Certificate Generation In PKI: A New Approach

Report Submitted

by

Subhash Bhagat
M.Tech (CS)-II
Roll No. CS0902
Indian Statistical Institute

AS A PARTIAL FULFILLMENT OF THE DISSERTATION

Under The Guidance
of

Dr. Kishan Chand Gupta
Applied Statistics Unit

INDIAN STATISTICAL INSTITUTE, KOLKATA
203 B.T. ROAD,
KOLKATA-700108

Indian Statistical Institute

Kolkata-700 108

CERTIFICATE

This is to certify that the thesis entitled ”**Certificate Generation In PKI: A New Approach**” is submitted in the partial fulfilment of the degree of M.Tech. in Computer Science at Indian Statistical Institute, Kolkata. It is fully adequate in scope and quality as a dissertation for the required degree.

The thesis is faithfully record of bona fide research work carried out by Subhash Bhagat under my supervision and guidance. It is further certified that no part of this thesis has been submitted to any other University or Institute for the award of any degree or diploma.

Dr. Kishan Chand Gupta

(Supervisor)

Countersigned

(External Examiner)

Date:

Acknowledgement

My first and foremost heart-felt gratitude goes to my wonderful supervisor of this dissertation work, Dr. Kishan Chand Gupta. His instructions and continuous suggestions have guided me towards the better learning of the relevant literature and understanding the problems. He has not only guided me towards the discovery of problems but also helped me in all aspects including the preparation of this manuscript. This work has been possible because of his continuous suggestion, motivation, inspiration, and guidance towards my learning process. He has also given me the full freedom to think and explore new ideas and implementing those ideas. I also take this opportunity to thank all my teachers who have taught me throughout the semesters in M.Tech. course. I would like to convey a special thank to Mr. Sumit Kumar Pandey (Sumit Da) for helping me a lot through out my dissertation work. Finally I thank my family and friends for their endless support.

Place: Kolkata

Date:14.07.2011

Subhash Bhagat

Contents

Chapter 1

Introduction

Cryptography is an art of storing and transmitting data in a form that only those it is intended for can read and process. It is a science of protecting information by encoding it into an unreadable format. Cryptography is an effective way of protecting sensitive information as it is stored on media or transmitted through network communication paths. Cryptography comes from Latin words "*Crypt*" meaning "*secret*" and "*Graphia*" meaning "*writing*". So, Cryptography literally means "*secret writing*". The fundamental objective of cryptography is to enable two people, usually referred as Alice and Bob, to communicate over an insecure channel in such a way that an adversary, Oscar, can not understand what is being said. The information that Alice wants to send to Bob is called the "plaintext". Alice encrypts the plaintext using a predetermined key and sends the resulting "ciphertext" over the channel. Oscar, upon seeing the ciphertext in the channel by eavesdropping tries to determine the plaintext, but fails if the underlying cryptosystem is secure enough. Bob, who knows the encrypted key can decrypt the ciphertext and reconstructs the plain text. The Following schematic diagram shows the basic Cryptographic Model.

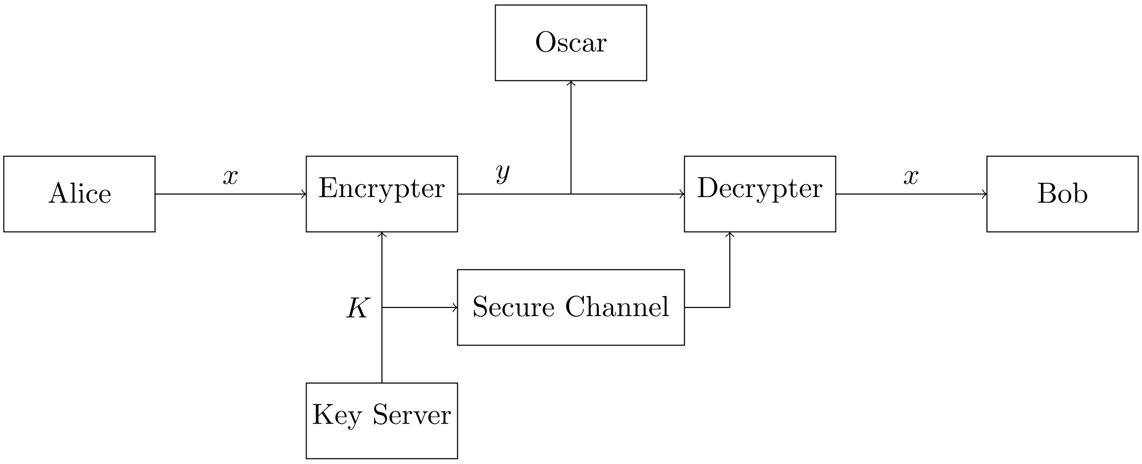

Figure 1.1: Basic Cryptographic Model

The following are the main four goals of modern cryptography:

- **Confidentiality:** *Prevent an unauthorized exposure of the information.* When transmitting data, one does not want an eavesdropper to understand the contents of the transmitted messages. The same is true for stored data that should be protected against unauthorized access.

- **Authentication:** *Prove that a message actually originates with its claimed originator.* The recipient should be able to verify from the message, the identity of the sender, the origin or the path it travelled (or combinations) so to validate claims from emitter or to validated the recipient expectations.
- **Integrity:** *Prove that a message has not been altered in unauthorized ways.* . The recipient should be able to determine if the message has been altered.
- **Non-repudiation:** *Prevent an originator from denying credit (or blame) for creating or sending a message.* The emitter should not be able to deny sending the message.

Definition 1. [1] *A Cryptosystem is a five tuple* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*, where the following conditions are satisfied:*

1. $\mathcal{P}$ *is a finite set of possible plaintext.*
2. $\mathcal{C}$ *is a finite set of possible ciphertext.*
3. $\mathcal{K}$*, the keyspace, is a finite set of possible keys.*
4. *For each* $K \in \mathcal{K}$*, there is an encryption rule* $e_K \in \mathcal{E}$ *and a corresponding decryption rule* $d_K \in \mathcal{D}$*. Each* $e_K : \mathcal{P} \to \mathcal{C}$ and $d_K : \mathcal{C} \to \mathcal{P}$ *are functions such that* $d_k(e_K(x)) = x$ *for very plaintext* $x \in \mathcal{P}$*.*

Note that to avoid ambiguous decryption, the function e_K should be an *injective function.*

Cryptography, depending on their encryption schemes can be broadly classified into two types, the *symmetric (or secret key)* schemes and *asymmetric (or public key)* schemes.

1.1 Symmetric Schemes.

In symmetric key schemes, both Alice and Bob share the same key for encryption and decryption. To provide privacy, this key needs to be kept secret. Once somebody else gets to know the key, it is not safe any more. Symmetric key schemes have the advantage of not consuming too much computing power. But the major drawback of symmetric key schemes is that it requires the prior communication of the secret key between Alice and Bob, using a secure channel, before any ciphertext is transmitted. In practice, this may very difficult to achieve(e.g. when they live far apart from each other). Also, for a system having n entities, we need $\binom{n}{2}$ private keys. Hence key management is a big issue in symmetric schemes. A few well-known examples are: DES, Triple-DES (3DES), IDEA, CAST5, BLOWFISH, TWOFISH.

1.2 Asymmetric or Public key schemes.

The concept of asymmetric key schemes was proposed to resolve the key management problem present in symmetric key schemes. Asymmetric key schemes use pairs of keys. One is used for encryption and the other one for decryption. The decryption key is typically kept secretly, therefore called *private key* or *secret key*, while the encryption key is spread to all who might want to send encrypted messages, therefore called *public key.* Everybody having the public key is able to send encrypted messages to the owner of the secret key. The secret key can't be reconstructed from the public key. Whenever Bob wants to send any plaintext to Alice, he simply encrypt the plaintext using public key of Alice and send the resulting ciphertext to Alice. Upon receiving the ciphertext from Bob, Alice uses her private key to decrypt the received ciphertext to get back the original plaintext. The idea of asymmetric key schemes was first published 1976 by Diffie and Hellmann. Well-known asymmetric key schemes are RSA, DSA, ELGAMAL.

Asymmetric key schemes seem to be ideally suited for real-world use: As the secret key does not have to be shared, the risk of getting known is much smaller. Every user only needs to keep one secret key in secrecy and a collection of public keys, that only need to be protected against being changed. To protect the public key against being changed, the concept of Certificates has been introduced in public key systems. We shall study the approaches regarding this in next chapter.

Asymmetric key schemes cost computationally more than the symmetric keys schemes. Therefore, in many applications, a combination of both is being used. The asymmetric keys are used for authentication and after this has been successfully done, one or more symmetric keys are generated and exchanged using the asymmetric encryption. This way the advantages of both algorithms can be used. We shall study this approach in chapter 4.

In next section we shall study the basic concepts of digital signature schemes which are used to sign the digital messages.

1.3 Signature Schemes

A signature scheme is a method of signing a message stored in electronic form. As such, a signed message can be transmitted over a computer network. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery or tampering. For messages sent through a non-secure channel, a properly implemented digital signature gives the receiver reason to believe the message was sent by the claimed sender. Digital signatures are equivalent to traditional handwritten signatures in many respects; properly implemented digital signatures are more difficult to forge than the handwritten type. Digital signatures can also provide non-repudiation, meaning that the signer cannot successfully claim they did not sign a message, while also claiming their private key remains secret; further, some non-repudiation schemes offer a time stamp for the digital signature, so that even if the private key is exposed, the signature is valid nonetheless. In 1976, Whitfield Diffie and Martin Hellman first described the notion of a digital signature scheme, although they only conjectured that such schemes existed. Soon afterwards, Ronald Rivest, Adi Shamir, and Len Adleman invented the RSA algorithm, which could be used to produce primitive digital signatures(RSA signatures are not secure). Following is the formal definition of a signature scheme.

Definition 2. [1] *A signature scheme is a five-tuple* $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, *where the following conditions are satisfied:*

1. $\mathcal{P}$ *is a finite set of possible messages.*

2. $\mathcal{A}$ *is a finite set of possible signatures.*

3. *$\mathcal{K}$, the keyspace, is a finite set of possible keys.*

4. *For each$K \in \mathcal{K}$, there is a signing algorithm $\mathbf{sig}_K \in \mathcal{S}$ and a corresponding verification algorithm $\mathbf{ver}_K \in \mathcal{V}$. Each $\mathbf{sig}_K : \mathcal{P} \rightarrow \mathcal{A}$ and $\mathbf{ver}_K : \mathcal{P} \times \mathcal{A} \rightarrow \{true, false\}$ are functions such that the following equation is satisfied for every message $x \in \mathcal{P}$ and for every signature $y \in \mathcal{A}$:*

$$ver_K(x,y) = \begin{cases} true & \text{if } y{=}\mathbf{sig}_K(x) \\ false & \text{if } y \neq \mathbf{sig}_K(x) \end{cases}$$

A pair (x, y) with $x \in \mathcal{P}$ and $y \in \mathcal{A}$ is called a signed message.

Now as an example of signature scheme we provide RSA signature scheme(although this scheme is not secure).

1.4 RSA Signature Scheme

Let $n = pq$, where p and q are two large primes. Let $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$ and define

$\mathcal{K} = \{(n, p, q, a, b) : n = pq, p, q$ are large primes $ab \equiv 1 \pmod{\phi(n)}\}$.

The values n and b are the public key, and the values p, q, a are the private key.

For $K = (n, p, q, a, b)$, define

$$\mathbf{sig}_K(x) = x^a \bmod n$$

and

$$\mathbf{ver}_K(x,y) = true \Leftrightarrow x \equiv y^b \pmod{n}$$

$(x, y \in \mathbb{Z}_n)$

Chapter 2

Public Key Infrastructure

2.1 Introduction

Here we shall study basic concepts of Public Key Infrastructure (PKI), i.e., the infrastructure needed to support public key cryptography. Public Key Infrastructure (PKI) is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates. A PKI is an arrangement that binds public keys with respective user identities by means of a certificate authority (CA). The user identity must be unique within each CA domain. The binding is established through the registration and issuance process, which, depending on the level of assurance the binding has, may be carried out by software at a CA, or under human supervision.

2.2 Brief History

The concept of Public Key Infrastructure (PKI) dates back to the work done by Diffie and Hellman in cryptography. In their 1976 paper, New directions in Cryptography [9], they introduced public key cryptography and claimed that the key management problem was solved. This was done by means of a modified telephone directory, which they called Public File. Instead of entries, each with name, address and phone number, the Public File would contain entries with name, number and public key. To send a confidential message, one would find the recipients public key by looking him up (by his name) in the Public File, and then encrypt the message with that public key before sending it to the recipient. Only the recipient, presumably, holding the corresponding private key can decrypt the message. As a result of the properties of public key cryptography, the public key need not be kept secret. The difficult problem of key management was solved but an equally difficult problem was introduced, namely, the problem of naming and name management.

In 1978, Kohnfelder, in his bachelor thesis at MIT [10], took up the problem of Public File and its performance in a network setting. To solve this, he proposed to take each entry of the Public File, namely the name and the public key, and digitally sign them. He coined the term certificate for this digitally signed version of the entries in the Public File. These certificates could then be distributed to anyone who wanted them.

In the 1980s, ITU (International Telecommunication Union) started an effort about building a directory like the one proposed by Diffie and Hellman. The directory was to cover all the people and devices in the world, and gather all information in one place. The result was a standard, known as X. 500 [11], defining all characteristics of such a directory. For authentication purposes, e.g., for granting permission to somebody to change an entry in the directory, a companion standard, X. 509, defining a certificate format was produced. A

X.509 certificate binds a public key to a Distinguished Name (DN), which can be thought of as a pathname into the X. 500 directory, and was supposed to be globally unique. For the signing of certificates, the notion of Certification Authority (CA) was introduced. This was supposed to be some trustworthy authority with a public key of his own, publicly available, who would then digitally sign the X.509 certificates.

The Privacy Enhanced Mail (PEM) [12] of ITEF made use of X. 509 certificates for the identification of mail recipients. This effort was carried out around 1990. PEM failed, mainly, because of lack of infrastructure; there were no Certification Authorities (CAs) in place to issue X. 509 certificates. To provide for certification without CAs, Pretty Good Privacy (PGP) [13] proposed another scheme. PGP allowed any keyholder to sign the key of any other keyholder, i.e., to issue certificates, thus forming a web of trust. The assumption was that multiple independent signatures on a certificate would be as trustworthy as the single signature of a CA on the same certificate.

Despite these various initiatives, still, PKI was not commonplace. In late 1990s, three independent initiatives (SDSI [14], SPKI [15], PolicyMaker [16]) started out, all based on the assumption that the PKI model itself was the problem. What they had in common was the way in which they differed from the traditional PKI model; they used the public key itself as the identifier of the keyholder. Of these more recent efforts, SPKI is still very active and gaining ground.

Lately, in an effort to make PKI more useful, some initiatives such as United States Federal Bridge [21], have tried to tackle the interoperability problems of the traditional PKI. But, a lot of open issues still need to be addressed

2.3 Traditional PKI

In this section we discuss the traditional approach to Public Key Infrastructure (PKI) and the different components that comprise a general purpose PKI.

2.3.1 PKI and its Components

A general purpose PKI is very complex and involves issues of different natures. All of them arise from the simple fact that in order to use a public key, one should have an assurance about the authenticity of the key, i.e., a guarantee that the public key in fact belongs to the entity that claims to own it. This guarantee of authenticity is achieved by means of a certificate, i.e., a digitally signed document binding the identity of the keyholder to its public key. Digital certificates stand therefore at the heart of PKI.

The introduction of certificates leads to the question of who is liable for the guarantee of authenticity of a public key? The answer is a Certification Authority (CA). Although conceptually simple, the certificate-CA pair involves quite a few technical and organizational challenges that necessitate the use of an appropriate infrastructure, i.e., a PKI. The main components of PKI are detailed in the remainder of this section.

- **Digital certificates**

  As mentioned above, a digital certificate associates an identity with the private-public key pair of the owner of the identity; therefore, the identity must be unique.

The widely used certificate formats are all based on X. 509v3 [17]. The basic information contained in a certificate is:

- Subject: the individual or entity being identified by the certificate.
- Public key: the public key of the subject, corresponding to its private key.
- Issuer: the trusted authority that has generated and signed the certificate.
- Serial number: a unique identifier for the certificate.
- Validity period: a date indicating the earliest time the certificate can be used and a date indicating the expiration of the certificate.
- Usage: the description of the usage for which the corresponding private-public key pair is valid.
- Digital signature: the digital signature of the issuer.

Although all based on the same basic format, certificates from different CAs are according to different profiles, use different extensions and ascribe different semantics to the attributes in a certificate. This situation creates problems with respect to naming. The lack of standards for naming, apart from the defined fields and attributes for encoding names, makes the task of processing names very hard. Actually, the directory and the naming issues are considered by many experts in the field as the major issues in PKI, in general, and as an obstacle to interoperability in particular [18].

- **Certification authorities(CA).**
  The main task of a CA is to issue certificates. In order to do so, a CA needs a private-public key pair for the applicant and means to properly identify the applicant. The process begins with the user providing the CA with sufficient information about his identity. After a satisfactory verification of the supplied identity credentials, the CA generates a public-private key pair for the user1, and creates a certificate for the generated public key; the private key must be transferred in a secure way to the applicant who must store it in a secure storage for later use. Currently, most of this process is software supported.
  Many systems require the ability to recover a lost key in order to access information previously encrypted by that key. In such cases, the users private keys are backed up by the CA or a separate key recovery system.
  As mentioned earlier, certificates expire; CAs have therefore the renewal of certificates as part of their task. The process of renewing a certificate is much simpler. The user presents the certificate it has and by proving access to the corresponding private key, his identity can be verified. The certificate can then be renewed immediately if there is no change in the identification information. Another task of the CAs is to revoke certificates when necessary, make the fact known to all possible users of the certificate, and manage the revocation status for all the certificates they have issued. This is discussed in more details in a later section.

- **Registration Authorities(RA).**
  Generally, it is assumed that a CA operates out of a vault where his (private) signing key is very strongly protected. The cost of such a facility is very high and there

cannot be very many of them. A scenario in which the users, i.e., those applying for certificates, should present themselves to the CA, with proof of their identities, would be too expensive for the users who might then have to travel a long way. Moreover, the larger the number of certificate-users that a CA must manage, the harder it becomes to verify their identities. In other words, the verification of an identity is more trustworthy when the CA is close to the actual user. To remedy this situation, a Registration Authority (RA) was introduced. That is, there are many RAs for each CA such that the users can find one close at hand. The task of the RAs is to verify the identity credentials that the users present and, if approved, start the certification process with a CA.

- **Repository.**
  A repository is a database of active digital certificates for a CA system. The main business of the repository is to provide data that allows users to confirm the status of digital certificates for individuals and businesses that receive digitally signed messages. These message recipients are called relying parties. CAs post certificates and CRLs to repositories.

- **Archive.**
  An archive is a database of information to be used in settling future disputes. The business of the archive is to store and protect sufficient information to determine if a digital signature on an old document should be trusted.

- **PKI users.**
  PKI Users are organizations or individuals that use the PKI, but do not issue certificates. They rely on the other components of the PKI to obtain certificates, and to verify the certificates of other entities that they do business with. End entities include the relying party, who relies on the certificate to know, with certainty, the public key of another entity; and the certificate holder, that is issued a certificate and can sign digital documents. Note that an individual or organization may be both a relying party and a certificate holder for various applications.

2.3.2 Deciding on the length of the key

Cryptography is based on mathematical problems that are extremely hard to solve. Given the proper amount of time and resources, attackers can break any cryptographic key. In general, the longer the key, the harder it is to break; therefore, the choice of the length of he key has very much to do with how long the information to which it is applied should be protected. If the information needs to be protected, e.g. , for five years, then the length should be chosen such that, given the current technology, it would take more than five years to break it.

2.3.3 Trust models and certification paths

Users must trust the CA, which includes the ability to verify the CAs signature on the certificate. That requires safe knowledge of the authenticity of the CAs public key.
Ideally, a single CA, trusted by all users, would issue all the certificates. In this case, the

users could get the CAs public key in a safe way from the CA and use it for certificate validation. However, in the real world, the model with a single CA is not achievable for both organizational and technical reasons, e.g. , different countries having different laws, different needs in different application domains, or technical problems for a single CA to manage a large global population. It is therefore best to have many CAs each being responsible for a subset of the user population. The model with multiple CAs while solving the problems involved in the single CA model, introduces problems of its own. That is, users will have to trust many different CAs, and must be able to verify signatures from all of them. The burden of managing multiple CAs is not acceptable for most users; the remedy to this situation is to build trust relationships between different CAs. Trust relationships between CAs means that CAs not only issue certificates to users and other entities such as application servers and network routers, but, also to other CAs, i.e., a CA will certify the identity of other CAs. In all trust models, the certificate user or relying party must have an initial trust in some entity of the model. This initially trusted entity is called the trust anchor for that relying party. Each relying party has the public key of his trust anchor.
To validate a received certificate which is not issued by the relying parties trust anchor, the relying party should follow a set of trust relationships from the CA that issued the received certificate to the CA that is his trust anchor. This results in following a chain of certificates, corresponding to the CAs in the set of trust relationships, also called certification path. The length of this path is critical in the performance of a deployed PKI.

- **Hierarchical model**
  The most widely used trust model is a strict hierarchy, where the subordinate CAs are certified by the parent CA, but not vice versa. In this model, the root CA of the hierarchy is trusted by all relying parties, i.e., it is the sole trust anchor in the model. It has a self-signed certificate and all relying parties have a copy of its public key. This model has several benefits:

  - All certificate paths terminate with the root CA certificate; the length of the certification path depends therefore only on the depth of the hierarchy2.
  - There is only one certification path for each end-entity. This makes it possible to provide a path to the root CA to the relying party by having the end-entities include the certificates of all the CAs on the path along with their own certificates.

  The main drawback of this model is that it is not possible for the whole population of certificate users to agree on a single root CA, which will be the common trust anchor. This model fits best smaller organizations with a hierarchical structure; it has been used in Privacy Enhanced Mail (PEM) and more recently, by the U. S. Department of Defence [22].

- **Peer-to-peer model**
  In a peer-to-peer trust model, there is no hierarchy and thereby no root CA and no single trust anchor; any CA can establish a trust relationship to any other peer CA and thus issue a certificate for that CA, i.e., cross-certify the other CA. In such a model, users trust local CAs.
  In a fully connected mesh of cross-certifications, where each CA cross-certifies all the other CAs, the certification path is very short, but the proportion of the number of

needed cross-certificates to the number of CAs is of the order of n^2. This model is not a very useful one and presents problems with respect to certificate distribution.
A more useful cross-certification model is one that allows longer certification paths and a partially connected mesh. Cross-certification is most useful where subordination cannot be applied and between different trust domains, e.g. , different organizations. This model also has some drawbacks. One is that since certification paths are longer and traverse several domains, the same level of trust cannot be guaranteed implying decrease in trust in certificates, e.g. , some trust domains have a more strict identification process than others. As there might exist several paths between two end-entities, another drawback is the complexity involved in the selection of an optimal certification path. A third drawback is that as new trust relationships are made and new CAs are added to the mesh by existing CAs, it becomes harder to see whether ones trust relationship has been extended to some not fully trusted entity. This brings about the issue of authorization. That is, in addition to PKI establishing identities, there is a need for an authorization system that lets one assign different access rights to ones system to different certificate holders. Authorization is handled by Privilege Management Infrastructure (PMI). It manages all aspects of users rights and privileges by issuing, distributing, revoking, and storing Attribute Certificates (ACs). Attribute certificate (AC) binds attributes to an AC holder. The public key certificate must first be used to perform authentication, then the AC is used to associate attributes with the authenticated identity.
An attempt to deploy a cross-certification model is the US federal bridge initiative [21]. This is a project where different government agencies can authenticate each others using the bridge CA. They operate with several levels of assurance, and the approach is standard-driven.

- **Hybrid trust models**
  A more flexible trust model can be obtained by mixing hierarchical and cross-certification models. For example, enterprises with hierarchical structures can deploy a hierarchical trust model internally, while conducting inter-enterprise business by deploying a cross-certification model. Another example of use of a hybrid model is in dealing with very large hierarchies; some often-used, long certification paths can be optimized by establishing a direct cross-certification link between the two leaf-CAs involved in the path.

2.3.4 Certificate revocation and validation

A certificate is valid only within a period of time indicated in the certificate. But, it can be revoked, i.e., declared invalid by its issuer, before its expiration time. There are different reasons for revoking a certificate, e.g. , a key being compromised as a result of a security attack on a system or the owner of the certificate leaving the company that had issued the certificate to him. It is therefore crucial to verify the validity of a received certificate with respect to both its validity period and its revocation status. Certificate validation is the process that determines whether a certificate can be accepted as being valid. It involves checking different aspects of the certificate:

- The digital signature on the certificate must be verified both to ensure that the certificate has not been tampered with and that the signature is that of the authority that issued it.

- The time at which the certificate is being checked must be within the validity period of the certificate, which is usually one to two years.
- The revocation status of the certificate must be checked to ensure that it is not revoked.
- Syntax and semantics of the certificate must be checked to ensure that the format is right, all the mandatory fields and critical extensions are present. It is important that all critical fields are well understood.
- It must be checked that the use of the certificate use is according to the purpose for which it was created.

As for the revocation process, when notified of some ground requiring the revocation of a certificate, a CA must take action to revoke the certificate and must advise all potential users of the certificate of the fact. The most widely used mechanism for revocation is based on Certificate Revocation Lists (CRL) as described in [36]. A CA must publish periodically a CRL containing the certificates it has revoked. Each CRL entry consists of the serial number of the revoked certificate along with the revocation date and reason. To ensure the integrity of CRL, it is signed by the CA (or some trusted revocation service). In order to allow for the use of the freshest release, the CRL contains the date it has been published and a date for the next release.

To check for the revocation status of a certificate, the recipient of a certificate downloads the CRL, checks whether the CRL is up-to-date (not an old copy), checks the CAs signature on the CRL, and lastly, checks whether the certificates serial number is on the list.

Provided that each received certificate should be validity-checked, this mechanism will easily bring the server providing the CRL service to its knees, creating a denial of service situation. The Online Certificate Status Protocol (OCSP) [19] was designed to ease this situation. OCSP provides more timely information about the revocation status of a certificate and is supposed to be faster than the CRL mechanism. An OCSP service is provided either directly by a CAs or by an authorized responder. For each status request, the service checks the status of the corresponding certificate directly with the CA, and can therefore provide a more timely status than that provided by periodical CRLs. Note that the status returned by the OCSP responder pertains only to the revocation of the certificate and indicates nothing about the validity status of the certificate, i.e., whether the certificate is still within its validity period.

2.3.5 Certificate policy and certificate practice statement

A CA operates based on a Certificate Policy (CP) and/or Certification Practice Statement (CPS) covering legal and technical aspects of the certificates and the process of issuing them. X. 509 [17] defines a Certificate Policy as:

"A named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. For example, a particular certificate policy might indicate applicability of a type of certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range."

The Certification Practice Statement (CPS) is defined by the American Bar Association [20] as follows:
“A certification practice statement may take the form of a declaration by the certification authority of the details of its trustworthy system and the practices it employs in its operations and in support of issuance of a certificate, or it may be a statute or regulation applicable to the certification authority and covering similar subject matter. It may also be part of the contract between the certification authority and the subscriber. A certification practice statement may also be comprised of multiple documents, a combination of public law, private contract, and/or declarations.”

2.4 Usages

PKIs of one type or another, and from any of several vendors, have many uses, including providing public keys and bindings to user identities which are used for:

- Encryption and/or sender authentication of e-mail messages (e.g. using OpenPGP or S/MIME).
- Encryption and/or authentication of documents (e.g. the XML Signature or XML Encryption standards if documents are encoded as XML).
- Authentication of users to applications (e.g. smart card logon, client authentication with SSL). There’s experimental usage for digitally-signed HTTP authentication in the Enigform and modopenpgp projects.
- Bootstrapping secure communication protocols, such as Internet key exchange (IKE) and SSL. In both of these, initial set-up of a secure channel (a ”security association”) uses asymmetric key (i. e public key) methods, whereas actual communication uses faster symmetric key (i. e secret key) methods.
- Mobile signatures are electronic signatures that are created using a mobile device and rely on signature or certification services in a location independent telecommunication environment.

2.5 Conclusion

A public key infrastructure (PKI) allows public key cryptography to be employed on a broad scale. With a PKI, parties who have not met in person are able to engage in verifiable transactions. The identity of the originator of a message can be traced to the owner of the private key as long as there is strong binding between the owner and the owners public key. A PKI provides the means to bind public keys to their owners and helps in the reliable distribution of public keys in large heterogeneous networks. Public keys are bound to their owners by public key certificates. These certificates contain information such as the owners name and the associated public key and are issued by a reliable Certification Authority (CA).
A PKI is often composed of many CAs linked by trust paths. The CAs may be linked in several ways. They may be arranged hierarchically under a ”root CA” that issues certificates to subordinate CAs. The CAs can also be arranged independently in a network. Recipients of a signed message with no relationship with the CA that issued the certificate

for the sender of the message can still validate the senders certificate by finding a path between their CA and the one that issued the senders certificate.

The confidence that can be placed on the binding between a public key and its owner depends much on the confidence that can be placed on the CA that issued the certificate that binds them. Here we have studied an overview of PKI functions and their applications.

Chapter 3

Identity Based Encryption and Signature Schemes

3.1 Introduction

In 1984, Shamir [6] proposed a concept of identity-based cryptography. In this new paradigm of cryptography, users' identifier information such as email or IP addresses instead of digital certificates can be used as public key for encryption or signature verification. As a result, identity-based cryptography significantly reduces the system complexity and the cost for establishing and managing the public key authentication framework known as Public Key Infrastructure (PKI).

Although Shamir [6] easily constructed an identity-based signature (IBS) scheme using the existing RSA function, he was unable to construct an identity-based encryption (IBE) scheme, which became a long-lasting open problem. Only recently in 2001, Shamir's open problem was independently solved by Boneh and Franklin [5] and Cocks [23].

Identity based encryption (IBE), introduced by Shamir, enables the computation of a public key for an entity, given only some general scheme parameters and a string identifying the entity (e.g. an e-mail address, a telephone number, etc.). A private key generator (PKG) computes private keys from a master secret and distributes these to the entities participating in the scheme. This eliminates the need for certificates as used in a traditional public key infrastructure.

Shamir's original motivation for identity-based encryption was to simplify certificate management in e-mail systems. When Alice sends mail to Bob at "bob@company.com" she simply encrypts her message using the public key string "bob@company.com". There is no need for Alice to obtain Bob's public key certificate. When Bob receives the encrypted mail he contacts a third party, which we call the Private Key Generator (PKG). Bob authenticates himself to the PKG in the same way he would authenticate himself to a CA and obtains his private key from the PKG. Bob can then read his e-mail. Note that unlike other existing secure e-mail infrastructure, Alice can send encrypted mail to Bob even if Bob has not yet setup his public key certificate. Also note that key escrow is inherent in identity-based e-mail systems: the PKG knows Bob's private key. Here we shall study the generic construction of the IBE and IBS(Identity Based Signature) systems and also the IBE scheme proposed by Boneh and Franklin and IBS proposed by Shamir .

3.2 Basic Concepts of Identity-Based Encryption Scheme

As mentioned earlier, in the IBE scheme, the sender Alice can use the receiver's identifier information which is represented by any string, such as email or IP address, even a digital image, to encrypt a message. The receiver Bob, having obtained a private key associated with his identifier information from the trusted third party called the "Private Key Generator (PKG)", can decrypt the ciphertext. Summing up, we describe an IBE scheme using the following steps. (Figure 1 illustrates a schematic outline of an IBE scheme).

- **Setup:** The PKG creates its master (private) and public key pair, which we denote by sk_{PKG} and pk_{PKG} respectively. (Note that pk_{PKG} is given to all the interested parties and remains as a constant system parameter for a long period.)
- **Private Key Extraction:** The receiver Bob authenticates himself to the PKG and obtains a private key $sk_{ID_{Bob}}$ associated with his identity ID_{Bob}.
- **Encryption:** Using Bob's identity ID_{Bob} and the PKG's pk_{PKG}, the sender Alice encrypts her plaintext message M and obtains a ciphertext C.
- **Decryption:** Upon receiving the ciphertext C from Alice, Bob decrypts it using his private key $sk_{ID_{Bob}}$ to recover the plaintext M.

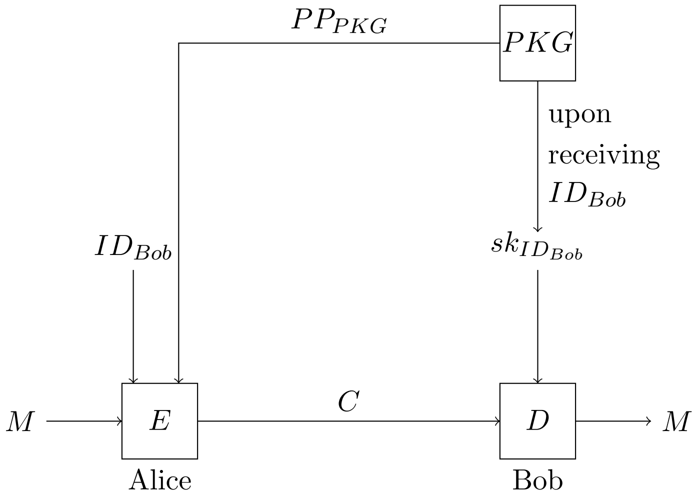

Figure 3.1: Identity Based Encryption

3.3 Basic Concepts of Identity-Based Signature Scheme

As a mirror image of the above identity-based encryption, one can consider an identity-based signature (IBS) scheme. In this scheme, the signer Alice first obtains a signing (private) key associated with her identifier information from the PKG. She then signs a message using the signing key. The verifier Bob now uses Alice's identifier information to verify Bob's signature. No needs for Bob to get Alice's certificate. More precisely, an IBS scheme can be described using the following steps. (Figure 2 illustrates a schematic outline of an IBS scheme).

- **Setup:** The Private Key Generator (PKG), which is a trusted third party, creates its master (private) and public key pair, which we denote by sk_{PKG} and pk_{PKG} respectively.
- **Private Key Extraction:** The signer Alice authenticates herself to the PKG and obtains a private key $sk_{ID_{Alice}}$ associated with her identity ID_{Alice}.
- **Signature Generation:** Using her private key $sk_{ID_{Alice}}$, Alice creates a signature σ on her message M.
- **Signature Verification:** Having obtained the signature σ and the message M from Alice, the verifier Bob checks whether σ is a genuine signature on M using Alice's identity ID_{Alice} and the PKG's public key pk_{PKG}. If it is, he returns "*Accept*". Otherwise, he returns "*Reject*".

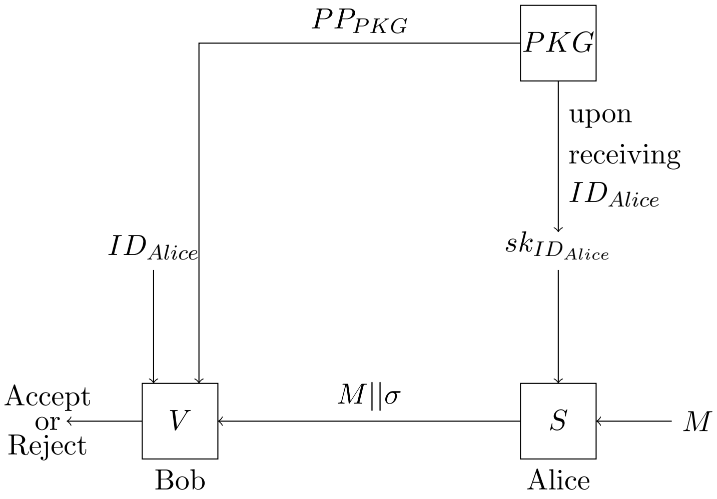

Figure 3.2: Identity Based Signature

3.4 Boneh-Frankline IBE Scheme

Bilinear Pairing and the Bilinear Diffie-Hellman Assumption:
The admissible bilinear pairing $\hat{e}$ is defined over two groups of the same prime-order q denoted by $\mathcal{G}$ and $\mathcal{F}$. (By $\mathcal{G}^*$ and $\mathbb{Z}_q^*$, we denote $\mathcal{G}\backslash\{O\}$ where O is the identity element of $\mathcal{G}$, and $\mathbb{Z}_q\backslash\{0\}$ respectively.)We will use an additive notation to describe the operation in $\mathcal{G}$ while we will use a multiplicative notation for the operation in $\mathcal{F}$. In practice, the group $\mathcal{G}$ is implemented using a group of points on certain elliptic curves, each of which has a small MOV exponent [27], and the group $\mathcal{F}$ will be implemented using a subgroup of the multiplicative group of a finite field. The admissible bilinear map, denoted by $\hat{e} : \mathcal{G}\times\mathcal{G} \rightarrow \mathcal{F}$, has the following properties.

- *Bilinear:* $\hat{e}(aR_1, bR_2) = \hat{e}(R_1, R_2)^{ab}$, where $R_1, R_2 \in \mathcal{G}$ and $a, b \in \mathbb{Z}_q^*$.
- *Non-degenerate:* $\hat{e}$ does not send all pairs of points in $\mathcal{G} \times \mathcal{G}$ to the identity in $\mathcal{F}$. (Hence, if R is a generator of $\mathcal{G}$ then $\hat{e}(R, R)$ is a generator of $\mathcal{F}$.)
- *Computable:* For all $R_1, R_2 \in \mathcal{G}$, the map $\hat{e}(R_1, R_2)$ is efficiently computable.

we will simply use the term "bilinear pairing" to refer to the admissible bilinear pairing defined above.

Bilinear Diffie-Hellman Assumption. The above bilinear pairing gave rise to the following computational problem called "Bilinear Diffie-Hellman(BDH)" problem:

- Given $(\mathcal{G}, q, \hat{e}, P, aP, bP, cP)$ where a, b, and c are chosen at random from $\mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc}$.

The BDH assumption means that the above problem is computationally intractable. the security of many identity-based cryptographic schemes in the current literature depends on the BDH assumption (or its variation).

BDH Parameter Generator. We say that a randomized algorithm $\mathcal{H}$ is a BDH parameter generator if (1) $\mathcal{H}$ takes a security parameter $k \in \mathbb{Z}^+$, (2) $\mathcal{H}$ runs in polynomial time in k, and (3) $\mathcal{H}$ outputs a prime number q, the description of two groups $\mathcal{G}, \mathcal{F}$ of order q, and the description of an admissible bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$. We denote the output of $\mathcal{H}$ by $\mathcal{H}(1^k) =< q, \mathcal{G}, \mathcal{F}, \hat{e} >$. The security parameter k is used to determine the size of q; for example, one could take q to be a random k-bit prime. we assume that the descriptions of the groups $\mathcal{G}$ and $\mathcal{F}$ contain polynomial time (in k) algorithms for computing the group action in $\mathcal{G}$ and $\mathcal{F}$ and contain a generator of $\mathcal{G}$ and $\mathcal{F}$ respectively. The generators of $\mathcal{G}$ and $\mathcal{F}$ enable us to generate uniformly random elements in $\mathcal{G}$ and $\mathcal{F}$ respectively. Similarly, we assume that the description of $\hat{e}$ contains a polynomial time algorithm for computing $\hat{e}$.

We now describe Boneh and Franklin's famous IBE scheme. We let $\mathcal{H}$ be some BDH parameter generator.

- **Setup:** Given a security parameter $k \in \mathbb{Z}^+$, the algorithm works as follows:
  Step 1: Run $\mathcal{H}$ on input k to generate a prime q, two groups $\mathcal{G}, \mathcal{F}$ of order q, and an admissible bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$. Choose a random generator $P \in \mathcal{G}$.
  Step 2: Pick a random $s \in \mathbb{Z}_q^*$ and set $pk_{PKG} = sP$, the public key of the PKG.
  Step 3: Choose a cryptographic hash function $H_1 : \{0, 1\}^* \rightarrow \mathcal{G}^*$. Choose a cryptographic hash function $H_2 : \mathcal{F} \rightarrow \{0, 1\}^l$ where l denotes the lenght of the plaintext. The security analysis will view H_1, H_2 as random oracles. The message space is $\mathcal{M} = \{0, 1\}^l$. The ciphertext space is $\mathcal{C} = \mathcal{G}^* \times \{0, 1\}^l$. The system parameters are params $=< q, \mathcal{G}, \mathcal{F}, \hat{e}, l, P, pk_{PKG}, H_1, H_2 >$. The **master-key** is $s \in \mathbb{Z}_q^*$.

- **Private Key Extraction:** For a given string $ID \in \{0, 1\}^*$ the algorithm does:
  (1) computes $Q_{ID} = H_1(ID) \in \mathcal{G}*$, and
  (2) sets the private key d_{ID} to be $d_{ID} = sQ_{ID}$ where s is the master key.

- **Encryption:** To encrypt $M \in \mathcal{M}$ under the public key ID do the following:
  (1) compute $Q_{ID} = H_1(ID) \in \mathcal{G}^*$,
  (2) choose a random $r \in \mathbb{Z}_q^*$, and
  (3) Compute $U = rP$, , $g_{ID} = \hat{e}(Q_{ID}, pk_{PKG}) \in \mathcal{F}$ and $V = M \oplus H_2(g_{ID}^r)$. Set the ciphertext to be $C =< U, V >$

- **Decryption:** Let $C =< U, V >\in \mathcal{C}$ be a ciphertext encrypted using the public key ID. To decrypt C using the private key $d_ID \in \mathcal{G}^*$ compute:

$$V \oplus H_2(\hat{e}(d_{ID}, U)) = M$$

This completes the description of Boneh-Franklin IBE-Scheme. We first verify consistency. When everything is computed as above we have:
1. During encryption M is bitwise exclusive-ored with the hash of: g_{ID}^r.
2. During decryption V is bitwise exclusive-ored with the hash of: $\hat{e}(d_{ID}, U)$.
These masks used during encryption and decryption are the same since:

$$\hat{e}(d_{ID}, U) = \hat{e}(sQ_{ID}, rP) = \hat{e}(Q_{ID}, P)^{sr} = \hat{e}(Q_{ID}, pk_{PKG})^r = g_{ID}^r$$

Thus, applying decryption after encryption produces the original message M as required. Note that the value of g_{ID} in Algorithm *Encryption* is independent of the message to be encrypted. Hence there is no need to recompute g_{ID} on subsequent encryptions to the same public key ID.

3.5 Shamir's IBS

In his introductory paper of IBE and IBS, Shamir proposed a signature scheme which is based on the famous RSA algorithm. Following is the scheme proposed by Shamir. Though in his original paper Shamir used the term Key Generation Center(KGC) instead of PKG(having same functionality), we shall stick by the term PKG.

- **Setup(1^k):** Given a security parameter $k \in \mathbb{Z}^+$, the algorithm works as follows:
  Step 1. Choose two large primes p and q such that $2^{k-1} < p < q < 2^k$
  Step 2. Compute $n = pq$
  Step 3. Compute $\phi(n) = (p-1)(q-1)$
  Step 4. Choose e such that $gcd(e, \phi(n)) = 1$
  Step 5. compute d such that $ed \equiv 1 \bmod \phi(n)$
  Step 6. choose h a cryptographic hash function.

  Public parameters are $< n, e, h >$ and d is the secrete *master-key*

- **Private Key Extraction:** For a given string $ID \in \{0,1\}^*$ the algorithm does:
  Step 1. sets the private key b_{ID} to be $b_{ID} = ID^d \bmod n$ where d is the master key.

- **Signature Generation:** To sign a message M under the identity ID do the following:
  Step 1. Choose random number r
  Step 2. compute $t = r^e \bmod n$ and $s = b_{ID}.r^{h(t||M)} \bmod n$
  The signature is $\sigma =< t, s >$.

- **Signature Verification:** After receiving the message M and the signature $\sigma =< s, t >$ on M under the identity ID do the following:
  Step 1. Check $s^e = ID.t^{h(t||M)} \bmod n$
  Step 2. If above verification is true then *accept* otherwise *reject.*

We now verify the consistency of the above signature scheme. When every thing is computed as above, recipient knows ID, M, public parameters $< n, e, h >$ and signature $\sigma < t, s >$ where $t = r^e \bmod n$ and $s = b_{ID}.r^{h(t||M)} \bmod n$. Then

$$\begin{aligned} &s^e{=}(b_{ID}.r^{h(t||M)})^e \bmod n \\ &\equiv (ID^d.r^{h(t||M)})^e \bmod n. \text{ [as } b_{ID} = ID^d \bmod n \text{]} \\ &\equiv ID^{de}.r^{h(t||M)e} \bmod n \\ &\equiv ID.(r^e)^{h(t||M)} \bmod n. \text{ [as } ed \equiv 1 \bmod \phi(n)] \\ &\equiv ID.t^{h(t||M)} \bmod n. \text{ [as } t = r^e \bmod n] \end{aligned}$$

Hence the signature scheme is consistent.

3.6 Issues in IBE.

IBE eliminates the use of certificates which was necessary in traditional PKI. But it does not come with smaller cost. All proposed IBE schemes cost computationally more than the traditional PKI. Following are two major draw-backs of IBE schemes other than this.

Key Escrow Problem. Unfortunately, all identity-based cryptographic schemes have inherent weakness, a ”key escrow” property. Recall that in IBE and IBS schemes, the PKG issues private keys for user using its master secret key. As a result, the PKG is able to decrypt or sign any messages. Hence the use of identity-based cryptography may be limited to the environment where the PKG is unconditionally trusted, for example, inside of a company or a particular organization.

Revocation Problem. In non-identity-based cryptography, the revocation of the public key is a big problem in that users who want encrypt messages or verify signatures should first check whether the concerning public keys have been revoked or not. To do this, current PKI requires to maintain Certificate Revocation List (CRL). Management of CRLs may be one of the factors that slows down the deployment of PKI. In identity-based schemes, this problem no longer exists as any identities can be served as public keys. However, another kind of revocation problem occurs in identity-based cryptography. Suppose that Bob wants others to use his email address to encrypt messages and the private key associated with Bob's email address has been compromised, so he cannot use his email address as a public key any more. Does he have to obtain new email address?

3.7 Conclusion

Shamir's original motivation for identity-based encryption was to simplify certificate management in e-mail systems and that has been achieved via IBE schemes. But it also has its own weaknesses like Key escrow problem and Revocation problem. But it has opened a new area of research in Cryptography with many promises to be achieved further.

Chapter 4

Certificate Generation In PKI: Our Approach

4.1 Introduction

In previous chapter we have studied an overview of public Key Infrastructure(PKI) and its traditional components. The main component of PKI is digital certificate that binds public keys with respective user identities by means of a certificate authority (CA). Here we shall study the different techniques of issuing certificates. The development of Public Key Cryptography has widen the field of applications of secure cryptosytems. Currently, the most widely used application using a PKI solution is the web-browser. When online retailers appeared on the World Wide Web, experts realized that this new form of cryptography (public-key cryptography) could allow these businesses to conduct secure transactions. The used PKI solution is greatly simplified. The browsers have a pre installed set of root certificates of trusted CAs and the web-servers they communicate with may have a certificate issued by one of those trusted CAs. Those CAs are mainly commercial ones and have a self-issued root certificate. It is the implementor of the web-browser that has taken the decision of which CAs to trust. Note that, the end-users can install root certificates of other CAs that they trust in their web-browser. Most browsers check the validity of the web-servers certificate with respect to its validity period but do not support certificate revocation. The commonly used security protocol for exchange of information between the web- browsers and servers is the Secure Socket Layer (SSL) protocol(see [7] for details). Here we shall study the existing approach of issuing certificates in SSL protocol along with one proposed approach based on IBE(Identity Based Encryption)to replace certificates in SSL. Here we also propose a new approach to generate certificates in PKI based on IBS scheme(Identity-Based Signature schemes) and compare its efficiency and security with those two previously mentioned approaches.

4.2 Overview of CA based SSL Protocol

1. Browser creates a random value and sends it to the server along with supported cipher methods.

2. Server creates a random value and sents it, its public key certificate(s)and selected cipher method to the browser.

3. Brower validates server's certificate and determines if communicating with the correct server.

4. Browser generates a random value and encrypts it with the server's public key
5. Browser sends that encrypted value to the server.
6. Server uses its private key to decrypt the received value
7. Browser and server uses that shared secret value and two publicly exchanged random values to create shared secret values for encryption(symmetric key encryption) and integrity.
8. Browser sends MAC of messages(a, b & e) to the server.
9. Server sends MAC of the messages(a, b & e) to the browser
10. A secure pipe between the server and the browser has been established i.e. server and browser have agreed upon some secret symmetric key which they use now to complete the required transaction.(see figure 4.1)

For further details see [7].

4.2.1 Certificate issuance in CA based SSL.

In SSL, certificates are issued to the web-servers by the trusted CA. Followings are the steps in this regard:

- CA issues a certificate containing the Identity of the server, Public key of the server, signature on the public key of the server using CA's private key, Public parameters and other essential informations. CA provides this certificate to the server. CA also provides the public key of CA corresponding to the private key used for signing the public key of the web-server to the web-browsers.
- On receiving the certificate from CA, server sends it to the web-browser whenever browser requests for it. Web-browser verifies the signature of CA on public key of server contained in the certificate using the CA's corresponding public key.
- After Verification, browser then extracts the public key of the server and uses it to encrypt the necessary data needed for the transaction and send it to the server. On receiving this encrypted data from the browser, server uses its private key to decrypt the encrypted data. Server and browser execute the steps stated in 4.2 to agree upon some symmetric key. They use this symmetric key to complete the transaction.

4.3 Main drawbacks of CA based SSL:

1. Certificates are issued by CA. So, CA has to maintain a log file of issued certificates so that if at any future instance any party which has been issued a certificate wants to verify the authenticity of his certifcate, then CA can verify the authenticity of the certificate by viewing the log file. Also CA has to maintain a list of revoked certificates i. e. a Certificate revocation list(CRL) containing the status information of the certificates indicating whether the certificates are currently valid. So on CA side there are much overhead of maintaining, updating and providing these list in regular intervals besides generating the certificates.
2. If we compromise the private key of the CA by which it signs on public keys of the servers, then adversary can impersonate any server which relies on it whenever he wants. Adversary simply choose a private and public key pair. Then, He signs on the chosen public

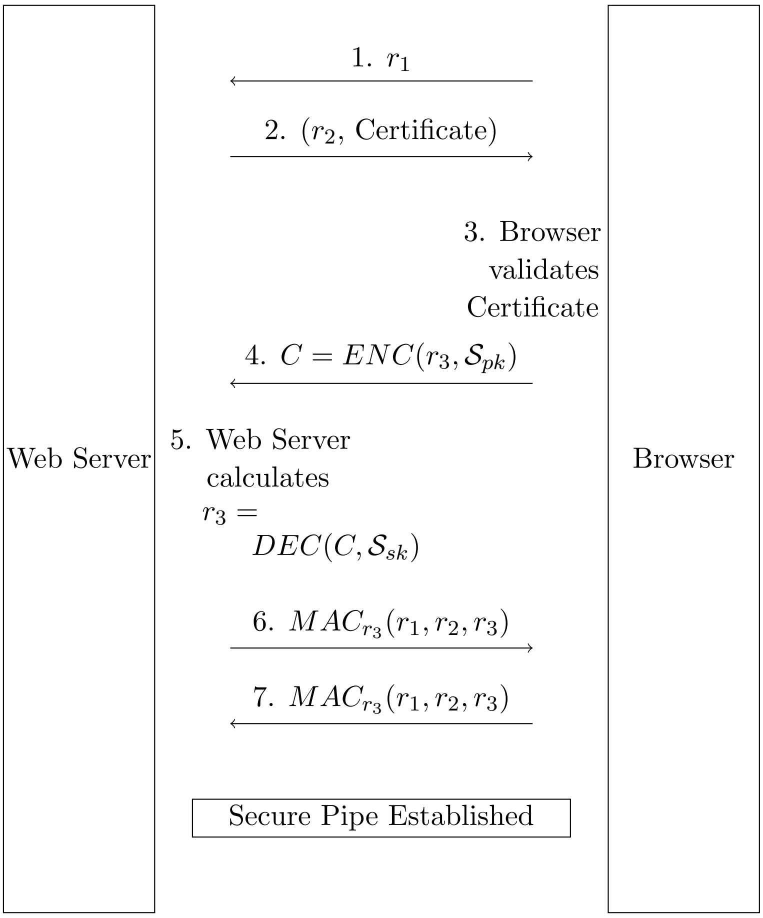
Web Server
Browser
1. r_1
2. (r_2, Certificate)
3. Browser validates Certificate
4. $C = ENC(r_3, \mathcal{S}_{pk})$
5. Web Server calculates $r_3 = DEC(C, \mathcal{S}_{sk})$
6. $MAC_{r_3}(r_1, r_2, r_3)$
7. $MAC_{r_3}(r_1, r_2, r_3)$
Secure Pipe Established

Figure 4.1: Handshake Protocol (SSL)

key by the CA's private key and generates a new certificate containing this public key and the corresponding signature on it. This is a valid certificate signed by the CA's private key. So, in this approach when we compromise the private key of the CA, the trust put on the CA compromises the security of that system on which the particular servers rely on. So, if we compromise this private key of CA, then adversary can impersonate any server that relies on it whenever it wants.
/

4.4 IBE-SSL Approach

In [8], an alternate approach was proposed which uses the IBE. They have omitted the need of certificate on the server side and introduced the master certificate embedded in web browsers for the PKG. Following are the steps in this regards:

- Instead of CA there is a trusted PKG(Public Key Generator) and PKG issues a master certificate which will be embedded in web browsers for the PKG.

- PKG sets the private key of the server and provides it to the server secretly.

- when web-browser wants to send some data to the server, it then encrypt data with the URL of the server (e.g. amazon. com) and send the encrypted text to the server. The server would get the encrypted message from the client and decrypt it with its private key that it has received from the PKG. After this they exchange the requisite information between them to agree upon some symmetric key. Using this symmetric key they completes the required transaction.(see figure 4.2 & figure 4.3)

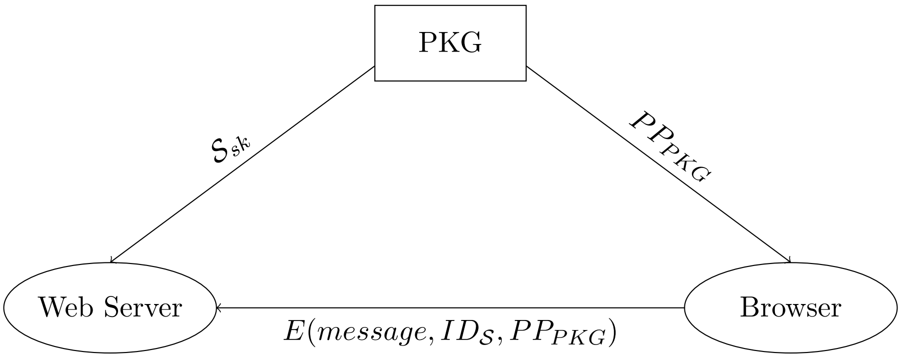

Figure 4.2: Diagram of IBE-SSL interactions

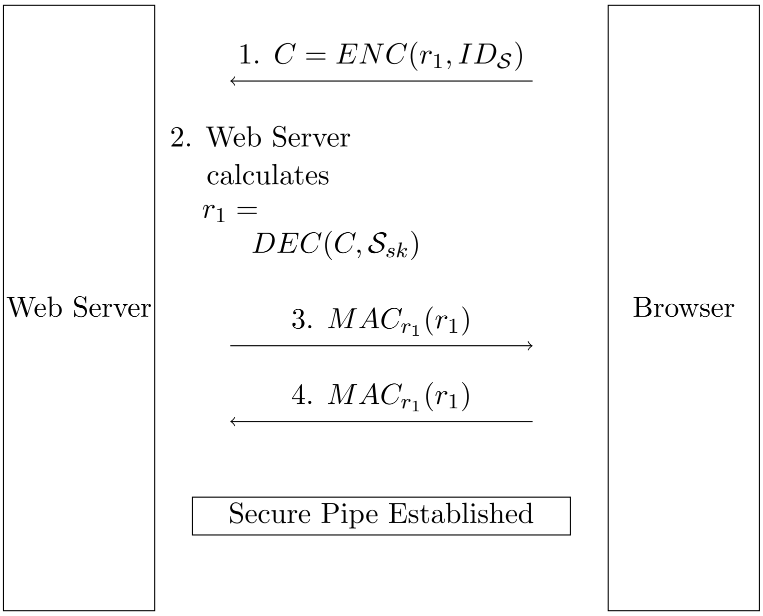
Web Server
Browser
1. $C = ENC(r_1, ID_{\mathcal{S}})$
2. Web Server calculates $r_1 = DEC(C, \mathcal{S}_{sk})$
3. $MAC_{r_1}(r_1)$
4. $MAC_{r_1}(r_1)$
Secure Pipe Established

Figure 4.3: Handshake Protocol (IBE-SSL)

4.5 Main drawbacks of IBE-SSL:

1. Suppose we compromise the master secret key of PKG. Then the following scenario can be happened:

- Suppose $\mathcal{S}$ is a server and $\mathcal{T}_{PKG}$ is the PKG on which server $\mathcal{S}$ relies to provide the secret key. The server $\mathcal{S}$ will use this secret key to establish a secure session with the browsers. Now suppose a browser $\mathcal{B}$ wants to communicate with the server $\mathcal{S}$ securely. So first $\mathcal{B}$ has to establish a secure pipe with $\mathcal{S}$(i. e. $\mathcal{S}$ and $\mathcal{B}$ have to agree upon a secret symmetric session key). To do this $\mathcal{B}$ sends to the server $\mathcal{S}$ the requisite information encrypted with the public parameters provided by $\mathcal{T}_{PKG}$ and the identity of the server $\mathcal{S}$. The adversary gets those information simply by listening the channel shared between the server $\mathcal{S}$ and the browser $\mathcal{B}$. Now, the adversary who holds the master secret key of the PKG $\mathcal{T}_{PKG}$ generates the private key of the server $\mathcal{S}$ using the master secret key. Then the adversary, uses this private key of $\mathcal{S}$ to decrypt these information to get the private session key shared between the server $\mathcal{S}$ and the browser $\mathcal{B}$ and $\mathcal{S}$ and $\mathcal{B}$ can never know about that. Since the adversary now knows the private session key shared between $\mathcal{S}$ and $\mathcal{B}$, he can passively read all the transaction between $\mathcal{S}$ and $\mathcal{B}$ simply by decrypting the ciphers sent between $\mathcal{S}$ and $\mathcal{B}$ using the private session key(which is symmetric key). So, whenever the adversary wants, he can passively read all the transaction between $\mathcal{S}$ and $\mathcal{B}$ until $\mathcal{T}_{PKG}$ changes its master secret key.

- Further, if the adversary plays an active role, he can actively change the actual cipher texts communicated between the server $\mathcal{S}$ and the browser $\mathcal{B}$ to wrong cipher text(meaningful) by tapping the channels because the adversary knows the private session key shared between $\mathcal{S}$ and $\mathcal{B}$. The adversary simply takes control of the channels and whenever $\mathcal{S}$ and $\mathcal{B}$ communicate with each other via some messages encrypted with the private session key, the adversary discards this intended cipher and injects his own cipher text encrypted with the session key into the channel. Even if $\mathcal{S}$ and $\mathcal{B}$ uses MAC, since the adversary knows all the secret values agreed upon by $\mathcal{S}$ and $\mathcal{B}$, he can do the same.

- In addition to that the adversary can also impersonate the server $\mathcal{S}$ whenever it wants. The adversary takes control of the channel and presents himself as $\mathcal{S}$ to the browser $\mathcal{B}$. The adversary decrypts with the private key of $\mathcal{S}$ all the messages sent by $\mathcal{B}$ which is trying to communicate with $\mathcal{S}$ and sends requisite information to convince $\mathcal{B}$ that he is the intended server $\mathcal{S}$. In this way the adversary can establish a secure pipe with $\mathcal{B}$ by the name of $\mathcal{S}$.
  Hence if adversary has access to the master secrete key of the PKG $\mathcal{T}_{PKG}$ then he can get private key of any server that relies on $\mathcal{T}_{PKG}$. So, if we compromise the master secret key of $\mathcal{T}_{PKG}$ then the security of the whole system based around $\mathcal{T}_{PKG}$ is gone. This is a very fatal disadvantage.
  Note that even if the adversary only gets access to the private key of the server $\mathcal{S}$ given by the PKG, then also he can do the above forgeries to the server $\mathcal{S}$.

2. All the existing IBE schemes are costly compared to non-IBE based PKCs. In, IBE-SSL, browser $\mathcal{B}$ uses an IBE scheme to encrypt the requisite information to send the server $\mathcal{S}$ for establishment of an secure transaction pipe via which they complete the transaction. So the computational cost is high in case of IBE-SSL.

4.6 Our Approach.

Here we introduce a new approach for issuance of the certificates which is based on Identity-Based-Signature(IBS) scheme. Following are the steps in this regards:

1. We replace CA with PKG. Suppose $\mathcal{C}_{PKG}$ is a PKG. First $\mathcal{C}_{PKG}$ will run its own setup algorithm and generate public parameters and a *master secret key.* $\mathcal{C}_{PKG}$ keeps the *master secret key* securely and make public the public parameters.

2. Suppose a server $\mathcal{S}$ approaches to $\mathcal{C}_{PKG}$ after verification from Registration authority(RA). Then following steps are executed:

   step *(i).* $\mathcal{S}$ provides its identity and the public key $\mathcal{S}_{pk}$ to $\mathcal{C}_{PKG}$.
   step *(ii).* $\mathcal{C}_{PKG}$ using its *master secret key* and identity of $\mathcal{S}$ generates a private key $\mathcal{S}^{sk}_{PKG}$ for the server $\mathcal{S}$. Further more, $\mathcal{C}_{PKG}$ using its *master secret key* generates signature $\sigma_{\mathcal{C}_{PKG}}$ on the public key $\mathcal{S}_{pk}$. Then $\mathcal{C}_{PKG}$ provides $\mathcal{S}^{sk}_{PKG}$ and the signature $\sigma_{\mathcal{C}_{PKG}}$ on $\mathcal{S}_{pk}$ to the server $\mathcal{S}$.
   step *(iii).* $\mathcal{S}$ keeps the private key $\mathcal{S}_{sk}$ corresponding to $\mathcal{S}_{pk}$ and the private key $\mathcal{S}^{sk}_{PKG}$ given by $\mathcal{C}_{PKG}$ securely. The server $\mathcal{S}$ stores the signature $\sigma_{\mathcal{C}_{PKG}}$, $\mathcal{S}_{pk}$ and $\mathcal{S}_{sk}$ securely for future uses. $\mathcal{S}$ will use $\sigma_{\mathcal{C}_{PKG}}$ and $\mathcal{S}^{sk}_{PKG}$ for authentication and $\mathcal{S}_{pk}$ and $\mathcal{S}_{sk}$ to establish a secure session with browser, say, $\mathcal{B}$.

3. After the interaction with the PKG $\mathcal{C}_{PKG}$, the server $\mathcal{S}$ executes the following steps:

   step *(i).* The server $\mathcal{S}$ uses its identity , the private key $\mathcal{S}^{sk}_{PKG}$ given by $\mathcal{C}_{PKG}$ and the public parameters of $\mathcal{C}_{PKG}$ to generate a signature $\sigma_{\mathcal{S}}$ on the public key $\mathcal{S}_{pk}$.
   step *(ii).* $\mathcal{S}$ then generates a certificate of its own containing its identity, public key $\mathcal{S}_{pk}$, its signature $\sigma_{\mathcal{S}}$ on public key $\mathcal{S}_{pk}$ and other essential information.

4. Suppose browser $\mathcal{B}$ tries to connect with the server $\mathcal{S}$ first time. Then following steps are executed

   step *(i).* $\mathcal{B}$ sends a random number and two null string to the server $\mathcal{S}$ and asks for the certificate of the server $\mathcal{S}$ and the signature $\sigma_{\mathcal{C}_{PKG}}$ of $\mathcal{C}_{PKG}$ on the public key $\mathcal{S}_{pk}$.
   step *(ii).* On receiving the request from $\mathcal{B}$, watching the null strings, server $\mathcal{S}$ understands that $\mathcal{B}$ is trying to connect with it for the first time. Then, $\mathcal{S}$ sends requested items and a list $\mathcal{L}^{pk}_{\mathcal{S}}$ of public keys with time stamp used by the server $\mathcal{S}$ to the browser $\mathcal{B}$.
   step *(iii).* After receiving the certificate from $\mathcal{S}$, browser $\mathcal{B}$ uses the public parameters of the PKG $\mathcal{C}_{PKG}$ to verify the signature $\sigma_{\mathcal{C}_{PKG}}$ of $\mathcal{C}_{PKG}$ on the public key $\mathcal{S}_{pk}$ and uses identity of $\mathcal{S}$, public parameters of $\mathcal{C}_{PKG}$ to verify the signature $\sigma_{\mathcal{S}}$ of the server $\mathcal{S}$ on the public key $\mathcal{S}_{pk}$.
   step *(iv).* After a successful verification, browser $\mathcal{B}$ first encrypts the requisite messages with the public key $\mathcal{S}_{pk}$ of the server $\mathcal{S}$ and sends the encrypted messages to $\mathcal{S}$. After receiving the encrypted messages from $\mathcal{B}$, server $\mathcal{S}$ decrypt the ciphers with its private key $\mathcal{S}_{sk}$ to get back the original messages sent by $\mathcal{B}$. They perform the necessary steps(as stated in 4.2) to establish a secure communication pipe between them(i.e. they agree upon some symmetric key called the *session key*) to complete the the transaction.(see figure 4.4)

5. After a successful first transaction between $\mathcal{S}$ and $\mathcal{B}$, $\mathcal{B}$ saves the list $\mathcal{L}^{pk}_{\mathcal{S}}$ sent by the server $\mathcal{S}$. When next time the browser $\mathcal{B}$ will try to communicate with the server $\mathcal{S}$, following steps are executed:
   step *(i).* Firstly $\mathcal{B}$ will choose randomly a public key $\mathcal{S}^{r}_{pk}$ from the list $\mathcal{L}^{pk}_{\mathcal{S}}$ and a random number r_b.
   step *(ii).* Then $\mathcal{B}$ encrypts this random number r_b with the chosen public key $\mathcal{S}^{r}_{pk}$ to generate r^{c}_{b}.

step *(iii).* $\mathcal{B}$ then sends the public key $\mathcal{S}^r_{pk}$ chosen from the list $\mathcal{L}^{pk}_{\mathcal{S}}$ with time stamp and the encrypted random number r^c_b to $\mathcal{S}$ and asks $\mathcal{S}$ to sign on r_b with $\mathcal{S}^r_{sk}$ and to send the signature on r_b to $\mathcal{B}$.
step *(iv).*Then $\mathcal{S}$ decrypts r^c_b with the private key $\mathcal{S}^r_{sk}$ corresponding to the public key $\mathcal{S}^r_{pk}$ to get back r_b and put its signature on r_b using the private key $\mathcal{S}^r_{sk}$ and sends only the signature to $\mathcal{B}$. $\mathcal{S}$ stores the random number r_b and sends it's MAC to $\mathcal{B}$ just before the establishment of the secure pipe along with other MAC values(see steps h & i in section 4.2).
step *(v).* On receiving this signature, $\mathcal{B}$ will verify the signature with the decided public key $\mathcal{S}^r_{pk}$ to verify whether it is trying to connect with the indended server $\mathcal{S}$ or not. After this verification $\mathcal{B}$ and $\mathcal{S}$ performs the all the steps stated above to establish a secure communicational pipe and to complete the required transaction.(see figure 4.5)
Note that in second and onwards interaction between $\mathcal{S}$ and $\mathcal{B}$, $\mathcal{S}$ does need not to send the signature $\sigma_{\mathcal{C}_{PKG}}$ of $\mathcal{C}_{PKG}$ on the first public key $\mathcal{S}_{pk}$ until PKG $\mathcal{C}_{PKG}$ changes its master secret key and public parameters. Hence until $\mathcal{C}_{PKG}$ changes its master secret key and public parameters, $\mathcal{S}$ will use the first private-public key pair ($\mathcal{S}_{sk}$, $\mathcal{S}_{pk}$) and the signature $\sigma_{\mathcal{C}_{PKG}}$ on $\mathcal{S}_{pk}$ for the browsers which try to connect it firstly and asked for the $\sigma_{\mathcal{C}_{PKG}}$ from the server $\mathcal{S}$. Otherwise $\mathcal{S}$ uses its current private-public key pair for the browser with which $\mathcal{S}$ has interacted already. If $\mathcal{C}_{PKG}$ changes its master secret key and public parameters, then the whole approach will be repeated.

6. Server $\mathcal{S}$ has to maintain a list $\mathcal{L}^{prev}_{\mathcal{S}}$ of private-public key pairs(with time stamp) used by it. $\mathcal{S}$ updates this list whenever it changes its private-public key pairs or whenever needed. Also $\mathcal{S}$ has to generate the list $\mathcal{L}^{pk}_{\mathcal{S}}$ from the list $\mathcal{L}^{prev}_{\mathcal{S}}$ containing the public keys (with time stamp) used by $\mathcal{S}$ and has to provide $\mathcal{L}^{pk}_{\mathcal{S}}$ to the browsers. $\mathcal{S}$ updates this list whenever needed and sends the updated list to the browsers. Browsers maintain the list $\mathcal{L}^{pk}_{\mathcal{S}}$ and update it when modified list is available. Initially the list $\mathcal{L}^{prev}_{\mathcal{S}}$ contains the $\mathcal{S}_{sk}$ and $\mathcal{S}_{pk}$ with the corresponding time stamp and the list $\mathcal{L}^{pk}_{\mathcal{S}}$ contains $\mathcal{S}_{pk}$ with the corresponding time stamp. In sub sequential time whenever $\mathcal{S}$ changes its private-public key pairs, it just simply put the replaced pairs with time stamp in the list $\mathcal{L}^{prev}_{\mathcal{S}}$ and updates the list $\mathcal{L}^{pk}_{\mathcal{S}}$ accordingly.

7. The PKG $\mathcal{C}_{PKG}$ also maintain a list $\mathcal{L}^{revok}_{\mathcal{C}_{PKG}}$ of revoked identities. This list contains the revoked identities of the servers due to change of identities of servers or found in some violation of the acts or caught in doing frauds or some other reasons. $\mathcal{C}_{PKG}$ provides this list $\mathcal{L}^{revok}_{\mathcal{C}_{PKG}}$ to the browsers and updates it whenever it is changed by $\mathcal{C}_{PKG}$. So, before starting communication with a server, browser consults with this list for the validity of the identity of the server with which it intends to communicate.

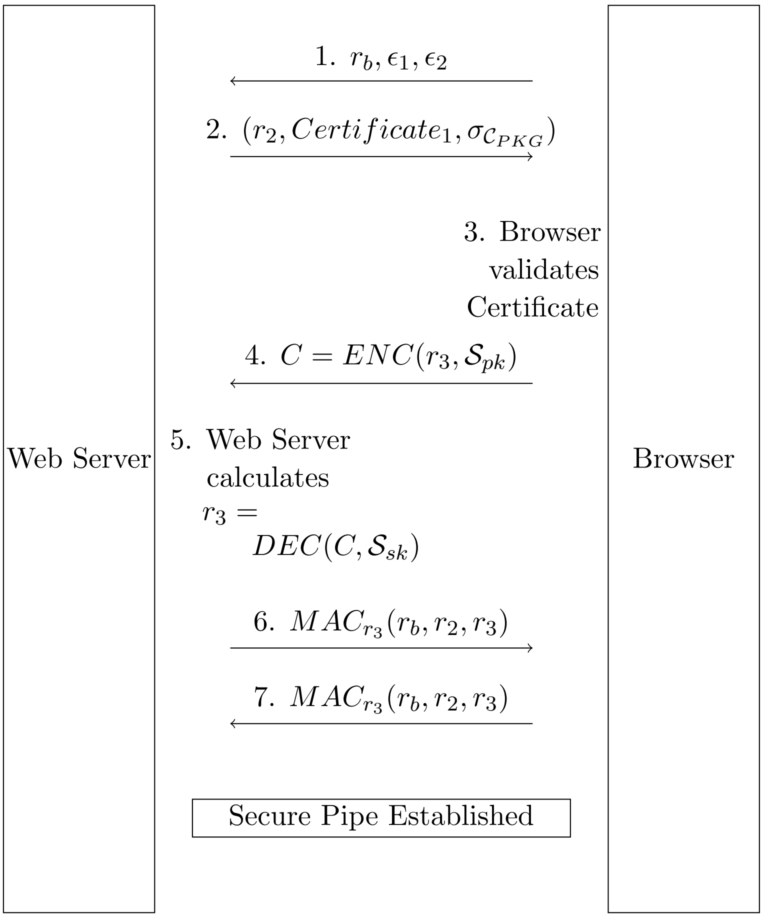
1. $r_b, \epsilon_1, \epsilon_2$
2. $(r_2, Certificate_1, \sigma_{\mathcal{C}_{PKG}})$
3. Browser validates Certificate
4. $C = ENC(r_3, \mathcal{S}_{pk})$
Web Server
5. Web Server calculates $r_3 = DEC(C, \mathcal{S}_{sk})$
Browser
6. $MAC_{r_3}(r_b, r_2, r_3)$
7. $MAC_{r_3}(r_b, r_2, r_3)$
Secure Pipe Established

Figure 4.4: Handshake Protocol (Proposed SSL) : First Session

4.7 Advantages of Our approach over the CA based SSL

1. In our approach server $\mathcal{S}$ can generate his own certificates whenever it wants. So the overhead on the side of PKG is reduced compared to the overhead of CA in CA based SSL. Even more, PKG has not to maintain any log file of issued private keys to the servers for checking the validity of the issued key in future references. Whenever, server $\mathcal{S}$ wants to verify that if he has been provided a correct private key by the PKG $\mathcal{C}_{PKG}$ or not, server $\mathcal{S}$ can do it by itself. Sever $\mathcal{S}$ selects any plaintext p and encrypt it with the public parameters of $\mathcal{C}_{PKG}$ and its own identity to get cipher text c and finally he decrypt this cipher text c to see that whether he is getting back the original plaintext p or not. By this verification method any server can verify the validity of the private key provided by the PKG. So, the overhead on the side of PKG is reduced.
2. We have seen that if we compromise the private key of CA, then adversary can impersonate any server that relies on the CA. In our scheme after one successful establishment of secure pipe between the server $\mathcal{S}$ and the browser $\mathcal{B}$, suppose an adversary get access to the master secret key of the PKG. Then the adversary can not impersonate the server $\mathcal{S}$ in sub sequential time whenever $\mathcal{B}$ will try to communicate with $\mathcal{S}$. After establishment of one secure pipe, in the next interaction with the server $\mathcal{S}$, $\mathcal{B}$ firstly starts executing the steps stated in **5**(section 4.6) of our approach to verify if $\mathcal{B}$ is trying to communicate with the intended server $\mathcal{S}$ or not.

4.8 Advantages of Our Approach over IBE-SSL

Here by $\mathcal{C}_{PKG}$ we denote the PKG of our approach.

1. We have seen that if we compromise the master secret key of the PKG in IBE-SSL, then adversary can impersonate the server whenever he wants until the PKG changes its master secret key. In our scheme after one successful establishment of secure pipe between the server $\mathcal{S}$ and the browser $\mathcal{B}$, suppose an adversary get access to the master secret key of the PKG. Then the adversary can not impersonate the server $\mathcal{S}$ in sub sequential time whenever $\mathcal{B}$ will try to communicate with $\mathcal{S}$. Because after establishment of one secure pipe, in the next interaction with the server $\mathcal{S}$, $\mathcal{B}$ firstly starts executing the steps stated in **5** of our approach to verify if $\mathcal{B}$ is trying to communicate with the intended server $\mathcal{S}$ or not. Note that in IBE-SSL if we compromise only the private key of $\mathcal{S}$ given by the PKG, then also adversary can impersonate the server $\mathcal{S}$ whenever he wants until PKG changes its master secret key and public parameters. But in our approach if adversary gets access only to the private key $\mathcal{S}^{sk}_{PKG}$ of $\mathcal{S}$, then he can not impersonate the server $\mathcal{S}$ at any time because he can not get the signature of $\mathcal{C}_{PKG}$ on his chosen public key which is essential for authentication to browsers.

2. Again we have seen that if adversary knows the private key of the server given by the PKG in case of IBE-SSL, he can passively read all the transaction between the server and browser. He can also actively change the messages shared between the server and the browser whenever he wants. In our approach the PKG $\mathcal{C}_{PKG}$ provides private key sk_{PKG} corresponding to the identity of the server and a signature on the public key $\mathcal{S}_{pk}$ to the server $\mathcal{S}$ and public parameters of the system. Server $\mathcal{S}$ keeps the private key corresponding to $\mathcal{S}_{pk}$ securely. Suppose that the adversary only get access the private key $\mathcal{S}^{sk}_{PKG}$ of the server $\mathcal{S}$. Then, in our approach, the adversary can not read any encrypted information exchanged between $\mathcal{S}$ and $\mathcal{B}$ in the process of establishment of the secret session key sk_{ss}, since the adversary does not know the private key $\mathcal{S}_{sk}$ of $\mathcal{S}$. Hence he can not get the secret session key. Thus, if we compromise only the private key $\mathcal{S}_{sk}$ of the server, adversary can not read passively or change actively any data exchanged between the server $\mathcal{S}$ and the browser $\mathcal{B}$ when the actual transaction is being proceeded.

3. Our approach has lower computational cost because it uses IBS and in literature there exists cost efficient IBS schemes which have less computational cost compared to existing IBE schemes.

4.9 Conclusion.

Here we have studied the overview of the approaches to generate certificates in PKI and studied the disadvantages of the existing two approaches, CA based SSL and IBE-SSL. We have proposed a new approach to generate certificates in PKI and discussed the advantages over the existing CA based SSL and IBE-SSL. Our approach has more communication over head with increased security compared to the other two stated approaches.

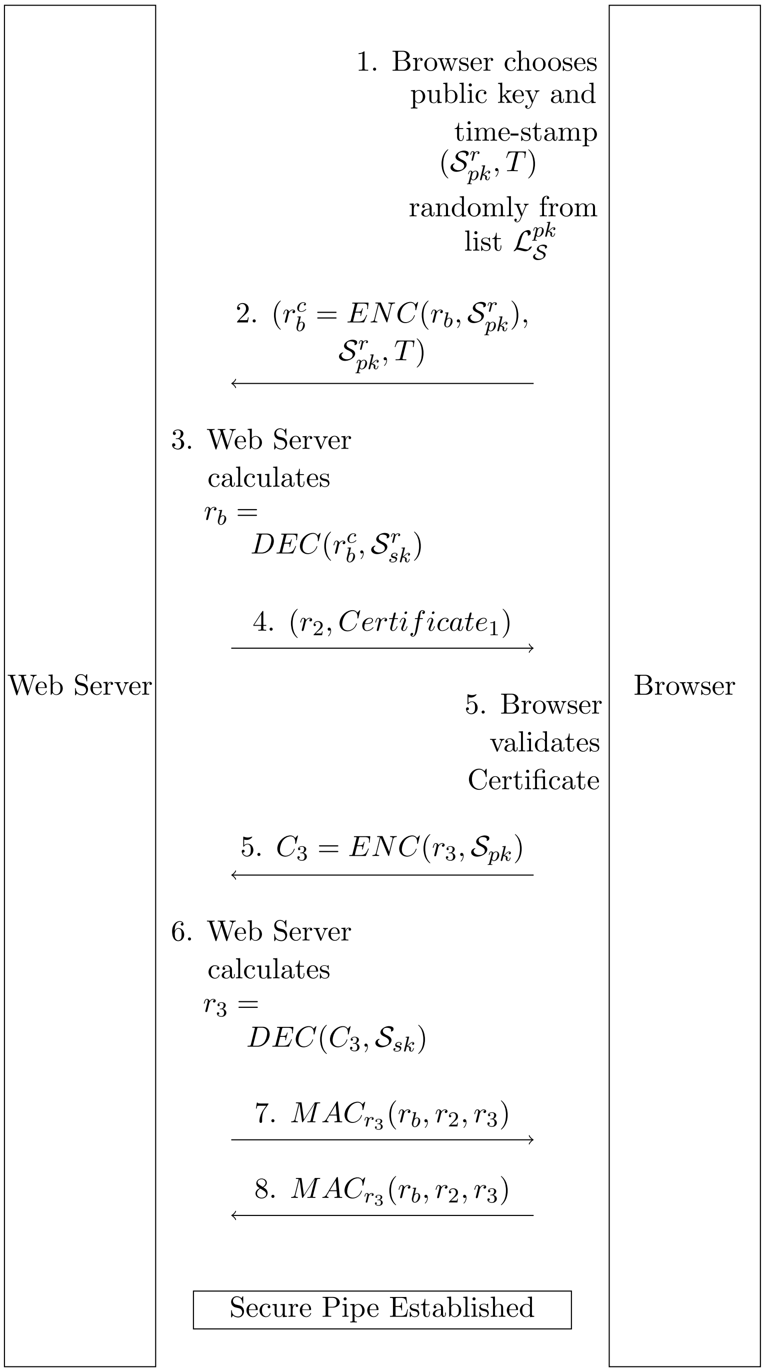
Web Server
Browser
1. Browser chooses public key and time-stamp $(\mathcal{S}^{r}_{pk}, T)$ randomly from list $\mathcal{L}^{pk}_{\mathcal{S}}$
2. $(r^{c}_{b} = ENC(r_b, \mathcal{S}^{r}_{pk}), \mathcal{S}^{r}_{pk}, T)$
3. Web Server calculates $r_b = DEC(r^{c}_{b}, \mathcal{S}^{r}_{sk})$
4. $(r_2, Certificate_1)$
5. Browser validates Certificate
5. $C_3 = ENC(r_3, \mathcal{S}_{pk})$
6. Web Server calculates $r_3 = DEC(C_3, \mathcal{S}_{sk})$
7. $MAC_{r_3}(r_b, r_2, r_3)$
8. $MAC_{r_3}(r_b, r_2, r_3)$
Secure Pipe Established

Figure 4.5: Handshake Protocol (Proposed SSL) : After First Session

Bibliography

[1] D. R. Stinson. *Cryptography theory and practice.* Chapman and Hall/CRc, 3rd edition.

[2] S. Mazaher and P. Re, *A survey of state of the art in public key infrastructure*, August 2003.

[3] D. R. Kuhn, V. C. Hu, W. T. Polk, and S. -J. Chang, *Introduction to public key technology and the federal PKI infrastructure*, http: //www. csrc. nist. gov/publications/nistpubs/800-32/ sp800-32. pdf, NIST, February 2001.

[4] J. Baek, J. Newmarch, R. Naini and W. Susilo, *A Survey of Identity-Based Cryptography* AUUG 2004, pp. 95-102, 2004.

[5] D. Boneh and M. Franklin. *Identity-based encryption from the Weil pairing.* SIAM J. Computing, 32(3): 586615, 2003. Extended abstract in Proceedings of Crypto 2001.

[6] Adi Shamir. *Identity-based cryptosystems and signature schemes.* In Crypto 84, LNCS Vol. 196, pages 4753. Springer, 1985.

[7] Franck Martin.*SSL Certificates HOWTO.* http://www.gtlib.cc.gatech.edu/pub/linux/docs/ HOWTO/other-formats/html single/SSL-Certificates-HOWTO.html.

[8] J. Adam Sowers. *Using Identity-Based Encryption to Eliminate Certificates in SSL Transactions.* A Thesis In TCC 402,University of Virginia,March 2002

[9] Whitfield Diffie and Martin E. Hellman.*New directions in cryptography.* IEEE Transactions on Information Theory, IT-22(6):644654, 1976.

[10] Kohnfelder, L. M., *Towards a Practical Public Key Cryptosystem.* MIT S.B. Thesis, May 1978.

[11] ITU-T, The Directory - overview of concepts, models and service. *International Telecommunications Union, X.500 series of Recommendations*, 1993.

[12] Kent, S. T., Internet Privacy Enhanced Mail.*Communications of the ACM*, August 1993

[13] Zimmermann, P. R., *PGP users Guide.* MIT, October 1994.

[14] Rivest, R. L. and Lampson, B., SDSI *A Simple Distributed Security Infrastructure.* Presented at CRYPTO'96 Rump session, 1996. http:// citeseer.nj.nec.com/rivest96sdsi.html.

[15] Ellison, C. et al., *SPKI Certificate Theory.* IETF RFC 2693, September 1999. http://www.ietf.org/rfc/rfc2693.txt

[16] Chokani, S. and Ford, W., *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, IETF PKIX WG, RFC2527, March 1999.

[17] ITU-T /ISO, OSI The Directory: Authentication Framework. ITU-T X.509 / ISO/IEC 9594-8, 1997.

[18] Ellison, C., *Improvements on Conventional PKI Wisdom.* Proceedings of the 1st Annual PKI Research Workshop, August 2002.

[19] Myers, M., Ankney, R., Malpani, A. and Adams, C.,*X.509 Internet Public Key Infrastructure Online Certificate Status Protocol OCSP. IETF PKIX WG, RFC 2560*, June 1999.

[20] American Bar Association. *Digital Signature Guidelines: Legal Infrastructure for Certification Authorities and Secure Electronic Commerce*, August 1996. http://www.abanet.org/scitech/ec/isc/digital_signature.html

[21] Federal Bridge Certification Authority Homepage, http://www.cio.gov/fbca/

[22] Green, R. M: and Harris, B., *United States DoD Public Key Infrastructure: Deploying the PKI Token.* 1st Annual PKI Research Workshop, August 2002.

[23] C. Cocks, *An Identity Based Encryption Scheme Based on Quadratic Residues*, Cryptography and Coding - Institute of Mathematics and Its Applications International Conference on Cryptography and Coding, Proceedings of IMA 2001, LNCS 2260, pages 360-363, Springer-Verlag, 2001.