International Journal of Engineering Research and Development
ISSN: 2278-067X, Volume 1, Issue 3 (June 2012), PP.08-14
www.ijerd.com

Public Key Cryptography by Centralized offline Server in Mission-Critical Networks

Ch. Krishna Prasad[1], G.Srinivasa Rao[2], V. Raja Sekhar[3]
[1]*Assoc. Prof. Anurag Engineering College, Kodad, Andhra Pradesh, India*
[2]*Assoc. Prof., Head Of The C.S.E Department, Anurag Engineering College, Kodad, Andhra Pradesh, India*
[3]*Asst. Prof, Sana Engineering College, Kodad, Andhra Pradesh, India.*

***Abstract*—**Mission-Critical networks show great potential in assisted living system, automotive networks, emergency rescue and disaster recovery system, military applications, critical infrastructure monitoring system. To build a secure communication system in that network, usually the first attempt is to employ cryptographic keys. Cryptographic key management is challenging due to the things like unreliable communications, limited bandwidth, network dynamics, largescale, resource constraints in wireless ad-hoc communications. Public-Private key pair in Mission-Critical networks to fulfill the required attributes of secure communications, such as data integrity, authentication, confidentiality, non-repudiation and service availability. So we go for an Self-Contained Public Key management scheme called Scalable method of cryptographic key management(SMOCK),which achieves almost Zero communication overhead for authentication and offers high service availability and also allows a mobile node to contain all of the necessary information for authentication locally. SMOCK provides a small number of cryptographic keys are stored offline at individual nodes before they are deployed in the network. The scheme allows a combinatorial design of Public-Private key pairs, to use good scalability in terms of the number of nodes and storage space. That means nodes combine more than one key pair to encrypt and decrypt messages.

***Keywords*—**Identity-Based Encryption, pair wise key distribution schemes, Rivest Shamir and Adleman (RSA), SMOCK (Scalable method of cryptographic key management),symmetric key techniques.

I. INTRODUCTION TO THE PUBLIC KEY INFRASTRUCTURE (PKI)

It has grown more important to ensure the confidentiality and integrity for data communication where an organization's network contains intranets, extranets, and Internet Web sites. Because of the connectivity of networks today, an organization's network is exposed to unauthorized users who could possibly attempt to access and manipulate mission critical data or the confidential data of its clients. The need to authenticate the identities of users, computers and even other organizations, has led to the development of the public key infrastructure (PKI). A public key infrastructure (PKI) can be defined as a set of technologies which control the distribution and utilization of unique identifiers, called public and private keys, through the utilization of digital certificates.

The set of technologies that constitute the PKI is a collection of components, standards and operational policies. The PKI process is based on the use of public and private keys to provide confidentiality and integrity of an organization's data as it is transmitted over the network. When users partake in the PKI, messages are encoded using encryption, and digital signatures are created which authenticate their identities. The recipient of the message would then decrypt the encoded message. For a PKI implementation to operate, each computer in the communication process must have a public key and private key. The public key and private key pair is used to encrypt and decrypt data, to ensure data confidentiality. When two parties use a PKI, each one obtains a public key and a private key, with the private kept only being known by the owner of that particular key. The public key on the other hand is available to the public. Before delving into the components and operations of the PKI, let's first look at what a properly designed and implemented PKI achieves:

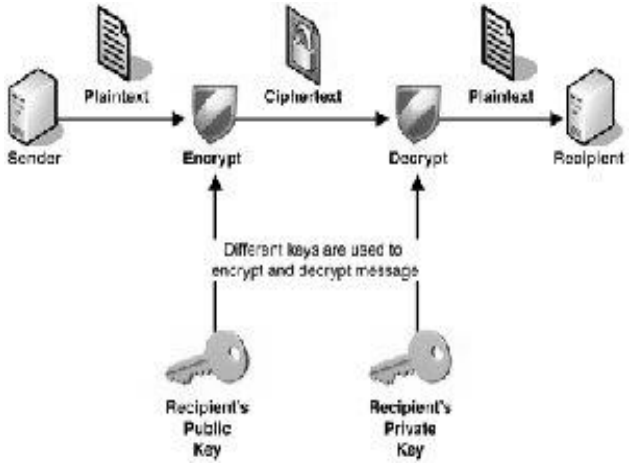
1. **CONFIDENTIALITY**: A PKI implementation ensures confidentiality of data transmitted over the network between two parties. Data is protected through the encryption of messages, so even in cases where the data is intercepted; the data would not be able to be interpreted. Strong encryption algorithms ensure the privacy of data. Only the parties which possess the keys would be able to decode the message.

2. **AUTHENTICATION:** The PKI also provides the means by which the sender of the data messages and the recipient of the data messages can authenticate the identity of each other. Digital certificates which contain encrypted hashes are used in authentication, and to provide integrity.

3. **INTEGRITY:** Integrity of data is assured when data has been transmitted over the network, and have not been fiddled with, or modified in any manner. With PKI, any modification made to the original data, can be identified

4. **Non-repudiation:** In PKI, non-repudiation basically means that the sender of data cannot at a later stage deny sending the message. Digital signatures are used to associate senders to messages. The digital signature ensures

that the senders of messages always sign their messages. This basically means that a particular person cannot, at a later stage, deny sending the message.



II. PKI COMPONENTS

In a PKI, there are several different entities or components. These components may be implemented separately, but are commonly integrated and delivered through what are called Certificate Servers.

1. Certificate Authority (CA) is the most fundamental component that will authorize and create digital certificates. A certificate authority (CA) server issues,manages, and revokes certificates. The CA's certificate (i.e., public key) is well known and trusted by all the participating end entities. The CA can delegate its authority to a subordinate authority by issuing a CA certificate, creating a certificate hierarchy. This is done for administration (e.g., different issuance policies) and performance reasons (e.g., single point of failure and network congestion). The ordered sequence of certificates from the last branch to the root is called a certificate chain. Each certificate contains the name of that certification's issuer (i.e., this is the subject name of the next certificate in the chain). A self-signed certificate means that the signer's public key corresponds to it's private key (i.e., the X.509v3 issuer and subject lines are identical).

2. The second core component of a PKI is the Registration Authority (RA), which provides the mechanism and interface for submitting users' public keys and identifying information in a uniform manner, in preparation for signing by the CA.

3. The third component is a Repository (Directory Server) in which certificates and certificate revocation lists are stored in a secure manner for later retrieval by systems and users. Lightweight Directory Access Protocol (LDAP) was originally designed to make it possible for applications running on a wide array of platforms to access X.500 directories. LDAP is defined by RFCs 1777 and 1778 as an on-the-wire bit protocol (similar to HTTP) that runs over TCP/IP. It creates a standard way for applications to request and manage directory information (i.e., no proprietary ownership, or control of the directory protocol). The directory entries are arranged in a hierarchical treelike structure that reflects political, geographic, and/or corporation boundaries.

4. PKI Applications are those use public-key technology. In most cases, the application would provide underlying cryptographic functions (e.g., public/private key generation, digital signature, and encryption) and certificate management. Certificate management functions include creating certificate requests, revocations,and the secure storage of a private key(s). Examples of PKI applications include Netscape's SSL 3.0 browser/server, Deming's Secure Messenger, and GlobeSet's Secure Electronic Transaction (SET) Wallet, Microsoft Outlook mail system.

III. PKI COMPONENT SECURITY REQUIREMENTS

PKI components each share a set of security requirements (i.e., baseline) with each other. The baseline corporate PKI security requirements are as follows:

- Reliable software (i.e., a comfortable level of assurance that security software is implementing the cryptographic controls properly).
- Secure/trusted communications between components (e.g., IPSec, SSL 3.0).
- PKI specific security policies that are derived from the existing set of corporate security policies.

Most PKI software/hardware is built upon cryptographic toolkits (e.g., RSA's B-Safe). The application that calls the lower level functions in the toolkit is still prone to human errors. Every other month Microsoft and Netscape release bug fixes for their Internet product sets. If the browser wars continue, there will be shorter quality assurance cycles to meet the current time to market constraints, hence produce a lower quality of software. PKI components require authenticated and private communication among each other. This prevents active or passive threats (e.g., eavesdropping, spoofing) from occurring. Most current implementation of PKI components supports SSL 3.0. Each component has a security criterion it must meet to be part of a PKI. This criterion is based on the level of protection necessary to perform the business objectives within the acceptable level of risk. The security mechanisms used to meet this criterion usually falls into physical,platform, network, and application categories. These categories are not all included in the PKI applications and have to be supplemented. Examples of these are network firewalls, disabling NFS exports, authenticated naming services, and tight administrator controls (e.g., root user).

IV. CERTIFICATE AUTHORITY

The certificate authority security requirements are:

- Certificate generation, issuance, inquiries, and revocation, renewal, and storage policies.
- Certification Practice Statement (CPS).
- Certificate attributes or extension policies.
- Certificate administration, audit journal, and data recovery/life-cycle support.
- Secure storage of private keys.
- Cross certification agreements.

The applicability and/or usage of the certificate the CA manages are defined in the Certificate Policy (CP). A security policy must exist for each CA function (e.g., generation, issuance, revocation list latency, etc.). These policies are the foundation upon which all the CA security related activities are based on Certification Practice Statement (CPS) is a detailed statement by the CA as to its certificate management practices. The certificate end entities and subscribers need to be well aware of these practices before trusting the CA. The CPS also allows the CA to indemnify itself to protect its relationships. One of the major improvements to version 3 of X.509 is the ability to allow flexible extensions to the certificate structure. These extensions include additional key and policy information, user and CA attributes, and certification path constraints. The CA must document, by way of a policy, the certificate attributes and extensions it supports. In addition, to allow interoperability outside the corporation, one must register the extension object identifiers (OID) with the American National Standards Institute (ANSI).The CA must maintain an audit journal of all key management operations it performs. All certificate management functions must be audited (e.g., issuance, revocation, etc.) in case of a dispute. In conjunction with this auditing function, a data recovery and certificate life cycle plan must also exist. The CA administrator interface must enforce the least privilege principal for all administrator actions. The certificate authority must provide for the adequate protection of the private key that it uses to sign certificates. The machine that the CA runs on must be protected from network and physical intrusions. Optionally, the CA's private key used to sign certificates can be stored in a tamperproof hardware module (e.g., meets FIPS PUB 140-1 level 3). Cross certification certificates are issued by CAs to form a non-hierarchical trust path. Two certificates are necessary for a mutual trust relationship (i.e., forward, and reverse directions). These certificates have to be supported by an agreement between the CAs. A cross- certification agreement details the obligation of liability between partners if a certificate turns out to be false or misleading.

V. DIRECTORY SERVER

The directory server security requirements are as follows:

- Supports network authentication through IP address/DNS name, and user authentication through LDAP user name and password, or a X.509 version 3 publickey certificate.
- Controls the users' ability to perform read, write, search, or compare operations down to the attribute level.
- Provides message privacy (SSL) and message integrity for all communications.

The directory server contains corporate and user personal attribute information. Access to this information must be controlled at the most granular level Possible. Directory administrators must be able to restrict particular users from performing specific directory operations (e.g., read, write, search, and compare).

Authentication must support conventional username/passwords and/or certificates. Additional filtering should be provided using IP address/DNS name.

Network access to the directory server must be able to be protected between all PKI components.

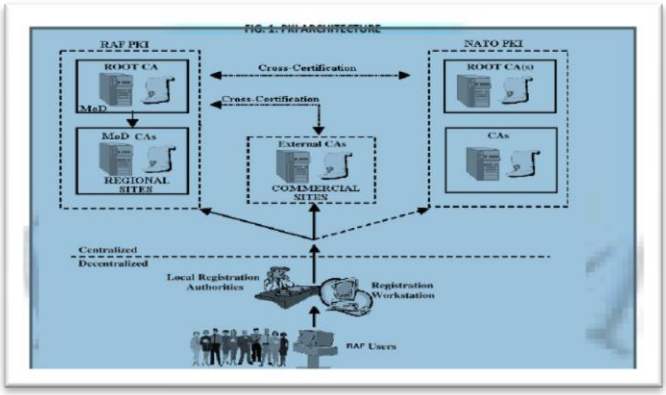
VI. PKI CLIENTS

All PKI clients, at a minimum, must be able to generate digital signatures and manage certificates. PKI client requirements are as follows:

- Generate a public/private key pair.
- Create a certificate request (PKCS#10).

- Display certificate.
- Verify certificate.
- Delete certificate.
- Enable or disable multiple certificates.
- Request a certificate revocation.
- Secure storage of certificates (e.g., password, and hardware).
- Secure exporting certificates (e.g., PKCS #12).
- Select algorithm, key strength, and password controls.
- Configure security options (e.g., sign/encrypt whenever possible).

The process begins with a PKI client generating a public/private key pair locally. The software used to generate the public/private key pair must use a nondeterministic algorithm. Once the key pair is generated, the public portion needs to be bound inside a certificate structure. The PKI client must then generate a certificate request adhering to the PKCS#10 syntax and submit that information to a CA. Once the CA fulfils the request, the message response sent back to the client is in PKCS#7 syntax (i.e., signed envelope). All network traffic is kept private between the client and the CA. The PKI client must have the ability to manage multiple certificates. This includes viewing the certificate structure (e.g., subject, issuer, serial number, fingerprint, and validity dates); deleting it, if necessary; choosing (i.e., enabling) what certificate to use or query the user; or requesting the CA to revoke it. A large portion of public cryptography is based on the protection of the private key. The PKI client must protect their private key commensurate with the risk associated with the loss of all the transactions it processes. This will require encrypted storage of the key using an application authentication challenge (e.g., organization compliant password), or hardware token or smart card, and the user physically protecting their desktop (e.g., password protected screen saver).Due to the infancy of this technology, certificates are bound to the PKI client application software and hence the host that the software resides on. An emerging public key cryptographic standard (PKCS) called personal information exchange syntax standard (i.e., PKCS #12) details the transfer syntax for personal identity information. This includes private keys,certificates, miscellaneous secrets, and extensions. This will allow PKI clients to import and export personal identity information across multiple platforms and applications. The most secure method includes a privacy and integrity mode that requires the source and destination platforms to have trusted public/private key pairs available for digital signatures and encryption. The least secure method protects personal identity information with encryption based on a password.



VII. EXISTING SYSTEM

In secure communication, wireless sensor networks use symmetric key techniques. In symmetric key techniques, secret keys are pre distributed among nodes before their deployment. A challenge of the key distribution scheme is to use small memory size to establish secure communication among a large number of nodes and achieve good resilience. Public-key (certificate)-based approaches were originally proposed to provide solutions to secure communications for the Internet, where security services rely on a centralized certification server. The certificate-based approaches to ad-hoc networks and present a distributed public key-management scheme for ad-hoc networks, where multiple distributed certificate authorities are used. To sign a certificate, each authority generates a partial signature for the certificate submits the partial signature to a coordinator that calculates the signature from the partial signatures.

VIII. DISADVANTAGES

Lack of support for authentication and confidentiality. Single-point failure of the centralized server is able to paralyze the whole network, which makes the network extremely vulnerable to compromises and denial-of-service attacks. Total number of keys held by each user is O(n) traditional key-management schemes.

IX. PROPOSED SYSTEM

In this system we propose to support secure communications with the attributes of data integrity, authentication, confidentiality, no repudiation, and service availability. To build a secure communication system, usually the first attempt is to employ cryptographic keys. In SMOCK, let us assume a group of people in an incident area, who want to exchange correspondence securely among each other in a pair-wise fashion.2.The key pool of such a group

consists of a set of private–public key pairs, and is maintained by an offline trusted server. Each key pair consists of two mathematically related keys. The **I**th key pair in the key pool is represented by (**priv ^i, pub^i**) . To support secure communication in the group, each member is loaded with all public keys of the group and assigned a distinct subset of private keys. Each person keeps a predetermined subset of private keys, and no one else has all of the private keys in that subset. For a public–private key pair, multiple copies of the private key can be held by different users. A message is encrypted by multiple public keys, and it can only be read by a user who has the corresponding private keys.

X. ADVANTAGES

1. To Support secure communications among the users in Mission-Critical Wireless Ad-Hoc Networks.
2. In SMOCK-management scheme, which scales logarithmically with network size O(log n), with respect to storage space.
3. In SMOCK to provide two encryption and decryption standard. In Decryption using a private key set.
4. Key Management System provide at offline-line centralized server.

XI. SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual design that defines the structure and/or behavior of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system.

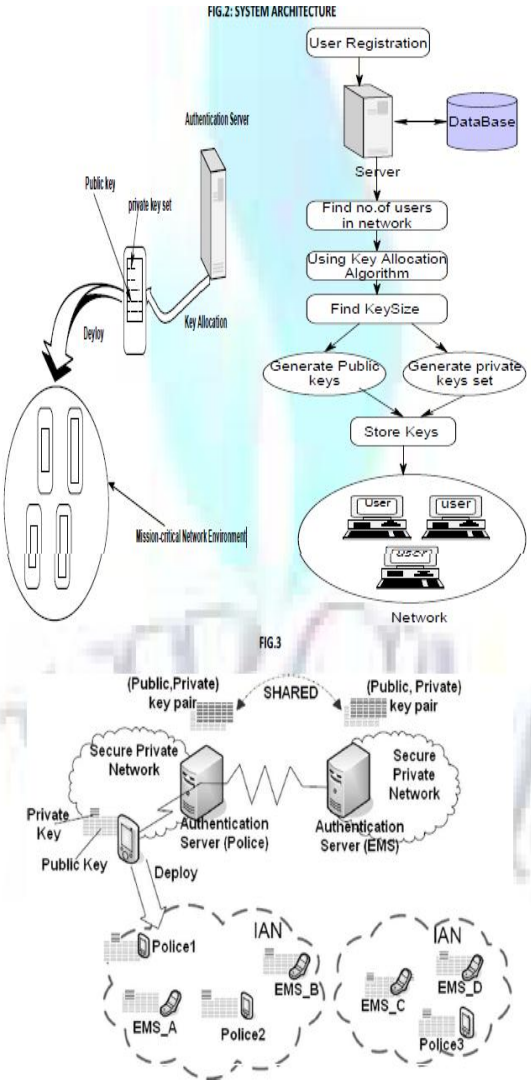


Fig.3. Two agencies [police department and emergency medical service (EMS)] maintain the same private and public key pool through a secure connection.

Before deployment, agencies predistribute keys to devices. After devices are dispatched into the incident areas, it is costly and unsafe to communicate with agencies. So all of the devices authenticate messages according to the predistributed keys.

XII. ALGORITHM DESIGN

KEY ALLOCATION ALGORITHM

This algorithm calculates the minimum number of memory slots to store public keys in order to support the secure communication among n users.

1. Initialize the number of users n;
2. Initialize j=2,a=1;
3. initialize key=j;
4. loop:

for(i=1;j;i<=n)
satisfied condition
if(n<=a)
satisfied condition
return key;
else
Not Satisfied
a=a+j;
increment i,j;
End loop;
Based on the key value the number of public keys generated.

PRIVATE KEY SET ALLOCATION TO USERS

1. Initialize the number of keys n users;
2. Initialize i=1,k=2,id=1,j=k;
3. store private keys in list.
4. loop

for(i; i <=nusers;i++)
satisfied condition
loop
for(j;j<=nusers;j++)
satisfied condition
get i position value from list
get j position value from list
allocate that i , j position key values to (id)user
id++;
end loop;
k++;
end loop;

XIII. RSA ALGORITHM

KEY GENERATION

1. Generate two large prime numbers, p and q
2. Let n = p q
3. Let m = (p-1)(q-1)
4. Choose a small number e, co-prime to m
5. Find d, such that de % m = 1

Publish e and n as the public key.
Keep d and n as the secret key.
Encryption
C = Pe % n
Decryption
P = Cd % n
x % y means the remainder of x divided by y
Key Generation
1) Generate two large prime numbers, p and q
2) Let n = p q
3) Let m = (p - 1) (q - 1)
4) Choose a small number, e co-prime to m
5) Find d, such that de % m = 1
6) Generate Public key, Secret key.

XIV. SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users,which it will work efficiently and effectively. It involves careful planning, investigation of the current System and its constraints on implementation, design of methods to achieve the change over, an evaluation, of change over methods. In our implementation, nodes receive their subset of private keys, unique SMOCK IDs and all SMOCK public keys via the SSL channel from a trusted authority before secure communication. When a node wants to send a message to another node (the receiver), it sends a plain-text message (along with its SMOCK ID). The receiver then encrypts its SMOCK ID with the sender's public keys, and sends the encrypted message to the sender. The sender can then encrypt the message by using the receiver's SMOCK keys. And the receiver can then decrypt the message using its SMOCK private keys. We measured the encryption and decryption process time that was taken to encrypt and decrypt a message.

XV. IMPLEMENTATION PLAN

The implementation can be preceded through Socket in java but it will be considered as one to all communication. So java will be more suitable for platform independence and networking concepts. For maintaining the load we go for SQL-server as database back end. In existing system, single-point failure of the centralized server is able to paralyze the whole network, which makes the network extremely vulnerable to compromises and denial-of-service attacks. we need a self-contained key-management scheme, which allows a mobile node to contain all of the necessary information for authentication locally. A realistic assumption about mission-critical applications is that before mobile devices are dispatched to an incident area, they are able to communicate securely with the trusted authentication server in their domain center, and get prepared before their deployment.

XVI. CONCLUSION

We depict a self-contained key-management scheme, which requires significantly less key storage space than traditional schemes and almost zero communication overhead for authentication in a mission-critical wireless ad-hoc network with nodes. The scheme also achieves controllable resilience against node compromise by defining required benchmark resilience. We generalized the traditional public-key-management schemes. And in SMOCK turned out to be the traditional public-key infrastructure. We can also see that SMOCK fulfills the secure communication requirements in terms of integrity, authentication,
confidentiality, no repudiation, and service availability.

XVII. SCOPE FOR FUTURE ENHANCEMENT

We can further extend the idea of SMOCK to other applications, such as broadcast authentication, Message signing and verification use all the hash chains associated with the senders' identity. With the combinatorial design, we expect better scalability and less delay than traditional broadcast authentication schemes. We will investigate this deeply in future works.

REFERENCES

[1]. S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in Proc. 10th ACM Conf.Computer and Communications Security, Oct. 2003.

[2]. D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," SIAMJ. Computing. vol.32, no. 3, pp. 586–615, 2003.

[3]. D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure For key distribution in Tiny OS based on elliptic curve cryptography," presented at the 1st IEEE Int. Conf. Sensor and Ad Hoc Communications and Networks, Santa Clara, CA, Oct. 2004.

[4]. A Pairwise Key Predistribution Scheme for Wireless Sensor Networks by Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney."IEEE Trans.July 2004.

[5]. Identity-Based Encryption from the Weil Pairing by Dan Boneh, Matthew Franklin."IEEE Trans. Feb 2005.

[6]. Efficient Security Mechanisms for Large Scale Distributed Sensor Networks by Sencun Zhu, Sanjeev Setia, Sushil Jajodia."IEEE Trans.Aug 2004.