Using Public Key Technology—Issues of Binding and Protection

James M. Galvin <galvin@tis.com>
Sandra L. Murphy <murphy@tis.com>

Abstract

Public key technology is ideal for supporting digital signatures. Given a data object, a signature object, and a public key, it can be determined if the three elements are related. Specifically, at the conclusion of the signature verification calculation it will be known if the private key corresponding to the given public key was used to create the signature object of the data object. However, the calculation only answers half the question. What is missing is the identity of the user claiming to control access to the private key component that was used to create the signature. Further, believing the identity requires knowing how well the private key component is protected from disclosure. These issues are examined.

1 Introduction

Public key technology is well suited to providing digital signatures. Often, the signature for a data object is created by:

1. hashing the object to be signed with a one-way hash function and
2. encrypting (signing) the hash value with the private key component of a public/private key pair.

The hash value is encrypted instead of the data itself because the encryption function is typically very slow compared to the time it takes to complete the hash of the data. The object created by these two steps is called the signature and is bound to the data object according to an application specific mechanism.

In conjunction with the public key component of the correct public/private key pair, the signature of a data object can be verified by:

1. decrypting the signature object with the public key component to expose the original hash value,
2. re-computing a hash value over the data object, and
3. comparing the exposed hash value to the re-computed hash value.

If the two values are equal the signature is often considered valid. Specifically, at the conclusion of the calculation it will be known if the private key corresponding to the given public key was used to create the signature object of the data object. From a mechanical (and programmatic) point of view, the signature is valid. However, from a user's point of view, the calculation only answers half the question.

When a user creates a digital signature, the user is asserting *ownership* of the data object. The ownership would be defined according to the context in which the signature was created. The digital signature is intended to be a mechanism with which other users can verify the ownership of the data.

In the paper world, a signature has two important properties: it identifies the signatory and it is unforgeable. The signature identifies the signatory insofar as the mark used represents the signatory's name. The signature is unforgeable insofar as the mark used has a very low probability of being accurately recreated by any person other than the signatory. The ownership asserted by a digital signature must satisfy these same two properties.

The mechanical validation described above does not satisfy either of these two properties. The problem is, it does not identify the owner of the public/private key pair, i.e., the identity of the user claiming to control access to the private key component that was used to create the digital signature. A signature is of little value when it is not known who created it since the significance of the data object can not be quantified without knowledge of its origin. Also, if an owner does not protect a private key from disclosure, then there is no way of knowing if it was really the owner who created the signature.

When the user knows the identity of the owner of a public/private key pair and how well that owner protects the private key from disclosure, the user knows the probability that the owner's identity is the signatory and that it was the signatory who created the signature.

Within a small community of users, it is reasonable to assume that all users know the public keys of all other users and that all users know the degree to which all other users protect their private keys from disclosure. Individual users may construct their own quasi-small communities by contacting other users and exchanging public keys and other relevant information via various ad hoc and semi-formal mechanisms. This model has served and continues to serve a growing community of users concerned about the security of their communications. However, it falls far short of being a solution for the global internet.

The global internet needs a mechanism with which users can acquire the public key of any other user and be able to quantify two things:

1. the assurance that the identity bound to the public key correctly identifies its owner and
2. the degree to which the owner protects the corresponding private key from disclosure.

Quantifying these values is essential to developing application programs that can process data without explicit and direct user interaction, e.g., application programs that can process data in a batch mode. With this information, interactive user programs will be better able to support their users.

Users who do not determine whether the public key used to verify a signature is the correct public key are no better off than if the data object was not digitally signed. This is because there is nothing to prevent an interloper from substituting a public key for which the interloper controls the private key. Such an interloper could assert the identity of any owner and claim to own whatever is believed to be interesting.

To date, there has been very little examination of these subtle but important issues. The OSI Directory community defined certificates in X.509 [1] but provided little guidance on how to use them effectively and how to protect the corresponding private key. In the Internet, Privacy Enhanced Mail (PEM) [6] restricted the general rules of X.509 and specified a strict hierarchical organization for certificates, with enough rules to ensure that signatures and identities could be evaluated programmatically [2]. PEM also included a discussion of some of the issues related to protecting the corresponding private keys. Pretty Good Privacy (PGP) [3] includes a mechanism that lends itself to individual users and conveniently supports the bottom-up construction of public key databases. It also includes a discussion of some of the issues related to protecting the corresponding private keys. The MIME Object Security Services (MOSS) [4] protocol supports all of the above suggestions but does not add to the discussion of these issues.

Each of these applications provides a similar mechanism for mechanically validating digital signatures, the first half of verifying a digital signature. However, each provides different guidance on the issue of validating the binding between the owner's name and the public key and none have any guidance for the recipients of signed data objects about evaluating the protection accorded private keys by their owners.[1]

Succeeding sections will examine the issues associated with binding identities to public keys and protecting the corresponding private keys. First, the next section will present what is available for use today to represent the identity of the owner, to protect the private key of a public/private key pair, and to bind public keys to identities.

[1] There is another binding that is important but not included in this discussion: the binding between the physical user and the computing environment. Space does not permit a complete discussion of these issues.

2 Identities, Protection, and Bindings

In order to examine binding and protection issues, users must understand the mechanisms used to identify owners of public keys, to protect the private components of public/private key pairs, and to bind the identities to the public keys.

2.1 Signatory Identities

Traditionally, a person's identity is indicated by their name as given to them by their parents. In small communities of users, using this name as a user's identity and binding it to a public key would probably serve to uniquely identify the public key's owner. However, as the community of users increases in size, the probability that this identity will be unique decreases and, as a result, additional information is required to be part of the identity to guarantee its uniqueness. For a user creating a digital signature, the signature object must contain at least the user's identity, which must provide enough information such that the public key of the user can be retrieved by those who need it.[2]

For a user validating a digital signature, however, the signatory's identity is only required if the user does not have the public key immediately available, since, presumably, the identity contains enough information that the public key can be retrieved. Once the user has retrieved the public key, if the user expects to regularly received signed objects from the signatory, the user will store the public key and its identity in a local database. If the identity is lengthy or otherwise cumbersome to work with, a user may also assign a local handle, or alias, to the database entry, to facilitate using the public key and quickly recognizing the signatory.

These two perspectives demonstrate that an identity serves two purposes. For a signatory, the identity must provide enough information to users validating signatures to retrieve the public key if they do not have it; the identity must be globally unique and unambiguous. For a user, the identity must provide enough information that the signatory can be easily recognized; the identity must be unique and unambiguous within the user's database of public keys. There are three identity marks in common use today.

[2] Current technology does not typically distribute the public key with the signature. There are two principal reasons for this: the size of the identity is often significantly smaller than the size of the public key and the identity is useful to human users while the public key appears to be little more than a string of random characters.

In the case of the OSI Directory, an identity is represented by a distinguished name, an object that typically conveys quite a bit of information in addition to what would ordinarily be considered a user's name, e.g., perhaps an employer's name or a home address. Distinguished names are by definition globally unique and unambiguous. Their adoption in the internet has been limited, in part due to the fact that they are lengthy and cumbersome. Most of the usability issues could be addressed by more intelligent user interfaces but none have been forthcoming.

In the case of PGP, an identity is represented by an arbitrary string chosen by the signatory. This has the advantage of supporting short and easy to use marks that address the principal disadvantage of distinguished names. However, it has the disadvantages of a lack of structure and no guarantee of uniqueness. Thus, arbitrary strings are only suitable for use in small communities or closed communities with a centrally managed database of identities.

In addition to supporting the two identity marks described above, the MOSS protocol introduces a third mark. The internet community has a great deal of experience with email addresses. Users are often recognized by other users by their email addresses. Email addresses have the advantages of being short, easy to remember and use; having structure suitable for application processing without direct user interaction; and being globally unique and unambiguous. The disadvantage is that there is still an installed base of users who do not have RFC822 [5] compliant mail systems or gateways that support acceptable communications.

From the perspective of a signatory, the distinguished name or the email address is the most suitable mark. Both are structured to provide sufficient information that users can retrieve public keys if they do not already have them. The email address has the additional advantage of being short and easy to use and remember.

From the perspective of a user validating a signature, the arbitrary string or the email address is the most suitable mark. Both are short and easy to use and remember. The email address has the additional advantage of being structured and suitable for processing by applications without direct interaction with the user.

2.2 Private Key Protection

Protecting the private component of the public/private key pair from disclosure is essential to the correct operation of the security services. If the private key is ever disclosed any digital signature created with it is invalid.

Ideally, the private key should be created, stored, and used within a closed environment, e.g., in a trusted system or in a PCMCIA card. In both cases, the functionality being enforced is that the private key is only accessible to authorized users. In both cases, a physical user performs an authentication step to ensure that only the correct user has access to the private key. The trusted system has the additional advantage that the software used is not vulnerable Trojan horses or substitution. Unfortunately, in both cases, specialized hardware and software is required for which, for large numbers of users, the cost may be prohibitive. In any case, there are two alternatives to protecting private keys that are in common use today. Both involve storing the private key in a file: in one case off-line and in one case on-line.

In the off-line case, the private key is placed in a file on a removable media, e.g., a diskette. This has the advantage of a low initial cost since no special hardware is needed; diskette drives are standard on most hardware configurations. It has the disadvantage of allowing the private key to be disclosed if the diskette is ever lost. However, this disadvantage can be addressed by including support for the encryption of the private key as provided in the on-line case.

Many users do not have personal workstations that are used only by themselves. Invariably, the hardware is shared either by design (e.g., in the case of a multi-user computing environment) or by necessity (e.g., borrowing a colleague's single-user environment because it includes applications not available in a user's own environment). Also, a user may not have easy access to the diskette drive, particularly in a multi-user environment.

In this situation, the only option available to a user may be to store the private key in an on-line file. Obviously, the contents of this file must be protected from disclosure. All computing environments provide a mechanism with which at least one user has access to the contents of all the files. To protect the contents of a file from this user, a signatory must encrypt the contents with a password known only to the signatory. In this way, if anyone acquires the contents of the file, they will not be able to read the private key.[3]

2.3 Public Key and Identity Bindings

Given a public key and a purported identity of its owner, a user needs to verify that the identity identifies the owner of the public key. By doing so,

[3] However, in a multi-user environment, encrypting the private key file may be insufficient to guarantee non-disclosure of the private key. The applications that are used to access the private key must also be validated, since they necessarily capture the password a user enters to decrypt the private key. If the applications can be replaced by an interloper that not only behaves as expected but also logs the captured passwords, then a signatory is vulnerable to the disclosure of the private key. Trusted Systems do not have this vulnerability.

the user will know, when a signature is validated, who is the signatory of the data object.

In the simplest case, an implicit binding is created when a user receives a public key directly from a signatory. The user takes the public key as received and associates the signatory's identity with it, e.g., by storing the public key with the identity in a file on disk. It is essential that the file containing the public key be protected from modification since otherwise an interloper could substitute a public key for which the interloper has control of the private key. This would allow the interloper to create signatures that would appear to have been created by the signatory.

The threat of modification is significantly reduced if a user creates an explicit binding between the public key and identity. An explicit binding establishes a cryptographic relationship between the public key and the identity, similar to the cryptographic relationship established between a data object and an identity by a digital signature. A user could create a data object composed of a public key and the identity of its owner, sign the data object, and store the data with the signature. As long as the user's private key is protected and the binding is verified each time the public key is used, the user is assured that the identity purported when a signature validates is the identity of the signatory.

There are two common mechanisms in use today for providing explicit bindings between identities and public keys. The OSI Directory defines a certificate, which is an embodiment of a binding between an identity (a distinguished name) and a public key. A certificate is composed of several components intended to facilitate the management of its binding. Two of the components are the signature over the contents of the certificate and the identity of the signatory, called an issuer. A user validates a certificate by obtaining the public key of its issuer and verifying its signature. If the signature in the certificate is valid then its public key is believed to be owned by the user identified by its distinguished name. (See also Section 3.) According to the Directory, issuers are well known, easily identifiable users trusted to create valid certificates. PEM uses certificates as defined by the Directory but adds specific requirements to users acting in the role of an issuer.

The PGP and MOSS protocols both define alternative embodiments of bindings.[4] A user exchanges public keys with a second user and then each creates a signature binding the other's public key with the other's identity. If a third user acquires the first user's embodiment and the third user has also exchanged public keys with the second user, the third user can validate the first user's public key without direct interaction. In both protocols a user may solicit signatures from many other users, thus increasing the probability that new users will be able to validate their public key without direct interaction. PGP differs from MOSS in that it allows a user to include all of the signatures obtained in a single embodiment.

[4] It should be noted that MOSS also supports the use of certificates as defined by the Directory.

3 Validating Public Keys

Mechanical validation, as described above, establishes a relationship between a data object, a signature object, and a public key. Specifically, the relationship is that the private key corresponding to the given public key was used to create the signature object of the data object. This validation is ideally suited to be performed automatically, i.e., by an application without direct interaction with a user. It is the validation of the identity of the owner of the public key that may not be conveniently performed automatically.

In the paper world, a user evaluates the validity of a signature by recognizing the mark used and associating a signatory with it. If the mark is recognized, the user completes the validation by evaluating whether the signature is valid in the context in which it was used.

If the mark is not recognized, the user must find the signatory or someone the user trusts to establish the validity of the signature. If the mark is legible, the user can read the name and may be able to find the signatory based on it. Alternatively, a colleague may recognize the mark and be able to complete the validation for the user. The latter option may be iterated: the colleague may know another colleague who may recognize the mark and be able to complete the validation. The iteration may continue as many times as are necessary or until the user decides the signature should be considered invalid.

The interative process creates a chain of trusted users between a signatory and a user. The user must evaluate the length of the chain in the context in which the signature was created. Presumably, the user trusted the first colleague's evaluation of the validity of the signature. Depending on the context, this trust may or may not extend to other colleagues trusted by this first colleague.

In the digital world, given that a mechanical validation has completed successfully, a user must now determine if the public key used in the mechanical validation was the correct public key. The public key used corresponds to the mark used by a signatory. Validating that the public key used was the correct public key corresponds to associating a signatory with it, i.e., the identity of the user who has control of the corresponding private key and evaluating the degree to which the private key is protected from disclosure.

In the simplest case, a user stores the public keys of signatories in a file with their identities, thus binding the two values together. The binding is assumed to be valid as long as the file is protected from modification. If the user stores the file in a single user environment that is inaccessible to all other users, no additional validation is necessary. This method of validating public keys corresponds to the paper world case of a user immediately recognizing a mark and associating a signatory with it. The validation process is complete after the user evaluates whether the signature is valid in the context in which it was used. It is suitable for users who:

- work in a single user environment,
- maintain a small number of public keys of other users, i.e., communicate with a small number of signatories,
- do not mind direct interaction with other users in order to obtain their public keys and identities.

Users who do not meet one or more of the requirements above must have mechanisms available to them to create and manage explicit embodiments of bindings between public key and identity pairs. In the simplest case, a user creates each embodiment locally and stores it with its respective pair. The validation of the public key requires the additional step of validating the embodiment. Including this additional step, this method of validating public keys also corresponds to the paper world case of a user immediately recognizing a mark and associating a signatory with it.

The digital world supports the case where a user must ask a colleague to complete the validation by allowing other users, called issuers, to create embodiments that are trusted. A user stores these embodiments with their respective public key and identity pairs. The process by which these embodiments are used to validate public keys is as follows.

1. The embodiment of the public key to be validated is retrieved.
2. The public key of the issuer who created the embodiment is retrieved.
3. The issuer's public key is used to mechanically validate the public key from Step 1.
4. The issuer's public key must now be validated by starting again at Step 1 with it.

The sequence of steps is iterated until the issuer is the user who began the validation process, i.e., a chain of trusted users is created between the signatory and the user, or the process can not continue, e.g., an issuer's public key may not be available. The most popular implementation of each of PGP, PEM, and MOSS supports mechanisms for constructing, evaluating, and displaying a chain of embodiments. The validation process is completed by the user evaluating the significance of the length of the chain in addition to whether the signature is valid in the context in which it was used.

The significance of the length of the chain of trusted users depends entirely on context. As in the paper world, a user may choose to trust a particular issuer, but whether that trust extends to issuers trusted by the first issuer is context dependent.

PEM is unique because it includes a specification of rules for its certificate-based key management infrastructure that provide a mechanism with which a user can decide if an issuer in the infrastructure is trusted. The infrastructure makes use of a single, global, hierarchically structured graph of all embodiments with a single root. When validating public keys the constructed chain must form a path from the signatory to the root of the infrastructure. The root of the infrastructure is trusted by all users within the infrastructure. The root creates an embodiment for each issuer who is required to establish and publish a policy by which all issuers in their sub-tree must abide when issuing embodiments. A user decides if an issuer is trusted by reviewing the policy of the sub-tree in which the issuer appears. As a result, the length of the chain is of little consequence to users of certificate-based key management.[5]

PGP implementations typically include a mechanism with which a user may specify the degree to which an issuer is trusted. The default, of course, is no trust. This requires a user to explicitly evaluate each issuer of interest, as opposed to evaluating a much smaller set of policies as is possible in the PEM case.

The MOSS protocol specification explicitly declares public key validation a separate and independent topic. As a result, a MOSS implementation could easily support both the PEM and PGP paradigms.

4 Protecting Private Keys

There is no explicit mechanism that a user can use to judge how well the private key component of a signatory is protected from disclosure. The mechanisms described above are implementation specific. Further, there is no way to be sure that a signatory makes use of the features available for protecting private keys. The significance of this must not be underestimated. It must be re-iterated that a digital signature is only valid if the private key used to create it is protected from disclosure and only used by the user to which it belongs. As a result, associating less trust in lengthy validation chains, especially chains with more than 3 links: the signatory, a trusted user, and the user performing the validation, is inevitable. A user must choose trusted users very carefully, expecting them to be as careful as the user when creating embodiments for others.

[5] A long chain increases the number of issuers who must protect their private keys from disclosure, thus increasing the probability of at least one being compromised.

5 Conclusions

Although mechanical validation is necessary and straightforward, it does not and must not be assumed to validate a digital signature. An essential component of the validation process is making sure the correct public key is used to validate the signature. Establishing that the correct public key was used to validate the signature ensures that the user has identified the signatory.

Users and vendors must consider how to address this issue. Initially, users will not be concerned, since the number of public keys that must be managed will be small. However, initial complacence on the part of users ultimately results in non-compliance on the part of vendors. As digital signatures become an integral part of the workplace, both users and vendors have a responsibility to do it right.

In the future, the legal significance of digital signatures will be determined. The informal acceptance and usage of digital signatures now will shape it. More research and usage of digital signatures are needed to fully understand the issues.

References

[1] The Directory—Authentication Framework. X.509, 1993. Developed in collaboration and technically aligned with ISO 9594-8.

[2] Steve Kent. Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. RFC1422, BBN Communications, February 1993.

[3] Simson Garfinkel. PGP: Pretty Good Privacy. O'Reilly & Associates, Inc. ISBN: 1-56592-098-8.

[4] Steve Crocker, Ned Freed, Jim Galvin, Sandy Murphy. MIME Object Security Services. Work in progress.

[5] David Crocker. Standard for the Format of ARPA Internet Text Messages. RFC822, University of Delaware, August 1982.

[6] John Linn. Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures. RFC1421, IETF PEM Working Group, February 1993.

James M. Galvin is a Senior Computer Scientist at Trusted Information Systems, Glenwood, MD, where his responsibilities emphasize network security. Dr. Galvin has been active in the development of both the PEM and MOSS protocols and has been responsible for reference implementations of both. He is active in the Security Area of the IETF serving on the Security Area Directorate, co-authoring the SNMP Security Protocols, and participating in the security review of Internet protocols. He is currently a project leader for TIS's Internet Infrastructure Protection task, with responsibility for cryptographic protection of the domain name system. Dr. Galvin received his Ph.D. from the University of Delaware, where he concentrated on communications security.

Sandra L. Murphy is a Senior Computer Scientist at Trusted Information Systems, Glenwood, MD, where she works in protocol and network security analysis. She is currently a project leader for TIS's Internet Infrastructure Protection task, with responsibility for cryptographic protection of routing. Prior to her joining TIS, Dr. Murphy received a Ph.D. in 1990 from the University of Maryland for work in network protocol specification and verification, and worked as a computer scientist at the Defense Communications Agency (1973-1982).