International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064

A Novel Anonymous and Certificate Less Public Key Infrastructure For MANETs

G. Anandhi[1], S. K. Srivatsa[2]

[1]Research Scholar, Vels University St. Joseph's College of Engineering, Chennai, India

[2]Senior Professor, Vels University St. Joseph's College of Engineering, Chennai, India

Abstract: ***This paper proposes a novel Anonymous and Certificateless Public-Key Infrastructure (AC-PKI) for MANETs. Our contributions are mainly threefold. First, we apply Shamir's secret-sharing technique to distribute the system trust, essentially a system master-key, across a pre-selected set of nodes, called distributed private-key generators (D-PKGs). D-PKGs collaboratively offer a prerequisite private-key-generation (PKG) service during network operation. Second, we propose offering D-PKGs anonymity protection to defend against pinpoint attacks that are quite easy to conduct and may cause devastating consequences in MANETs. Last, we determine the optimal secret-sharing parameters for achieving the maximum security.***

Keywords: CA (Certificate Authority), CRL, Keychains

1. Introduction

In a conventional public-key infrastructure (PKI), a centralized Certification Authority (CA) is indispensable for managing public key certificates used to generate confidence in the legitimacy of public keys. However, it is difficult to deploy such a certificate-based PKI in MANETs for the lack of infrastructure and other centralized services. Although the secret-sharing technique could be employed to distribute the CA's role to a pre-selected set of nodes, termed distributed CAs, resource-constrained ad hoc networks might be still unable to afford the rather complicated certificate management, including revocation, storage and distribution, and the computational costs of certificate verification.

The infrastructure less nature and network dynamics of ad hoc networks make the conventional certificate based public-key solutions less suitable. To tackle this problem, we propose a novel Anonymous and Certificate-less Public-Key Infrastructure (AC-PKI) for ad hoc networks. AC-PKI enables public-key services with certificate-less public keys and thus avoids the complicated certificate management inevitable in conventional certificate-based solutions. In a conventional public-key infrastructure (PKI), a centralized Certification Authority (CA) is indispensable for managing public key certificates used to generate confidence in the legitimacy of public keys. However, it is difficult to deploy such a certificate-based PKI in MANETs for the lack of infrastructure and other centralized services.

The following example (Figure 1) can help understand how public-key encryption/decryption services are accomplished in AC-PKI. Suppose nodes Alice and Bob are both legitimate members of Ψ. When having some secret information msg for Bob, Alice no longer needs to obtain from anywhere Bob's public-key certificate and verifies it in advance, as what she has to do in a conventional certificate-based PKI. Instead, she can directly use Bob's identifier IDB as his public key and generate the ciphertext as IBE (IDB, msg). Here IBE () represents an identity-based encryption (IBE) function built on the above public system parameters. Many such IBE functions as have been proposed in the literature, which can guarantee that no other node than Bob, which must hold the valid private key corresponding to "IDB", can correctly decrypt the ciphertext.

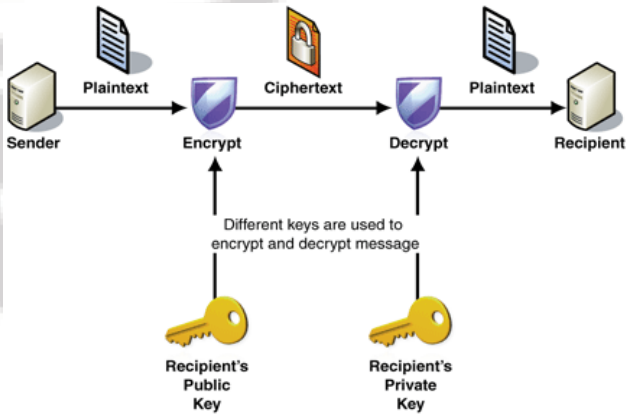

Figure 1

2. PKI Components and Functions

- The Certificate Authority (CA), an entity which issues certificates. One or more in-house servers, or a trusted third party such as VeriSign or GTE, can provide the CA function.
- The repository for keys, certificates and Certificate Revocation Lists (CRLs) is usually based on an Lightweight Directory Access Protocol (LDAP)-enabled directory service.
- A management function, typically implemented via a management console.[7]

2.1 PKI Functions

The most common PKI functions are issuing certificates, revoking certificates, creating and publishing CRLs, storing and retrieving certificates and CRLs, and key lifecycle management. Enhanced or emerging functions include time-stamping and policy-based certificate validation.

2.1.1 Issuing certificates

The CA signs the certificate, thereby authenticating the identity of the requestor, in the same way that a notary

public vouches for the signature and identity of an individual. In addition, the CA "stamps" the certificate with an expiration date. The CA may return the certificate to the requesting system and/or post it in a repository

2.1.2 Revoking certificates

A certificate may become invalid before the normal expiration of its validity period. For instance, an employee may quit or change names, or a private key may be compromised. Under such circumstances, the CA revokes the certificate by including the certificate's serial number on the next scheduled CRL.

2.1.3Storing and retrieving certificates and CRLs

The most common means of storing and retrieving certificates and CRLs is via a directory service, with access via LDAP. Other options include X.500 compatible directories, HTTP, FTP, and e-mail.

2.1.4 Providing trust

Each public key user must have at least one public key from a CA that the user trusts implicitly. Organizations can establish and maintain trust within a single security management domain through a thorough audit of the CA's policies and procedures, repeated at regular intervals.

3. How Applications Work With A PKI

The PKI manages the keys and digital certificates used to implement cryptography within applications such as e-mail and messaging, Web browsers and Web servers, electronic data interchange (EDI); in applications that establish secure network transactions or communications sessions over the Web or in VPNs using protocols such as S/MIME, SSL and IPSEC; and in functions such as digitally signed document or code. In addition, applications developed in-house can be PKI-enabled [8].

3.1 E-Mail and Messaging

Secure e-mail and messaging use key pairs for encryption of messages and files, and for digital signatures. For instance, e-mail programs like Microsoft Exchange and IBM's Notes Mail are increasingly using encryption to carry sensitive information. The same is true of messaging-based groupware programs, such as Novell's GroupWise. EDI systems support financial transactions that require authentication, privacy and data integrity. The most common secure e-mail/messaging protocol is Secure Multipurpose Internet Mail Extensions (S/MIME), which extends the Multipurpose Internet Mail Extensions (MIME) standard.

3.2 Web Access

Browsers and Web servers use encryption for authentication and confidentiality, and for applications like online banking and online shopping. Typically, using Secure Sockets Layer (SSL), servers authenticate themselves to clients. SSL also encrypts traffic. Client authentication is also an option. SSL is not limited to Hypertext Transfer Protocol (HTTP) but also supports protocols such as File Transfer Protocol (FTP) and Telnet.

3.3 VPN

Encryption and authentication are the main technologies used to convert standard Internet links into Virtual Private Networks (VPNs), used either for site-to-site privacy (router-to- router) or for secure remote access (client-to-server). These functions are implemented in the context of a tunnelling protocol that wraps (or "encapsulates") one protocol in another protocol. For instance, the encapsulated protocol may be Point-to-Point Protocol (PPP), while the encapsulating protocol may be Internet Protocol (IP). The emerging standard for site-to-site tunnelling is the IP Security (IPSEC) protocol from the IETF.

3.4 Digitally Signed Code and Files

Increasing reliance on downloaded programs and files has raised security concerns, particularly in the area of virus control. Technologies like Microsoft's Authenticode use RSA digital signatures to verify the source and the integrity of the content. A PKI is utilized in order to scale this approach to the huge numbers of users and programs requiring such services.

3.5 Standards That Rely on a PKI

Most major security standards are designed to work with a PKI. For instance, Secure Sockets Layer (SSL), Transport Layer Security (TLS), Secure Multipurpose Internet Mail Extensions (S/MIME), Secure Electronic Transactions (SET) and IP Security (IPSEC), all assume, require or allow the use of a PKI.

3.6 S/MIME

S/MIME is the IETF standard for secure messaging. S/MIME assumes a PKI for digitally signing messages and to support encryption of messages and attachments, without requiring prior shared secrets. Because e-mail is the most mature of the popular Internet applications, the S/MIME committee has led the way in implementing and extending PKI standards, taking advantage of the PKIX standards when possible, and filling in where additional standards were necessary. The most important standards developed by the S/MIME committee are Cryptographic Message Syntax, Message Specification, Certificate Handling, and Certificate Request Syntax.

3.7 SSL and TLS

SSL and the emerging IETF standard, TLS, which is based on SSL, are the most important standards for providing secure access to Web servers. SSL and TLS are also being used for general client/server security in a variety of non-Web applications. Both rely on a PKI for certificate issuance for clients and servers.

3.8 Secure Electronic Transactions (SET)

SET facilitates secure electronic bank card payments. SET uses keys for authentication, confidentiality and data integrity. PKI is a critical underpinning for authentication of the parties involved in a payment transaction.

3.9 IPSEC

The IETF Internet Protocol Security Protocol (IPSEC) standard defines protocols for IP encryption, and is one of the primary protocols used for deploying VPNs. IPSEC requires keys for IPSEC are emerging, and a PKI is the most scalable way of managing IPSEC keys. Use of IPSEC is still fairly limited. However, the need for PKI will grow with IPSEC deployment.

3.10 A Basic Private-Key-Generation Scheme

In the previous example, Bob's identifier is used as his public key. In fact, any type of string can be a public key in AC-PKI. For instance, Alice can encrypt a message using as his public key Bob's identifier concatenated with any desired information, e.g., "*IDB*|| current-date || role = captain", where "||" denotes the concatenation of messages. By doing this, Alice attempts to make sure that if and only if Bob is a captain who holds the valid private key on the specified date, could he decrypt the ciphertext. An interesting property here is that Bob does not need to possess the corresponding private key beforehand. He can request the private key from the TA after receiving the ciphertext. Such on-demand means of private-key requests coincides well with the dynamic, resource-constrained nature of MANETs. Obviously, we can accomplish more nice properties that do not exist in a conventional PKI setting by concatenating the destination identifier with different information. Such a nice feature, however, poses the demand for a PKG scheme during network operation: the destination may not have the needful private key in hand so that it should be able to obtain it from somewhere problematic to use a single TA in MANETs because it may become the single point of failure. To enable a robust PKG scheme, Shamir's (*t,n*) secret-sharing technique can be employed to distribute the TA functionality among a set of pre-selected nodes such that as long as there are no less than *t* such nodes being functional, mobile nodes can still ask for private keys from them.[6]

3.11 Anonymous and Certificateless Public-Key Infrastructure (AC-PKI) for MANETs.

Our contributions are mainly threefold. First, we apply Shamir's secret-sharing technique to distribute the system trust, essentially a system master-key, across a pre-selected set of nodes, called distributed private-key generators (D-PKGs). D-PKGs collaboratively offer a prerequisite private-key-generation (PKG) service during network operation. Second, we propose offering D-PKGs anonymity protection to defend against pinpoint attacks that are quite easy to conduct and may cause devastating consequences in MANETs. Last, we determine the optimal secret-sharing parameters for achieving the maximum security.[3]

We now present our main contribution, the design of a fully decentralized public key infrastructure, Key-Chains. This PKI is built on top of the Local Minima Search (LMS) protocol, which is capable of efficient storage and retrieval of data over a network. We begin with a brief discussion of the LMS protocol and then describe the modifications to this protocol needed to realize our PKI. A brief overview of LMS provides only the general concepts and terminology. The semantics of LMS are similar to those of a DHT: peers and objects are mapped into an *identifier space* using consistent hashing, and objects are stored at peers determined by the distance between objects' and peers' identifiers in this space. Each peer knows the identifiers of peers within h hops of it in the network, which needs its h-*hop neighborhood.* Rather than storing an object at the peer with the globally smallest distance between their identifiers, LMS stores multiple *replicas* of the object at *local minima*: peers with the smallest identifier distance in their neighborhoods. A peer performing either storage or lookup sends a number of *probes* into the network. These probes are forwarded to local minimum using _rst a _xed-length random-walk (mixing) phase followed by a deterministic phase. Forwarding is always done along undirected links between peers; peers never contact one another directly except according to the overlay graph. Local minima are selected randomly, and the performance of the protocol relies on storing enough replicas and performing enough searches that there is a high probability of locating at least one replica.

The properties of LMS are nearly ideal for implementing a PKI. First, because LMS runs over arbitrary topologies it can be run on a peer-to-peer system where the topology reproduces the web of trust certificate graph.. Furthermore, due to the way probe messages are forwarded in LMS, using LMS to search for public keys means that whenever a probe locates a target public key, a certificate chain from the initiator to this target can be reconstructed directly from the path taken by the probe. Actually creating this certificate chain requires significant modifications to LMS, including extending the protocol to directed overlay graphs. We assume that each *principal* (user) is associated with a *peer*, a host that runs the KeyChains protocol. KeyChains does not impose specific trust relationships between peers and principals. Multiple principals may be associated with a single peer; e.g., a departmental server can hold certificates for all department members. In the rest of the discussion, however, for simplicity we assume that there is a one-to-one mapping between peers and principals. Peers never need to know the private keys of their principals, so compromising a peer does not compromise its principal. Principals generate certificates for one another using some out-of-band mechanism, and relay these new certificates to their peers. The REPLICA-PLACE message for key PKV is sent from initiator v along reverse validated edges, collecting the certificates corresponding to the edges on the path. Finally, the message arrives at local minimum t, who stores PKV together with the certificate path. Search involves forwarding a message along validated edges until it reaches the local minimum t, which then responds with the stored replica and constructs the certificate path back to w pair h .B's public key is PKB and B is trustworthy.; _Ai, where _A denotes A's signature on the statement. The PGP web of trust is simply the union of all certificates existing in the system. Principals can associate a measure of trust (that is, how likely a principal is to issue *correct* certificates) with a certificate, and trustworthiness can then be computed from a certificate chain [19]. PGP allows for different levels of trust, though these values are not exported to the keyservers. Principals and peers are distinct, each having its own public key. This means that when principals exchange public keys, they must also exchange the public keys of their peers. This

allows one peer to authenticate another when they connect, and we say that this in turn *validates* the link between the principals. The set of all validated links needs the *trust graph*. A directed edge from A to B exists in this graph if and only if there exists a certificate certA(B; PKB) and A's peer can authenticate its connection to B's peer. [4]

3.12 Attacks on the PKI

In this section, we argue that Key Chains has very strong security properties: the system is resilient to a wide range of attacks. We describe potential attacks in terms of an adversary A. A particularly powerful adversary is one that corrupts principals and is therefore able to insert itself into the trust graph and create certificates with whatever public keys it chooses bound to any principal (in the network or not). Note that such an adversary is, in effect, attacking the web of trust model rather than our particular system. The decentralized nature of the PKI, however, mitigates the damage that even such an adversary can cause. Peers accept bogus certificates only if they are unable to need more trustworthy correct certificates, so only the peers in the immediate neighborhood of corrupted principals are expected to be significantly impacted. Because this is a more general attack against the model (and is, in fact, inevitable in the absence of a centralized authority), rather than present the details here. For the attacks against the protocol, we restrict A to actively interfering with communication between peers, but not corrupting any peers. We need only consider message deletion attacks, since message insertion and modification attacks are easily dealt with using well-known techniques (recall that messages are routed only between peers who share a secure channel). In a message deletion attack, A targets some of the peers and deletes most or all of the messages to and from them, effectively cutting them off from the rest of the network. If A can cause specific subsets of nodes to fail, it can partition the network. We simulate this attack and show that KeyChains provides resilience against different forms of this attack.[10]

A more powerful adversary might be able to corrupt peers. The adversary is unable to generate new certificates, since it does not know any principals' private keys. It can, however, cause the corrupted peers to either refuse to participate or attempt to bias the behavior of the protocol. Refusal to participate is the same as the previously discussed attack. Biasing the protocol behavior can take the form of either a denial of service attack or a *path-biasing* attack on the certificate chains returned during successful public key retrievals. In the denial of service attack, the corrupted peers assert that any placement probe forwarded to them has succeeded even though no replica is stored and the remaining path from the corrupt peer might be entirely bogus. These peers then silently drop any search probes they receive. In the path-biasing attack, A returns valid certificate chains distributed differently from the certificate chains that KeyChains would naturally return. For instance, A may cause only long certificate chains to be returned with the assumption being that users will view such chains as untrustworthy. In order for this attack to be effective in a large network, A has to compromise a sizable fraction of the peers. This is because (in an undirected network), if the adversary controls only a small fraction of the peers, the probability that any individual probe of an uncorrupted peer v is affected is very small for all but a small fraction of the honest nodes .

The PKI implementation is split into two separate programs: the peer and the user interface client. Multiple clients can connect to a single peer, with different users, though one user is identified as the *owner* of the peer. The peer maintains an access control list indicating what operations (key storage, key retrieval, and peer management) are permitted to a particular user. Each peer and user is associated with a unique 160-bit identifier. A peer's identifier is the SHA-1 hash of its 1024-bit RSA public key. A user's identifier is the SHA-1 hash of his or her email address, as embedded in an X.509 certificate also containing the user's public key [3].

The final issue we address is processing time. The processing time for a message is taken as the time between when a peer receives a message and when it forwards the next message, as reported by get time of day. This time averages 30:0 _ 0:4 ms, with a slight increase of 0:2 ms/KB as message size increases or a processing capacity of over 30 messages per second (assuming only one processor is available to the peer). We note that the processing time for a single message is essentially independent of the overall size of the network.

4. Conclusions

This paper presented our preliminary results about the applications of identity-based public-key cryptography in MANETs. Specifically, we proposed AC-PKI, a novel Anonymous and Certificate-less Public-Key Infrastructure to efficiently and securely provide public-key services without using public-key certificates. To satisfy the demand of private keys during network operation, we designed a distributed private key generation scheme by utilizing Shamir's *(t,n)* secret sharing technique to distribute a system master-key among a set of pre-selected nodes, called D-PKGs. In addition, D-PKGs were offered anonymity protection to defend against pinpoint attacks, which makes AC-PKI more secure than previous applications of the secret-sharing technique in MANETs. We also determined the optimal secret-sharing parameters (*t, n*) to achieve the maximum security and designed a novel protocol to dynamically adjust (*t, n*) to accommodate dynamic node join/leave. As the future research, we intend to evaluate and justify the efficacy of the proposed schemes through simulations and practical implementations.

Public key cryptography and certificates are emerging as the preferred enablers of strong security for a number of applications, including e-mail, Web access, VPNs and digitally signed code. A PKI manages keys and certificates for people, programs and systems. PKI standards, such as the PKIX specifications, allow multiple PKIs to interoperate, or multiple applications to use a single PKI. This makes PKI consolidation possible, facilitating a manageable, scalable PKI. To successfully deploy a PKI, organizations must develop a sound strategy, plan for interoperability, determine how applications will interface with the PKI, size the initial project correctly, and plan for technical staff requirements.

References

[1] Shamir, "How to Share a Secret," *Communications of the ACM*.
[2] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Networks*.
[3] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad hoc networks," *IEEE ,* Nov. 2001.
[4] S. Yi and R. Kravets, "Moca: mobile certificate authority for wireless ad hoc networks,"Apr. 2003.
[5] M. Narasimha, G. Tsudik and J.H. Yi, "On the utility of distributed cryptography in p2p and manets: the case of membership control," Nov. 2003.
[6] M. Bechler, H.-J. Hof, D. Kraft, F. Pahlke, and L. Wolf, "A cluster-based security architecture for ad hoc networks," in *IEEE INFOCOM'04*, Mar. 2004.
[7] Shamir, "Identity based cryptosystems and signature schemes," in *Proc. CRYPTO'84*, 1984.
[8] D. Boneh and M. Franklin, "Identify-based encryption from the weil pairing," 2001.
[9] P. S. L. M. Barreto, H. Y. Kim, B. Bynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems,", August 2002.
[10] KeyChains: A Decentralized Public-Key Infrastructure_Ruggero Morselli Bobby Bhattacharjee Jonathan Katz Michael Marsh ,*University of Maryland*
[11] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Anonymous communications in mobile ad hoc networks," .
[12] M. Pease, R. Shostak and L. Lamport, "Reaching agreement in the presence of faults," Apr. 1980.
[13] J.C. Cha and J.H. Cheon, "An identity-based signature from Gap Diffi-Hellman groups," 2003.
[14] Perkins, E. Belding-Royer and S. Das, "Ad hoc on-demand distance vector (AODV) routing," July 2003.
[15] Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: how to cope with perpetual leakage," extended abstract, IBM T.J. Watson Research Center, Nov. 1995.