SECURITY SYSTEM FOR DNS USING CRYPTOGRAPHY

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

UNDER THE SUPERVISION OF

PROF. DR. SATYA PRAKSH GHRERA

BY

SAKSHI– 111282

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT, SOLAN- 173234, HIMACHAL PRADESH

CERTIFICATE

This is to certify that project report entitled “SECURITY SYSTEM FOR DNS USING CRYPTOGRAPHY”, submitted by SAKSHI(111282) in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Prof. Dr. Satya Prakash Ghrera
(Head Of The Department ,CSE)
Department of CSE & IT
Jaypee University of Information Technology
Waknaghat, Solan , H.P- 173234

Date:

ACKNOWLEDGEMENT

On the very outset of this report, I would like to extend my sincere & heartfelt obligation towards all the personages who have helped me in this endeavor. Without their active guidance, help, cooperation & encouragement, I would not have made headway in the project.

I would like to show my greatest appreciation to Mr. Prof. Dr. Satya Prakash Ghrera. I feel motivated every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I feel fortunate to be taught by Prof. Dr. Satya Prakash Ghrera, who gave me his unwavering support. Besides being my mentor, he taught me that there is no substitute for hard work.

In the light of new developments and recent findings, I devote the task that was asked from me at Jaypee University of Information Technology to **"SECURITY SYSTEM FOR DNS USING CRYPTOGRAPHY".**

Date: Sakshi

111282

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

To reach another person on the Internet we have to type an address into our computer - a name or a number. That address has to be unique so computers know where to find each other. ICANN coordinates these unique identifiers across the world. Without that coordination we wouldn't have one global Internet. When typing a name, that name must be first translated into a number by a system before the connection can be established. That system is called the Domain Name System (DNS) and it translates names like www.icann.org into the numbers – called Internet Protocol (IP) addresses. ICANN coordinates the addressing system to ensure all the addresses are unique.

Recently vulnerabilities in the DNS were discovered that allow an attacker to hijack this process of looking some one up or looking a site up on the Internet using their name. The purpose of the attack is to take control of the session to, for example, send the user to the hijacker's own deceptive web site for account and password collection.

These vulnerabilities have increased interest in introducing a technology called DNS Security Extensions (DNSSEC) to secure this part of the Internet's infrastructure. DNSSEC will ensure the end user is connecting to the actual web site or other service corresponding to a particular domain name.

CHAPTER 1

INTRODUCTION

1.1 SCOPE OF THE PROJECT

The Domain Name System(DNS) has become a critical operational part of the Internet Infrastructure, yet it has no strong security mechanisms to assure Data Integrity or Authentication. Extensions to the DNS are described that provide these services to security aware resolves are applications through the use of Cryptographic Digital Signatures. These Digital Signatures are included zones as resource records.

The extensions also provide for the storage of Authenticated Public keys in the DNS. This storage of keys can support general Public key distribution services as well as DNS security. These stored keys enables security aware resolvers to learn the authenticating key of zones, in addition to those for which they are initially configured. Keys associated with DNS names can be retrieved to support other protocols. In addition, the security extensions provide for the Authentication of DNS protocol transactions.

The DNS Security is designed to provide security by combining the concept of both the Digital Signature and Asymmetric key (Public key) Cryptography. Here the Public key is send instead of Private key. The DNS security uses Message Digest Algorithm to compress the Message(text file). The message combines with the Private key to form a Signature using DSA Algorithm, which is send along with the Public key.

The receiver uses the Public key and DSA Algorithm to form a Signature. If this Signature matches with the Signature of the message received, the message is Decrypted and read else discarded.

1.2 WHAT IS DNS?

DNS is the shorthand for the Domain Name System. The Domain Name System provides a mechanism of conversion with a double functionality: it translates both symbolic host names to IP addresses and IP addresses to host

names.

The DNS has three major components:

• The first category contains:

– the **Domain Name Space** and

– the **Resource Records**, that are specifications for a tree structured name space and the data associated with these names.

• **Name Servers** are server programs which maintain the information about the DNS tree structure and can set information. A name server may cache information about any part of the domain tree, but in general it has complete information about a specific part of the DNS. This means the name server has authority for that sub domain of the name space - therefore it will be called *authoritative*.

• **Resolvers** are programs that extract the information from name servers in response to client requests.

.A more detailed presentation of the DNS can be found in the appendix

1.3 DNS SECURITY PROBLEMS

It is known the fact that DNS is weak in several places. Using the Domain Name System we face the problem of trusting the information that came from a non authenticated authority, the name-based authentication process, and the problem of accepting additional information that was not requested and that may be incorrect.“Many of the classic security breaches in the history of computers and computer networking have had to do not with fundamental algorithm or protocol flaws, but with implementation errors. While we do not intend to demean the efforts of those involved in upgrading the Internet protocols to make security a more realistic goal, we have observed that if BIND would just do what the DNS specifications say it should do, stop crashing, and start checking its inputs, then most of the existing security holes in DNS *as practiced* would go away.” - Paul Vixie, founder of ISC and main programmer of BIND.

1.3.1. Misdirected Destination: Trusting Faked Information

Suppose the following scenario: a user wants to connect to host A by means of a telnet client. The telnet client asks through a resolver the local name server to resolve the name A into an IP address, it receives a faked answer, and then initiates a TCP connection to the telnet server on the machine A (so it thinks). The user sends his login and password to the fake address. Now, the connection drops and the user retries the whole procedure this time to the correct IP address of the host A. He might ignore what just happened but the malicious attacker that spoofed the name of the host A is now in control of his login and password. This happened because the present routers have no capacity to disallow packets with fake source addresses. So, if the attacker can route packets to someone, then he is capable of forging those packets to look as if they come from a trustworthy host. Therefore,

in our case the attacker predicts the time when a query will be sent and he starts to flood the resolver with his fake answers. With a firewall for the user's network the resolver would not be reachable from the outside world, but his local name server would. So, if the local name server can be corrupted in the same manner as described above then the attacker can redirect such application with vital information towards hosts con- trolled by him and capture these information. Following these assumptions, we observe that in this case we have the possibility of a Denial of Service (DoS) attack. In case of such an attack, if the name server can be spoofed and the attacker's machine can impersonate the true name server then it can maliciously provide that certain names in the domain do not exist. Later on, we present a way in which such an attack is annihilated in DNSSEC.

1.3.2. Name Based Authentication/Authorization

Some applications, unfortunately spread all over the Internet, make use of an extremely insecure mechanism: name based authentication/authorization. It is the case, for example, of the Unix "r-commands" such as rlogin, rshor rcpthat use the concept of "remote equivalence" to allow the remote access to a computer.

In these networks, system administrators or, even worse, users can declare the remote equivalence of two accounts on two different machines (e.g., by means of the files /etc/hosts.equiv or .rhosts). This equivalence associates two users of two different hosts simply on the basis of their names. The access to a remote computer is then granted if the remote user is declared equivalent to a local user, and if the requesting hostname matches the one contained in the equivalence definition. No other authentication mechanisms are used, so we can talk of name based (weak) authentication. As an example, user joe can login as the user doe to the computer host.mydomain.com from the computer otherhost.mydomain.com if

the file /etc/hosts.equiv contains the equivalence between the local user doe and the user joe@otherhost.mydomain.com.

Remote commands have been designed at the dawn of the Internet for the use in trusted local network, where all the users were known to the system administrator, and the network was not connected to the big Internet. Unfortunately, remote commands survived to the Internet growth and they are still present and used in many networks.

If name based authentication/authorization is used, it is possible to access to a remote machine simply spoofing the name of a host. Also, if the local network is protected by a firewall, all the hosts that use name based authentication/authorization are at risk if an attacker can get control of a single machine of the firewall-protected network. The attacker can monitor network traffic learning the equivalences used in that network, and spoof the IP address of an equivalent host (e.g., performing a denial of service at- tack on that machine, or simply waiting for the machine to shut-down). Now, the attacker's host is completely equivalent to the spoofed host for all the computers using remote equivalence.

1.3.3. Trusting Supplementary : Non-Authoritative Information

This is another side of the DNS weakness. For the goal of efficiency the DNS was designed to have the additional section in its standard message format. Therefore, in certain cases when a supplementary information is considered necessary to speed up the response for a given query, it is included in the additional section. One example, if we ask our name server, i.e. *ns.mydomain.com*, to retrieve the mail exchange RR for the domain *comp-craiova.ro*, the server responds with a *mail.gate.comp-craiova.ro* RR in the answer section of the DNS message and in

the additional section the name and the IP address of the name server authoritative for the *comp-craiova.ro* domain are included.

Remember that this information was not explicitly asked by us, rather it was cached by our name server, in its pursuit for solving our query, in order to avoid further lookups for the name server authoritative for that domain.

The type of attack possible in this case is called **"cache poisoning"**. How does this happen? Suppose the following situation described in figure 1. An attacker controlling the name server for his domain *evil.com* wants to poison the cache of another name server called *ns.broker.com* used by a broker's agency in order to impersonate the machine *www.bank.com* that is often accessed by the users in the domain *broker.com*. From his machine the attacker asks the name server *ns.broker.com* for a name under the authority of his own name server, e.g. *anyhost.evil.com*. The name server *ns.broker.com* will contact the attacker's name server - authoritative for that name. This name server will answer the query and will also get the query ID which it stores for later use. This query ID is placed in the header section of any DNS message and is assigned by the pro- gram that generated the query (i.e., the target name server). This identifier is copied in the corresponding reply and can be used by the requester to match up replies to outstanding queries - as mentioned in the RFC 1035 [2]. The at- tack continues with another query from the attacker's side. He knows that the broker's agency is frequently contacting a certain bank site whose name he is willing to spoof. Therefore, he will ask the *ns.broker.com* name server for the address of the *www.bank.com*. Normally, the name server *ns.broker.com* will contact the DNS server authoritative for the domain *bank.com* (e.g., *ns.bank.com*). At this point, the attacker will start to flood the *ns.broker.com* server with replies in which the address of the attacker's machine is mapped to the name *www.bank.com* before

the true response can arrive from the authoritative name server (that is *ns.bank.com*). He also can predict correctly the query and reply ID, since he already has the last query ID generated by *ns.broker.com*. In this way, the server *ns.broker.com* receives an information which is not proper and also caches it after responding to the former attacker's query for *www.bank.com*. Now, the trap is set and all the attacker has to do is wait until a connection from *broker.com* domain is made to *www.bank.com*. Since the IP address of the attacker's machine is mapped incorrectly, in the server's cache (*ns.broker.com*), to the name *www.bank.com* all the connections to the bank will be directed to the attacker's machine. The name server will not try to query again for *www.bank.com*, it will just use the information it cached during previous DNS lookups. This is another reason why data received by the name servers in the DNS need origin authentication and integrity verification.

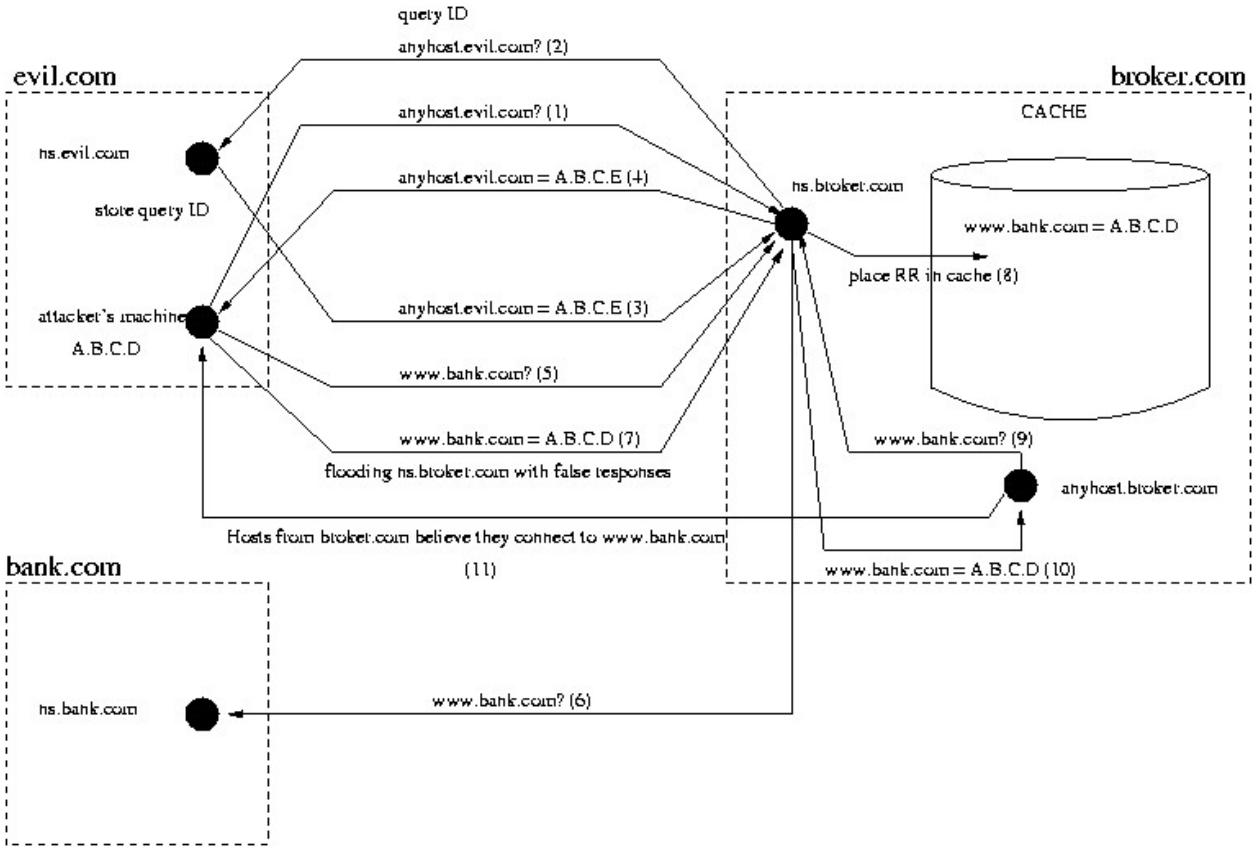

Figure 1. A cache poisoning example

1.4. DNS SECURITY EXTENSIONS

“DNS implementation and specification” - the security considerations were not forgotten since it is emphasized that the cache integrity is of maxi- mum importance. Despite this statement, the need for performance has pushed the present implementations to the situation of adding unauthorized records to the additional section and - lacking a strong authentication mechanism believing that all information provided by DNS is trust- worthy.

1.4.1. Involving Cryptography

The need for security extensions to DNS was acknowledged and standardized in an organized manner within the DNSSEC IETF working group. The first step is to provide data authentication of the resource records travelling back and forth in the internet. With authentication come also data integrity and data source authentication. The authentication is obtained by means of cryptographic digital signatures. The public key algorithms used for authentication in DNSSEC are **MD5/RSA and DSA**. The digital signatures generated with public key algorithms have the advantage that anyone having the public key can verify them. Each resource record in the DNS messages exchanged can be digitally signed providing data origin authentication and integrity of the message.

1.5 DNSSEC Objectives

A fundamental principle of the DNS is that it is a public service. It requires correct and consistent responses to queries, but the data is considered public data. As such,

the need for authentication and integrity exists, but not for access control and confidentiality. Thus, the objectives of DNSSEC are to provide authentication and integrity to the DNS. Authentication and integrity of information held within DNS zones is provided through the use of cryptographic signatures generated through the use of public key technology. Security aware servers, resolvers, and applications can then take advantage of this technology to assure that the information obtained from a security aware DNS server is authentic and has not been altered.

Although the DNSSEC WG chose not to provide confidentiality to DNS transactions, they did not eliminate the ability to provide support for confidentiality. Other applications outside of the DNS may choose to use the public keys contained within the DNS to provide confidentiality. Thus the DNS, in essence, can become a worldwide public key distribution mechanism. Issues such as cryptographic export are not, and may never be, solved worldwide; however, the DNS provides mechanisms to have multiple keys, each from a different cryptographic algorithm for a given DNS name, as a means to help alleviate this problem.

1.5.1 Performance Considerations

Performance issues are a concern for the security extensions to the DNS protocol and several aspects in the design of DNSSEC are targeted to avoid the overhead associated with processing the extensions. For instance, formulating another query that asks for the signature belonging to the RRSet just retrieved is not necessarily the most efficient way to retrieve a signature for the RRSet. This additional query is avoided whenever possible by allowing information retrieved from secured

zones to be accompanied by the signature(s) and key(s) that validate the information.

1.5.2 DNSSEC Scope

The scope of the security extensions to the DNS can be summarized into three services: key distribution, data origin authentication, and transaction and request authentication.

1.5.3 Key Distribution

The key distribution service not only allows for the retrieval of the public key of a DNS name to verify the authenticity of the DNS zone data, but it also provides a mechanism through which any key associated with a DNS name can be used for purposes other than DNS. The public key distribution service supports several different types of keys and several different types of key algorithms.

1.5.4 Data Origin Authentication

Data origin authentication is the crux of the design of DNSSEC. It mitigates such threats as cache poisoning and zone data compromise on a DNS server. The RR Sets within a zone are cryptographically signed thereby giving a high level of assuredness to resolvers and servers that the data just received can be trusted.

DNSSEC makes use of digital signature technology to sign DNS RR Set. The digital signature contains the encrypted hash of the RR Set. The hash is a cryptographic checksum of the data contained in the RR Set. The hash is signed (i.e., digitally encrypted) using a private key usually belonging to the originator of the information, known as the signer or the signing authority. The recipient of the RR Set can then check the digital signature against the data in the RR Set just

received. The recipient does this by first decrypting the digital signature using the public key of the signer to obtain the original hash of the data. Then the recipient computes its own hash on the RR Set data using the same cryptographic checksum algorithm, and compares the results of the hash found in the digital signature against the hash just computed. If the two hash values match, the data has integrity and the origin of the data is authentic [CHAR].

1.5.5 DNS Transaction and Request Authentication

DNS transaction and request authentication provides the ability to authenticate DNS requests and DNS message headers. This guarantees that the answer is in response to the original query and that the response came from the server for which the query was intended. Providing the assurance for both is done in one step. Part of the information, returned in a response to a query from a security aware server, is a signature. This signature is produced from the concatenation of the query and the response. This allows a security aware resolver to perform any necessary verification concerning the transaction.

Another use of transaction and request authentication is for DNS Dynamic Updates. Without DNSSEC, DNS Dynamic Update does not provide a mechanism that prohibits any system with access to a DNS authoritative server from updating zone information. In order to provide security for such modifications, Secure DNS Dynamic Update incorporates DNSSEC to provide strong authentication for systems allowed to dynamically manipulate DNS zone information on the primary server [RFC 2137].

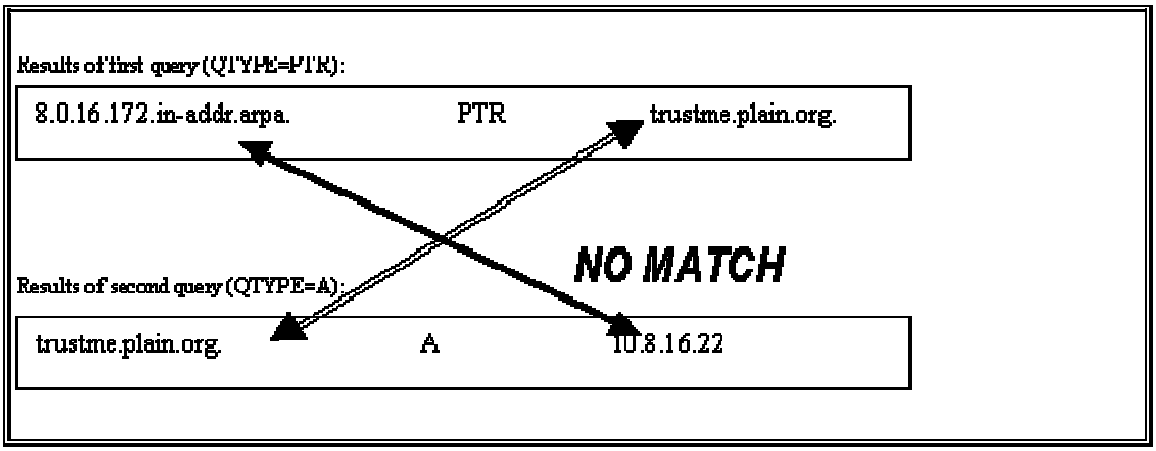

Figure 2. Example of a DNS cross check that fails

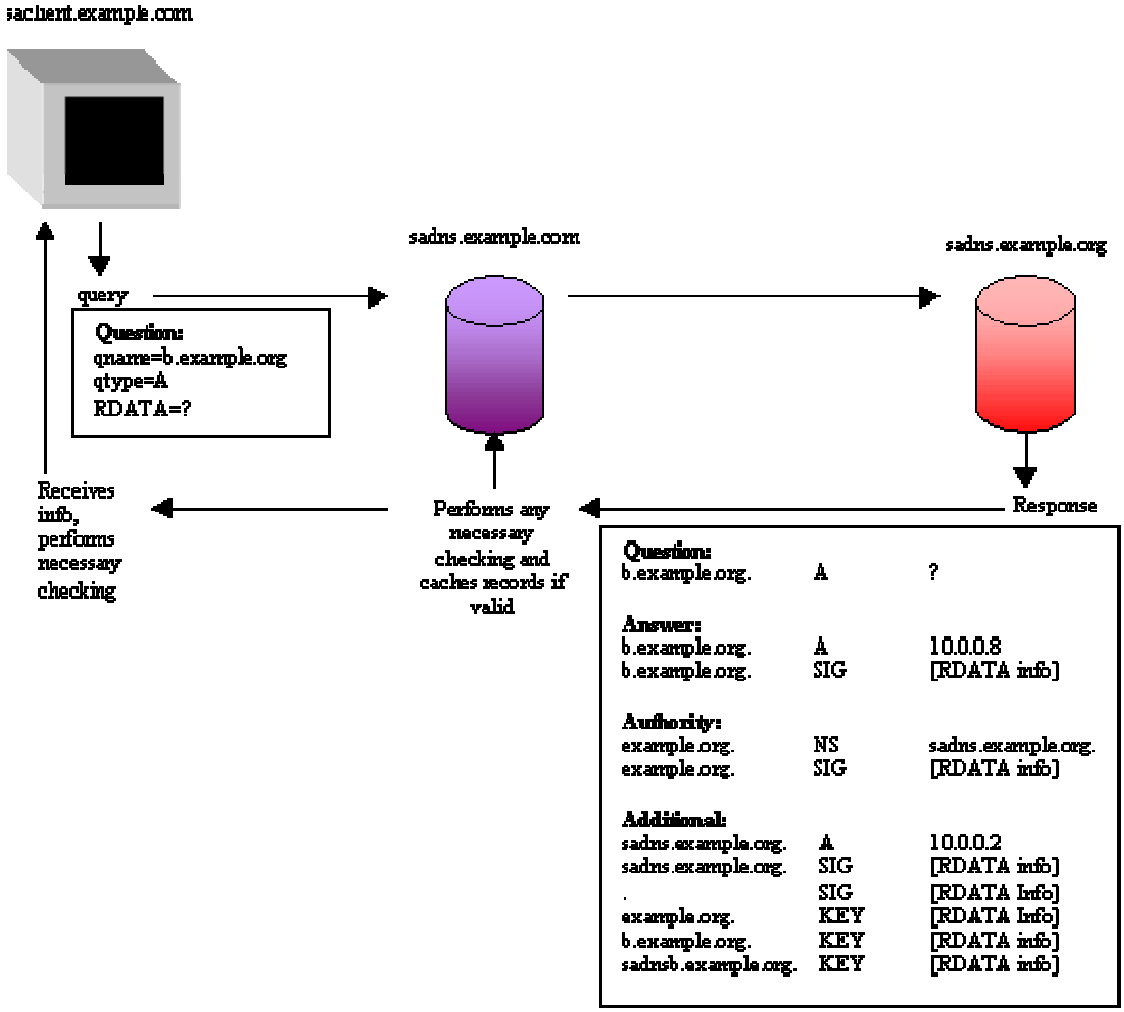

Figure 3. DNSSEC query & response messages

1.5.6 Public Key Retrieval

Resolvers can obtain public keys of zones in one of two ways. Resolvers can utilize the DNS to query for the public key or they can be statically configured with the key. Regardless of the method used, problems exist with both. In the case where keys are obtained through the DNS, the issue of trusting the key arises. In order to trust the retrieved key, it must be signed and this signature must be reliable. Providing the assurance that the signature on the key is reliable means that the public key of the signing authority must also be obtained, be signed, and be found reliable and so on. The solution to ending this recursive chain of events is to configure the resolver with the public key that authenticates the signed keys below it. In other words, a trusted zone key can be used as a starting point for verifying all keys found below it. A likely set of trusted public keys with which a secure zone can be statically configured are those of the root zone.

The static configuration of a resolver with the public keys from many different zones has one advantage in that the compromise of one of the zone's private key does not result in the compromise of the keys for all the other zones. The disadvantage of statically configuring each resolver with keys for many different zones is that it does not scale well. If one key for one zone must change, then all the resolvers must be configured to reflect this change.

CHAPTER 2

LITERATURE REVIEW

2.1. TITLE -

‘A New Approach To DNS Security(DNSSEC)’

SUMMARY –

The proposed model has following features-

(1) Secure DNS is a big change but inevitable.DNS has been extended to provide security services (DNSSEC) through symmetric-key cryptography. Offers the highest level of security while reducing network traffic. In addition, it reduces storage requirements and enables efficient mutual authentication.

(2)Here proposal for DNSSEC is that, when properly implemented, offers the highest level of security while reducing network traffic. In addition, it reduces storage requirements and enables efficient mutual authentication.

DNSSEC can not provide protection against threats from information leakage. This is more of an issue of controlling access, which is beyond the scope of coverage for DNSSEC. Adequate protection against information leakage is already provided through such things as split DNS configuration.

DNSSEC demonstrates some promising capability to protect the Internet infrastructure from DNS based attacks. DNSSEC has some fairly complicated issues surrounding its development, configuration, and management.

CHAPTER 3

PROBLEM STATEMENT

Authenticity is based on the identity of some entity. This entity has to prove that it is genuine. In many Network applications the identity of participating entities is simply determined by their names or addresses. High level applications use mainly names for authentication purposes, because address lists are much harder to create, understand, and maintain than name lists.

Assuming an entity wants to spoof the identity of some other entity, it is enough to change the mapping between its low level address and its high level name. It means that an attacker can fake the name of someone by modifying the association of his address from his own name to the name he wants to impersonate. Once an attacker has done that, an authenticator can no longer distinguish between the true and fake entity.

CHAPTER 4

PROPOSED MODEL

PROPOSED SYSTEM

Taking the above prevailing system into consideration the first step is to provide data authentication of the resource records travelling back and forth in the internet. With authentication come also data integrity and data source authentication. The authentication is obtained by means of cryptographic digital signatures. The public key algorithms used for authentication in DNSSEC are **MD5/RSA and DSA**. The digital signatures generated with public key algorithms have the advantage that anyone having the public key can verify them. Each resource record in the DNS messages exchanged can be digitally signed providing data origin authentication and integrity of the message.

Each time the System get the message, the Destination System generates Signature using Public Key and DSA Algorithm and verifies it with received one. If it matches it Decrypts otherwise it discards.

The Following functions avoids the pitfalls of the existing system.

Fast and efficient work

Ease of access to system

Manual effort is reduced

DIAGRAM

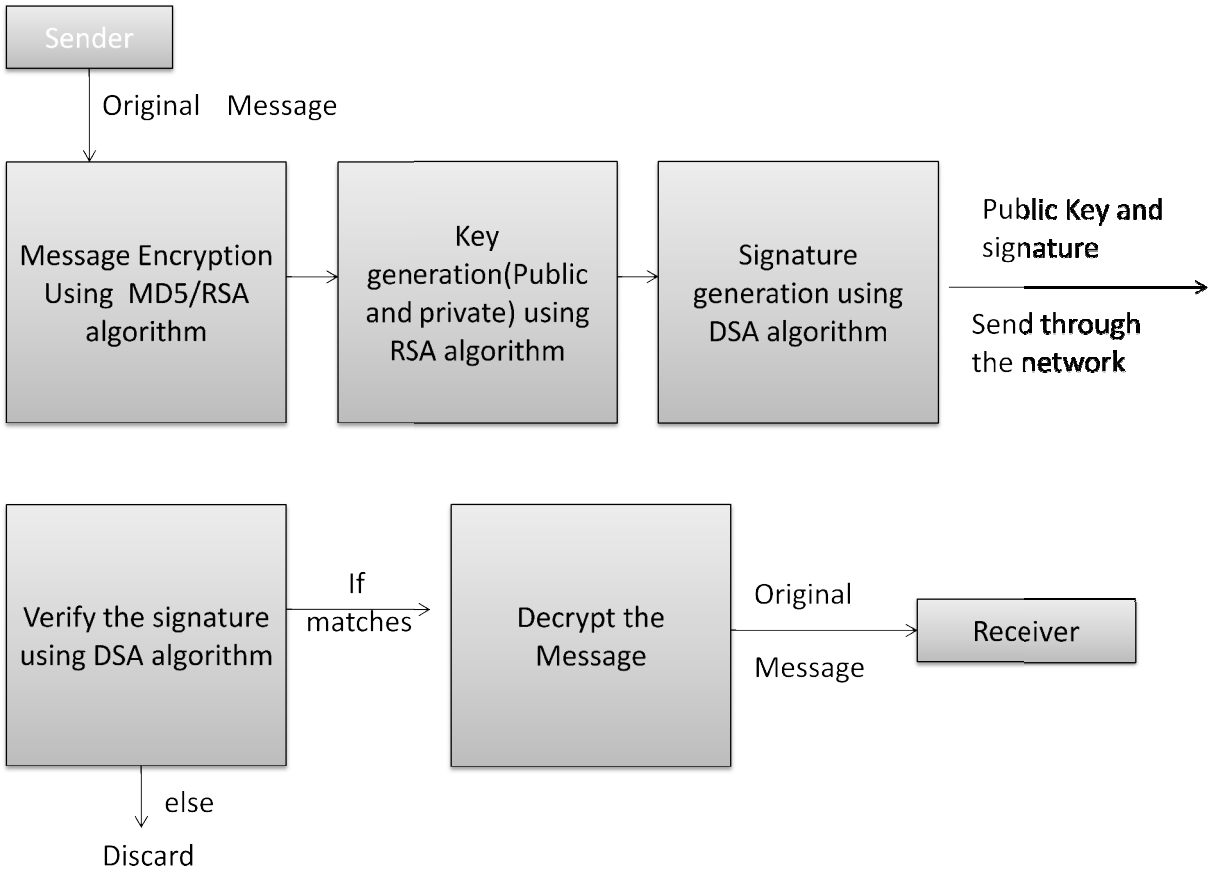

Figure 5 : Project model

4.1 FUNCTIONALITY

4.1.1 MD5

Message digest algorithm is used to provide data integrity. The MD5 (message-digest algorithm) is a widely used cryptographic hash function producing a 128 bit (16-byte) hash value, typically expressed in text format as a 32 digit hexadecimal number. MD5 has been utilized in a wide variety of cryptographic applications, and is also commonly used to verify data integrity.

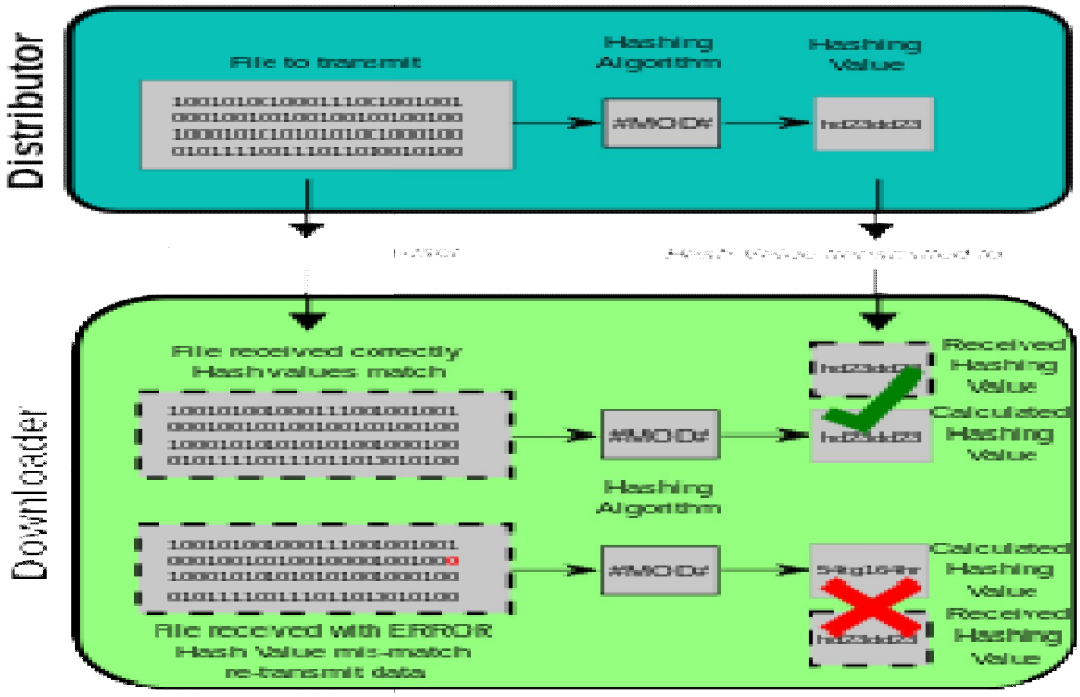

Figure 5. Working of MD5

4.1.2 RSA

RSA is used to create public and private key. RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977.

In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem

A user of RSA creates and then publishes a public key based on the two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime numbers can feasibly decode the message.

4.1.3 DSA

DSA is used to provide data authentication. DSA is a United States Federal Government standard for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS).

A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit.

The first part of the DSA algorithm is the public key and private key generation, which can be described as:

(1)Choose a prime number q, which is called the prime divisor.

(2)Choose another primer number p, such that p-1 mod q = 0. p is called the prime modulus.

(3)Choose an integer g, such that 1 < g < p, g**q mod p = 1 and g = h**((p–1)/q) mod p. q is also called g's multiplicative order modulo p.

(4)Choose an integer, such that 0 < x < q.

(5)Compute y as g**x mod p.

The second part of the DSA algorithm is the signature generation and signature verification, which can be described as:

(1)To generate a message signature, the sender can follow these steps:

(2)Generate the message digest h, using hash algorithm like.

(3)Generate a random number k, such that 0 < k < q.

(4)Compute r as (g**k mod p) mod q. If r = 0, select a different k.

(5)Compute i, such that k*i mod q = 1. i is called the modular multiplicative inverse of k modulo q.

(6)Compute s = i*(h+r*x) mod q. If s = 0, select a different k.

To verify a message signature, the receiver of the message and the digital signature can follow these steps:

(1)Generate the message digest h, using the same hash algorithm.

(2)Compute w, such that s*w mod q = 1. w is called the modular multiplicative inverse of s modulo q.

(3)Compute u1 = h*w mod q.

(4)Compute u2 = r*w mod q.

(5)Compute v = (((g**u1)*(y**u2)) mod p) mod q.

(6)If v == r, the digital signature is valid.

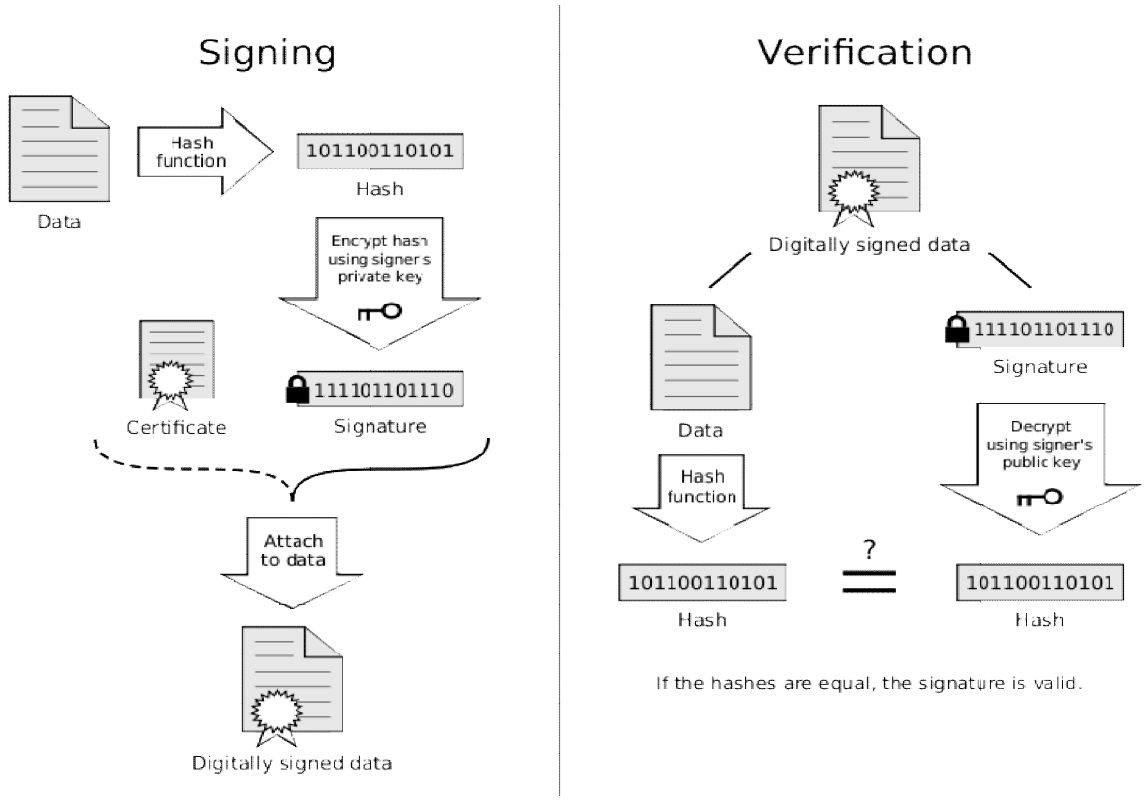

Figure 6: Working of DSA

4.2 SCREEN CAPTURES

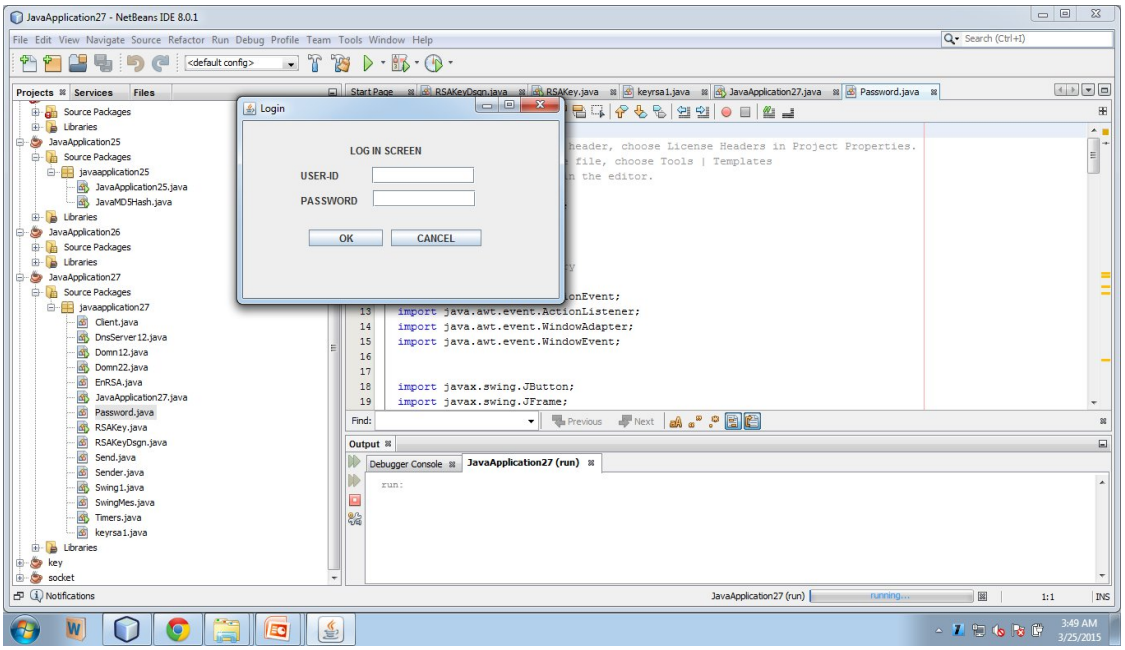

Figure 7: User login screen

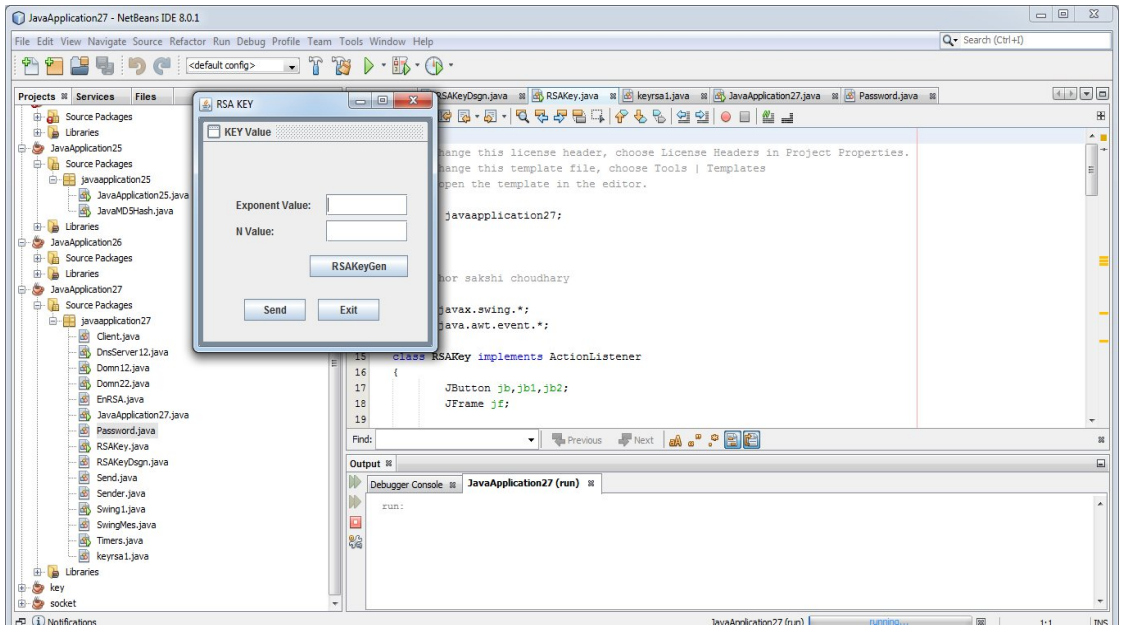

Figure8: RSA key generation

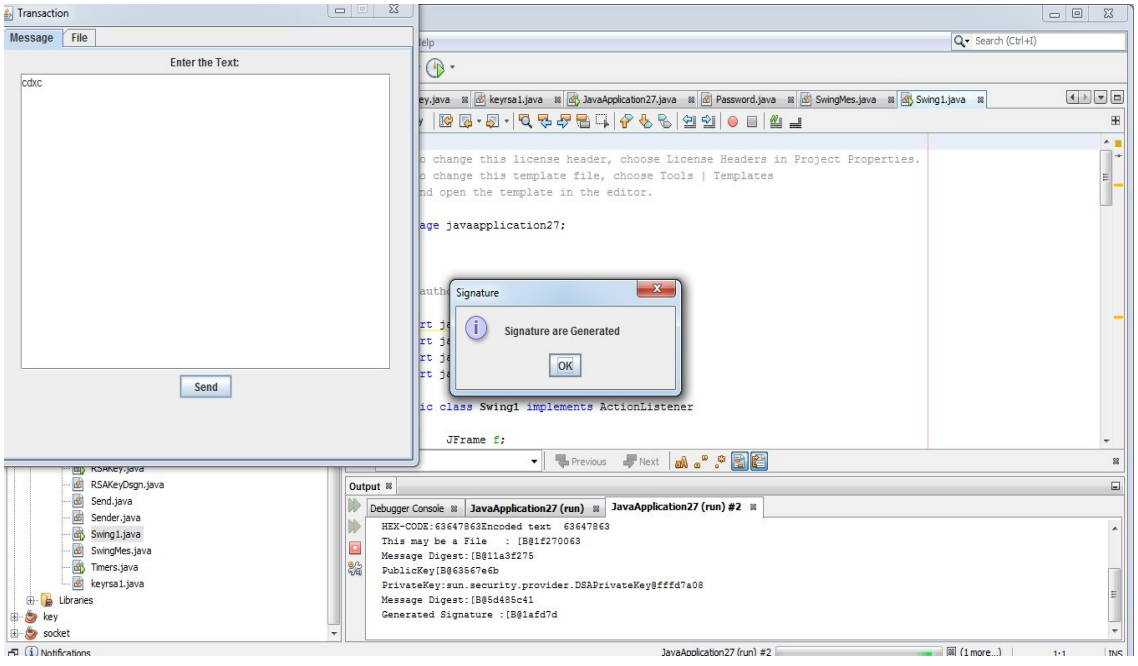

Figure 9:Signature generation

CHAPTER 5

TESTING

Software Testing is a process of executing program within the intent of finding an error. Software testing is a critical element of software quality assurance and represents The ultimate review of system specification, design, coding. Testing is last chance to uncover the error defects in the software and facilities delivery of quality system.

5.1 SYSTEM TESTING REQUIREMENTS

Software testing is not an activity to take up when the product is ready. An effective testing begins with a proper plan forms the user requirement stage itself. Software testability is the case with which a computer program is tested. Metrics can be used to measure the testability of a product.

5.2 PHASES OF THE TESTING

Several testing strategies and lead to the following generic characteristics:

Testing begins then unit level and works “outward” toward the integration of the entire system.

Different testing techniques are appropriate at different points of software development cycle.

5.2.1 UNIT TESTING

System security refers to the technical innovations and procedure applied to the hardware and operating system to product against deliberate or accidental damage.

Data security refers to the protection of data from loss, disclosure, modification and distraction. Privacy defines the rights of the users or organization to determine what information they willing to share with others and protect the information to minimize the possible invasion of privacy. To achieve all the above objectives.

5.2.2 INTERGRATION TESTING

System integrity refers to the proper functioning of hardware and software, Appropriate physical security and safety against external threats like wiretapping. Data integrity makes sure that data do not differ from their original form.

5.2.3 SYSTEM TESTING

After the Integration testing gets over the system has a whole is tested for validation. Here the testing is done by a complete tour of all the modules in a sequence. In case of further development of the system in the future, the programmer has to know t he logic involved. Documents to a programmer are like Road map to a traveler on the move.

Having the above facts in mind, a lot of care was taken in documenting at every stage of the project. By reading these documents the logic's involved in the programs will be crystal clear to the Programmer, in future. These documents will be of extensive use for debugging if any bugs are detected in the future.

5.2.4 PERFORMANCE TESTING

The system is very much user friendly and has a good user interface. This has been tested. Every user who needs to access this system is given an user Id and password and no one else can access. This too has been tested. It has been tested whether the loading of the screens of the application is fast and the migration from one form to another took less time. The time taken for this had been calculated.

The application is designed in such a way that it occupies less memory space; the database is also designed in such a way that it avoids duplication of records -i.e. the database avoids redundancy in all possible ways. Redundancy in storing the same data multiple times leads to several problems. Due to this storage place is also wasted. The files that represent the same data may be inconsistent. All these problems are looked after and rectified for efficient execution of the application.

5.2.5 VALIDATION TESTING

Here in this validation testing, all the values entered in each and every module are tested for correctness and validation as it has been entered before updating to the back end system.

The developed prototype of is verified and validated upon the software testing methods.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The DNS as an Internet standard to solve the issues of scalability surrounding the hosts.txt file. Since then, the widespread use of the DNS and its ability to resolve host names into IP addresses for both users and applications alike in a timely and fairly reliable manner, makes it a critical component of the Internet. The distributed management of the DNS and support for redundancy of DNS zones across multiple servers promotes its robust characteristics. However, the original DNS protocol specifications did not include security. Without security, the DNS is vulnerable to attacks stemming from cache poisoning techniques, client flooding, dynamic update vulnerabilities, information leakage, and compromise of a DNS server's authoritative files.

In order to add security to the DNS to address these threats, the IETF added security extensions to the DNS, collectively known as DNSSEC. DNSSEC provides authentication and integrity to the DNS. With the exception of information leakage, these extensions address the majority of problems that make such attacks possible. Cache poisoning and client flooding attacks are mitigated with the addition of data origin authentication for RR Sets as signatures are computed on the RR Sets to provide proof of authenticity. Dynamic update vulnerabilities are mitigated with the addition of transaction and request authentication, providing the necessary assurance to DNS servers that the update is authentic. Even the threat from compromise of the DNS server's authoritative files is almost eliminated as the SIG RR are created using a zone's private key that is kept off-line as to assure key's integrity which in turn protects the zone file from

tampering. Keeping a copy of the zone's master file off-line when the SIGs are generated takes that assurance one step further.

DNSSEC can not provide protection against threats from information leakage. This is more of an issue of controlling access, which is beyond the scope of coverage for DNSSEC. Adequate protection against information leakage is already provided through such things as split DNS configuration.

DNSSEC demonstrates some promising capability to protect the Internet infrastructure from DNS based attacks. DNSSEC has some fairly complicated issues surrounding its development, configuration, and management.

APPENDICES

(I) INTRODUCTION TO DNS

A bit of History

DNS is the shorthand for the Domain Name System. It represents the set of protocols and services on a TCP/IP network which allow users of the network to use hierarchical user-friendly names when looking for other hosts instead of having to remember and use their IP addresses. This system is used almost by any other application and protocol that is involved in network communication (e.g., web browsing, ftp, telnet or other TCP/IP utilities on Internet).

At the beginning of Internet, the name resolution was performed by means of *"hosts"* files (e.g., */etc/hosts* in UNIX) which contained the complete list of names and their associated IP addresses. These files were administered centrally, by the Network Information Center (NIC), and each computer connected to the Internet had to update its file periodically. With the exponential growth of the Internet, this became a burden for system administrators, so a better solution was needed. And it was given by prof. Paul Mockapetris the main designer of the Domain Name System.

So, the best known function of DNS consists in mapping symbolic names to IP addresses and viceversa. One exam- ple, if we need to connect to a certain web site, we need to know the IP address of the machine that supports this service, (for example, something like this 131.87.24.29), instead of this sequence of ciphers, not so easy to re- member and use, we could use the more suggestive name *www.mydomain.com*. This is where DNS gets involved.

In the ISO/OSI hierarchy, DNS finds itself on the ap- plication level, even though its usage is transparent to the users that simply refers to names instead of IP

addresses, and it can use either TCP or UDP as transport protocols. Usually, the resolvers are mainly relying on UDP (since the DNS queries and responses are well-suited for this protocol), but TCP might be used whenever truncation of the returned data occurs.

The Domain Name Space: Overview

The DNS is a hierarchical tree structure whose root node is known as the root domain. A label in a DNS name directly corresponds with a node in the DNS tree structure. A label is an alphanumeric string that uniquely identifies that node from its brothers. Labels are connected together with a dot notation, ".", and a DNS name containing multiple labels represents its path along the tree to the root. Labels are written from left to right. Only one zero length label is allowed and is reserved for the root of the tree. This is commonly referred to as the root zone. Due to the root label being zero length, all FQDNs end in a dot [RFC 1034].

As a tree is traversed in an ascending manner (i.e., from the leaf nodes to the root), the nodes become increasingly less specific (i.e., the leftmost label is most specific and the right most label is least specific). Typically in an FQDN, the left most label is the host name, while the next label to the right is the local domain to which the host belongs. The local domain can be a subdomain of another domain. The name of the parent domain is then the next label to the right of the subdomain (i.e., local domain) name label, and so on, till the root of the tree is reached.

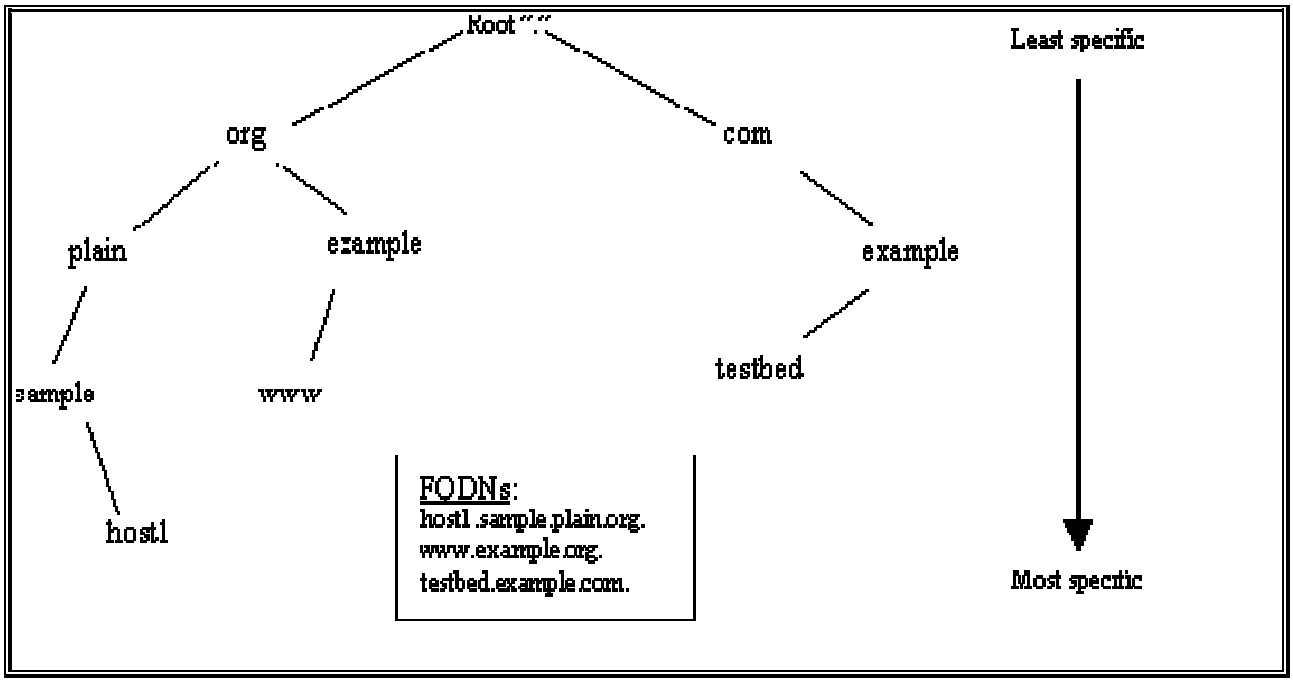

Figure 10. Domain Name Space example

When the DNS is used to map an IP address back into a host name (i.e., inverse resolution), the DNS makes use of the same notion of labels from left to right (i.e., most specific to least specific) when writing the IP address. This is in contrast to the typical representation of an IP address whose dotted decimal notation from left to right is least specific to most specific. To handle this, IP addresses in the DNS are typically represented in reverse order. By doing this, using IP addresses to find DNS host names are handled just like DNS host name lookups to find IP addresses.

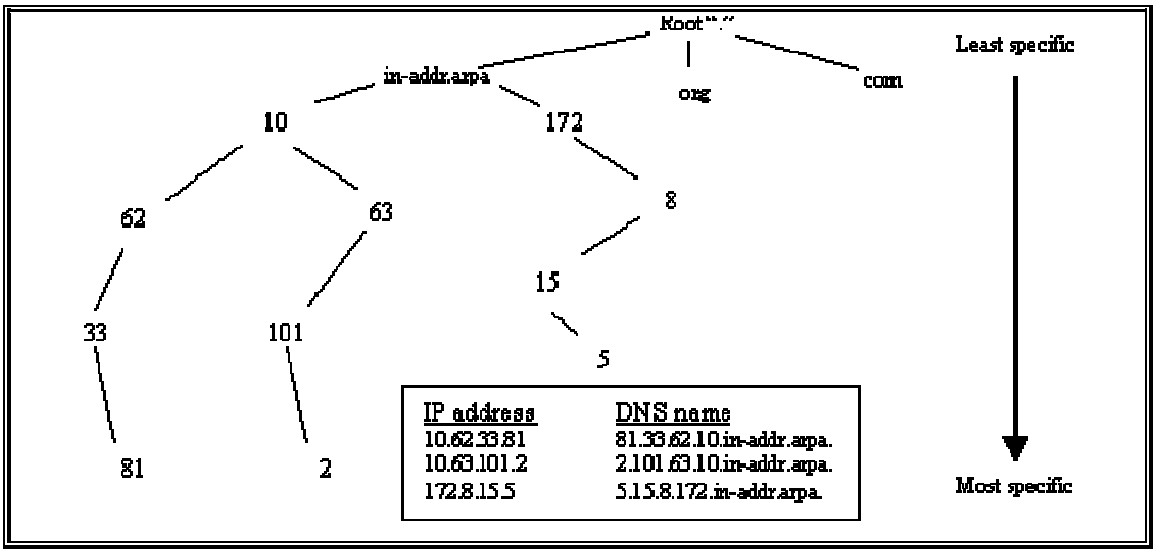

Figure 11. Example of inverse domains and the Domain Name Space

1.2.4 DNS Transactions

DNS transactions occur continuously across the Internet. The two most common transactions are DNS zone transfers and DNS queries/responses. A DNS zone transfer occurs when the secondary server updates its copy of a zone for which it is authoritative. The secondary server makes use of information it has on the zone, namely the serial number, and checks to see if the primary server has a more recent version. If it does, the secondary server retrieves a new copy of the zone.

A DNS query is answered by a DNS response. Resolvers use a finite list of name servers, usually not more than three, to determine where to send queries. If the first name server in the list is available to answer the query, than the others in the list are never consulted. If it is unavailable, each name server in the list is consulted until one is found that can return an answer to the query. The name server that receives a query from a client can act on behalf of the client to resolve the query.

Then the name server can query other name servers one at a time, with each server consulted being presumably closer to the answer. The name server that has the answer sends a response back to the original name server, which then can cache the response and send the answer back to the client. Once an answer is cached, a DNS server can use the cached information when responding to subsequent queries for the same DNS information. Caching makes the DNS more efficient, especially when under heavy load. This efficiency gain has its tradeoffs; the most notable is in security.

Source code

Java source codes used in the project-

SOURCE

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.util.zip.*;
import java.security.*;
import java.security.SecureRandom.*;
import java.security.spec.*;
import javax.swing.*;
```

```
public class SwingMes extends JFrame implements ActionListener
{
	JTabbedPane tp ;
	JButton send,browse,send1;
	JTextField tf1;
	JTextArea ta,ta1;
	JScrollPane sp,sp1;
	String g1=" ";
	byte str[];
	byte realSig[];
	String y="";
	String st="";
	Client ct;

	public SwingMes(Client ct)
	{
		super("Transaction");
		this.ct=ct;
		setSize(500,500);
		Container c = getContentPane();
		JPanel jp = new JPanel(new FlowLayout());
		JLabel jl = new JLabel("Enter the Text: ");
```

```
ta = new JTextArea(20,40);
sp = new JScrollPane(ta);
send= new JButton("Send");
jp.add(jl);
jp.add(sp);
jp.add(send);
jp.setVisible(true);
JPanel jp1 = new JPanel(new FlowLayout());
JLabel jl1 = new JLabel("Enter the File Name ",JLabel.LEFT);
tf1 = new JTextField(15);
browse = new JButton("Browse");
send1= new JButton("Send");
send.addActionListener(this);
browse.addActionListener(this);
send1.addActionListener(this);
jp1.add(jl1);
jp1.add(tf1);
jp1.add(browse);
jp1.add(sp1);
jp1.add(send1);
jp1.setVisible(true);
tp = new JTabbedPane();
tp.addTab("Message",null,jp,"Enter the Message here");
```

```
        tp.addTab("File",null,jp1,"Enter the File Name here");
        c.add(tp);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
        setResizable(false);
}


/*        MESSAGE DIGEST ALOGORITHM         */

private byte[] content;
public byte[] digestValue(byte[] in_text1)
{
        byte[] in_text=in_text1;
        content=in_text;
        MessageDigest mg1=null;

        try
        {
                mg1=MessageDigest.getInstance("SHA1");
        }
        catch (Exception e)
        {
```

```
                System.out.println(e);
            }
            mg1.update(content);
            byte[] digest1=mg1.digest();
            System.out.println("Message Digest:"+digest1);
            return digest1;
        }
        public void keys()
        {
            try
            {
KeyPairGenerator keygen = KeyPairGenerator.getInstance("DSA","SUN");

             /* cryptographically strong pseudo-random number generator
(PRNG) */
SecureRandom random = SecureRandom.getInstance("SHA1PRNG","SUN");
                keygen.initialize(1024,random);
                KeyPair keypair = keygen.generateKeyPair();
                PrivateKey privatekey = keypair.getPrivate();
                PublicKey publickey = keypair.getPublic();
                FileOutputStream pubkeyfos = new
FileOutputStream("public.txt");
                pubkeyfos.write(publickey.getEncoded());
```

```
                    System.out.println("PublicKey"+publickey.getEncoded());
                    pubkeyfos.close();
                    FileOutputStream prikeyfos = new
FileOutputStream("private.txt");
                    prikeyfos.write(privatekey.getEncoded());
                    System.out.println("PrivateKey:"+privatekey);
                    prikeyfos.close();
                    JOptionPane.showMessageDialog(null, "keys are
created","Keys", JOptionPane.INFORMATION_MESSAGE);
                }
                catch(Exception e)
                {
                    System.out.println(e);
                }
        }
        public void met2(String dis2)
        {
                System.out.println("FileName:"+dis2);
                try
                {
                    int i;
                    String g=" ";
                    char x[]=new char[50];
```

```
char d,s;
String s1;
InputStream in = new FileInputStream(dis2);
str=new byte[in.available()];
in.read(str, 0, str.length);
System.out.println(new String (str));
ta1.setText(new String (str));
FileOutputStream fos=new FileOutputStream("input.txt");
fos.write(str);
int len=str.length;
System.out.println("StringLength:"+len);
int a[]=new int[len];
System.out.print("HASH-CODE:");
for(i=0;i<len;i++)
{
        int aa= str[i];
        System.out.print("\t"+ aa);
        a[i]=aa;
}
System.out.print("HEX-CODE:");
for(i=0;i<len;i++)
{
```

```
                    g=" ";
                    b[i]=a[i]%16;
                    a[i]=a[i]/16;
                    g1=g1+g.trim();
                }
                str=g1.getBytes();
                digestValue(g1.getBytes());
            }
            catch(IOException e)
            {
                System.out.println(e);
            }
            keys();
            genSig("input.txt");
            String inputValue = JOptionPane.showInputDialog("Enter the System
Name for Domain1");
            try
            {
                Socket s1 = new
Socket(InetAddress.getByName(inputValue),7878);
                PrintStream ps = new PrintStream(s1.getOutputStream());
                ps.println(ct.SenderName.getText());
                System.out.println("SenderName
:"+ct.SenderName.getText());
```

```
                    ps.println(ct.SenderPassword.getText());
                    ps.println(ct.ReceiverName.getText());
                    System.out.println("ReceiverName
:"+ct.ReceiverName.getText());
                    System.out.println("Encrypted Data:"+ new String(str));
                    ps.flush();
                    s1.close();
            }
            catch(Exception ex)
            {
                    System.out.println(ex);
            }
    }
    public void genSig(String fname)

    {
            try
            {
                    byte[] md;
                    String args=fname;
                    FileInputStream fin=new FileInputStream(args);
                    byte[] in_text=new byte[fin.available()];
                    fin.read(in_text);
```

```
                fin.close();
                md=digestValue(in_text);
                FileInputStream fins=new FileInputStream("private.txt");
                byte[] enc_priv=new byte[fins.available()];
                fins.read(enc_priv);
                fins.close();
                Signature
dsa=Signature.getInstance("SHA1withDSA","SUN");
                PKCS8EncodedKeySpec privKeySpec=new
PKCS8EncodedKeySpec(enc_priv);
                KeyFactory
keyFactory=KeyFactory.getInstance("DSA","SUN");
                PrivateKey
priv=(PrivateKey)keyFactory.generatePrivate(privKeySpec);
                FileOutputStream sigfos = new
FileOutputStream("realSign.txt");
                sigfos.write(realSig);
                sigfos.close();
                JOptionPane.showMessageDialog(null, "Signature are
Generated","Signature", JOptionPane.ERROR_MESSAGE);
            }
            catch(Exception e)
            {
                System.out.println(e);
            }
```

```
}
public void actionPerformed(ActionEvent e)
{
        if(e.getSource() == browse)
        {
                JFileChooser fc = new JFileChooser();
                int option = fc.showOpenDialog(SwingMes.this);
                if(option == JFileChooser.APPROVE_OPTION)
                {
                        tf1.setText(fc.getSelectedFile().getAbsolutePath());
                }
        }
        else if(e.getSource() == send)
        {
                if (ta.getText() == null)
                else
                {
                        int i;
                        int b[]=new int[5000];
                        int a[]=new int[5000];
                        String g=" ";
                        char x[]=new char[50];
                        char d,s;
```

```
String h=ta.getText();
System.out.println("Message Entered:"+h);
int len=h.length();
System.out.println("StringLength:"+len);
for(i=0;i<len;i++)
{
        int aa= h.charAt(i);
        System.out.print("\t"+ aa);
        a[i]=aa;
}
for(i=0;i<len;i++)
{
        g=" ";
        b[i]=a[i]%16;
        a[i]=a[i]/16;
        g1=g1+g.trim();
}
System.out.println(g1);
str=g1.getBytes();
System.out.println(str);
st=new String(str);
digestValue(g1.getBytes());
keys();
```

```
                    genSig("input.txt");
                    try
                    {
                        Socket s1 = new
Socket(InetAddress.getByName(inputValue),7878);
                        PrintStream ps = new
PrintStream(s1.getOutputStream());
                        ps.println(ct.SenderName.getText());
                        ps.println(new String(str));
                        ps.println("public.txt");
                        ps.println("realSign.txt");
                        ps.flush();
                        BufferedReader br=new BufferedReader(new
InputStreamReader(s1.getInputStream()));
                        {
    JOptionPane.showMessageDialog(null,"Valid
Password","Password",JOptionPane.ERROR_MESSAGE);
                        }
                }
            }
    }
}
```

DESTINATION

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.util.zip.*;
import java.security.*;
import java.security.SecureRandom.*;
import java.security.spec.*;
import javax.swing.*;

public class SwingRMes extends JFrame implements ActionListener
{
        JTabbedPane tp ;
        JLabel jl,jl1,jl2,jL;
        JButton jb,jb1,jb2;
        JTextField tf1,tf2,tfd,tfd1;
        JTextArea tf,ta1;
        JScrollPane sp,sp1;
        static String g1=" ";
        static String n,n1,n2;
```

```
        static String y="";
        static String str6,str61,str2,str12,str13;
        static String t,st="",st1="",vj="";

        public SwingRMes()
        {
                super("Reception");
                String
inf="com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
                try
                {
                        UIManager.setLookAndFeel(inf);
                }
                catch(Exception e){}
                setSize(400,450);
                Container c = getContentPane();
                JPanel jp = new JPanel();
                jp.setLayout(null);
                jl = new JLabel("Sender Name ",JLabel.LEFT);
                jL = new JLabel("Receiver Name ",JLabel.LEFT);
                tfd = new JTextField(15);
                tfd1 = new JTextField(15);
                tf = new JTextArea(20,40);
```

```
jb= new JButton("OPEN");
jp.add(jl);
jp.add(tfd);
jp.add(jL);
jl.setBounds(25,10,150,25);
tfd.setBounds(150,10,150,25);
jL.setBounds(25,40,150,25);
tfd1.setBounds(150,40,150,25);
JPanel jp1 = new JPanel();
jp1.setLayout(null);
jl1 = new JLabel("Sender Name ",JLabel.LEFT);
jl2 = new JLabel("Receiver Name ",JLabel.LEFT);
tf1 = new JTextField(15);
sp1 = new JScrollPane(ta1);
jb2= new JButton("Retrive");
jl1.setBounds(25,10,150,25);
tf1.setBounds(150,10,150,25);
jl2.setBounds(25,40,150,25);
tf2.setBounds(150,40,150,25);
jp1.add(jl1);
jp1.add(tf1);
jp1.add(jl2);
jp1.setVisible(true);
```

```
            tp = new JTabbedPane();
            setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
        public static void ServSoc()
        {
            try
                {
                    ServerSocket ss = new ServerSocket(5555);
                    Socket s11 = ss.accept();
                    System.out.println(" Server Started..... ");
                    BufferedReader ps2=new BufferedReader(new
InputStreamReader(s11.getInputStream()));
                    str2 = ps2.readLine();
                    str2=str2.trim();
                    System.out.println("File Name    : " +str2);
                    char a[]=new char[50000];
                    char res[]=new char[9];
                    char s1[]=new char[5];
                    String w =new String();
                    String r =new String();
                    System.out.println("Verify Signature are Generated");
                                        try
                                        {
```

```
                verifySig(str12);
            }
            catch(Exception e)
            {
                System.out.println(e);
            }
        InputStream in3 = new
FileInputStream(str12);
        FileOutputStream out2=new
FileOutputStream("Sign1.txt");
        byte[] str3=new byte[512];
        while(in3.read(str3)!=-1)
        {
            String r1=new String(str3);
            out2.write(str3);
        }


    /*  Converting Hexcode into Bits    */

            for(i=0;i<len;i++)
            {
                char aa=
((str2).charAt(i));
```

```
a[i]=aa;
if(a[i]=='A')
{
        st="1010";
        st1=st;
}
else if(a[i]=='B')
{
        st="1011";
        st1=st;
}
else if(a[i]=='C')
{
        st="1100";
        st1=st;
}
else if(a[i]=='D')
{
        st="1101";
        st1=st;
}
else if(a[i]=='E')
{
```

```
        st="1110";
        st1=st;
}
else if(a[i]=='F')
{
        st="1111";
        st1=st;
}
else if(a[i]=='0')
{
        st="0000";
        st1=st;
}
else if(a[i]=='1')
{
        st="0001";
        st1=st;
}
else if(a[i]=='2')
{
st="0010";
        st1=st;
}
```

```
else if(a[i]=='3')
{
        st="0011";
        st1=st;
}
else if(a[i]=='4')
{
        st="0100";
        st1=st;
}
else if(a[i]=='5')
{
        st="0101";
        st1=st;
}
else if(a[i]=='6')
{
        st="0110";
        st1=st;
}
else if(a[i]=='7')
{
        st="0111";
```

```
                st1=st;
            }
            else if(a[i]=='8')
            {
                st="1000";
            }
            else if(a[i]=='9')
            {
                st="1001";
            }
            w=w+st;
        }

        char w_ch[]=w.toCharArray();
        String x=w.toString();
        System.out.print(x);

System.out.println("Length:"+len);

    public static boolean verifySig(String fname) throws Exception

        {
```

```
byte[] md;
byte[] sign;
byte[] realSig;
String args=fname;
FileInputStream fin=new FileInputStream(args);
byte[] in_text=new byte[fin.available()];
fin.read(in_text);
fin.close();
//SHA sha=new SHA(in_text);
md=digestValue(in_text);

FileInputStream finPublic=new FileInputStream(str12);
byte[] enc_pub=new byte[finPublic.available()];
finPublic.read(enc_pub);
finPublic.close();

FileInputStream sigfis=new FileInputStream(str13);
sign=new byte[sigfis.available()];
sigfis.read(sign);

Signature dsa1 =
Signature.getInstance("SHA1withDSA","SUN");
```

```
                    PublicKey
pub=keyFactoryPub.generatePublic(pubKeySpec);
                    dsa1.initVerify(pub);
                    dsa1.update(md);
                    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource() == jb2)
        {
            tf1.setText(str6);
            tf2.setText(str61);
            ta1.setText(vj);

        }
        else if(e.getSource() == jb)
        {
            JOptionPane.showMessageDialog(null, "Signature are
Verified","Verify Sign", JOptionPane.INFORMATION_MESSAGE);
            tfd1.setText(str61);
            tf.setText(vj);
        }
    }
}
```

REFERENCES

[1]Antonio Lioy, Fabio Maino, Marius Marian, Daniele Mazzocchi*"DNS Security"*, Department of Control and Computer Engineering Turin (Italy)

[2]Charishma G Shivaratri *"Domain Name System with Security Extensions"*Computer Science and Engineering University of Texas at Arlington

[3]*"Domain Name System Security Extensions"*, Donald Eastlake, IBM, March 1999.

[4]*"DSA KEYs and SIGs in the Domain Name System(DNS)"*, Donald Eastlake, IBM, March 1999.

[5]*"DNS Security Operational Considerations"*, Donald Eastlake, IBM, March 1999.

[6] Internet Draft *"Secret Key Transaction Signatures for DNS (TSIG)"*, Paul Vixie (Ed.) (ISC), Olafur Gudmundsson (NAIL- abs), Donald Eastlake (IBM), Brian Wellington (NAILabs), July 1999.

[7]Brian Wellington, *"An Introduction to Domain Name System Security"*, TIS Labs, January 1999.

[8]*"Domain Name System - Concepts and Facilities"*, Paul Mockapetris, ISI, November 1987.

[9]*"Domain Name System - Implementation and Specifi- cation"*, Paul Mockapetris, ISI, November 1987.

[10]*"Domain Name System Security Extensions"*, Donald Eastlake, IBM, C. Kaufman, January 1997.

[11]Paul Albitz, Cricket Liu,*"DNS and BIND"*, Third Edition, O'Reilly, Sebastopol, CA, 1998, ISBN 1-56592-512-2

[12]http://www.ijert.org/view.php?id=2533&title=security-system-for-dns-using-cryptography

[13]http://www.cs.jhu.edu/~ateniese/papers/dnssec.pdf

[14]http://www.sans.org/reading-room/whitepapers/dns/security-issues-dns-1069

[15]William Stallings,'Cryptography and Network Security' Prentice Hall of India publication