

# PGPticket - A Secure Authorization Protocol

Vinnie Moscaritolo, Chief Consulting Engineer  
Network Associates - Total Network Security Division  
<[vinnie@vmeng.com](mailto:vinnie@vmeng.com)>

## Abstract

*The control of user access through secret passwords and centralized authentication databases dates back to early timeshare systems. However, this strategy is no longer scaleable or secure enough for today's highly distributed, Internet based services.*

*This paper discusses the limitations of traditional user authentication and authorization methods and offers a single sign-on alternative using strong-cryptography and the same PGP key infrastructure that is widely deployed on the Internet today. PGPticket, a lightweight but very secure authorization protocol based on the SPKI and OpenPGP standards is designed to control access of services over a public network. PGPticket grants and transfers user access privileges through authorization certificates signed with strong public key cryptography.*

## Rethinking User Authentication

It's time to rethink how we perform user authentication and authorization – the process by which we establish the genuineness of a user's identity and what we allow them to access over a network.

The widespread acceptance of the Internet has introduced changes in how we access computer services. Timeshare systems have been replaced by Internet-based web servers, and data communication sessions are no longer sheltered by the privacy of direct connections. User authentication information which was previously private is now broadcast over public networks and available to anyone with a packet sniffer and the right opportunity. Consequently, many of the existing computer security practices are insufficient to handle the broader threats found in an open network.

We can no longer live with the false sense of security that our existing infrastructures are adequately protected. To securely participate on the Internet today, one must appreciate that any data not protected by strong encryption is open to access by nefarious individuals.

## Attacking Network Services

Ensuring the secure authentication of the user – the human being sitting at the keyboard – is one of the most critical factors in protecting computer services from unwanted intrusion. Yet many of the authentication systems used today by network

services are vulnerable to attack with techniques that are both well-known to hackers and difficult to defend against<sup>[CERT]</sup>.

## Password Guessing

Although generally considered a crude form of attack, automated password guessing can sometimes be an effective way to break into network services. A guessing attack can use a brute force method – trying every combination of consecutive characters – or a more sophisticated dictionary attack where the program applies only valid words and names.

Password guessing attacks can be performed either on or off line. In an on-line attack, a program simulates a user performing a network server login, trying each password combination until it's successful. In most cases, there are easy ways to foil this kind of attack, such as locking out the account or alerting the system administrator after a number of consecutive unsuccessful attempts.

Off-line attacks, however are a whole different matter. Off-line password guessing attacks occur when the attacker is first able to obtain a copy of the servers password database. This usually involves exploiting an indirect server security weakness, such as reading the database off a system backup set or using a web server cgi-script. Once acquired by the attacker, the database can be analyzed for potential passwords in the relative safety of his own machine. In fact, these password databases are such prime targets for hackers that quite a number of utilities

have appeared on the net to decode and crack password files.

### Replay Attacks

The wide availability of low-cost packet sniffer and analysis software today enables even a novice hacker to easily wiretap and capture passwords from a network. The attacker, located somewhere on the same wire as the user and the server, simply records the user entering his password. Later the login session is played back to the host computer, sometimes from a location different from the original recording.

### Session Stealing

The attacker watches for a user to initiate a network login sequence. The attacker then performs a denial of service attack on the user by sending a forged reset packet to the user's TCP connection, causing it to close immediately. The attacker then hijacks the user's authenticated session to the server, and is able to access all services for which the victim was authorized.

### Infrastructure Penetration

The infrastructure used to carry Internet communications depends on software-controlled routers and name-servers to direct network traffic. An attacker will target these devices and reprogram them to forward or redirect a specific user's network traffic to an attacker's computer.

Once the communications infrastructure is compromised, the attacker is clear to mount a "Man-in-the-Middle" attack. In this attack, the user's client software is tricked into believing that it's communicating with the server, but in reality the packets are directed to the attacker's computer. Each message that the user sends is recorded and relayed to the intended recipient. Man-in-the-Middle attacks are even effective when client-server communications are encrypted. The user notices no loss of service and suspects nothing, but the attacker is able to read all traffic, and possibly even prompt the user to give up its secret passwords.

### Device Penetration

Device penetration attacks often involve the use of a virus or Trojan horse application. One very effective attack is to modify the operating system to capture all of a user's keystrokes and send them to a remote attacker for analysis. For this reason, many sites now employ anti-viral software.

An attacker may try to circumvent these security precautions by modifying a local DNS server to point to a counterfeit web site. The user then downloads what they think is a bona-fide software distribution, but in actuality they have loaded a version of the intended software that contains a Trojan horse. The defense for this scenario is to get vendors to digitally sign their distributions.

### Social Engineering and Rubber Hose Attacks

The weakest link in any security system is typically the users themselves. People can be fooled, coerced, or intimidated into giving up secret information. Passwords can also be obtained by shoulder-surfing – watching someone type his/her password into a keyboard. In practice, defending against these low-tech attacks can be very difficult, because it requires management acknowledge the threat and support threat awareness education of the users..

## User Authentication Methodologies

Fortunately many of these techniques for sidestepping security measures have been documented and a number of approaches have been proposed to prevent unauthorized access<sup>[FIPS190]</sup>. In order for us to begin formulating a security strategy, we must first understand the strengths, weaknesses, and tradeoffs of the various user authentication methods available.

### Local & Remote Authentication

Authentication can be performed on a local or remote basis. Which scheme is used can have subtle but profound security implications.

In a local-authentication scheme, the authentication material never exits the user's computer and hence is less likely to be revealed to an attacker. For example when a user enters a passphrase to mount an encrypted PGPdisk volume, all authentication is done locally on the user's computer.

However, in remote-authentication systems, for instance mounting a network volume, the authentication material must first travel to the server and then be compared against a stored template. This procedure requires that a copy of the authentication material reside on the server as well. In this case, the user has relinquished control of his/her secret material to an outside party. Thus the secrecy of this authentication material not only depends on the security of the method used to transport the key material but also the integrity of the server.

Unfortunately, experience has proven that a secret known by more than one person isn't really a secret at all.

If an attacker manages to obtain even read-only access to a server's password database, they will then have the tools needed to masquerade as any of that server's authorized users. This provides a vulnerability upon which attackers can concentrate their efforts.

## Methods of Authentication

There are three factors used to establish a user's identity or authenticity to a computer, listed in increasing levels of security: something one has (possession of a key or badge), something one knows (a secret password), or something that is unique to the individual (a fingerprint). These factors can be used alone or in combination with each other.

### Something One Has

Most people today already carry on their person some form of hardware token which they use to access a restricted facility or service. These tokens are usually personal, mobile, and convenient (they fit in your pocket), such as a corporate ID badge, ATM access card, or even a one-time password generator.

However, hardware tokens are seriously limited when used as a sole form of authentication – they only provide proof that the device is in the hand of a user at the moment of authentication, but not necessarily the owner of the token. If a token is stolen, it can be used by an unauthorized individual. Unless hardware tokens are combined with some other form of authentication such as a secret password, they are usually considered low security.

Hardware tokens are available in either passive or active forms. Passive tokens are really nothing more than a convenient key or password storage device. Since these keys are static, they are very susceptible to a number of attacks, especially the replay attack. On the other hand, active tokens can be used to reply to a random cryptographic challenge. Properly implemented, the challenge will be unpredictable and immune to replay attacks.

Smartcards and recently the iButton<sup>1</sup> are a class of active hardware token that in addition to acting as repository for encryption and digital signature keys,

<sup>1</sup> The Crypto iButton™ is a hardware token device the size of a watch battery manufactured by Dallas Semiconductor. See <http://www.ibutton.com>

actually have on board crypto-engines. These devices can perform all crypto and signing functions on the card and never require the secret key material to leave the device.

Since they also require the addition of a password, this kind of token prevents misuse by an unauthorized person like a pickpocket.

### Something One Knows

Using something one knows – such as a secret password or PIN – to authenticate a remote user is a method that has been in use since the earliest timeshare systems and is today's most commonly-adopted authentication technique. Password-based authentication systems are also well understood and easy to implement, however when they are used for remote authentication they are also the easiest method to breach. One reason for this is that secret passwords share many of the same inherent key management and security issues as secret-key cryptography.

For example, here is a list of some problems associated with password-based authentication schemes.

- Passwords in transit are very susceptible to packet sniffing and replay attacks, and therefore should never be transmitted as cleartext. This problem can be overcome by augmenting the appropriate network protocol with some form of encryption. APOP and SPEKE<sup>2</sup> are two such examples.
- Simple text passwords are extremely vulnerable to dictionary attacks.
- Complex passwords are difficult for users to remember and manage.
- Passwords are vulnerable to social engineering attacks.
- Although using a secret password locally can be sufficiently secure, using a secret password to remotely authenticate results in that password being known by at least one other entity – in this case the server – and thus no longer “secret”.

<sup>2</sup> Strong Password-Only Authenticated Key Exchange (SPEKE) uses cryptographic techniques similar to the Diffie-Hellman key exchange method for transporting passwords across an insecure communications channel. For more information on SPEKE see David Jablon's paper at <http://world.std.com/~dpj/>

## Something One Is

Using unique, measurable characteristics or traits of human beings to automatically recognize or verify identity is called biometric authentication. Fingerprints, retina scans, and voice recognition systems are all examples of biometrics in use today.

These technologies attempt to verify a person's authenticity by recording the measurements of human body parts or personal traits and later comparing them against a stored template.

Although biometrics are the most secure of the three methods, biometric identifiers are static and can be subject to replay attacks. It is best to combine biometrics with another authenticator such as a secret password.

## Multi-factor Systems

By far the best security tactic is to combine more than one authentication method. Corrupting a multi-factor authentication system requires separate attacks on each of the methods. For example, although social engineering or some technical attack might be able to acquire a secret password without the attacker ever physically approaching the victim, gaining access to a hardware token calls for a physical attack or theft. This exposes the attacker to a whole other level of risk. Of course, to be truly secure, a strong-authentication system would require that the user employ all three, a memorized password, a token, and a biometric.

SecurID is an example of a multi-factor system. SecurID uses a credit card-sized device to calculate the response to a new time-based password challenge every 60 seconds. The user enters both their secret password and the one that is currently displayed on the device (a hash of the current time combined with a secret key). This tactic of combining a hardware token with another authenticator such as a password is an great improvement over a token alone.

Although there are a few sophisticated attacks<sup>3</sup> that can compromise the SecurID system, its biggest drawback is that SecurID requires that its servers reside in a physically secure location. In addition users must maintain a separate authentication device for each server site they wish to access. These

<sup>3</sup> The SecurID system is open to denial of service and race attacks. There are also communication weaknesses when the SecurID authentication server must support remote content servers. For more information see "Weaknesses in SecurID" <ftp://ftp.secnet.com/pub/papers/securid.ps>

constraints are somewhat impractical for large scale deployment.

## Single Sign-On Systems

One of the security challenges of password-based authentication is defending against automated password guessing and dictionary attacks. Current practice suggests that passwords be composed of complex variations of alphanumeric and punctuation characters, periodically changed, and not used on more than one server at a time. Not surprisingly, many corporations have instituted security policies which specify this very password usage etiquette.

The reality is that the limitations of human memory make it unrealistic to expect that users will memorize more than one or two passwords reliably. In fact, these complex password guidelines actually encourage people to resort to grossly insecure methods of remembering their passwords such as writing them down and storing them in a convenient but unprotected location.

Experience shows that minimizing the number of secret passwords a user needs to remember not only greatly improves the user experience, but also boosts the overall security of a networked system. This presents us with what seems to be a dilemma – we wish to impede password-guessing attacks, but at the same time we need to decrease demands on the user.

What is needed is an integrated single sign-on subsystem, in which the user authenticates themselves to the local machine thus giving it the authority to negotiate all subsequent authentication to remote services autonomously.

## Password Caches and Keychains

One early single sign-on implementation was the PowerTalk Keychain<sup>4</sup>. The Keychain was a piece of code that would intercept server logins, and record the most commonly used passwords in a (weakly-encrypted) local cache. Once unlocked by a user's master password, the Keychain transparently provided

<sup>4</sup> Apple Computer provided the "Keychain" as part of AOCES System (PowerTalk), which shipped beginning with System 7.1.1. This keychain was used primarily to store AppleShare and PowerShare server passwords. Although the implementation was slow and awkward, its single point of authentication for multiple services was widely desired. In fact, many users installed PowerTalk exclusively to use the keychain. After its demise with the release of Mac OS 7.6, a number of independent implementations of the Keychain were developed from both within and outside Apple.

authentication when the user later wanted to access a server.

However, password caches – in addition to sharing the same vulnerabilities as password-based authentication systems – must also tackle the following issues:

- Many users juggle between more than one computer, such as an additional laptop or home office. The password cache must be kept synchronized across those systems.
- Unless the applications that use the password database are designed from the ground up with security in mind, any advantages gained from encrypting the passwords can be lost. For instance, what good does encrypting a user password do if the file-server login can occur in cleartext?
- Integration of single sign-on and two-factor systems is either awkward or insecure.
- Unless the password cache database is strongly encrypted, the theft of a laptop computer containing a user's password cache can have serious consequences for both the user and the organization.
- If the application programming interface is not properly designed, the password database could be accessed by a rogue application or virus designed to export clandestinely all passwords to a remote attacker.

These weaknesses place such limits on the utility of password caches that they become a risk for highly secure applications. Fortunately, there are other ways to implement single sign-on systems.

## Kerberos

The Kerberos authentication system is a popular single sign-on solution adopted by many higher education installations. Kerberos, developed at the Massachusetts Institute of Technology, uses a combination of symmetric or secret-key encryption and distributed databases to authenticate user login from any computer located on a local network.

Although an ingenious design, its dependence on symmetric encryption technology introduced a number of key management constraints – the overall security of a Kerberos system depends on the trust of

its authentication servers<sup>5</sup>. Even with the proposed public-key extensions, a Kerberos system still requires synchronization of its participant clocks. Thus to prevent compromising the entire system, the Kerberos authentication and time servers must be kept in a restricted area.

These restrictions, although quite suitable in a tightly managed network environment like a college or corporate campus, make Kerberos an inappropriate solution for the needs of small businesses or large scale Internet deployment.

## Beyond Passwords

Since the inherent security weaknesses and management complexities of password based authentication systems make them inadequate for remote user authentication over public networks, it is time that we look beyond passwords.

A modern user authentication system should achieve the following goals:

1. Provide a single sign-on experience; a user needs to only remember one password. This password is used to authenticate him to a local system which is delegated the authority to sign on to remote systems.
2. Strong user authentication, extendable to multi-factor methods. The only copy of the authenticating (secret) material is in the possession of the user.
3. The system does not depend on a trusted server base.
4. If a server is compromised it should have no effect on other servers.
5. A system should build on existing and well accepted infrastructures and must scale to fit a large base of users and servers.

Fortunately there are a number of cryptographic technologies available today that can help accomplish these goals.

---

<sup>5</sup> See Bellare, S., and M. Merritt, "Limitations of the Kerberos Authentication System", Computer Communications Review, October 1990.

## Authentication with Cryptographic Signatures

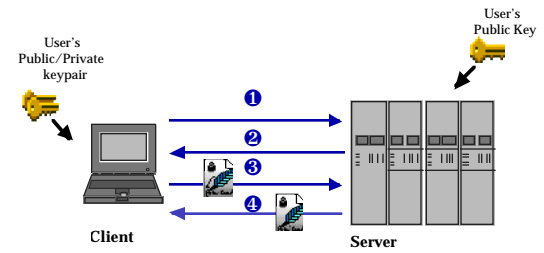
E-mail communication on the Internet faces a security challenge similar to network user authentication. Since e-mail packets are broadcast through public networks, these messages can also be eavesdropped on and counterfeited by unauthorized individuals. However e-mail vendors have been able to address this problem by leveraging public key cryptography to digitally sign and authenticate e-mail messages.

In a public key cryptosystem, each user or *principle* generates a pair of mathematically-related cryptographic keys. These keys are created in such a way that it is computationally infeasible to derive one key from the other. One of the keys is made publicly available to anyone who wishes to communicate with that user. The other key is kept private and never revealed to anyone else. This private key is secured by either placing it in a hardware token or further encrypting it to a passphrase, or both. The private key is then used by the principal to digitally sign data. This digital signature can later be checked with the matching public key to ensure that the data has not been tampered with and that it originated from the holder of the private key.

Since the holder of the private key is the only entity who can create a digital signature that verifies with the corresponding public key, there is a strong correlation between a user's identity and the ability to sign with that private key. Thus the public key can function as a principle's identity in cyberspace<sup>6</sup>.

Because they provide such strong testimony to the authenticity of the sender, public-key cryptographic systems are becoming very popular among today's Internet e-mail users.

The same digital signature technology that guarantees the identity of the e-mail sender can be used to strongly authenticate users of network services, too. By having the user sign a random challenge message from the server at login, the server can verify the identity of the user with his public key. The server randomly creates the challenge message to prevent replay attacks. This process is illustrated in the following scenario :



1. The user initiates network service access.
2. The server looks up the user's public key in its authentication database. The server then generates a random challenge string and sends the challenge to the client.
3. The client digitally signs the challenge string and returns the cryptographic signature to the server. The client also sends a counter challenge string which is used to verify the server's authenticity.
4. The server then checks the client's signature and, if successful, grants access. It also signs and returns the client's counter-challenge.

This process provides several significant advantages over password based user authentication systems:

- Since the same key and passphrase used to sign e-mail messages is also used for server access, the user need only remember one passphrase. This is done without the risks associated with passphrase caches.
- Strong user authentication is provided by cryptographic digital signatures.
- Since the user maintains the only secret material in the system, compromising a server's user database results in limited damage.

While similar systems have been proposed by a number of researchers <sup>[KEMP][FTPDSA]</sup> they lacked the availability of a widely-accepted key infrastructure.

Systems like PGP<sup>7</sup> already have a large key infrastructure.

<sup>6</sup> In practice, a secure hash of the public key, also known as a key fingerprint, is used to identify the key holder.

<sup>7</sup> OpenPGP is a proposed internet standard that uses strong public-key and conventional cryptography to provide security services for electronic communications and data storage. These services include confidentiality, key management, authentication and digital signatures. A subset of OpenPGP is widely in use today thanks to the popularity of PGP for encrypting Internet mail

Implementation notes.

- ⇒ Replace UAM/PAM to use public keys, store copy of fingerprint or even key-id in access control database.
- ⇒ system needs only keep copy of key fingerprint can access key from key server and verify authenticity by comparing hash.
- ⇒ cache keys client keys, since same users come back.

## User Authorization Databases

Maintaining the appropriate level of security can also introduce a number of management challenges to server administrators; Keeping track of user authorization databases can quickly become cumbersome when more than a handful of servers are involved.

When a new user wishes to gain access to a server, the system administrator adds their name and access information to the appropriate server database, but if the user wishes to access a number of servers, this process must be duplicated on each server.

When a user departs from an organization each of the servers must then updated to reflect this change. Often servers that are managed by different departments can be overlooked thus creating a security risk.

Although there are a number of automated systems in place to replicate or centralize the authorization databases, they typically share the following characteristics:

5. The authorization server itself must be physically secure and thus is the weakest link in the security chain.
  6. Each server must be in communication with the authorization server to verify user identity and authorizations. This can an unreasonable requirement for remote sites.
- The authorization server is an ideal target for denial of service attacks, since they affect all the servers managed by it.

### authorization wishlist

- ⇒ highly secure
- ⇒ simple yet flexible authorization
- ⇒ no synchronization issues

- ⇒ decentralization of authority, nottrusted servers
- ⇒ automatic expiration of auth
- ⇒ global revokation
- ⇒ the protocols should be secure even if the network is under the complete control of an adversary.
- ⇒ resilient to denial of service
- ⇒ Accessibility of remote servers is transient.

### Authorization through certificates.

- ⇒ Capabilities base system
- ⇒ SPKI<sup>8</sup> model for auth certs.
- ⇒ Policymaket<sup>[BFL]</sup>
- ⇒ authorization with a signed permission slip.
- ⇒ sys admin issues auth cert with Issuer, Subject, Delegation, Authorization and Validity.
- ⇒ cert is presented by user on access.
- ⇒ system needs to check sys issuer sig, challenge user to authorize.
- ⇒ servers only need admin pub key & ability to verify sig.

The parts of any certificate necessary for trust computations can be expressed as up to five fields: Issuer, Subject, Delegation, Authorization and Validity.

7. The ISSUER generates and signs the certificate.
8. The SUBJECT is the principal or set of principals to which the certificate grants its authorization.

---

<sup>8</sup> Simple Public Key Infrastructure or SPKI is a proposed internet standard intended to provide mechanisms to support security in a wide range of Internet applications, including IPSEC protocols, encrypted electronic mail and WWW documents, payment protocols, and any other application which will require the use of public key certificates and the ability to access them.

9. DELEGATION is a boolean, indicating if TRUE that the Subject is allowed to delegate the specified Authorization further.
10. AUTHORIZATION is a structured field expressing the authorization that this certificate grants to the Subject.
11. VALIDITY is some combination of dates or on-line tests specifying the validity period/conditions of the certificate

⇒ COMBINE PGP AND SPKI.

⇒ why use pgp format vs SPKI sexprs

⇒ pgp format well know and accepted , infrastructure in place

⇒ code exists for parsing pgp keys, crypto

⇒ tickets can be transported and identified in pgp message

⇒ why use pgp and not S/MIME

⇒ flexibility of key infrastructure

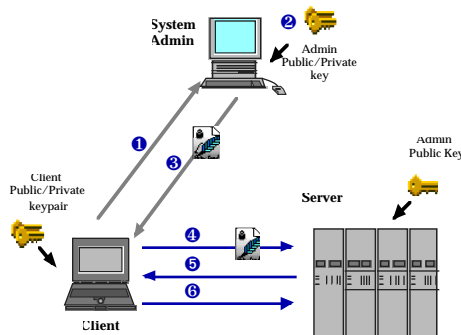
⇒ no need for CA. small sites can self CA

⇒ Although PGPticket defines auth certs, ident certs are possible also.

6. The user requests server access from the system admin (possibly a verbal request). The user either provides a copy of his public key or makes the key available on common key-server. The Admin verifies the validity of this key.
7. The administrator generates the PGPticket with appropriate authorizations and validity information, signing the ticket with the server admin key. The ticket is either posted in a public place or sent by email to the user.
8. The user retrieves the PGPticket and stores it in a local ticket database.
9. The user attempts to access a network service, the client searches its ticket database for any tickets pertaining to the server, and sends a copy of the tickets along with the access request.
10. The server checks validity of ticket, which entails verifying the admin signature and expiration date of ticket. The server then generates a random challenge string and sends the challenge to the client requesting that it be signed by the key specified in the ticket.
11. The client signs and returns the challenge string with a random nonce appended. The server then checks the clients signature and if successful grants access with the authorizations specified in the ticket

## A break with tradition

The PGPticket architecture introduces a change in how authorization is managed for network services. Instead of maintaining a per-server database of users names, passwords and their corresponding access rights, the system administrator generates a cryptographically signed authorization certificate or permission slip valid only for a specific users signature key. This authorization certificate is presented upon accessing a restricted service.



## PGPticket Architecture

⇒ packet format<sup>[PGP]</sup>

⇒ binary / ascii armored

⇒ ticket id

⇒ multiple subjects vs groups

⇒ delegation issues

## Possible Attacks

⇒ Preventing Session Stealing

⇒ any session over TCP/IP (with or without initial user authentication) can be hijacked unless it is protected by encryption or some continuous message authentication. (That's why so many sites now use OTPs over SSH.)

⇒ rfc2104 - HMAC: Keyed-Hashing  
for Message Authentication

⇒ encrypted links

## Example Applications

⇒ File Servers

⇒ WebServer proxy

⇒ Firewalls & VPNs

⇒ Screensharing

⇒ Linux PAM

## Implementations

⇒ PGPticketlib & pgpsdk

⇒ PGPticketmgr

## Futures

⇒ Ticket Management

⇒ auto ticket renewal

⇒ ticket revokation

⇒ key revokation

⇒ key sharing/ticket sharing

⇒ anon users.

## Acknowledgments

We are grateful for the helpful comments on this paper from Jon Callas, Kathy Massucci, Eric Gundrum

Carl Ellison, Hal Finney, Colin Plumb, Steve Schoenfeld, Robert Hettinga.

## About The Author

Vinnie Moscaritolo, chief consulting engineer for Total Network Security Division of Network Associates (formerly PGP) runs the PGPsdK Developer Services Group. It is rumored that he lives in a survivalist cabin somewhere deep in the woods of the Santa-Cruz mountains, with CZ the girl of his dreams and a dozen or so rabbits. All well out of the reach of those pesky black helicopters. He can be reached by sending email to [vinnie@vmeng.com](mailto:vinnie@vmeng.com) and his PGP fingerprint and other papers can be found on his very own bizarre web page: <http://www.vmeng.com/vinnie/>



---

[CERT] Computer Emergency Response Team, "IP Spoofing and Hijacked Terminal Connections", CA-95:01, January 1995, [ftp://info.cert.org/pub/cert\\_advisories/CA-95:01.IP.spoofing](ftp://info.cert.org/pub/cert_advisories/CA-95:01.IP.spoofing)

[FIPS190] NIST FIPS Pub 190, "Guideline for the Use of Advanced Authentication Technology Alternatives" 28-Sep-1994 , <http://csrc.ncsl.nist.gov/fips/fip190.txt>

[KEMP] D. Kemp, "The Public Key Login Protocol", draft-kemp-auth-pklogin-03.txt, 30-Jul-1997, <http://www.ietf.org/internet-drafts/draft-kemp-auth-pklogin-03.txt>

[FTPDSA] Russell Housley, William A. Nace, Peter Yee, "FTP Authentication Using DSA", <draft-ietf-cat-ftpsaauth-02.txt>, February 1998, <http://www.ietf.org/internet-drafts/draft-ietf-cat-ftpsaauth-02.txt>

[PGP] Jon Callas, Lutz Donnerhacke, Hal Finney, Rodney Thayer "OP Formats - OpenPGP Message Format" draft-ietf-openpgp-formats-00.txt Nov-1997.

[BFL] Matt Blaze, Joan Feigenbaum and Jack Lacy, "Distributed Trust Management", Proceedings 1996 IEEE Symposium on Security and Privacy.