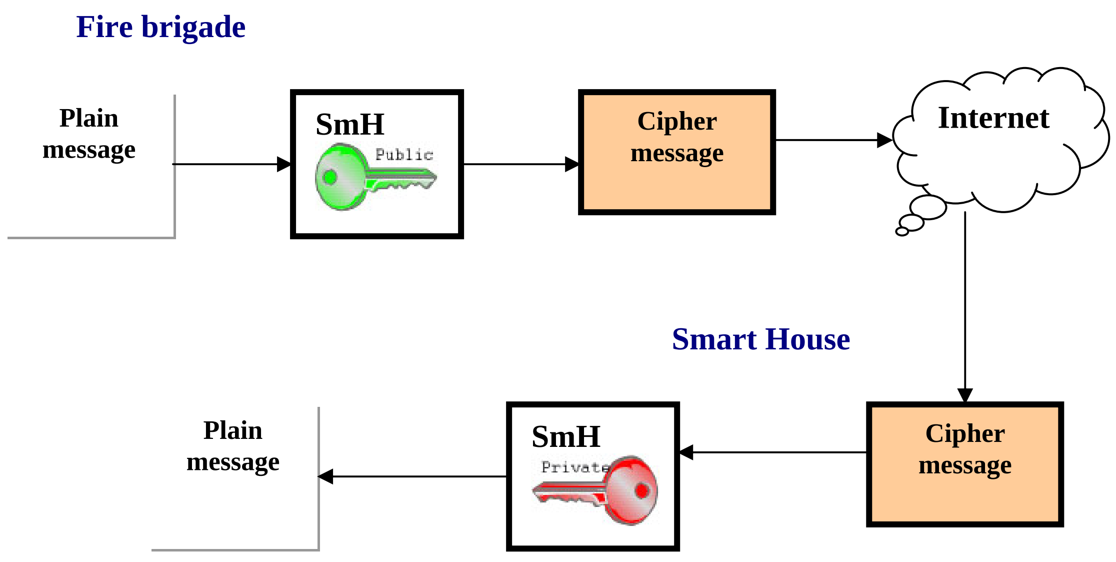


**Figure 6. 2 Encrypted and decrypted with SmH's public/private key.**

Since the Smart House is the recipient, the Smart House's browser would generate its public/private key pair in advance. The Smart House would then carefully protect its private key so nobody else in the world knew it. The Smart House would, however, make Smart House's public key available for everyone to use. It is like the phone directory for public keys, where everyone would go to look up someone else's public key.

The Fire brigade would look up Smart House's public key in that directory, and use Smart House's public key to produce the original cipher text. That cipher text would then be sent to the Smart House across the internet. Note that if the hacker hiding in the Internet were to intercept the cipher text, he could also look up Smart House's public key in the directory. But the only thing that can decrypt a cipher text created with a particular public key is the matching private key. The hacker would not be able to decrypt the cipher text with the public key.

When the cipher text finally delivered to the Smart House, the Smart House's private key would be used to decrypt the cipher text and recover the original plain text.

This would actually work. The Smart House finally has a way to get messages from the Fire brigade.

There are still some issues, which must be discussed. If the hacker wanted to be really disgusting, he could take some other plain text encrypt it using the Smart House's public key, and send it to the Smart House instead of Fire brigade's original message. The fact that Smart House's public key is public makes the Smart House vulnerable to the man in the middle attack, see next scenario. We need some way to authenticate that the message came from the Fire brigade and not modified or substituted by the hacker.

The combination of symmetric and asymmetric meets every one of the PKI security requirements. Therefore the PKI could help us authenticate the source of data origin [4].

The generation of a random symmetric key by the Fire brigade is where the process begins at the first time. The symmetric key is used to encrypt the message, producing the encrypted version of the message. The symmetric cipher is secure and fast, and the resulting cipher text is compact.

Now, the problem we had before with symmetric encryption was how to get the symmetric key to the Smart House. This is where public/private key crypto comes in.

We look up the Smart House's public key in the directory (see the Figure 6.3), and use it to encrypt the created random symmetric key.

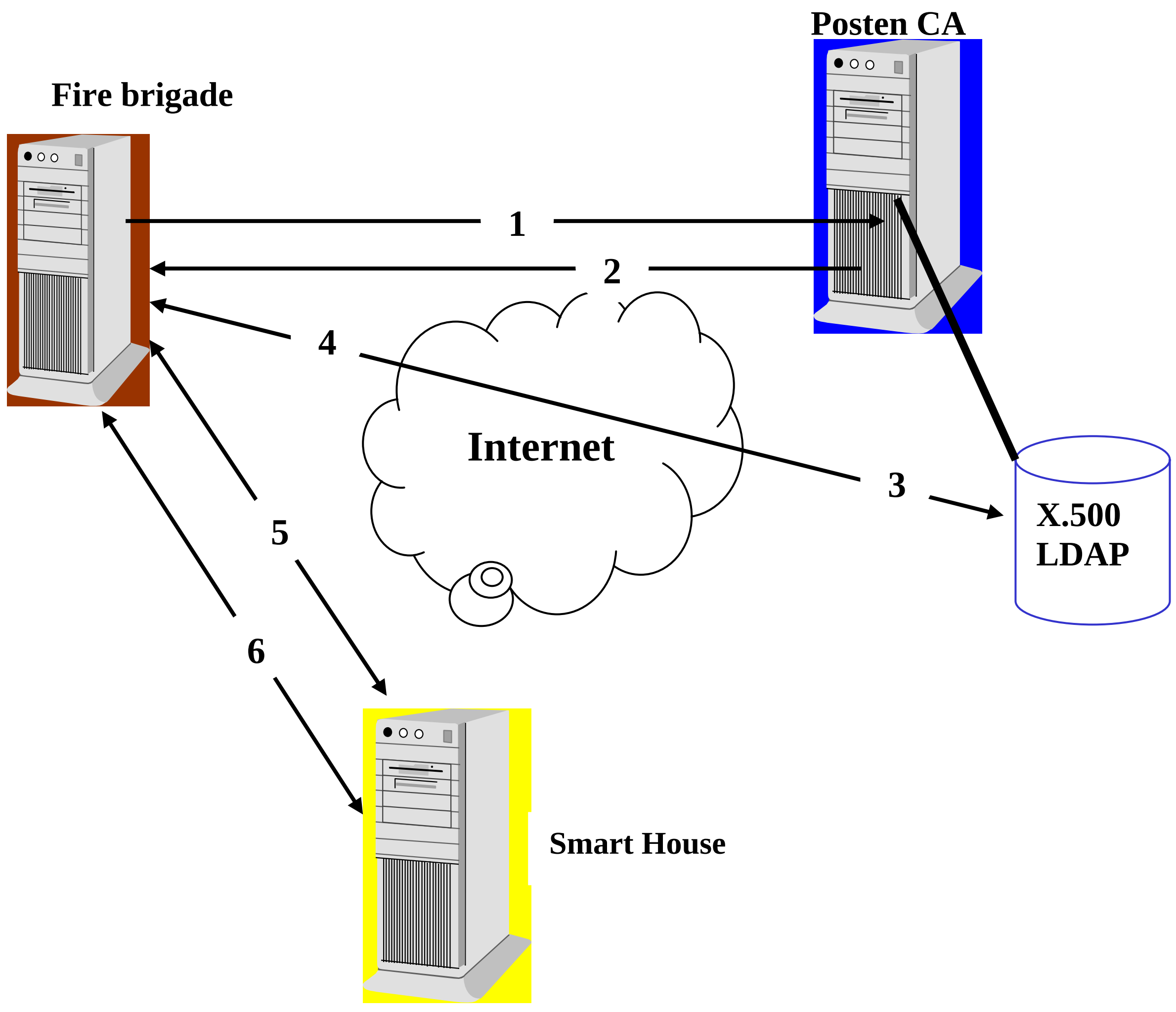


**Figure 6. 3. Looking up the Smart House's public key**

In this example, take a look at Figure 6.3. This Figure describes a simplified client certificate request and its response from the CA. The main aim of this figure is to show how the Fire brigade could retrieve a public key from a trusted third party's (Posten's) directory.

1. *The Fire brigade sends a certificate request to the Posten's web server which is a certificate authority (CA) directly or through a Registration Authority (RA).*
2. *The Posten creates the certificate, and sends the response to the Fire brigade directly or through a Registration Authority (RA). Now we assume that the two communicating end entities (the Fire brigade and the Smart House), both trust the Posten.*
3. *The Fire brigade looks up the Smart House's public key in the directory (X.500), to encrypt the symmetric key.*
4. *At this point we get the Smart House's public key from the directory.*
5. *A simplified handshake between the Fire brigade's server and the Smart House's browser takes place here.*

6. *The encrypted conversation between the Fire brigade's server and the Smart House's browser takes place.*

The Fire brigade uses the Smart House's public key to encrypt the random symmetric key, which it generated. The key encrypted with another key is called key wrapping operation.
The last step in this process is to attach the wrapped key to the cipher text in preparation for sending to the Smart House. The combination sometimes is referred as the digital envelope (see Figure 6.4). The digital envelope is sent to the Smart House across the Internet. If a hacker intercepts the digital envelope, it is of no use to him. It may even frustrate him that the symmetric encryption key he needs in order to get at the message is actually inside the digital envelope, but it is encrypted with the Smart House's public key and so remains not reachable.

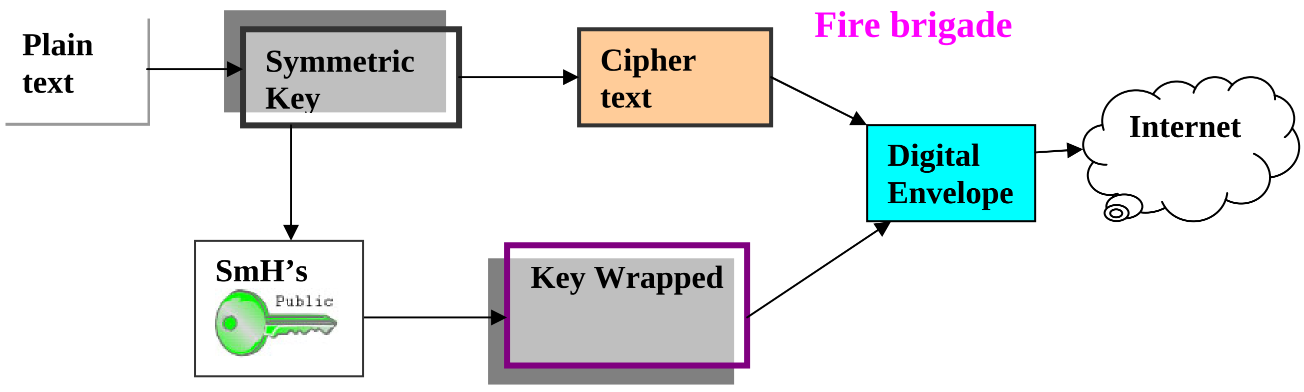


**Figure 6. 4. The digital envelope is sent to the Smart House.**

The use of public/private key encryption to wrap the symmetric key gives the solution scalability, protects against interception of the symmetric encryption key, does not require a prior relationship between the involved parties, and supports digital signature and none repudiation. Let's move at the Smart House's side of this process [8].

At the recipient the process starts with the reception of the digital envelope. The first step is to decompose the digital envelope into its constituent parties (see Figure 6.5), the cipher text and the wrapped key. We know that, the wrapped key is the symmetric key encrypted with the Smart House's public key. Since the Smart House cannot access the cipher text yet, the next step is to recover the symmetric key.  The wrapped key is decrypted using the Smart House's private key. The symmetric key is then used to decrypt the cipher text, recovering the original plain text. The symmetric key is then discarded; it is only one time use key.

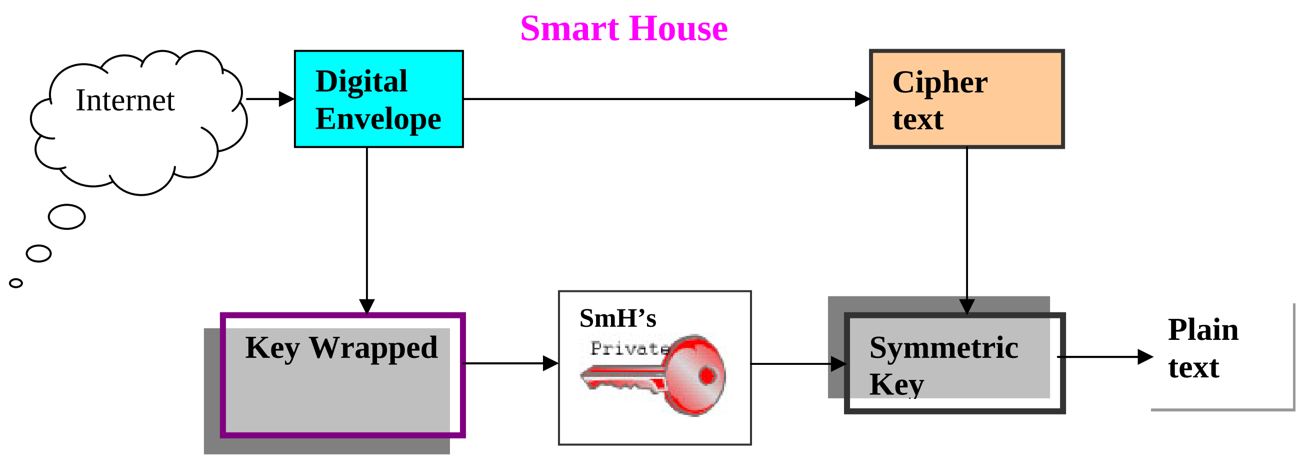


**Figure 6. 5. The Smart House receives the digital envelope**

A clever hacker could look up the Smart House's public key in the directory, and then encrypt some other message, use the Smart House's public key to encrypt the symmetric key, create the digital envelope and send it to the Smart House. The Smart House will hash the clear message, and decrypt the wrapped key using the Smart House's private key. Everything will look like normal, which means that the message came from the Fire brigade, while in reality, it came from the hacker. We need some way to guarantee that the message came from the Fire brigade and not from someone else; what we need here is digital signatures.

The following example fulfils the PKI security requirements needed for the messages between the Fire brigade and the Smart House. In this example, we would like to change the problem a little. Let's assume for this exercise that it is okay if anyone can see this message (message without privacy). In this case, it can be sent across the Internet as clear text. Allowing the message to remain in the clear text will simplify thing and allow us to focus on the digital signature process.

However we want to be sure the message is coming from the Fire brigade. We would like to prevent the problem discussed previously where a hacker attempts send the Smart House a different message.

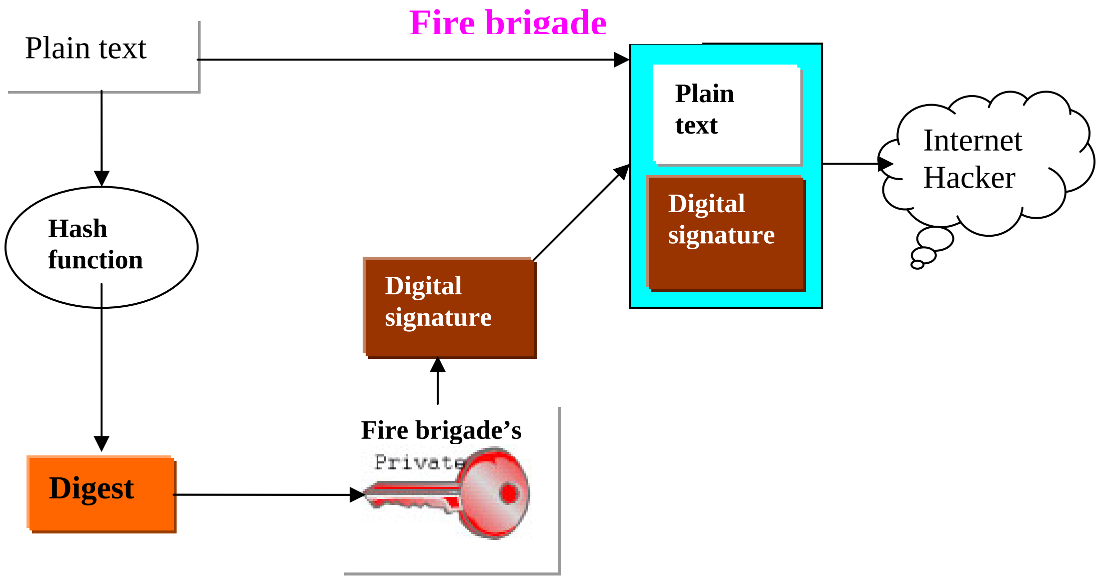


**Figure 6. 6. The Fire brigade sends digital signature to the Smart House.**

We start the process with the plain text (see Figure 6.6). At this point an appropriate hash algorithm is selected, and it is used to process the plain text producing the digest. Next, the Fire brigade's private key used to encrypt the digest creating the digital signature. The encrypted digest (the digital signature) is attached to the original plain text and sent to the Smart House across the Internet. One may wonder to see how this process gets us anywhere. It will all make sense when we see the digital signature verification process, so let's proceed (see Figure 6.7).

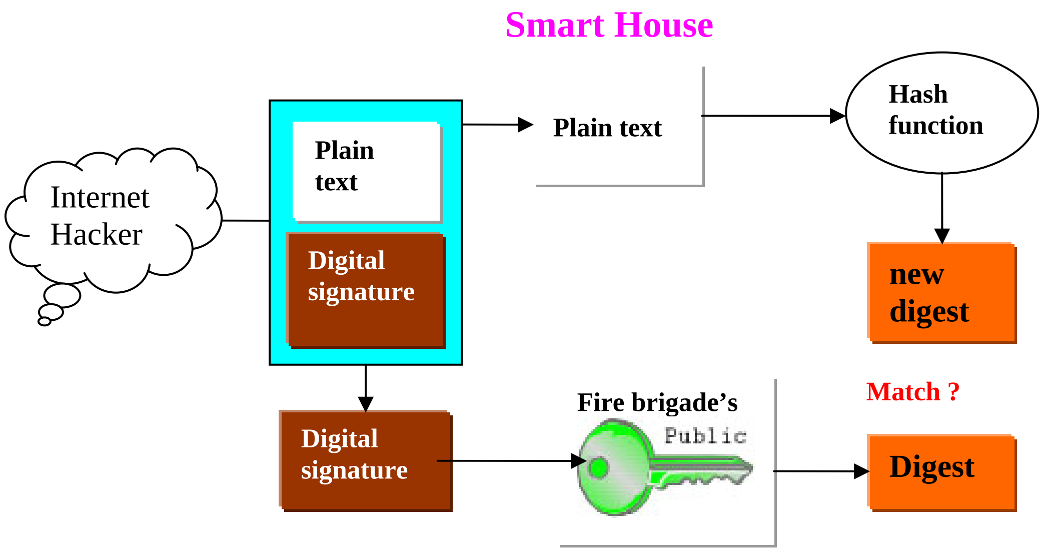


**Figure 6. 7. The Smart House verifies the Fire brigade's digital signature**

The plain text with the encrypted digest attached travels across the Internet to the Smart House. The Smart House's software would separate the clear message from the encrypted digest. The Fire brigade used its private key to encrypt the original digest. The Smart House will get its matching public key from the directory. Using the Fire brigade's public key, the Smart House will decrypt the digest, recovering the original digest.

Here comes the clever part. The Smart House now turns to the clear message. Using the same hash algorithm that Fire brigade used to create the original digest, the Smart House will take the received clear message and create a new copy of the digest, shown as (new digest) in Figure 6.7. As last step, the Smart House will compare the newly created (new digest), with the original digest the Smart House decrypted using Fire brigade's public key. If the two copies match, the Smart House knows for sure that the Fire brigade has sent the message, because it used the Fire brigade's public key. The Smart House also knows that the message was not modified in transit, because the new calculated hash and the original hash are equal.
There is one way to break this system that needs to be addressed. Let's assume that the hacker is particularly clever and decides to attack the system in another special way. Rather than attempt to find away to defeat the hash algorithm, or perhaps defeat the private key encryption, our clever hacker might turn to his attention elsewhere.
We know that, to verify a signature goes something like this:

1. *The Smart House looked up Fire brigade's public key in the directory.*
2. *The Smart House used that key to decrypt the encrypted digest.*
3. *The encrypted digest was created using Fire brigade's private key.*
4. *The Fire brigade has the only copy of the private key in existence.*
5. *Therefore, if the decrypted digest and the calculated digest (new digest) match, the message must have come from the Fire brigade.*

If the hacker is clever and able to reach into the directory and substitute his public key for the Fire brigade's public key, the whole process collapses.
Once the hacker substitutes his public key under Fire brigade's name in the directory, he can start with his own message, create digest, encrypt it with his private key, and then send the message with the encrypted digest to the Smart House. The Smart House will extract the hacker's public key from the directory under Fire brigade's name. Assuming that the Smart House has the correct key, the Smart House will proceed to perform the signature verification. In this case, new digest will match the decrypted digest and the Smart House will think it has got the correct key and every thing is under control. Because of the Smart House has cryptographically strong assurance it will believe that the message came from the Fire brigade!

We need some way to make sure that a particular public key belongs to a particular person or node. This is where digital certificates come in. See the digital certificates in chapter 3. Now we need to use digital certificates instead of naked public key in the directory, to be a bit more accurate (see Figure 6.8), we can redraw the digital signature verification process and update the Figure 6.7.

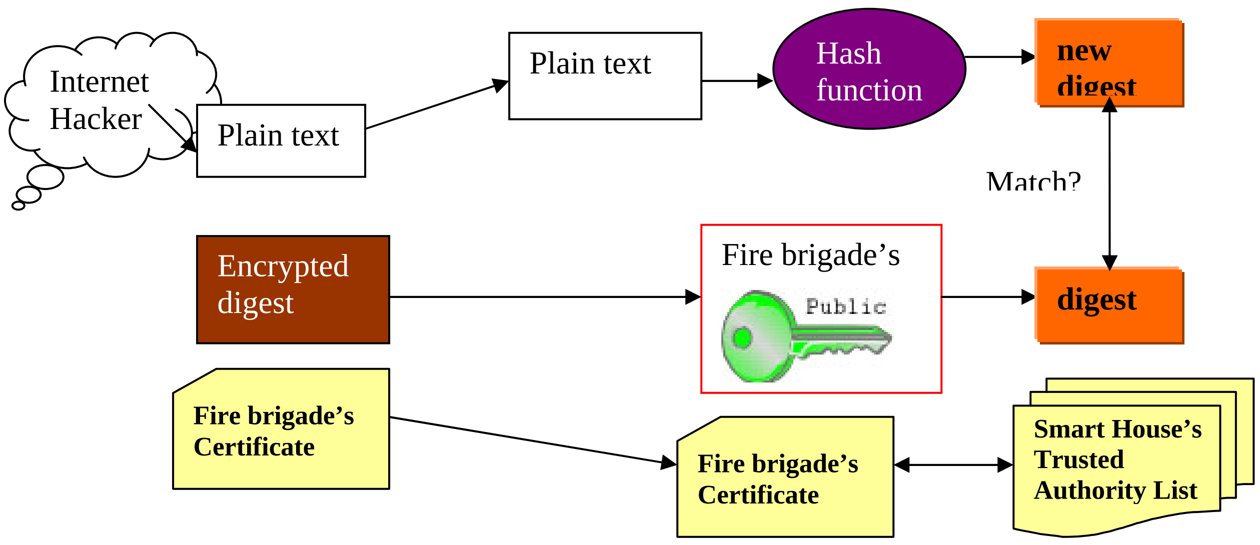


**Figure 6. 8. The Smart House really verifies the digital signature of the Fire brigade**

In reality the Fire brigade will include a copy of the digital certificate along with encrypted hash and the clear message. The first step is to separate the three components. To authenticate the binding between a public key and the server identified by the certificate that contains the public key, an SSL enabled Smart House browser must receive a "yes" answer to the four questions shown in Figure 6.9. Although the fourth question is not technically part of the SSL protocol, it is the Smart House's responsibility to support this requirement, which provides some assurance of the Fire brigade's identity and thus helps protect against a form of security attack known as "man in the middle [6]."

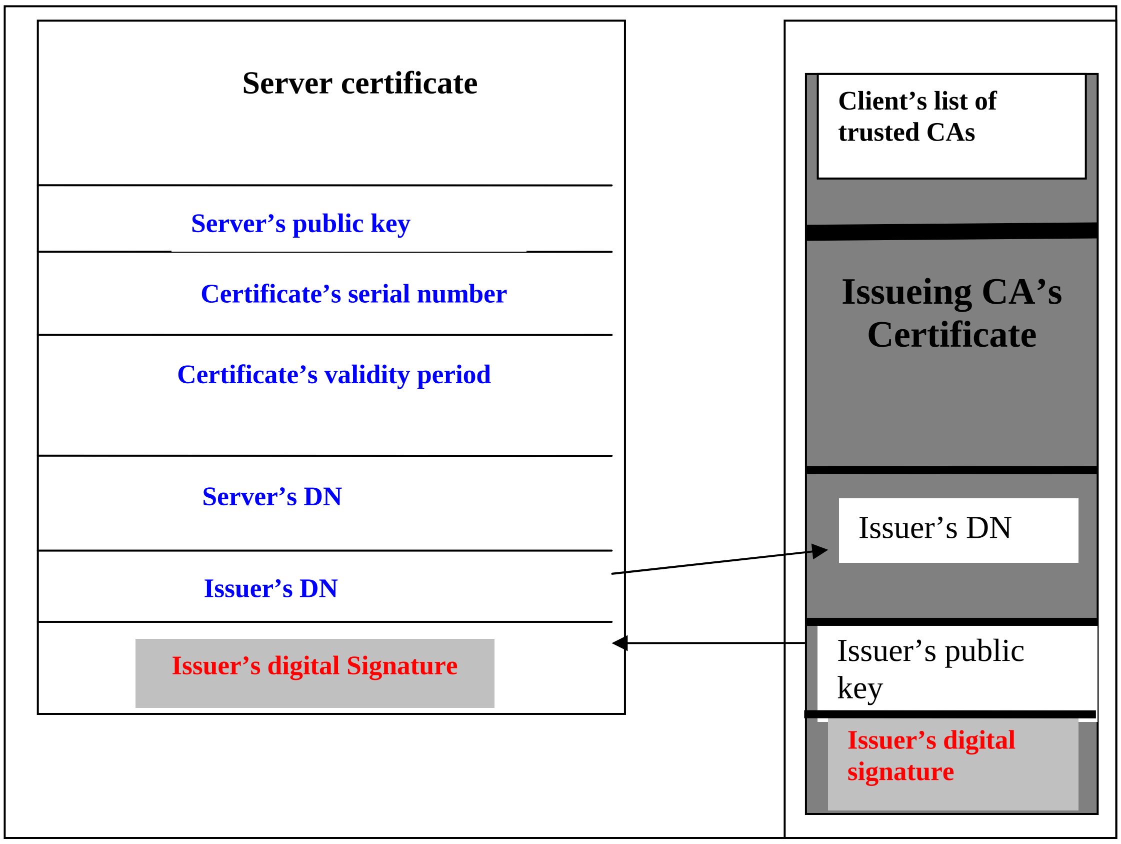

Server certificate
Server's public key
Certificate's serial number
Certificate's validity period
Server's DN
Issuer's DN
Issuer's digital Signature
Client's list of trusted CAs
Issueing CA's Certificate
Issuer's DN
Issuer's public key
Issuer's digital signature

**Figure 6. 9. Shows how a Client authenticates a Server certificate**

An SSL enabled Smart House goes through these steps to authenticate the Fire brigade's identity:

1. *Is the issuing CA a trusted CA? The SSL enabled Smart House maintains a list of trusted CA certificates, represented by the shaded area on the right side of Figure 6.9. This list determines which server certificates the Smart House will accept. If the distinguished name (DN) of the issuing CA matches the DN of a CA (Posten) on the Smart House 's list of trusted CAs, the answer to this question is yes, and the Smart House goes on to next step. If the issuing CA is not on the list, the server will not be authenticated unless the Smart House can verify a certificate chain ending in a CA (Posten) that is on the list.*
2. *Does the CA's public key validate the issuer's digital signature? The Smart House uses the public key from the Posten's certificate (which it found in its list of trusted CAs in step 2) to validate the CA's digital signature on the server's certificate being presented. If the information in the server's certificate has changed since it was signed by the CA or if the Posten certificate's public key doesn't correspond to the private key used by the CA to sign the server's certificate, the Smart House won't authenticate the server's identity. If the CA's digital signature can be validated, then it can continue to the next step. At this point, the Smart House has determined that the Fire brigade's certificate is valid.*

3. *Is today's date within the validity period? The Smart House checks the Fire brigade certificate's validity period. If the current date and time are outside of that range, the authentication process won't go any further. If the current date and time are within the certificate's validity period, the Smart House goes on to next step. It is the Smart House's responsibility to take step 4 before step 5.*
4. *Does the domain name in the server's certificate match the domain name of the Fire brigade? This step confirms that the Fire brigade's server is actually located at the same network address specified by the domain name in the server certificate. Although step 4 is not technically part of the SSL protocol, it provides the only protection against a form of security attack known as a man in the middle attack. The Smart House must perform this step and must refuse to authenticate the server or establish a connection if the domain names don't match. If the Fire brigade's actual domain name matches the domain name in the server certificate, the Smart House goes on to next step.*
5. *The Fire brigade is authenticated. The Smart House proceeds with the SSL handshake the handshake protocol in chapter. If the Smart House doesn't get to step 5 for any reason, the server identified by the certificate cannot be authenticated, and the Smart House will be warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the Fire brigade requires the Smart House authentication, the Fire brigade performs similar steps as the Smart House does to authenticate the Smart House.*

After the steps described here, the server must successfully use its private key to decrypt the premaster secret the Smart House sends in step 4 of the SSL handshake (see the SSL handshake protocol at section 5.1.1). Otherwise, the SSL session will be terminated. This provides additional assurance that the identity associated with the public key in the server's certificate is in fact the Fire brigade's server with which the Smart House is connected then the Smart House will extract the Fire brigade's public key from the certificate.

If the copy of the digital certificate is not included in the digital envelope, The Smart House must look up the Fire brigade's digital certificate in the directory (X.500). Then the Smart house must do all the steps mentioned above to validate the digital certificate except the first step that separates the three components.

## *6.3 Scenario 3: Securing Web transactions between Smart House and LoS AS*

In this scenario, I will build the security issues of the transactions between Los AS and the Smart House.

I will bring those concepts mentioned in the previous scenario together with real world example. When browsing the Web, there are times when you need to enter sensitive data such as personal information or credit card numbers. In these situations it is important that you authenticate the server you are sending the information to, because it makes no sense to send sensitive information if you don't know who is getting that information [13]. In addition to this, it is important that the communications between your Web browser and the Web server be encrypted so that attackers cannot gather the information as it flows across the Internet. The techniques we have been examining so far are used to achieve this security. Web servers support protocol named Secure Sockets Layer (SSL) that uses cryptographic techniques that the reader will no doubt now find familiar. Let's go through the process step by step (see Figure 6-10)

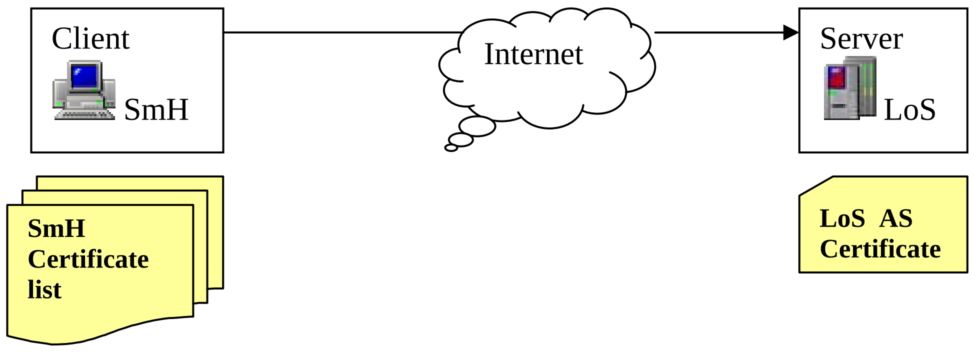


**Figure 6. 10. The Smart House sends certificate request to the LoS AS's server**

In this Figure 6.10, we see that the LoS AS has already bean issued digital certificate that contains the identity of the LoS AS, as well as the LoS AS's public key. The public key isn't shown here, is the fact that the LoS AS also is holding the matching private key. Note also that the Smart House has its preloaded table of trusted authorities.

To simplify this example, I will only show what is known as "server side SSL." As you will see, when you use server side SSL, the Smart House authenticates the LoS AS, and an encrypted channel is developed between the Smart House and the LoS AS. However, we assume that the LoS AS does authenticate the Smart House also, and the process is identical to that of server side SSL authentication. This is fairly common. If the Smart House (client) and the LoS AS (server) exchanging important messages like transactions, they have to authenticate each other. The Smart House must be sure that it is really connected to LoS server. LoS AS may not need to authenticate the Smart House, simply because it can identify the Smart House by other means, like time tokens. If it is necessary for the Smart House to authenticate itself to the LoS AS, the Smart House must get a digital certificate for verification from the Posten first. Remember in this case we are only performing server side SSL, but assuming that they are mutually authenticating each other. Client side SSL will not be shown in this work.

The first step in the process is for the LoS AS to send its digital certificate to the Smart House (see Figure 6.11). Recall that since all the information in the digital certificate is public information, it does not matter that if the certificate travels between the LoS AS and the Smart House in the clear.

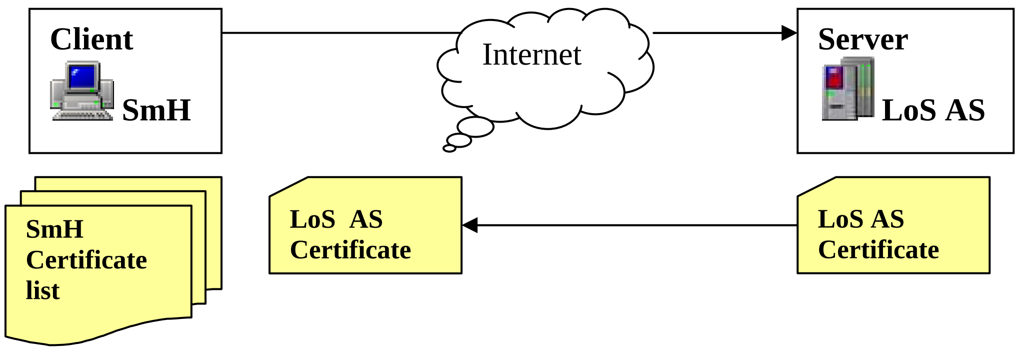


**Figure 6. 11. The LoS AS sends a digital certificate to the Smart House.**

In Figure 6.12, you see the Smart House extracting the LoS AS's public key from the certificate. Before the Smart House can trust the public key, it must validate the LoS AS's certificate. The Smart House will be able to see if the certificate is signed by source on the trusted authority list. Assuming that it is, the Smart House will then compute the hash of the certificate and compare it with hash in the certificate (decrypted using the trusted authority public key). If the hashes match, the Smart House knows the certificate has not been tampered with. Next, the Smart House will check the validity dates encoded in the certificate to be sure that the certificate has not expired. Assuming it has not, it will do one more special check associated with server certificate. Part of the identity information in the server certificate is the URL of the Web server. The Smart House will do an extra check to ensure that the node that sent the certificate has the same URL as with which the Smart House is trying to communicate, which is Los AS server.

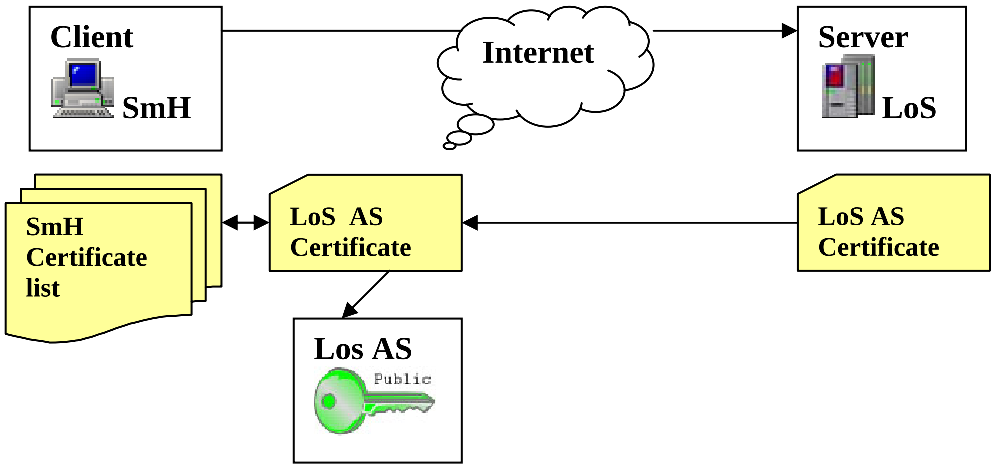


**Figure 6. 12. The Smart House extracts the LoS AS's public key from the certificate**

The Smart House application must check the server domain name specified in the server certificate against the actual domain name of the LoS AS with which the Smart House is attempting to communicate. This step is necessary to protect against a man in the middle attack, which works as follows [20].
The "man in the middle" is a rogue program that could intercept all communication between the Smart House and LoS AS with which the Smart House is attempting to communicate via SSL. The rogue program intercepts the legitimate keys that are passed back and forth during the SSL handshake, substitutes its own, and makes it appear to the Smart House that it is the LoS AS, and to the LoS AS that it is the Smart House (see Figure 6.13).
The encrypted information exchanged at the beginning of the SSL handshake is actually encrypted with the rogue program's public key or private key, rather than the Smart House 's or LoS AS's real keys. The rogue program ends up establishing one set of session keys for use with the LoS AS, and a different set of session keys for use with the Smart House. This allows the rogue program not only to read all the data that flows between the Smart House and the LoS AS, but also to change the data without being detected. Therefore, it is extremely important for the Smart House to check that the domain name in the server certificate corresponds to the domain name of the LoS AS with which the Smart House is attempting to communicate.

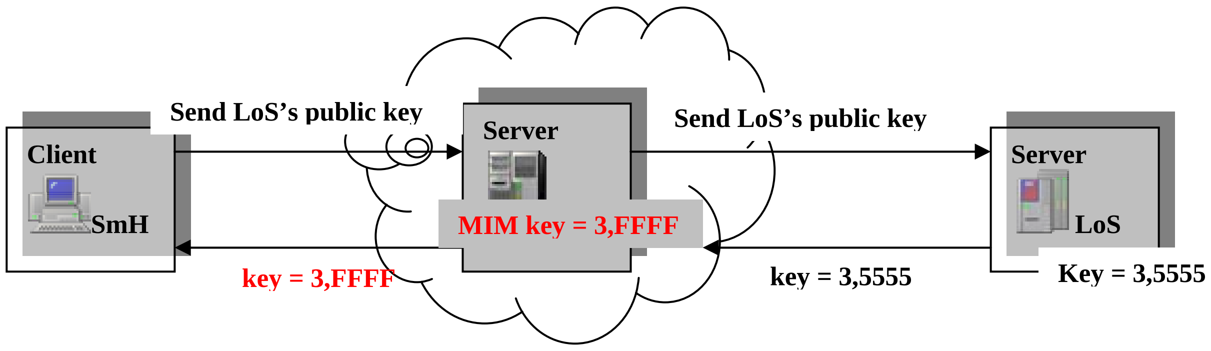


**Figure 6. 13. Man in the middle**

If all these checks match, the Smart House will then extract the public key of the LoS AS from the LoS AS certificate.
Once the Smart House has the LoS AS's public key, it then generates random symmetric encryption key. This key will be used to encrypt the conversation between the Smart House and the LoS AS. Recall that symmetric encryption algorithm is used because symmetric ciphers are fast, and they do not expand the data during the encryption operation [4]. In order to move the symmetric encryption key to the LoS AS, the Smart House performs key wrapping operation (see Figure 6.14). The symmetric key is encrypted using the LoS AS's public key that was extracted from the LoS AS's digital certificate.

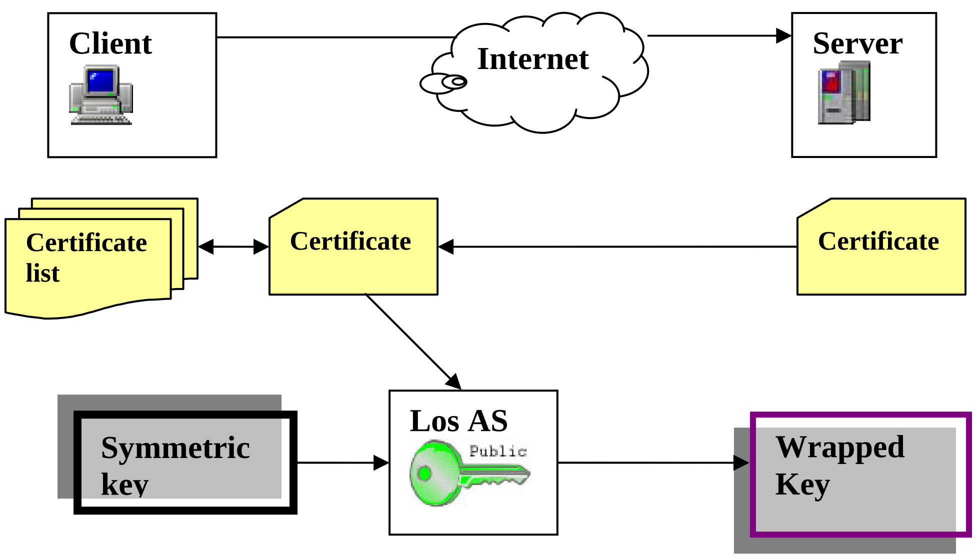


**Figure 6. 14. The Smart House sends the wrapped key to the LoS AS**

The Smart House sends the wrapped key to the LoS AS. Recall that since symmetric key is encrypted using the LoS AS's public key, and since the LoS AS is the only entity that has the matching private key, the hacker cannot extract the symmetric key.

Now that the LoS AS has the wrapped key, it can use the private key to decrypt the wrapped key (see Figure 6.15). This yields the original symmetric key that was randomly generated by the Smart House.
At this point, both the Smart House and the LoS AS have copy of the same symmetric encryption key.

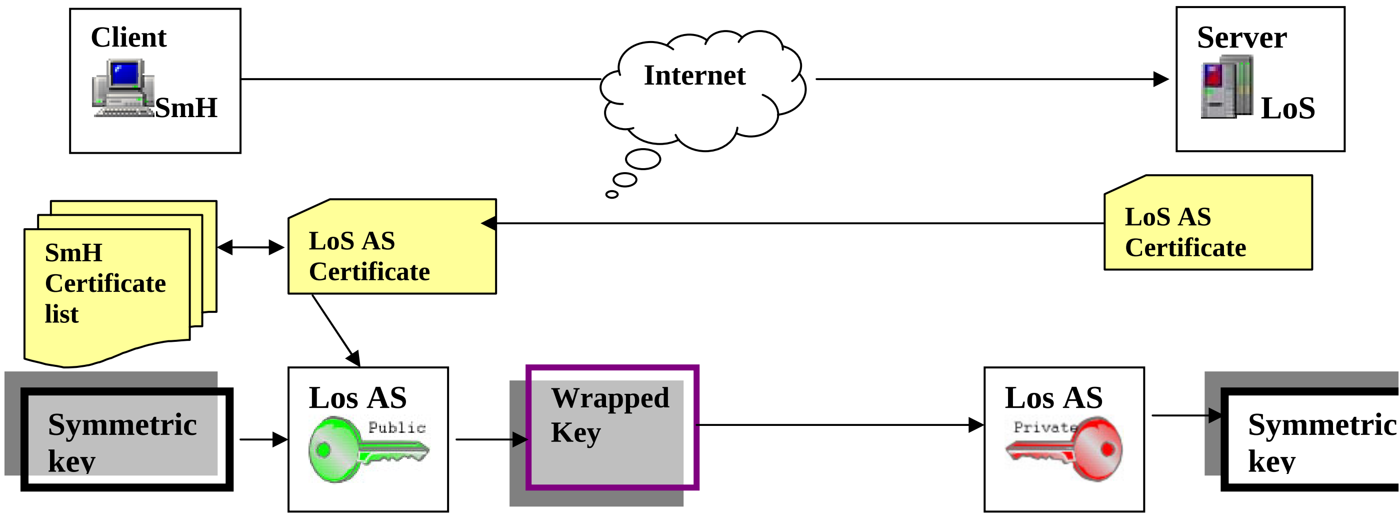


**Figure 6. 15. LoS AS's private key decrypts the symmetric key.**

In Figure 6.16, you can see that both ends of the conversation have the same symmetric key. They can now start an encrypted conversation using the exchange symmetric key to encrypt and decrypt the data from each other.

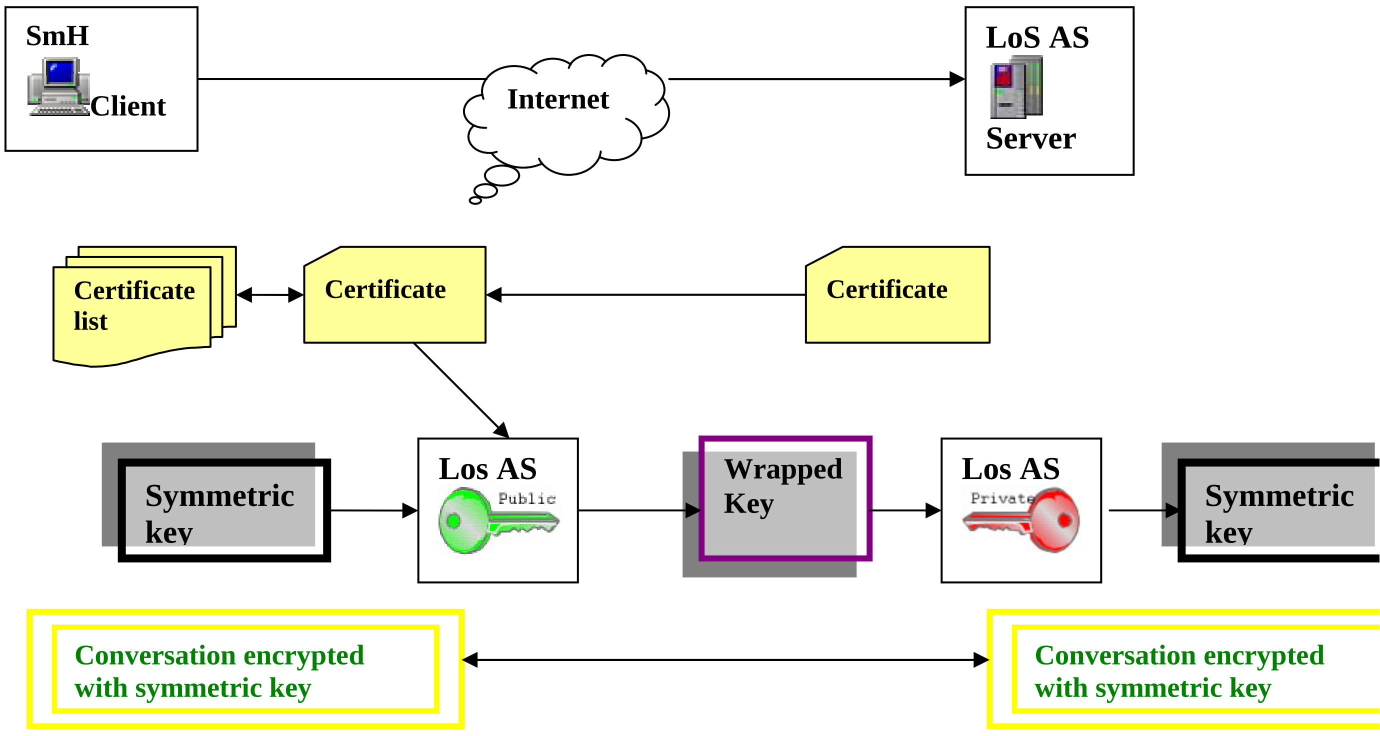


**Figure 6. 16. Conversation encrypted with the symmetric key.**

Something else happened in this exchange as well, and it is little subtle. As I mentioned at the start, the Smart House needs to be sure that it is talking to the correct LoS AS in other words; the Smart House needs to authenticate the LoS AS.

The check I mentioned in the certificate processing where the Smart House verifies the URL of the LoS AS is not sufficient check. This is because an evil web site could be spoofing as the real LoS AS. In this case, all traffic for the real web site will be redirected to the evil site. This type of attack is reasonably common in the Internet and is frequently accomplished when the attacker compromises a DNS sever and redirects traffic to the evil site [20].

The Smart House generated random symmetric encryption key, and then encrypted it using the public key of the LoS AS. The fact that the LoS AS was able to engage in an encrypted conversation with the Smart House told the Smart House that the LoS AS had successfully decrypted the wrapped key and extracted the symmetric key. This in turn told the Smart House that it was in fact talking to the real LoS AS because the real LoS AS is the only node in the universe that has the matching private key needed to perform the unwrapping operation.

### *6.4 Scenario 4: Simplified Wireless Public Key Infrastructure (WPKI)*

The following scenario will show a very simplified Wireless Public Key Infrastructure (WPKI) relation that will help us understand, how we could use certificates in the Wireless world [5]. In this scenario the SmH owner wants to have some kind of control with his/her Smart House's server, and uses his/her PKI enabled mobile telephone. This section is to answer to the third requirement point of the definition in this work. By using PKI enabled mobile fulfils all the requirements of this security level stage with condition that the owner needs to have an attribute certificate instead of public key certificate. Attribute certificates bind the characteristics of an entity (called attributes) to that entity through the signature of a so called Attribute Authority (AA) on a particular AC. Consequently, the major difference between a public key certificate and an attribute certificate is that the former includes a public key (with the key being certified), whereas the latter includes an attribute (with the attribute being certified). At this point, the owner gets full control over the Smart House including special privileges and authorization (see Figure 6.17).

1. SmH owner must have a certificate to take over the control of his own Smart House computer. Therefore SmH owner must send a certificate request to the Posten
2. The SmH owner gets activating code through the post office (out-of-band channel), and then the post office must validate the SmH owner's identity. The activating code uses to generate the public/private key par in the USIM. The SmH owner sends a certificate request together with the generated public key over the mobile network to the certification authority (Posten). The Posten puts the public key in a certificate, signs it and then sends a copy of it back to the SmH owner and keeps another copy in the certificate directory like (X.509 directory) for public use, not shown here. The SmH owner must protect his/her private key with PIN (personal identification number) or with hard to guess password. The private key must be saved in the USIM card. No body in the universe even the SmH owner himself could know the private key.
   This solution is one of many different ways, which certificate authorities could operate.
3. The Smart House gets enquiry from the SmH owner, who wants to take control of his home computer.
4. The Smart House must assure that the owner's certificate is valid.
5. SmH owner's certificate is approved, and now the contents of the message itself must be verified using the SmH owner's public key.
6. The SmH owner is informed that he/she is now authorized to have a special session connection with the Smart House.
7. The SmH owner and the Smart House really know each other by authenticating each other. The SmH owner gets now a secured access channel to have a control with his/her own home computer, using his mobile telephone. The SmH owner and the Smart House can now start an encrypted conversation using the symmetric key to encrypt and decrypt the data that flows between them.

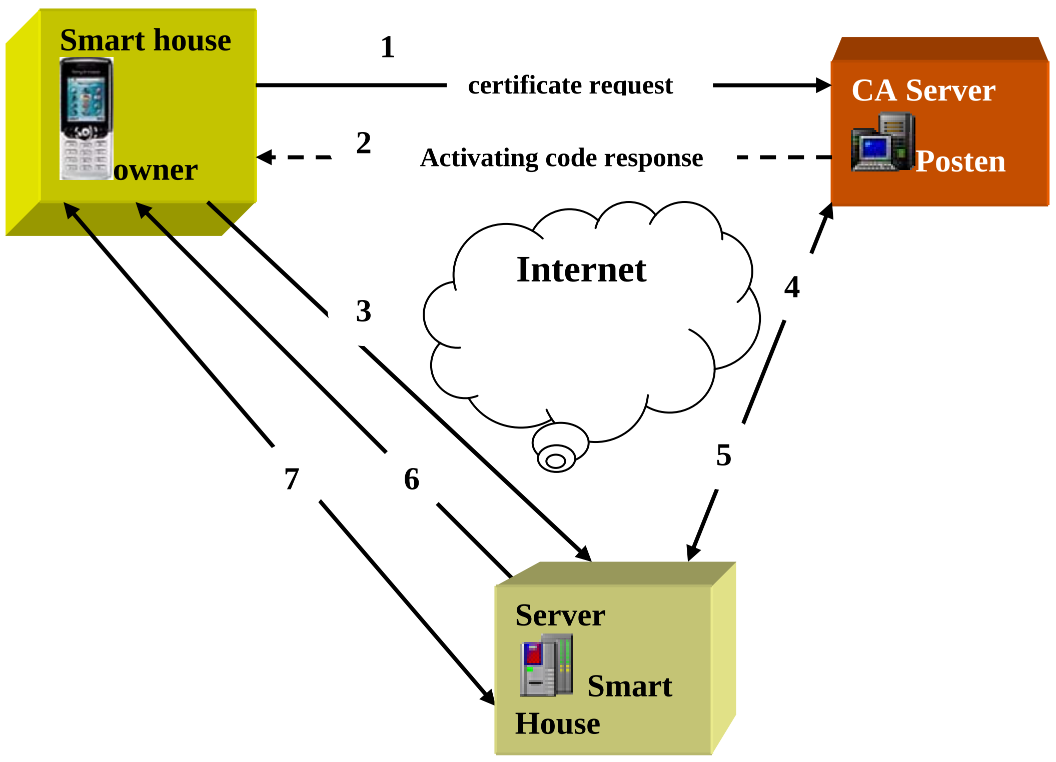


**Figure 6. 17. Wireless certificate request**

The transactions between the SmH owner and the Smart House will not be legally valid until they are digitally signed (see scenario 2 and 3 in this chapter). This Wireless PKI scenario does not describe how the data exchange happens (Example, to use WAP or SMS). It could be some differences but the main principle is the same. We assume also that the Smart House owner has installed SSL server software together with the client software at the Smart House's node.

## *6.5 Scenario 5: How to receive the digital certificate from the Posten (CA)*

How the Smart House and the other vendors can get a certificate from the Posten (CA) for the first time, or get another certificate from it at later time or renew the old certificate. This section explains how the Smart House can retrieve a digital certificate from the Posten (CA) directly or through a mediator SubPosten (RA).

Public Key Infrastructure (PKI) management transactions can be designed so that the Posten and the Smart House can implement the transaction without assistance. These are called two party transaction models. In other case, the transactions leverage a SubPosten to fill in the gap in the trust relationships between the Posten and the Smart House. These are three party transaction models. Different models will achieve different security objectives [1].

The simplest PKI transactions include two parties: the Posten and the Smart House. Figure 6.18, shows a simple two party transaction model between the Smart House and the Posten.

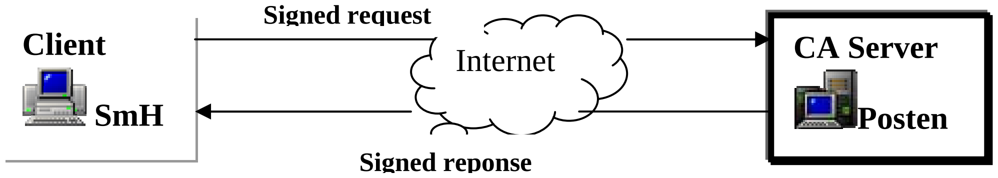


**Figure 6. 18. Two party transaction model**

**Process:**
1. The Smart House generates the signed request.
2. The Posten validates the Smart House's certification path and digital signature.
3. The Posten processes the request and generates a signed response.
4. The Smart House validates the Posten's path and digital signature, then processes the response

To use this model, the Smart House must be the subject of a valid signature certificate, and the Smart House must know the Posten's public key. Each message is digitally signed to authenticate the sender.

In this model, all information flows in the form of electronic messages. This type of transaction can be completed in a single round trip or may include additional confirmation messages.

The simple two party transaction model is widely used to implement both the basic certificate request and the revocation requests. The Posten authenticates the Smart House identity based on the signed request. For revocation requests, the signature of the Smart House or SubPosten is all the confirmation that the Posten needs to revoke the certificate. If the Smart House is not requesting a change in his or her name or other security relevant attributes, then the basic certificate request provides the Posten with all of the information needed to issue a new certificate. This type of certificate request is best suited for certificate renewal.

However, the simple two party model is insufficient for the initial certificate request. The Posten requires further confirmation in order to trust the information received from the Smart House. This model is also insufficient when the Smart House is requesting changes in particular attributes. For example, the Smart House might be requesting new name or might request a different certificate policy for the new certificate. To provide the Posten with additional reliable information, we can either extend the two party transaction model or turn to the three party transaction model.

The extended two party model is depicted as Figure 6.19. In this model, the Smart House generates a message, and then transmits it to the Posten. The Posten processes the message, and then generates a response. The Posten generates a random encryption key, called the authenticator, and then it uses to encrypt the response. The encrypted response is returned to the Smart House. The Smart House must obtain the authenticator to correctly decrypt the message. The Posten sends the authenticator to the Smart House through an out-of band channel.

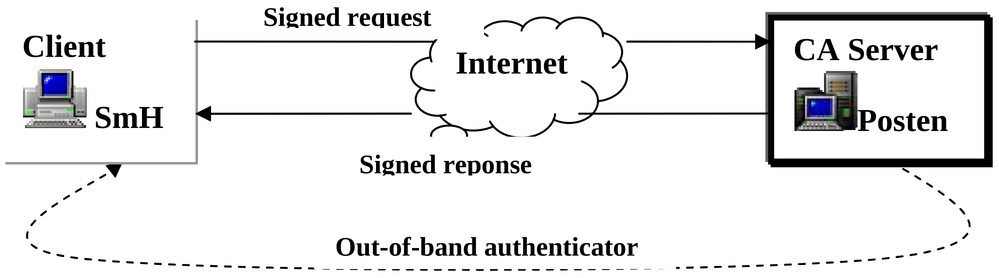


**Figure 6. 19. Extended two party transaction model**

**Process:**

1. Smart House generates the signed request.
2. Posten verifies integrity, processes contents, and generates authenticator.
3. Posten encrypts the response using the authenticator and returns it.
4. Smart House stores the encrypted response.
5. Posten sends the authenticator via out-of-band means.
6. Smart House decrypts the response using the authenticator

The out-of band channel is designed to confirm the information provided by the Smart House. For example, the Posten might confirm the electronic mail address of the Smart House by returning the authenticator in an e-mail message sent to that address. Similarly, the Posten might confirm the postal address of the Smart House by returning the message by surface mail. The Posten might confirm the Smart House's name by sending the authenticator via certified mail, where the owner of the Smart House presents identification to the postal clerk to retrieve the message. There is a variation to this model. Instead of encrypting the response, the Posten stores the response in an online database and protects access to the data with a password. In this case, the password is the authenticator. As discussed earlier, the Posten transmits the authenticator to the Smart House through an out-of-band channel. The selected out-of-band channel is designed to confirm the identity information in the request.

The extended two party models are widely used because of their simplicity. However, they have several undesirable properties. The Posten is generating a certificate without knowing whether the information it contains is valid. This action is in direct conflict with the Posten's responsibilities. The Posten cannot publish the certificate in a repository without confirmation that the Smart House was able to access the response. More satisfactory results can be achieved by delegating the burden for verifying the Smart House information to the SubPosten.

In this case, the Posten, SubPosten, and the Smart House all participate in a three party transaction model. The Posten counts on the SubPosten to verify the information it cannot accept directly from the Smart House. The Posten uses the SubPosten validated information to issue and revoke certificates. There are many different transactions that employ the three party model to issue certificates. Selecting appropriate model requires consideration of the type of information to be verified by the SubPosten. This information is defined in the certification policy. We take one illustrative example which is relevant to this thesis here.

The Smart House generates a public/private key pair, and then the Smart House owner presents physical and electronic credentials to the SubPosten. The SubPosten reviews the credentials, and then the SubPosten forwards the relevant information to the Posten. The Posten generates the certificate and returns it to the SubPosten which forwards it to the Smart House. This model is depicted in Figure 6.20.

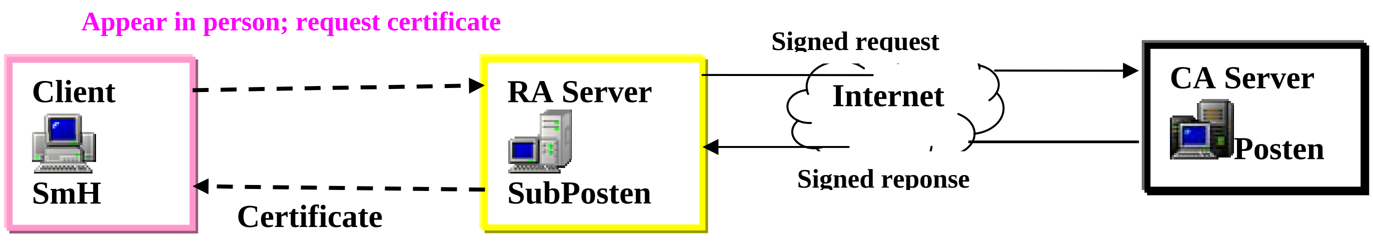


**Figure 6. 20. SubPosten performs in person authentication**

**Process:**

1. Smart House generates public/private key and the owner appears in person at the SubPosten.
2. SubPosten validates Smart House's credentials, then generates signed request to Posten.
3. Posten validates SubPosten's certification path, and digital signature.
4. Posten processes the request and generates a signed response.
5. SubPosten validates Posten's certification path and digital signature, then processes the response.
6. SubPosten validates the response and provides certificate the Smart House.

In Figure 6.20, the SubPosten authenticates the request in person, so the SubPosten can verify all the information in the certificate request. Once this is completed, the SubPosten signs and transmits the request to the Posten. The Posten verifies the signature on the request, knows all that information is trustworthy, and generates the certificate. The Posten may wish to verify that the requester has the private key, or it leave this validation to the SubPosten. This is the best suited model in our scenarios, because we can use regional post offices as the SubPosten and the main post office in Oslo as the Posten.

## *6.6 Scenario 6: IPsec VPN client using digital certificate for authentication*

We will use an example to show how two systems might use some of IPsec options. In our example, the Smart House owner has IPsec based virtual private network (VPN) client software installed on his laptop. When the Smart House owner's laptop connects to the Internet, the VPN client filters the traffic, watching for IP packets destined for the Smart House owner's laptop. It allows any traffic not going to the Smart House to pass through normally (unsecured). When the client sees a packet that is addressed to the Smart House, however, it intercepts it. It then uses IPsec services to transmit the packet securely to the Smart House and to ensure that all traffic back from the Smart House is also secure.

The first thing the VPN client does is to establish an IPsec security association with the Smart House's communications server. A security association (SA) defines a security context between two parties. The Internet Security Association and Key Management Protocol (ISAKMP) is the framework that defines how the VPN client and server set up this association [1]. With ISAKMP, they negotiate the encryption algorithm, the hash algorithm, the authentication mechanism, and the key establishment mechanism they will use for their IPsec services. The ISAKMP does not mandate particular algorithms or mechanisms so that it can provide maximum flexibility. It does, however, require the use of digital signatures within the authentication component. This means that the Smart House owner's laptop and the Smart House server must have IPsec public key certificates to be able to establish a security association. It also means that the Smart House owner’s laptop and the Smart House server need to know the security association options that the other supports: otherwise, the Smart House owner’s laptop and the Smart House server may not be able to negotiate common security association settings.

The first part of the ISAKMP SA entails the two parties negotiating a secure channel over which they will negotiate further SAs. Each subsequent SA is specific to a security protocol. ISAKMP’s current use is primarily for IPsec authentication header and Encapsulating Payload protocols. ISAKMP consolidates the authentication and key negotiations that are commonly done in secure protocols, making security operations more efficient.

At this point the Smart House owner’s laptop and his Smart House server have negotiated an ISAKMP SA for traffic from the Smart House owner’s laptop to the Smart House server. They then negotiate a second SA. Each SA is uniquely identified by the combination of a security parameters index (SPI) in each IP packet, the security protocol, and the destination IP address [4]. As a result, each SA is one way (one destination address, one SA). For the Smart House owner’s laptop and the Smart House server to exchange bidirectional IPsec traffic, they must therefore negotiate two security associations one for traffic from the Smart House owner’s laptop to the Smart House server (the Smart House server is the destination) and one for the traffic from the Smart House server to the Smart House owner’s laptop (the Smart House owner’s laptop is the destination).

The Smart House owner’s laptop and the Smart House server have now completed the first step in setting up IPsec communications. The SAs set the basic security contest: the Smart House owner’s laptop and the Smart House server have agreed upon the algorithms to use and have authenticated each other. They then complete a second phase of SA negotiation specific to the IPsec services they will use. To apply IPsec’s other security services to their communications, such as access control, connectionless integrity, data origin authentication, replay protection, or confidentiality, the Smart House owner’s laptop and the Smart House server must negotiate, the use of the authentication header (AH), the Encapsulating Security Payload (ESP), or normally a combination of the two.

Finally, the Smart House owner’s laptop with client VPN is sending sensitive contractual details back to the Smart House server, so the VPN client is preconfigured with a shared HMAC key, the Smart House owner’s laptop and the Smart House server agree to use keyed MACs and triple DES encryption to provide integrity, origin authentication, and confidentiality

# 7 Discussion

This chapter starts with a discussion of the most frequently used transaction protocols (SSL and IPsec). SSL is an application to application security protocol, while IPsec is network layer security protocol, which intends to guarantee confidentiality, authenticity and Integrity.

Then I discuss the main scenarios in this work, using an example of asymmetric cryptography and another one of the combination of symmetric and asymmetric cryptography.
The discussion will cover all the three main PKI scenarios we have seen in this work, in terms of:

- Authentication
- Integrity
- Confidentiality
- And non-repudiation

## *7.1 Comparing PKI based protocols SSL and IPsec VPNs*

Virtual Private Networks (VPNs) allow enterprises to build secure, private communications over public network infrastructures. Several different technologies are used to create VPNs, including Internet Protocol Security (IPSec), and Secure Sockets Layer (SSL). In this discussion we focus on these two technologies used to provide remote VPN access for mobile users IPSec and SSL. Each technology employs standards based encryption and authentication techniques that secure access to corporate data over the Internet.
Selecting the appropriate technology or a combination of both for their remote access will be discussed:

1. *What are the Similarities or differences between these two technologies?*
2. *Which Technology is best for Remote Access?*

### 7.1.1 What are the Similarities or differences between these two technologies?

IPSec and SSL are both effective ways to provide secure remote access to corporate resources over the Internet. The two technologies are similar yet different in their approach to VPNs, each having its advantages and disadvantages. Their differences can be[21]:

- Accessibility and Ease-of-Use
- Security
- Management Complexity
- Scalability and Performance
- Cost of Ownership

#### 7.1.1.1 Accessibility and Ease of Use

Users can only access the VPN using that specific IPSec client, IPSec VPN access is tied to a specific machine (laptop) often for a specific user. This can provide stronger security but may limit accessibility and mobility [6]. IPSec clients may also require manual configuration making them some what difficult to use for none technical workers like sales personnel.
IPSec's primary advantage is that it operates at the network layer, securing all data between two end points, including all applications. Remote users have access to corporate resources as if they were physically in the office connected to the corporate LAN. This makes IPSec ideal

for workers in branch offices. IPSec users can access the following applications: E-mai, File share, Web (HTTP), Client server, Databases, Terminal services.

SSL VPNs use standard web browsers like Internet Explorer and Netscape as the remote user's interface. A key advantage is that web browsers are familiar to just about all users and are embedded in every type of user device, mostly web browsers.
SSL's primary disadvantage is that it operates at the application layer, limiting access to only those resources that are browser accessible or for which the SSL VPN gateway has developed special purpose proxy capabilities. The common applications accessible using SSL are: E-mail, File share, Web (HTTP) [7].

**7.1.1.2 Security**
A major difference between IPSec and SSL is the security protection they provide. In many cases, security is used as the primary criteria for selecting which users and applications should use IPSec versus SSL. Both can play a role in an enterprise virtual network if applied appropriately.

The two main security components when comparing IPSec and SSL are: encryption and authentication. Both IPSec and SSL support the use of encryption but use different encryption algorithms. IPSec typically uses 56-bit DES or 112-bit or 168-bit Triple DES (3DES) encryption. SSL typically uses 40-bit or 128-bit RC4 encryption. Each of these cryptographic algorithms are similar in that they ensure data privacy over the Internet, but IPSec provides the stronger (3DES) encryption method.

IPsec devices must agree in advance on the security association in order to establish the tunnel between the end points. This is not always a feature of SSL VPNs, however. Some SSL implementations negotiate down to the lowest common denominator (40-bit encryption), and therefore enterprises cannot guarantee the use of strong encryption for their remote users. New, more advanced SSL VPN solutions provide the IT administrator with the ability to only allow browsers that support 128-bit encryption, overcoming this potential security weakness.

Like encryption, both IPSec and SSL support authentication to ensure validity of each end entity. The authentication techniques can be the same for both access types. Supported authentication technologies are largely dependent on the VPN provider, but both IPSec and SSL can employ username and password, username and token + pin, or X.509 digital certificates [4]. Digital certificate support can vary from using a certificate only on the server (VPN gateway) to both the client machine (user's PC) and the server.

Although both IPSec and SSL can use the same authentication technologies, IPSec requires a specific piece of client software be installed on a specific machine to access the network, whereas SSL users can potentially gain access from any device with a web browser. To overcome this security hole, we can utilize two-factor authentication technologies like RSA SecurID, which combines something you know (password) with something you have (token). This approach guarantees the identity of the user, not the machine. If we choose to use digital certificates we should understand that the most secure implementation is when both server and client side certificates are used. This is true for both IPSec and SSL.

A disadvantage to SSL is that authentication requires the end user to verify that the certificate being presented by the server is correctly representing the server's identity. There is risk that a hacker could successfully fool the user into accepting a bogus certificate, thus creating a secure communications channel with a hacker and exposing the corporate network to what's known as a "man-in-the-middle" attack. We should consider this risk when selecting SSL.

#### 7.1.1.3 Management Complexity

While IPSec is considered more secure than SSL, IPSec VPNs can be more complicated to deploy and manage. This is because IPSec requires special purpose VPN client software, whereas SSL VPNs are browser based. Another reason is that IPSec requires configuration of many networking parameters and security policies to create an end to end VPN tunnel.

Deploying an IPSec based VPN involves several steps, the first of which is distribution of IPSec client software to all remote users. Most require IT administrators to burn CDs and mail them to users. Another common approach is to make the software downloadable from a LAN based server.

Once users have received the IPSec software, they must successfully install it on their PC. This step alone is often the greatest causes for help desk calls, because the installation may be complicated or not succeed due to incompatibility issues. Many IT administrators take installation into their own hands. This approach not only is labour intensive, it also greatly slows down VPN deployment.

IPSec VPN products require IT administrators to become experts in tunneling and encryption technology because IPSec is more complex than SSL. IPSec manual configuration is so complex, many IT managers prefer by accepting the defaults, which significantly reduces the security of the network.

SSL VPNs work with existing software embedded in user operating systems, therefore they are often referred to as "client-less", although they do require technology on the server side that can accept SSL sessions. This saves us significant deployment cost, help desk support and headache. IT administrators can enrol users by simply enabling their username and password and providing them with the URL of the VPN gateway. With SSL, users are typically connected trouble free.

#### 7.1.1.4 Scalability and Performance

SSL VPNs are scalable in that they can be quickly deployed to remote users regardless of machine or location, but IPSec is more scalable in terms of its transparency to the network. From the user and application perspectives, the secure network (once established) is indistinguishable from a trusted LAN. Existing network accessible applications can be used through the VPN without modification. Changes to the applications are independent of changes in the VPN.

SSL VPNs require tight integration with the application, as it becomes the interface to the user. Therefore, e-mail and file-sharing applications are well suited to SSL access and offer users comparable performance levels to when they are in the office.

#### 7.1.1.5 Cost of Ownership

SSL and IPSec VPNs are comparable in terms of capital outlay. Both require VPN capable servers at the corporate site to terminate remote user sessions. But because SSL VPNs do not require client software and can be less of a deployment and management burden, their total cost of ownership is usually less.

### 7.1.2 Which Technology is best for Remote Access?

IPSec and SSL can both be used in an enterprise virtual network when applied appropriately. Each has its strengths and weaknesses that make the technology better suited to some remote access users and applications than others.

In general, IPSec is best suited to users that require access to all applications and resources as if they were physically connected to the corporate LAN. IPSec also supports stronger encryption strengths (3DES, AES) and guarantees the identity of the remote user because it requires the use of specially provisioned IPSec client software. While IPSec may take longer to deploy, as a system it is more scalable because it operates independent of the applications. Enterprises must balance their business needs with their requirements for a secure network. Many enterprises quickly jump to select IPSec VPNs without considering whether IPSec is actually too much security (and overhead) for their needs.

In general, SSL is best suited to users that need casual or mobile access to applications like email and file sharing. It is also ideal for extranet applications because SSL enabled browsers are prevalent and can be used to quickly and easily connect customers, partners and suppliers. However, for permanent or always on extranet connections between offices, IPSec VPN gateways are recommended.

## *7.2 Discussing the different scenarios of my work*

This discussion shows how we could apply the benefits of symmetric and asymmetric cryptography. It shows how we could take the advantages of the symmetric and asymmetric cryptography together, and apply them to two communicating end entities. Using of their advantages will allow us have the ideal solutions of PKI's security. For the sake of clarity, there will be only three main figures, and the text will be simplified. To combine and discuss the benefits of symmetric and asymmetric cryptography properties, the discussion will cover all the three main PKI scenarios we have seen in this thesis project. The first part of this discussion at section 7.2.1, combines scenario 1 and scenario 3, while in section 7.2.2, summarizes scenario 2.

### *7.2.1 The ideal solution of cryptography*

Our solution must qualify to the following requirements [4]:

1. The solution must be secure.
2. The speed of the encryption must be fast.
3. The cipher text must almost be the same size as the plain text.
4. The scaling must satisfy to a large population.
5. The key must not be vulnerable for interception.
6. The communicating parties must not require prior relationship.
7. The original data source must guarantee by the solution.
8. And finally it must detect if the message was modified in transit.

***The sending side (the LoS AS's side):***
The LoS AS server will generate a random symmetric encryption key. The key will be used to encrypt the message from the LoS AS. It is clear, that asymmetric cryptography is slow, but since the size of the symmetric key is very small (typically 128 bits) the actual time spent on an asymmetric encryption is small. The result of this encryption is a random symmetric encryption key, encrypted with an asymmetric public key. One key encrypted with another key. This is called a key wrapping operation. In order to get the symmetric key to the Smart House, we will need to perform a key wrapping operation. With key wrapping, we use the public key of the Smart House to encrypt the symmetric key, producing the wrapping key.

The process of creating digital signature is the next step. The first step of this process is to hash the clear text message and produce the original digest. Then to create the digital signature, we take LoS AS's private key to encrypt the digest we just created.
At last we take the cipher text, the wrapped key, the LoS AS's digital certificate and the encrypted digest (digital signature) and put them together in to the digital envelope and send it to the Smart House (see Figure 7.1 ) .

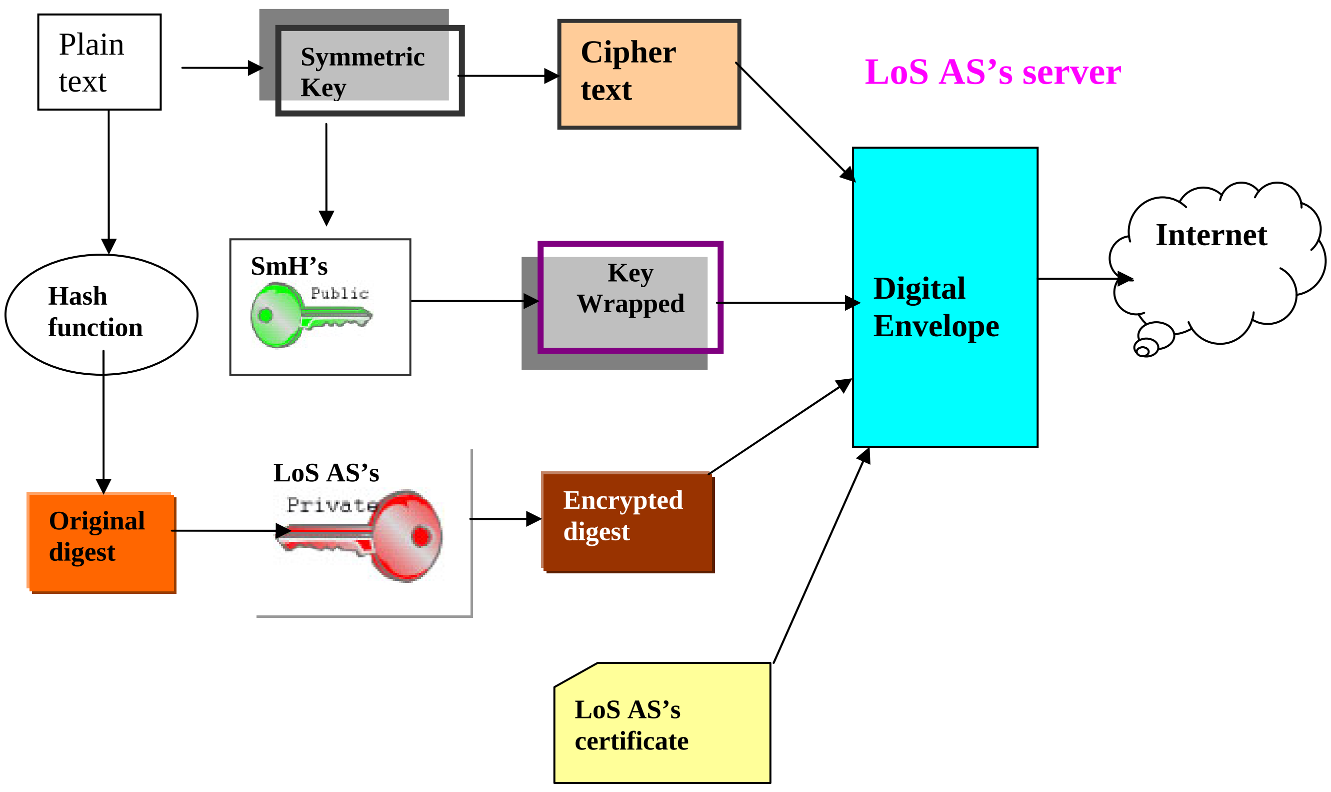


**Figure 7. 1. LoS AS Sends digital envelope to the Smart House**

***The receiving side (Smart House):***
When the package arrives at the destination, the first process will be to separate the three components and the digital certificate. If the public key certificate was not attached to the digital envelope, we would need to look up the public key certificate in a directory.

We use the Smart House's private key to decrypt the wrapped key. This will yield the symmetric encryption key used to encrypt the cipher text.
Now that we have the symmetric encryption key, we can use it to decrypt the cipher text, and recover the original clear text.

*This step is to check the digital certificate, which was attached to the digital envelope, to be sure that if the signature of the certificate is valid before we use it.*
*Validity starts if someone that the Smart House's software trusts has signed the digital certificate. This is done by checking the signer of the certificate against the trusted authority's list contained with in the software.*
*If the authority that signed the digital certificate is not included in the trusted authority list, the verification stops with an error. Applications could allow different procedures, how to proceed in this situation.*
*If the signed authority is in the trusted authority's list, then the LoS AS's digital certificate itself will be checked for validity. This includes comparing a new hash of the certificate data with the copy that was encrypted by the trusted authority when the certificate was created. To decrypt the trusted authority's (Posten's) digital signature, we use the certificate authority's (Posten's) public key. If the certificate passes this check, the validity dates will be examined. If the certificate has not expired, you will extract the LoS AS's public key.*

The next step is to use the LoS AS's public key that we have just extracted from the certificate to decrypt the LoS AS's digital signature, and get the original digest. Then we take the clear text message and run it through the same hash algorithm that was used by LoS AS to create the original digest. This process was the second copy of the digest, labeled the new digest. If the two copies of the digests match, the Smart House knows that the LoS AS has sent the message. This is assured because the Smart House used the LoS AS's public key to decrypt the encrypted digest, and since the Smart House recovered the correct original digest, the Smart House knows it was originally encrypted using the matching private key. Since the LoS AS's server is the only server in the universe which has the matching private key, the Smart House can guarantee that the LoS AS has sent the message.
If the two copies of the digest match, the Smart House also knows that the message was not modified as it traversed the Internet. If the hacker had attempted to modify or substituted the clear message as it passed by, the (new digest) version of the digest would have a different value than the decrypted version of the original digest created by the LoS AS before it sent the message into the Internet.
Recall that if even one bit of the clear message changes, the digest resulting from the hash algorithm will be different. This capability of hashes to detect the smallest change in the clear message is what makes them useful for verifying message integrity (See Figure 7.2).

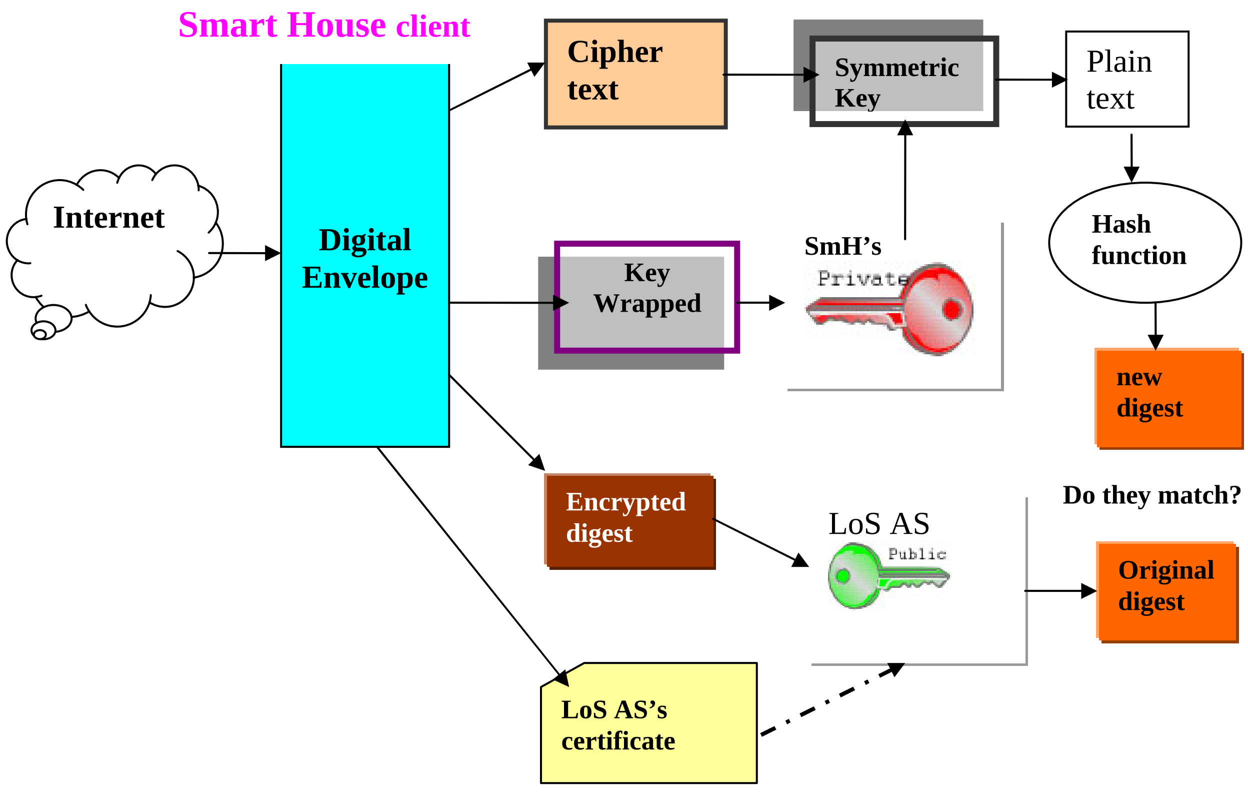


**Figure 7. 2. The Smart House receives the digital envelope**

In this first part of discussion we have used the symmetric cipher to perform a fast, compact, and secure encryption of the message. The encryption protected the message so that hackers on the Internet could not see it. Public/private key cryptography was used to wrap the symmetric encryption key, allowing us to have a scalable solution that is not vulnerable to interception, and which did not require prior relationship.

This first part of discussion we used the combination of symmetric and asymmetric cryptography, which we needed to protect our transactions between LoS AS and the Smart House. Therefore this part of discussion has fulfilled all PKI services required for those transactions:

- Authentication
- Integrity
- Confidentiality
- Non-repudiation

### *7.2.2 Messages that flow between the Fire brigade and the Smart House*

Finally, I discussed the communication that goes between, the fire brigade and the Smart House in detail in scenario 2. In that scenario we saw that the clear message with the digital signature travelled across the Internet. The message was not protected from eavesdropping, which means, hackers could read or see the clear message, but they couldn't modify it while the message was in transit. This is the requirement for the communication between SecuritAs, Fire brigade and the Smart House (see the definition of this work). The verification of the sender's digital signature (see Figure 7.3) fulfils all the PKI services required for the transactions between the Fire brigade, SecuritAs and the Smart House:

- authentication,
- integrity
- and none repudiation

For the sake of clarity, only the verification of the digital signature will be shown here.

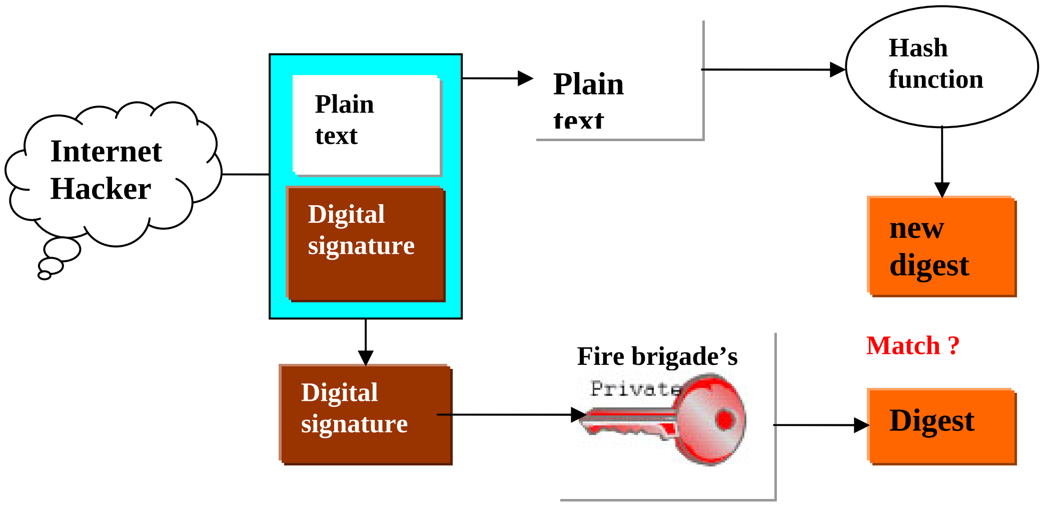


**Figure 7. 3. The Smart House verifies the digital signature**

The advantage of this last part of discussion is that the two parties don't need to encrypt the messages that go between them. Therefore it doesn't need to use the overhead of the symmetric algorithm at all, compared to our ideal solution of section 7.2.1 of this discussion. Even if we get a high speed and the cipher text is compact when using symmetric cipher, the speed will be even higher or faster, if we don't encrypt the messages at all and send them as clear text. The disadvantage is that it is vulnerable to eavesdropping, anybody who wants to can read it or even see it.

# 8 Conclusion

All throughout this work, we have seen different scenarios and discussed cryptography and the role of Public Key Infrastructure (PKI) in the protection of the information systems. The Public Key Infrastructure provides the most effective method of assuring communication integrity, authentication, confidentiality and non-repudiation between two communicating end entities.

**End to end transaction protocols (IPsec and SSL)**
IPSec and SSL are both effective, standards based technologies to use when deploying remote access VPNs. SSL is frequently used to secure web sessions at the application level, while the IPsec protocol secures communications between two nodes. Each technology has its advantages and disadvantages as well as strengths and weaknesses. Factors to consider include application and user accessibility, ease of use for none technical users, encryption and authentication security, deployment and management complexity, scalability and performance, as well as total cost of ownership.

The best suited end to end transaction protocol in this work is the SSL protocol, because it is dynamic and there is no special purpose client software to deploy as IPsec requires.

**Transactions sent in clear text to the Smart House**
According to this project's definition, the messages that flow between the SecuritAs, Fire brigade and the Smart House should have all the PKI services except the confidentiality (privacy) service. In scenario 2, I discussed this type of communication in detail, and it is clear that to send a message without encrypting it, has a disadvantage and an advantage.
The disadvantage is that the transactions are vulnerable to eavesdropping, because the transactions travel across the Internet in clear text. Anybody who likes can see or read it, but there is no way for the hackers to modify the transaction in transit without disclosing it by the receiving end entity.

The messages in scenario 2 and in the discussion in section 7.2.2 are sent in clear messages across the Internet. That means it is not so important to hide the content of these messages here, but it is more important to handle the integrity and authenticity in a secure way, according to the definition of this thesis project. The Smart House authenticates the Fire brigade but the Fire brigade may not need to authenticate the Smart House, simply because it can identify the Smart House by other means, like time tokens.

The advantage of sending messages in clear text saves us from the overhead of the symmetric encryption algorithm, which is not needed here.

**Transactions between the Smart House and the LoS AS**
These transactions use all the fundamental PKI security services: *Authentication, Integrity, Confidentiality and Non-repudiation.*
Using the combination of symmetric and asymmetric cryptography techniques, like those I have shown in scenario 3 and in the discussion section 7.2.1, we can achieve those fundamental security services mentioned above. Combining symmetric and asymmetric cryptography into a practical encryption gives us the ideal solution, which has the following principles: The solution is secure, the encrypted message can be transmitted more quickly, the cipher text is compact, it scales to a large population, it is not vulnerable to key interception, it

doesn't require prior relationship by the end entities, it guarantees the source of the message, and finally it can detect if the message was modified in transit.

It is very important to notice that, performing the wrapped key operation by the Smart House solves the key exchange problem we had in the symmetric cryptography. We know that symmetric encryption algorithm is used because symmetric ciphers are fast, and they do not expand the data during the encryption operation. The LoS AS is the only entity that has the matching private key, no one else can extract the symmetric key. It is also important to notice that the communication that flows between the Smart House and LoS AS is encrypted, by using this symmetric key. The attackers couldn't gather the information that flows between the two end entities.

The Smart House authenticates the LoS AS's server and since the LoS AS decrypted the wrapped key and engaged an encrypted conversation with the Smart House, the Smart House knows that it is connected the real LoS AS. We assume that the LoS AS does authenticate the Smart House also (mutual authentication). The Smart House must have a digital certificate to be authenticated by the LoS AS's server.

**Wireless PKI**
In scenario 4, I showed a simple example of real Wireless Public Key Infrastructure (WPKI). The condition here is that the Smart House must have SSL server software installed, and the owner must have an attribute certificate (AC) in order to have full authorization and special privileges for his home computer. Therefore he could use an SSL-enabled mobile phone or an SSL-enabled browser.

# 9 Abbreviations and Glossary

| | |
|---|---|
| PKI | Public Key Infrastructure |
| WPKI | Wireless Public Key Infrastructure |
| SmH | Smart House |
| IP | Internet Protocol |
| TCP | Transmission Control Protocol |
| SSL | Secure Socket Layer Protocol |
| IPsec | Internet Security Protocol |
| TLS | Transport Layer Security |
| LDAP | Light Directory Access Protocol |
| HTTP | Hyper Text Transport Protocol |
| IMAP | Internet Messaging Access Protocol |
| Posten | Norwegian main Post Office |
| WTLS | Wireless Transport Layer Security |
| PGP | Pretty Good Privacy |
| CA | Certificate Authority |
| RA | Register Authority |
| AA | Attribute Authority |
| TTP | Trusted Third Party |
| IT | Information Technology |
| CP | Certificate Policy |
| CPS | Certificate Policy Security |
| ICT | Information Communication Technology |
| HR | Home Register |
| VPN | Virtual Private Network |
| DES | Data Encryption Standard |
| RC4 | Rivest Cipher |
| ITU | International Telecommunication Union |
| X.509 | Public Key Certificate Standard |
| IETF | Internet Engineering Task Force |
| SPI | Security Parameter Index |
| SA | Security Association |
| ISAKMP | Internet Security Association and Key Management Protocol |
| AH | Authentication Header |
| ESP | Encapsulating Security Payload |
| MAC | Message Authentication Code |
| HMAC | Keyed Hashing for Message Authentication |
| AES | Algorithm Encryption Standard |
| LAN | Local Area Network |

## 10 References

| | |
|---|---|
| [1] Houseley, 2001 | " Planning for PKI"<br>Russ Houseley<br>ISBN 0-471-39702-4 |
| [2] Adams & Lioyd, 2003 | "Understanding PKI"<br>Carlisle Adams & Steve Lioyd<br>ISBN 0-672-32391-5 |
| [3] Alfred, 1997 | "Hand book of applied Cryptography"<br>Alfred Menezes<br>ISBN 0-8493-8523-7 |
| [4] Nash and Duane, 2001 | "PKI implementation & managing e-security"<br>Andrew Nash and William Duane<br>ISBN 0-07-213123-3 |
| [5] Larsen & Slettholt, 2002 | "Sikkerhet i UMTS - er PKI tilstrekkelig"<br>Dag Frode Larsen & Joakim Slettholt<br>NITH |
| [6] SMITH, 1997 | "Internet Cryptography"<br>Richard E. Smith<br>ISBN 0-201-92480-3 |
| [7] Suranjan, 2002 | "Public Key Infrastructure Implementation and design"<br>Suranjan Choudhury<br>ISBN 0-7645-4879-4 |
| [8] Ferguson, 2003 | "Practical Cryptography"<br>Niels Ferguson<br>ISBN 0-471-22357-3 |
| [9] Klaus, 2003 | "Cryptography and Public Key Infrastructure on the Internet"<br>Klaus Schmeh<br>ISBN 0 470 84745 X |
| [10] Garfinkel, 2002 | "Web Security"<br>Simson Garfinkel<br>ISBN 0-596-00045-6 |

[11] Stallins, 2003 ”Cryptography and networking”
William Stallins
ISBN 0-13-091429-0

[12] Rhee, 2003 “Internet Security”
Man Young Rhee
ISBN 0-470-85285-2

[13] Jay, 2002 ”Designing Security Architecture Solutions”
Jay Ramachandran
ISBN 0-471-20602-4

[14] Kapil, 2003 “PKI Security Solutions for the Enterprises”
Kapil Raina
ISBN 0-471-31529-X

[15] Terje Kolnes, 2003 “PKI dyttes”
Weekly newspaper of Telecom dated 6. mars 2003.

[16] Atif Ghauri, 2000 “PKI Honors Thesis”
http://ajconnections.net/pki/

[17] Dr. Stefan Brands, 2000 “Rethinking Public Key Infrastructures and Digital Certificates;”
ISBN 0-262-02491-8
http://www.xs4all.nl/~brands/chapter1.pdf

[18] INFOSEC Engineering, 1999 “Building a Corporate Public Key Infrastructure”
http://www.infoseceng.com/corppki.htm

[19] RSA Security, 2001 “Analysis of Symmetric and Asymmetric Key Lengths “
http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html

[20] Netscape Security, 1996 “SSL 3.0 SPECIFICATION”
http://wp.netscape.com/eng/ssl3/index.html

[21] Srinivasa Sivakumar, daily update “Sivakumar's Resource Site”

http://www3.brinkster.com/webguru/Sample.asp?start=151

[22] Nash and Duane, 2001 “PKI Implementation and Managing chapter 10”
http://www.osborne.com/networking_comm/0072131233/0072131233_ch10.pdf

[23] Nash and Duane, 2001 “PKI Implementation and Managing chapter 3”
http://www.osborne.com/networking_comm/0072131233/0072131233_ch03.pdf

[24] Douglas P. Barton, 1996 “Design Issues in a Public Key Infrastructure (PKI)”
http://www.csu.edu.au/special/auugwww96/proceedings/barmoroco/barmoroco.html