International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600

http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-203 PAGE NO: 1256-1262

Sri Vasavi College, Erode Self-Finance Wing *3rd February 2017*
National Conference on Computer and Communication *NCCC'17*
http://www.srivasavi.ac.in/ nccc2017@gmail.com

ROBUST AND ATTACK TOLERANCE SECURE SECRET DISTRIBUTION SYSTEM

Mr. Aruljothi.[1] Mr. B.Srinivasan[2]
[1]*Assistant Professor, Department of Computer Science, Gobi Arts and Science College,Gobi*
[2]*Associate Professor, Department of Computer Science, Gobi Arts and Science College,Gobi*

ABSTRACT- Robust and Attack Tolerance Secure Secret Distribution system has been designed to be one of the components in a Secured Networked Environment. This system stores and retrieves secrets by authorized clients. A relatively new approach to building security services that is both fault-tolerant and attack-tolerant. This approach includes the asynchronous model of computation using cryptography to improving confidentiality and integrity. This system is a distributed service for storage and dissemination of secrets. It was designed to be one of the components in a secure publish/subscribe communications infrastructure, providing the support for storing secret keys used to encrypt published information objects and ensuring that only authorized subscribers retrieve those secret keys. In this system client request the server to perform the operations (create, read, and write). The server responds to client to generate a pair of keys (public and public). The public key is transferred to client and it holds the private key. The client generates message and secret key. The message is encrypted by secret key and the secret key is encrypted by public key. Along with message digest transferred to the interface the interface after receiving the encrypted information it generate message digest and compare with the clients message digest for checking the integrity. Finally it stored as encrypt / decrypt information in the server.

Keywords: Client/Server Interface Protocol, Message Digest algorithm, RSA algorithm

1. INTRODUCTION

We build an intrinsic defense against denial of service attacks by designing the asynchronous model of execution. The client's secrets are stored in encrypted form and therefore can safely transfer to server and client. For encrypting the client secrets that are storing. For authenticating server responses to clients. The server checks authorization for every request by the client. The server designed in a secure communication based on asynchronous mode providing the support for storing secret keys used to encrypt information objects and ensuring that authorized subscriber retrieve that information.

Data that can be read and understood without any special measures is called plaintext or clear text. The method of disguising plaintext in such a way as to hide its substance is called encryption. Encryption plaintext results in unreadable gibberish called cipher text. We use encryption to ensure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting cipher text to its original plaintext is called decryption.

The primary benefit of private key cryptography is that it allows people who have no preexisting security arrangement to exchange message securely. No private key is ever transmitted or shared This paper thus should be seen not only as a discussion of how one might implement a service for storing and disseminating secrets but as an exercise in evaluating a promising recipe for building services that are both fault-tolerant and attack tolerant.

This system is a distributed service for storage and dissemination of secrets. It was designed to be one of the components in a secure publish/subscribe communications infrastructure, providing the support for storing secret keys used to encrypt published information objects and ensuring that (only) authorized subscribers retrieve those secret keys. Confidentiality, integrity.

The protocols to coordinate this system server avoid a large class making only weak assumptions about the execution environment. For example, correct operation of the protocols does not depend on assumptions about message delivery delays or processor execution times. Since denial of service attacks invalidate such assumptions about timing, we build into this system an intrinsic defense against certain denial of service attacks by designing for this asynchronous model of execution.

In particular, the confidentiality, integrity, and availability of secret keys stored by this system cannot be compromised by attacks to cause delays. But, adopting the asynchronous model brings challenges. Protocols for consensus and other replica-coordination problems arising in distributed systems require stronger assumptions.

This system cannot offer real-time guarantees to clients. Avail-ability is eventual. However, nothing about the CODEX protocols introduces unpredictable delays in process-sing client requests. Real-time bounds could therefore be derived for the case where the system and environment are not under attack.

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-203 PAGE NO: 1256-1262
Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

2. ALGORITHM

2.1 Server / Client Interface Protocol

The operations that are used to enable the clients to manipulate and retrieve bindings: Client protocol for the create operation, Client protocol for the write operation Client protocol for the read operation.

2.2 RSA Algorithm

RSA algorithm is developed by Ron Rivest, Adi Shamir, and Len Adleman. . It is a Public Key Cryptosystem It is asymmetric i.e., it uses a pair of keys (Public key Private Key).A public-key cryptosystem that offers both encryption and digital signatures (authentication), A mechanism which is done secretly to facilitate unauthorized access into the system which is normally done by the attackers.

2.2.1 Description Of The Algorithm

Encryption and decryption are the following form, for some plain text block M and ciphertext block C:

C=Me mod n

M=C d mod n=(M e)d mod n=M ed mod n

Both the sender and receiver must know the value of n. The sender the knows the value e, and only the receiver knows the value of d.Thus, this is a public key encryption.

2.3 Message Digest Algorithm

The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly.

This both algorithm using implements secrets to names. Bindings are writing-once only a single value is ever bound to each name. The three operations enable clients to manipulate and retrieve bindings: create introduces a new name, Write associates a value with a name, and. read returns the value associated with a name. 3. SYSTEM ARCHITECTURE

login through Admin and can create new user ID with password, giving rights to perform operation and also alter the rights. The user now login and sends an invocation message create (name) and wait for confirmation message with public key. After receiving the confirmation message we generate a secret key to encrypt the create (name), then by public key we encrypt the secret key and send the encrypted secret key and create (name) through TCP connection.

After creating the name in the server we invoke write (message) and wait for confirmation with public key. After receiving we generate secret key to encrypt the write (message), then by public key we encrypt the secret key and send through TCP connection.

3.1 INTERFACE

The interface receives the encrypted message, secret key and a message digest, which was created by client after encryption. Here we first create a message digest and compare with clients message digest, if any alteration happened immediately send an acknowledgement to resend the message. Or if there is no error then we decrypt the secret key by private key, which was created by server. And save the message and secret key in the server database. If the secret key has been deleted, information encrypted using that key becomes unavailable, the three operations can be Create - introduces a new name, Write - associates a value with a name, Read - returns the value associated with a name.

3.2 AUTHORIZATION

Whenever the client sends a request for accessing the particular operations in the server, the administrator verify the services provided to that client from the database and permit or denial of service.

When the request has come from the client for any operation, the server generate the pair (public and private) of keys and send the public key to the client to encrypt the secret key and the private key is send to the interface to decrypt the secret key.

Server responsible for received information from client. So it stores the data's in a secured database for future use. User ID, password, rights given to the user, the private key, the secret key binds with the names, operations allowed to user are stored in databases for storage and retrieval.

Insecure Links Assumption. Messages in transit may be disclosed to, deleted, or altered by adversaries; new messages may be injected. But, a message sent sufficiently often from one host to another will eventually be delivered. The only weaker assumption we can imagine is to offer no guarantee that hosts can use links to communicate, but without communication it would be impossible for clients to coordinate with CODEX or with each other.

Asynchrony Assumption. Message delivery and server computation times are unbounded. Note the Asynchrony Assumption is actually a nonassumption about timing. Finally, the only assumption we make about hosts is that not too many fall under control of the adversary.

Compromised Hosts. A host is either correct or compromised. A compromised host might deviate arbitrarily from its protocols and/or disclose information. But, less than one-third of the hosts running CODEX servers are assumed to be compromised at any time.

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-203 PAGE NO: 1256-1262
Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

If all hosts run the same software, then an adversary that compromises one replica will probably be able to exploit that same vulnerability at other replicas and compromise them too. A single exploit could then cause Compromised Hosts to be violated. One solution is to employ diversity, so that CODEX servers are not identical in their design or implementation and, therefore, do not have common vulnerabilities. For some system components, diverse implementations already exist.

3.2.1.Clients of CODEX can expect the following security properties to hold:

i) CODEX Availability

Authorized invocations of create, read, and write that are not concurrent with other Invocations involving the same CODEX name, if re-peated sufficiently often, cause the corresponding operations to be performed.

ii) CODEX Confidentiality

Executing read is the only way to learn a value that CODEX stores.

iii) CODEX Integrity

Executing write, giving a name that does not yet have a value associated, is the only way to bind a value to that name.

Maintaining Confidentiality and Integrity of Client Data.

iv)Unauthorized clients: Access control lists, stores clients' secrets (keys).

v) Compromised servers: Encryption with service public key, private key -> shared secret.

vi) Threshold decryption/signature: Does not use private key explicitly retrieval requires decryption.

vii) Codex distributed service

The protocols used by CODEX resemble those in COCA, since a reason for building CODEX was to explore how broadly applicable those COCA protocols are. Grouped according to function, here is a high-level description of the protocols.

viii) Servers and Service Authentication

In CODEX is each server has a public/private key pair, with the public key known to all servers Delegates can thus authenticate responses from servers and determine when responses have been received from a quorum. Clients do not know server Public keys. This allows server private keys to be changed without incurring an obligation to inform clients of the corresponding new public keys.

ix) Secure Links from Insecure Links

In CODEX, as in COCA, repeated message retransmission is used to overcome message loss admitted by the Insecure Links Assumption. Repeated retransmission of a given message is ended once the sender has been notified of

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600

http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-203 PAGE NO: 1256-1262

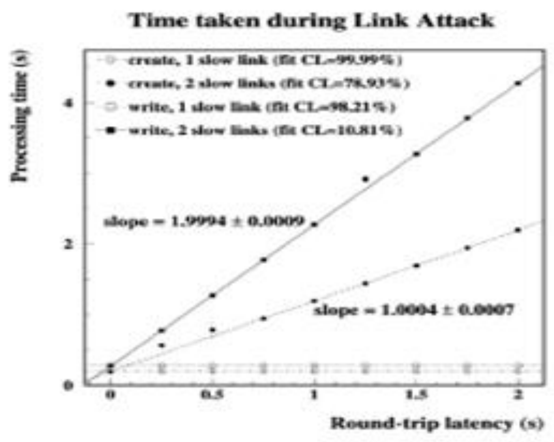
Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

successful receipt. This notification is usually signaled in a subsequent message from the receiver or from some other process that has received a message from the receiver

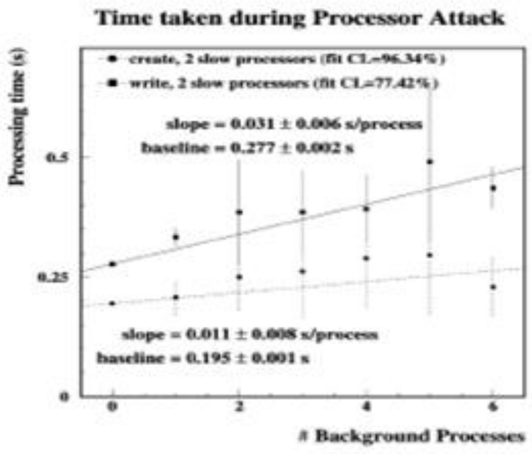
The direct comparisons with predictions of the costs for the various cryptographic operations. All measured ratios are consistent with our predictions, so we feel confident that modular exponentiations are indeed the dominant cost of the CODEX protocols in a LAN deployment. An adversary can launch a number of attacks on the service. We consider two, both attempted denials of service:

An attack that increases message latencies between servers. An attack that decreases CPU cycles available on servers.

For the first class of attacks, increased message latencies were simulated by modifying the CODEX binary so that message delivery could be delayed in a controlled way. We then simulated having one link under attack and then having two links under attack. Performance of CODEX under these attack scenarios is shown for create and write.



Time taken during Link Attack



Time taken during Processor Attack

4. CONCLUSION

A surprise in developing CODEX was that read and write and create invocations must include proofs of plaintext knowledge or else attackers can learn secrets CODEX is storing.

We expected that building CODEX would be a straight forward exercise in applying the architecture. CODEX, we have contributed a bit to better understanding which Services. Upcoming trends and technologies are mainly based on Distributed services, so implementation of this concept will provide a very good Environment.