PGPticket
INTERNET-DRAFT
Expires: 16-Feb-2000

Vinnie Moscaritolo
Apple Computer Corp.
Antonino N. Mione
Rutgers University Computing Services

PGPticket
<draft-moscaritolo-mione-pgpticket-03.txt>

Status of this Memo

Abstract

OpenPGP specifies message formats and certificate formats used for exchange of encrypted and/or authenticated objects. This document discusses methods of extending OpenPGP's message formats to support an authorization system. This system would use public key cryptography to authenticate a user to a server and establish the user's access permissions. The concept is that the user acquires a ticket signed by some issuer that specifies what they are entitled to do. That ticket is then submitted to a server. The server uses a challenge/response method to verify that the holder really has the matching private key. The server then allows the access specified.

Table of Contents

Status of This Document.....1
Abstract.....1

Table of Contents.....2

1. Introduction.....3
1.1 Target Audience.....3
1.2 Purpose and Motivation.....3
1.3 Related Formats and Protocols.....3
1.4 Attribute Certificates.....4
2. Security Considerations.....4
2.1 Public Key Cryptography Limitations.....4
2.1.1 Public Key Exchange and Key Trust.....4
2.1.2 Key ID vs. FingerPrint.....5
2.2 Expiration Dates.....5
2.3 Security Risks When Signing a Challenge.....5
2.4 Random Numbers.....5
3. Protocol Description.....6
3.1 PGPticket Format.....6
3.1.1 Overview of Ticket Structure.....6
3.2 PGPticket Field Descriptions.....7
3.2.1 Packet Tag.....7
3.2.2 Hashed Subpacket Length.....8
3.2.3 Certificate Validity Information [Part 1].....8
3.2.4 Certificate Issuer Information.....8
3.2.5 Certificate Validity Information [Part 2].....8
3.2.6 Certificate Authorization Information.....8
3.2.7 Certificate Subject Information.....8
3.2.8 Unhashed Subpacket Length.....9
3.2.9 Signature Verification Check.....9
3.2.10 Signature.....9
3.3 RADIX-64 Encoded PGPticket.....9
3.4 Ticket Acquisition.....10
3.4.1 Out-of-band.....10
3.4.2 Possible Inband Ticket Requests.....11
3.5 Ticket Use.....11
3.5.1 Ticket Presentation.....11
3.5.2 Ticket Verification.....12
3.5.3 Issue of the Challenge.....12
3.5.4 The Response.....12
3.5.5 Acceptance.....13
3.5.6 Rejection.....13
4. Examples.....14
4.1 Non-anonymous FTP with no remote account.....14
5. References.....18
6. Authors' Addresses.....18

1. Introduction

1.1 Target Audience

This document is intended for implementors of systems that will issue or verify a PGPticket. The document should also be reviewed by service administrators to help them determine reasonable and secure policies for the use of PGPticket.

1.2 Purpose and Motivation

There is an enormous need for secure but flexible authentication technologies. This is illustrated by many of the standardization efforts under way dealing with authentication, authorization, and Public Key Infrastructures. The complexity of these schemes grows exponentially when they are extended to a world-wide community.

The motivation for this document is to specify a method of creating PGP authorization certificates that may be used to grant various types of access to services. These methods are based on proven, secure cryptographic technology. The tickets defined in this documente a compatible variant of the OpenPGP stand-alone signature packet. They provide access to resources in a fashion similar to existing token card methods but without the drawback of requiring extra hardware.

1.3 Related Formats and Protocols

Several existing and developing standards incorporate similar concepts. The SPKI group is attempting to design an infrastructure based totally on attribute certificates. The primary purpose is to grant access or privileges to the holder of a private key regardless of knowing that holder's human readable name. The key (or hash of the key) is the identity. Its design focuses on use in electronic commerce (i.e. bank transactions, stock transactions, purchasing, etc.) There is insufficient experiential evidence to know of all the drawbacks that may be present in this system.

Kerberos authentication may be used to authenticate a user to a service (and vice-versa). Each principal's (user or server) secret key is kept in a database. This is used by the key distribution center to issue tickets when requested for a user to access a service. These tickets contain a random secret key for use between user and service. The user's key is based on a password known by the user. The user must type the password to a client program in order for that program to decrypt and use the ticket issued by the kdc. The main drawback in this system is that everyone must trust a single secure system. That system is a prime target for attack whether that attack is to access secret keys or to deny service to legitimate users.

OpenPGP uses strong public key and symmetric cryptography to exchange encrypted and/or signed (authenticated) objects. It uses a web-of-trust system to determine if a specific public key is

trusted (believed to belong to the named entity.) This differs from the typical hierarchy of certification authorities (CAs) in a traditional public key infrastructure although PGP's network model of trust can be reshaped to form a hierarchy as well. The network (web-of-trust) model has its own set of advantages and disadvantages. However, additionally, OpenPGP does not address a way use its objects or cryptographic facilities to determine access rights to a system or other resource.

PGPticket combines elements of the above technologies to specify an authentication mechanism free of some of the above mentioned drawbacks. Since the use of the tickets is intended for local operations, we are avoiding (i.e. not specifically addressing) the global naming issue nor the world-wide PKI issue.

1.4 Attribute Certificates

Attribute certificates provide a method of binding access information to a public key. The purpose of such a certificate is to assert that the entity holding the corresponding private key is assigned the attributes listed in the certificate and hence any named access permissions.

Attribute certificates may include an identity but many times do not. SPKI centers around attribute certificates.

2. Security Considerations

The following sections discuss some of the security issues involved with generating and using Public Keys (in general) and PGPticket specifically) for authentication.

2.1 Public Key Cryptography Limitations

2.1.1 Public Key Exchange and Key Trust

Although powerful, public key technologies have some limitations. The largest problem concerns key trust. Distribution of the keys is not a problem since public keys require no secrecy. However, one must have a reasonable assurance that a specific key belongs to the expected entity.

Public Key Infrastructures set up key trust based on statements made by a trusted certification authority (CA). The CA signs electronic documents called certificates in order to bind names and other attributes (access permissions, validity dates, etc) to public keys. Of course, the signature is made with the CA's public key. So that they must be verified by checking a different certificate and so forth. The entire process involves the user collecting a chain of certificates from his/her trusted CA to the CA of the target

certificate. They then check the signature and validity in each certificate in the chain. These trusted CAs are set up in a type of hierarchy where parent CAs certify child CAs and sometimes vice-versa. Problems with these techniques include certificate path

discovery, certificate path validation, and how to handle the compromise of a root CA key. This does not take into account the fact that not everyone will trust every CA to be competent in ascertaining identity before certifying public keys.

This process easily becomes long and complex. The Less complex but more reliable method to establish public key ownership is to use out-of-band methods. This means that you must use the phone or an in-person meeting to acquire and/or verify a key.

A PGPticket will be issued by a service administrator or administration application. The method of acquiring and verifying the requester's key is up to that person or application.

2.1.2 Key ID vs. FingerPrint

In general, keys are hashed to generate a summary called a fingerprint. Practice has included abbreviating that fingerprint to form a key id that is easy to index for searches.

Unfortunately, although the hashes used are 'collision proof' (i.e. it is computationally infeasible to derive a usable key with the same fingerprint), the shorter key ids are not. It has been shown that it is easy to generate an RSA key pair that produces a specific key id. This was called the 0xDEADBEEF attack.

Being able to produce a key with the same key id as someone else's key means that an attacker can masquerade as that user and trick others into encrypting for them. Thus, the key id should only be used for fast lookup. An application must compare full key fingerprints in order to verify it is using the correct key.

2.2 Expiration Dates

The longer a key or certificate exists, the more likely it is that a private key may be discovered (either by technical means or through social engineering.) Policies controlling dates in a PGPticket should be determined by the issuing authority. A good idea is for the service administrator to decide on a reasonable maximum life. They can then use the lesser of that lifetime or the lifetime given by the requester.

2.3 Security Risks When Signing a Challenge

There is a known attack involving acquiring a signature on a challenge. A bogus server can present a challenge which is a pre-computed message digest of some text. The response from the user is that message digest encrypted with their private key. This response would give the server a valid signature from the user on text they had never seen. The bogus server could then send the text

with the valid signature to another party and the original user can not prove that they did not originate the message.

PGPticket avoids this attack by having the user build another standalone signature packet. This packet will contain one hashed subpacket. That subpacket is a Notation field with the datatype 'CHALLENGE' and the data value being the random number issued by the server. Since the signature is made on a hash of the notation packet (not on the challenge itself), this circumvents the attack noted above. See section 3.5.4 for a complete description of the response.

2.4 Random numbers

The random challenge strings generated by the server should be cryptographically strong random numbers. There should be no reasonable way to predict a challenge string based on time, location, sequence, or any other criteria.

3. Protocol Description

This section specifies PGPticket formats and the protocol for generating and verifying a PGPticket.

3.1 PGPticket Format

3.1.1 Overview of Ticket Structure

A PGPticket are simply OpenPGP V4 signature packets with several required Notation subpackets and a couple of constraints.
The format is as follows:

```
Packet Tag
 - One-octet version number (4)
 - One-octet signature type [Standalone] (2)
 - One-octet public key algorithm id
 - One-octet hash algorithm id
Hashed Subpacket Length
 - Two-octet scalar octet count for following hashed subpackets
Certificate Validity Information
 - One-octet subpacket length (5)
 - One-octet subpacket type
       [Signature Creation Time, Critical] (82)
 - Four-octet time field
Certificate Issuer Information
 - One-octet subpacket length (9)
 - One-octet subpacket type
       [Creation Issuer key ID, Critical] (90)
 - Eight-octet key ID
Certificate Validity Information
 - One-octet subpacket length (5)
 - One-octet subpacket type
       [Signature Expiration Time, Critical] (83)
 - Four-octet time field
```

```
        Certificate Authorization Information
         - Two-octet or Five-octet subpacket length (N+13)
         - One-octet subpacket type
                [Notation, Critical] (94)
         - Four-octet flag field (0)
         - Two-octet name length (4)
         - Two-octet value length (N)
         - Four-octet name value ('AUTH')
         - N-octet authorization string
        Certificate Subject Information
         - Two-octet or Five-octet subpacket length (N+13)
         - One-octet subpacket type
                [Notation, critical] (94)
         - Four-octet flag field (0)
         - Two-octet name length (4)
         - Two-octet value length (N)
         - Four-octet name data ('SUBJ')
         - N-octet subject string data.
           For each subject, data includes:
            - Eight-octet key ID
            - One-octet public key algorithm
            - One-octet scalar octet count (M)
            - M-octet public key fingerprint data
        Unhashed Subpacket Length
         - Two-octet scalar octet count for unhashed subpackets (0)
        Hash Value Check
         - Two-octet integer holding left 16 bits of signed hash value
        Signature Value
         - One or more Multi-Precision Integers
                [Depends on Signature Algorithm]
```

The following sections detail the meaning of each field and subpacket. Since these tickets are implemented as OpenPGP V4 Standalone signature packets, the format of the various fields will match the definitions in [RFC2440].

The various fields and subpackets of a PGPticket are positional. Thus, they MUST be in the order specified above. Additionally, most or all fields use the 'Critical' bit of the Packet Type tag. In the OpenPGP specification, this normally means that if the application does not understand the subpacket type, it SHOULD consider it an error. This specification requires that the application MUST consider it an error.

3.2 PGPticket Field Descriptions

3.2.1 Packet Tag

The Packet Tag field contains four one-octet subfields. The first is

the version number and must be 4 since these are OpenPGP V4 packets. The second is the Packet Type and must be the value for Standalone signature (2). The next octet is the Public Key Algorithm id. The last octet is the hash algorithm id. These represent the hash and public key signature algorithms used to produce the issuer's

signature at the end of the packet. The various id values are listed in [RFC2440].

3.2.2 Hashed Subpacket Length

This field is a two-octet length. It represents the total number of octets in the following five (5) subpackets.

3.2.3 Certificate Validity Information [Part 1]

This packet is the first of two validity packets. Its contents indicate the time the ticket was created. Its packet length field must be 5. Its packet type must be 'signature creation time' with the critical bit on (0x82). The remaining field is a four-octet time field compatible with time fields from the OpenPGP Specification.

3.2.4 Certificate Issuer Information

This packet contains information about the issuer or signer's key. Its packet length field must be 9. Its packet type field is 'issuer key ID' with the critical bit on (90). The last field holds the eight-octet key ID of the issuer.

3.2.5 Certificate Validity Information [Part 2]

This packet is the second of two validity packets. Its packet length field must be 5. Its packet type must be 'signature expiration time' with the critical bit on (0x83). The remaining field is a four-octet time field compatible with time fields from [RFC2440].

3.2.6 Certificate Authorization Information

This packet is a Notation subpacket. It's length is 13 more than the length of the enclosed authorization string. If this packet length is less than or equal to 8383 octets, then a two-octet length field is used. If it is more, than a five octet length field is used.

The subpacket type is 'notation' with the critical bit on (94). The flag field MUST be zero (0) for this version of the spec. The following field is a two-octet name length which must be 4. Next is a two-octet value length that holds the actual length of the authorization string data. The following field is the name of the notation data. It must be 'AUTH'.

The last field is the authorization data string. The authorization string data itself is present for use by application level software. It is not specified or interpreted by PGPticket code.

3.2.7 Certificate Subject Information

This packet is a Notation subpacket. It's length is 13 more than the length of the enclosed subject data. If this packet length is less

than or equal to 8383 octets, then a two-octet length field is used. If it is more, than a five octet length field is used. Note that these length field rules are based on those specified in [RFC2440].

The subpacket type is 'notation' with the critical bit on (94). The flag field MUST be zero (0) for this version of the spec. The following field is a two-octet name length which must be 4. Next is a two-octet value length that holds the actual length of the subject data. The following field is the name of the notation data. It must be 'SUBJ'.

The last field is the subject data. The subject data may hold information on one or more subjects. The ticket grants the authorization from the above notation packet to each subject contained in this notation packet. For each subject mentioned in this packet, the following information MUST be present:

- An eight-octet key id for the subject's key
- one-octet public key algorithm id. This represents the algorithm used when encrypting or signing using the associated key.
- one-octet scalar octet count. This count gives the length of the key's full fingerprint.
- The fingerprint of the subject's key

3.2.8 Unhashed Subpacket Length

This contains a two-octet scalar octet count for any unhashed subpackets that follow. This field MUST be zero (0) for this version of the specification. There must be no unhashed subpackets.

3.2.9 Hash Check Value

This is a two-octet field that holds the high-order 16 bits of the hashed packet data. It is present as an additional integrity check. It may be used as a quick test to reject certain invalid signatures. See [RFC2440] for details on this field.

3.2.10 Signature

This field holds one or more Multi-Precision Integers (MPIs). See [RFC2440] for the format of MPIs. The precise format will depend on the needs of the specific signature algorithm. Refer to [RFC2440].

3.3 RADIX-64 Encoded PGPticket

The below protocol specifies the delivery of tickets using email, bulletin boards, or web servers. This requires ASCII encoded tickets. To convert a PGPticket to an ASCII string, use the RADIX-64 Armor

conversion described in [RFC2440]. The encoded ticket will read:

```
-----BEGIN PGP TICKET-----
Version: Fred's PGPticket 1.0
[Optional headers]

{RADIX-64 encoded PGPticket}
-----END PGP TICKET-----
```

This format matches that of PGP messages and keys given in [RFC2440] with the addition of the new Armor Header/Tail Line pair (BEGIN,END PGP TICKET).

3.4 Ticket Acquisition

It is recommended that each PGPticket be generated and delivered out-of-band. This allows service administrators to apply as many or as few checks as desired to meet the demands of whatever security policies are in place. This method of acquisition is described here. Note that PGPtickets are long term entities. They have a limited but potentially long lifetime and may be used more than once. The only times a PGPticket becomes invalid is either:

- When the expiration date is past.
- When a local administrator revokes a ticket. This is done through some local mechanism not specified by this draft.

3.4.1 Out-of-band

Typically, the person asking for access should simply send their public key (or server location of their public key) and the access they are requesting to the service administrator. The administrator then generates the PGPticket and signs it with the service administration private key. The resulting ticket is returned to the requester by any of the following:

- Private cleartext email
- Posting it to a bulletin board or news group.
- Placing it on a web server and returning the URL to the requester.

Other means may also be used providing they are known and well understood by both parties.

It is not a security risk to expose the contents of the PGPticket to any other party. This is true since anyone trying to use the ticket will be required to encrypt a challenge with the corresponding private key in order to use the ticket.

Note that it is a good idea to have some type of time synchronization between the machine generating the tickets and the server which will

be verifying those tickets. This prevents the generation of tickets that have a shorter than expected lifetime.

3.4.2 Possible Inband Ticket Requests

It is possible to develop some automated methods for requesting, generating and delivering tickets. Specifying such protocols are beyond the scope of this document.

3.5 Ticket Use

Once the ticket is retrieved, the client application may use it to gain access to the service. The client starts the protocol. Here is an overview of the procedure:

- The client sends the ticket to the server along with a request for some service.
- The server checks that the current date is in the range of validity dates in the ticket.
- The server uses its public key to verify the ticket signature field.
- The server verifies that the ticket contains permissions for the access requested.
- The server sends a random challenge string to the client. This is done regardless of whether or not the above checks passed in order to prevent leaking information about the ticket's validity to a potential attacker.
- The client generates and sends a response. The response is created by appending a random nonce to the challenge string. It then generates a signature on a new V4 standalone signature packet with the resulting string in a hashed notation packet.
- The server checks the signature in this packet. If the leading characters of the notation subpacket's data field match the sent challenge, it grants access to the client.

The following sections look at the protocol in more detail.

3.5.1 Ticket Presentation

Tickets may be sent to the service daemon as part of the service request or as part of some additional exchange after the initial request is made.

The method used to deliver the ticket will vary from one application protocol to another. These methods are beyond the scope of this document.

3.5.2 Ticket Verification

The server must perform a number of validity checks:

- The current time must be between the UTC times in the first and second Certificate Validity Information subpackets.
- The issuer's signature must be verified using the issuer's public key.
- The requested access sent with the ticket must be covered by the access in the ticket's AUTH notation subpacket.

Whether or not these conditions are met, the server issues a challenge string to the client. If one or more of the tests failed, the server will later reject the request after recieving the challenge response.

3.5.3 Issue of the Challenge

The challenge string is a series of characters (binary or ASCII), that must be signed with the user's matching private key before access is granted. There are no constraints on this string, however there are security considerations concerning its length.

3.5.4 The Response

The response is composed of another V4 standalone signature packet. The signature in this packet is generated by hashing and encrypting the one hashed notation subpacket. The entire packet looks like the following:

Packet Tag
- One-octet version number (4)
- One-octet signature type [Standalone] (2)
- One-octet public key algorithm id
- One-octet hash algorithm id

Hashed Subpacket Length
- Two-octet scalar octet count for following hashed subpackets

Certificate Authorization Information
- Two-octet or Five-octet subpacket length (N+18)
- One-octet subpacket type
  [Notation, Critical] (94)
- Four-octet flag field (0)
- Two-octet name length (4)
- Two-octet value length (N)
- Four-octet name value ('CHALLENGE')
- N-octet challenge string

Unhashed Subpacket Length
- Two-octet scalar octet count for unhashed subpackets (N+(15 or 18))

```
Certificate Subject Information
 - Two-octet or Five-octet subpacket length (N+13)
 - One-octet subpacket type
       [Notation, critical] (94)
 - Four-octet flag field (0)
 - Two-octet name length (4)
 - Two-octet value length (N)
 - Four-octet name data ('SUBJ')
 - N-octet subject string data.
    - Eight-octet key ID
    - One-octet public key algorithm
    - One-octet scalar octet count (M)
    - M-octet public key fingerprint data
Hash Value Check
 - Two-octet integer holding left 16 bits of signed hash value
Signature Value
 - One or more Multi-Precision Integers
       [Depends on Signature Algorithm]
```

By generating a signature on a hash of a complete notation subpacket, we avoid the attack described above in the security section.

The subject notation sub packet contains information only on the subject attempting to respond to the challenge given by the server. This subpacket SHOULD be in the unhashed subpacket section for efficiency.

3.5.5 Acceptance

Once the ticket is accepted, an informational message is returned to the client. The exact form of this message depends on the application protocol (FTP, Telnet, etc.)

After the client receives this message, the application protocol continues with the standard client request/server response cycle.

3.5.6 Rejection

One of a number of conditions can cause a PGPticket to fail verification. This standard minimally supports the reasons listed here:

- PGPTICKET_ISSUER_SIGNATURE_FAILED_VERIFY - This could occur because the packet was tampered with, or the issuer's key has expired.
- PGPTICKET_TIME_NOT_VALID - The current UTC time does not fall in the range of dates in the Certificate Validity Information subpackets.
- PGPTICKET_RESPONSE_SIGNATURE_FAILED_VERIFY - The signature in

the V4 standalone signature response packet could not be verified. This could mean that the subject's public key was not available or the response packet was corrupted or altered in transit.

- PGPTICKET_CORRUPTED_TICKET - This error should be issued if the hash check bytes do not match the decrypted signature.
- PGPTICKET_NO_SUCH_KEY - The user's public key does not exist.
- PGPTICKET_KEY_NOT_AVAILABLE - The user's public key is known to exist but is stored on a different server that cannot be reached at the current time.
- PGPTICKET_REJECTED - The server rejected the ticket for an unspecified reason. This error may be used to replace the specific error's listed above to prevent leaking ticket validity information to a potential attacker.
- PGPTICKET_UNKNOWN_ERROR - This error occurs if some other problem was noticed when checking the ticket or response.

The precise form in which an error is reported is not given here. That is up to the application protocol implementation. The errors should at least enable a client to distinguish between the above listed causes.

4. Example

Following is an example of the use of the PGPticket message format and protocol.

4.1 Non-anonymous FTP with no remote account

There are cases where you may wish to give someone temporary FTP access to a machine you run. Additionally, you may not trust or run anonymous FTP.

Using an FTP server modified to handle PGPticket access, you can issue a ticket to the person that is good only between a specific range of dates and to access a specific FTP directory or directories.

The AUTH notation subpacket would contain information on which directories the key holder has access to as well as the type of access. Log entries could be created showing the key id of the user and the time they accessed the server. This provides the auditing needed for site security requirements.

Here is an example (SPKI-style) ftp authorization cert done as a PGPticket (OpenPGP V4 Standalone Signature packet).

```
- PGP Ticket  194 Bytes -

- TicketID ---
         ADD4 4F94 BEDD 06B9 17DA 8746 B097 D031
         6DB7 F765 0000 0000 0000 0000 0000 0000

- Issuer ---
 KEYID:  0x9E07E39CD2828413
```

```
   - Subjects (1) ---
    1:  0x62B8442FF762CC18 DSA 514F 939A 16D6 ABF6 A5B3  D161 62B8 442F
   F762 CC18
```

```
   - Validity ---
    STIM:   Fri Mar 13 15:05:58 1998
    ETIM:   Fri Mar 13 16:05:58 1998

   - Assertion ---
    ADATA:  [60] |(tag (pkpfs //afp.vmeng.com/pub/vinnie/file4
                 (read execute))|

      0: 8A00 0000 BD04 0211 0200 8782 B8B3 C9D6   ................
     16: 83B8 B3D7 E690 9E07 E39C D282 8413 9400   ................
     32: 0000 0000 0400 1E53 5542 4A62 B844 2FF7   .......SUBJb.D/.
     48: 62CC 1811 1451 4F93 9A16 D6AB F6A5 B3D1   b....QO.........
     64: 6162 B844 2FF7 62CC 1894 0000 0000 0004   ab.D/.b.........
     80: 003C 4155 5448 2874 6167 2028 706B 7066   .<AUTH(tag.(pkpf
     96: 7320 2F2F 6166 702E 766D 656E 672E 636F   s.//afp.vmeng.co
    112: 6D2F 7075 622F 7669 6E6E 6965 2F66 696C   m/pub/vinnie/fil
    128: 6534 2028 7265 6164 2065 7865 6375 7465   e4.(read.execute
    144: 2929 0000 ADD4 00A0 9695 4EF9 13F2 4761   ))........N...Ga
    160: AC42 F68E CF97 AD6F 584A A24C 009E 32DF   .B.....oXJ.L..2.
    176: C501 484F 9747 D4ED 5EA4 F4DF 44E1 DA9B   ..HO.G..^...D...
    192: 169C
```

Here is a human readable description of the fields in the above PGPticket:

Field	Size	Hex-value	Desccription
Packet Header			
	1	8A	Ptag 10 0010 10 | | '-- 2 = 4 Byte length | '------- 2 = Signature Packet '----------- old packet format
	4	000000BD	Total packet Length
V4 Signature packet			
	1	04	version number: (4).
	1	02	Signature Types: Standalone sig
	1	11	public key algorithm (DSA)
	1	02	hash algorithm (SHA1)
Hashed SubPacket Data			
	2	0087	Hashed subpacket len
	1	82	2 = signature creation time (VALIDITY)
	4	B8B3C9D6	Signature creation time
	1	83	3 = signature expiration time

```
                4       B8B3D7E6      Signature expiration time

                1       90            16 = issuer key ID ( ISSUER )
                8       9E07E39C
                        D2828413      Key ID of issuer
```

```
   Notation SubPacket
   (SUBJECTS)
               1       94            20 = notation data
               4       00000000      (4 octets of flags)
               2       0004          - name length
               2       001E          - value length, (typ 30)
               4       5355424A      - name data  'SUBJ'
               8       62B8442F
                       F762CC18      - Key ID of subject
               1       11            - public key algorithm.
               1       14            - Key fingerprint length
               N       514F939A      - Key Fingerprint
                       16D6ABF6
                       A5B3D161
                       62B8442F
                       F762CC18

   Notation SubPacket
   (AUTHORIZATION)
               1       94            20 = notation data
               4       00000000      (4 octets of flags)
               2       0004          - name length
               2       003C          - value length,
               4       41555448      - name data 'AUTH'
               N       28746167      - value data
                       2028706B       '(tag
                       70667320         (pkpfs //afp.vmeng.com/pub...'
                       2F2F6166
                       702E766D
                       656E672E
                       636F6D2F
                       7075622F
                       76696E6E
                       69652F66
                       696C6534
                       20287265
                       61642065
                       78656375
                       74652929

  UnHashed SubPacket Data
               2       0000          UnHashed subpacket len (not used)
   Signature Data
               2       ADD4          - Two-octet field
                                        (left 16 bits of signed hash)
```

```
               N       00A09695      - One or more multi-precision
                       4EF913F2         integers comprising
                       4761AC42         the signature.
                       F68ECF97
                       AD6F584A
                       A24C009E
                       32DFC501
                       484F9747
                       D4ED5EA4
                       F4DF44E1
                       DA9B169C
```

The challenge and response sequence would look as follows:

```
hosta 2> ftp afp.vmeng.com
Connected to afp.vmeng.com
220 afp.vmeng.com FTP server (Version 4.4  Thu Mar 19 15:54:18 EST
    1998) ready.
Name (afp.vmeng.com:vinnie):
Ticket:
         ADD4 4F94 BEDD 06B9 17DA 8746 B097 D031
         6DB7 F765 0000 0000 0000 0000 0000 0000

         AUTH(tag.(pkpfs.//afp.vmeng.com/pub/vinnie/file4.
           (read.execute))
         [Y/N]?  Y
Passphrase for Private key "Vinnie Moscaritolo <vinnie@apple.com>
           (DSS/...)":
230 User vinnie logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

At the application protocol level, the exchange would look like the following:

```
220 afp.vmeng.com FTP server (Version 4.4 Thu Mar 19 15:54:18 EST
    1998) ready.
USER vinnie
333 Username ok, Need PGP Ticket
TICK ADD4 4F94 BEDD 06B9 17DA 8746 B097 D031 6DB7 F765
334 Challenge -----BEGIN PGP SIGNATURE-----;Version: PGP Personal
    Privacy 6.0.2;;iQEVAwUBNwThCisFUqBqnwv7A...;=xhSh;-----END PGP
    SIGNATURE-----
CRSP -----BEGIN PGP SIGNATURE-----;Version: PGP Personal
    Privacy 6.0.2;;iQA+AwUBNwUU+vUeIl4piG8AEQIXF...;=iXuy;-----END PGP
    SIGNATURE-----
```

```
   230 User vinnie logged in.
```

5. References

[RFC2440] "OpenPGP Message Format", Callas J., Donnerhacke L., Finney H., Thayer R., 1998/11/11 (65pp)

6. Authors' Addresses

Vinnie Moscaritolo
Worldwide Developer Technical Support
N&C/Hardware
Apple Computer, Inc.
3 Infinite Loop, MS: 303-2T
Cupertino, CA 95014

vinnie@apple.com
+1 408/974-5560

Antonino N. Mione
Manager of TD Network Services
Rutgers University Computing Services
Hill Center for the Mathematical Sciences
110 Frelinghuysen Rd.
Piscataway, NJ 08854

mione@noc.rutgers.edu
+1 732/445-0650