Public Key Cryptosystems, Certificates, and Certification Authorities.

Danny Smith

Australian Computer Emergency Response Team
c/- Prentice Centre
The University of Queensland
Qld. 4072. Australia

auscert@auscert.org.au
+61 7 365 4417 (AUSCERT Hotline)

10-October-1994

Abstract

This seminar and report discusses public key cryptosystems and some of the ways in which they can be used to verify the identity of a user. Problems using this technique are examined, along with some solutions to those problems.

These solutions make use of certificates and certification authorities. The concept of a trusted path is introduced, and one possible technique which may remove the requirement for an ultimate trusted source is examined.

Introduction

The purpose of this seminar is to examine the problem of using public key cryptosystems for private communication. To do so requires the ability to manage the keys in an efficient and trusted method. Public key cryptosystems provide a mechanism for authenticating users without the requirement for a user to reveal any secrets. Using this method requires the ability to determine the public portion of the authentication information with a high degree of trust.

Certificates are introduced as a mechanism for providing that level of trust. Certification Authorities issue certificates which guarantee the information they contain. However, to verify the certificate becomes a recursive problem, forming a certification path.

To terminate the certification path, a single point of trust must be established. Finally, a technique that can decrease the requirement for a single point of trust is examined.

Many of the terms and concepts found in this seminar can be obtained from [ISO7498] and [X.509].

Introduction

PKCS

Determining Public Keys

Certificates

Certification Authorities

Certification Paths

Mutual (Dis)Trust

What is a PKCS?

PKCS stands for "public key cryptosystem". These cryptosystems, also known as asymmetric cryptosystems use a pair of keys, one secret and one public, rather than a single key as in conventional cryptographic systems. With a single key system, the key used to encrypt information by the originator of a message is the same used to decrypt the message by a legitimate recipient. This means that both parties must possess knowledge of the key or communication to take place.

In public key cryptosystems, the keys come in pairs, one key of which is used for encrypting the message by the originator, and the other used to decrypt the encrypted message. Each key pair is associated with a particular user X. The dual key system of a PKCS allows a number of different techniques to be used when encrypting information, particularly when it comes to key management.

The most common key management technique is to make one of the keys known as the public key (X_p) widely known to anyone who wishes to use, and keep its corresponding pair as a secret key (X_s) known only to user X. It must be computationally infeasible to derive the secret key from knowledge of the public key. Using this technique, information may be encrypted using either key, resulting in differing services:

If a user (Y) encrypts information using X_p, then only the user possessing the corresponding secret key X_s can decrypt that information. This allows any user to communicate a piece of information which only X can find out, thus allowing that information to be kept secret. This is represented notationally as $D = X_s[\ X_p[\ D]]$. By extension, this allows two users to communicate in secret by using each other's public key to encrypt the information. Such an exchange can, as well as transferring secret information between the parties, serve to verify their identities. Specifically, a user is identified by the possession of their secret key

It is the property of some PKCS that the steps of encrypting and decrypting may be reversed. If user X encrypts the information using their secret key (X_s), then anyone in possession of the corresponding public key (X_p) can decrypt the information. However, only user X should possess the secret key, and therefore the encrypted information could only have come from user X. The provides data origin and data integrity authentication. This is represented notationally as $D = X_p[\ X_s[\ D]]$.

PKCS

public key cryptosystem

- public key
- secret key

DES versus RSA

Can't derive X_S from X_p

Confidentiality
Y uses X_p - $D = X_s[X_p[D]]$

Data Origin, Data Integrity
X uses X_s - $D = X_p[X_s[D]]$

Obtaining a user's public key

Given that PKCS keys are in pairs, and possession of one of those keys is used to verify the identity of a particular user, then we need to be careful of how we determine the public key for that user. A rogue user may generate a public-secret key pair, and then pretend to be another user, by giving you the public key portion of that pair, stating that it belongs to the user they are attempting to masquerade as.

What is required is a trusted way of determining the public key for any given user. If this trust breaks down, then the whole system of using a PKCS is unworkable.

Determining X_p

$D = X_p[\ X_s[\ D]]$

X_p/X_s pairs

identify X by possession of <u>An</u> X_s

use corresponding X_p

- need to determine X_p with trust

Certification Authorities

One method for determining the public key of a user with a high degree of authenticity is to trust some authority to provide that service. This service is generally provided by a Certification Authority (CA). Public keys may be detailed by the CA each time someone requests it, but is more usual for the CA to use the public key algorithm to certify the public key of a user by producing a certificate (discussed in the next section).

A Certification Authority produces the certificate of a user by digitally signing a collection of information, which includes the user's name and public key. Digital signatures involve appending an encrypted summary of the information to the actual information. The summary is produced by a one-way hash function (h), and the encryption is performed using the secret key of the signer (X_s). This is represented notationally as X{ Information} = Information, X_s[h(Information)]

The encryption using the secret key ensures that the signature cannot be forged. The one-way nature of the hash function ensures that false information, generated so as to have the same hash result (and hence signature), cannot be substituted. The recipient of the signed information verifies the signature by:

- applying the one-way hash function to the information;
- comparing the result with that obtained by decrypting the signature using the public key of the signer (X_p). Note that this requires trusted access to the signer's public key.

Thus the problem becomes recursive. The obtain a trusted copy of a user's public key we can obtain a certificate which was issued by a Certification Authority. To verify that certificate requires trusted access to the public key of the Certification Authority. This may be obtained for example, through another certificate issued by yet another CA. Eventually, we form a Certification Path (discussed in a later section).

Certification Authorities

trusted authority to "certify" Xp

Certification Authorities

- detail Xp on each request
- use PKCS to produce certificate

Digital Signature

X{ I} = I, Xs[h(I)]

need CAp

- recursive problem

Certificates

A Certificate is a collection of information which has been signed by a Certification Authority. In the X.509 standard, certificates are used to obtain a trusted copy of a user's public key, provided the CA issuing the certificate is trusted.

Certificates rely on each user possessing a unique name. The allocation of names is the responsibility of the Naming Authorities. Each user shall therefore trust the Naming Authorities not to issue duplicate names. If naming is performed in some hierarchical fashion, the delegation of name spaces can be easily achieved, and one Naming Authority does not require cooperation with other Naming Authorities to ensure that a given name is unique.

A certificate has the following properties:

- any user with access to the public key of the CA can recover the public key (and any other information) which was certified;
- no party other than the CA can modify the certificate without this being detected (certificates are unforgeable).

As certificates are unforgeable, they can be freely published (for example, in public databases or public communications) without the need to make special efforts to protect them.

In X.509 [X.509], the following certificate structure has been defined (using ASN.1 [X.208]):

```
Certificate ::= SIGNED SEQUENCE {
      version                    [0] Version DEFAULT v1,
      serialNumber               CertificateSerialNumber,
      signature                  AlgorithmIdentifier,
      issuer                     Name,
      validity                   Validity,
      subject                    Name,
      subjectPublicKeyInfo       SubjectPublicKeyInfo,
      issuerUniqueIdentifier     [1] IMPLICIT BIT STRING OPTIONAL,
                                         -- if present, version must be v2
      subjectUniqueIdentifier    [2] IMPLICIT BIT STRING OPTIONAL,
                                         -- if present, version must be v2 }

Version :== INTEGER { v1(0), v2(1)}

CertificateSerialNumber ::= INTEGER

Validity ::=
      SEQUENCE {
                 notBefore       UTCTime,
                 notAfter        UTCTime }

SubjectPublicKeyInfo ::=
      SEQUENCE {
                 algorithm       AlgorithmIdentifier,
                 subjectKey      BIT STRING }

AlgorithmIdentifier ::=
      SEQUENCE {
                 algorithm       OBJECT IDENTIFIER,
                 parameters      ANY DEFINED BY algorithm OPTIONAL }
```

Certificates

info signed by CA
- (letter from HoD)

require unique name
- naming authorities
- hierarchical

unforgeable

freely published

CA«A» vs CA{A}

CA«A» =
$CA\{V,SN,AI,CA,UCA,A,UA,Ap,T^A\}$

From this defined certificate structure, we can see the following:

- Certificates are numbered. This allows for a valid certificate to be identified, and possibly revoked before its expiration;
- Algorithms are identified. This allows a number of cryptographic algorithms to be used, and the specific one used to signed this certificate to be identified;
- Valid dates are included. This allows a certificate to have a start and end date and time, during which the certificate is valid. Certificates outside of those dates should not be trusted as providing reliable or valid information;
- Unique Identifiers are included. This allows a Naming Authority to reuse a name for either a principal, or a CA. When a name is reused, the unique identifier is adjusted, thus the combination of the name and unique identifier defines a particular instantiation of that name. This prevents confusion or incorrect results should a name or the name of a CA be reissued.

A certificate is notationally defined as:
CA«A» = CA{ V, SN, AI, CA, UCA, A, UA, A_p, T^A}

The notation CA«A» denotes the certificate containing the public key for A, which has been issued by CA(A) (the Certification Authority for A). Note that user A may obtain certificates from more than one CA (thus may possess CA^1«A», CA^2«A», and so on). The notation CA{ Information} denotes the digital signing of Information.

Certificates of the above form are defined by Recommendation [X.509], and are expected to be stored within an X.500 Directory [X.501]. However, there is no requirement that they must be stored in this manner. They can be stored using any facility, provided that they can be access. Another example of how this may be done can be found in [SPX].

Certificates

version
serial number
signature
issuer
validity
subject (owner)
subjectPublicKeyInfo
IssuerUniqueId
subjectUniqueId

Validity

- **not before**
- **not after**

Certification Path

As detailed previously, the problem we are trying to solve is to determine a user's public key, with a high degree of trust. The mechanism outlined above is to include the public key and the user's name in a certificate, issued by the CA for that user, and digitally signed by the CA thus rendering the certificate unforgeable. Digital signature techniques make use of a PKCS.

To verify the contents of the certificate requires access to the CA's (the certificate signer's) public key. This must also be obtained in a trusted manner so that we can verify any certificates issued by that CA. Failure in the ability to reliably obtain the public key of the CA means that any public/secret key pair may have been used to signed what appears to be a valid certificate. That certificate may contain an invalid public key for user X, and hence a rogue user may masquerade as user X through the use of an invalid public/secret key.

Therefore, we must trust the CA to issue valid certificates, and be able to trust the source of the public key for that CA. One method of obtaining a trusted copy of the public key for a CA is to have it issued in a certificate by another CA. This can be represented notationally as CA^2«CA^1», CA^1«X». This is the way that certification paths are formed. Ultimately, we must reach a single trusted point that is well known to issue valid certificates, and the public key can be trusted.

For two users to communicate with each other, the public key for each user must be made available. This may involve a certification path for each user until the other user is satisfied that they have reached a trusted point. Note that the certification path from user X to user Y may be totally different to the certification path from user Y to user X. For example:

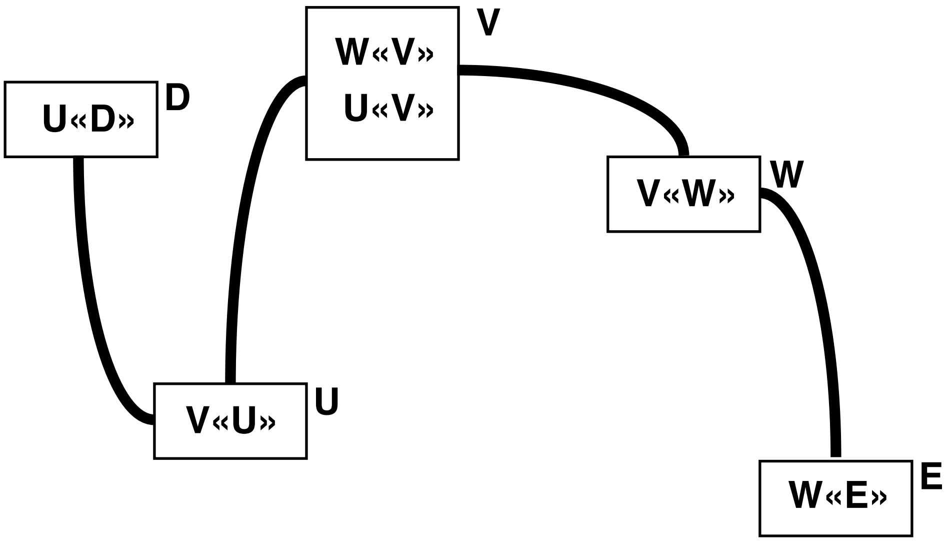
Z«A», A«B», B«C», C«X»
Z«D», D«E», E«Y»

Since the certificates are digitally signed and cannot be forged, they can be stored anywhere that is convenient to their use, without the requirement for special protection. A user may store two types of certificates: forward certificates of X that are generated by other CAs, and reverse certificates that are generated by X (as a CA) which are the certified public keys of other CAs. It is through the use of these certificates that certification paths are formed.

In the example diagram, we assume that V is a CA with whom CAs U and W have at some previous time exchanged public keys in a trusted way. As a result, certificates U«V», V«U», W«V», and V«W» have been generated and stored at the nominated points.

User D must find a certification path to E. Various strategies could be used. One such strategy would be to regard the users and CAs as nodes, and the certificates as arcs in a directed graph. In these terms, D has to perform a search in the graph to find a path from U to E, one such being U«V», V«W», W«E». When this path has been discovered, the reverse path W«V», V«U», U«D» can also be constructed.

Certification Path



rogue CAs

ultimate point of trust
- known to issue good certificates

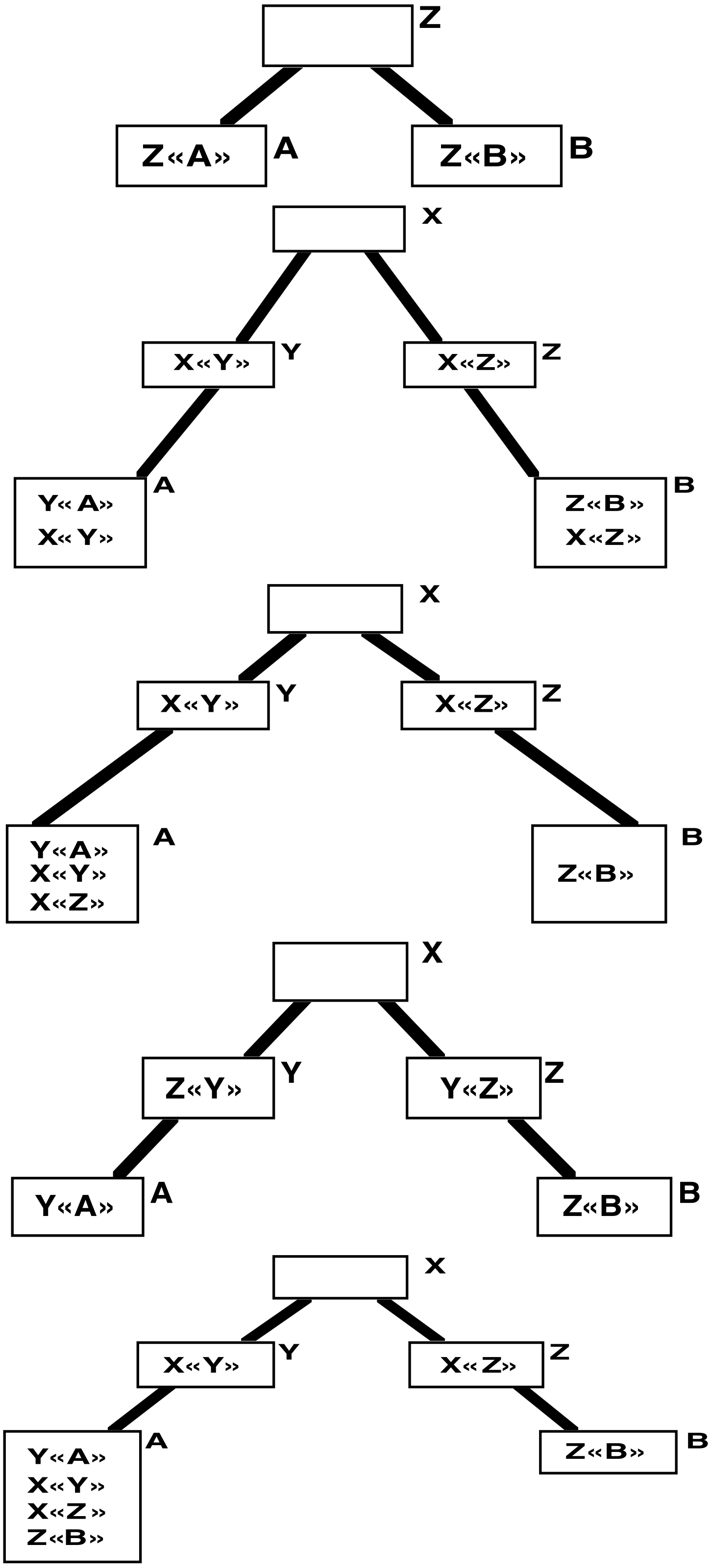
Certification Path Optimisations

Before users can mutually communicate using the PKCS and certificates to determine the valid public keys, a complete certification and return certification path must be determined. At first this seems to be a daunting task, but in practice the amount of information required can be reduced for a particular instance. For example:

- if two users that want to communicate are served by the same CA, then the certification path becomes trivial, and the users unwrap each others' certificates directly;
- if the CAs of users are arranged in some form of hierarchy, a user could store the public keys, certificates, and reverse certificates of all CAs between the user and some common point. The user would then only be required to obtain the certification paths from the common point of trust;
- if a user frequently communicates with users certified by a particular (other) CA, that user could learn the certification path to that CA and the return certification path from that CA, making it necessary only to obtain the certificate of the other user;
- certification authorities can cross-certify each other by some agreement, resulting in a shorter certification path. This works on the assumption that two rogue CAs are not working in collusion;
- if two users have communicated before and have learned one another's certificates (and verified them), they are able to authenticate without any for further verification (provided that certificate does not appear on a revocation list).

In any case, having learned each other's certificates from the certification path, the users must check the validity of the received certificates before trusted communication can take place.

Cert. Path Optimisations



Private Key Certificates

In general, non-repudiation services are provided using public key cryptosystems. However, it is possible to make use of private key cryptosystems to provide message authentication, thus providing for private key certificates [DS90].

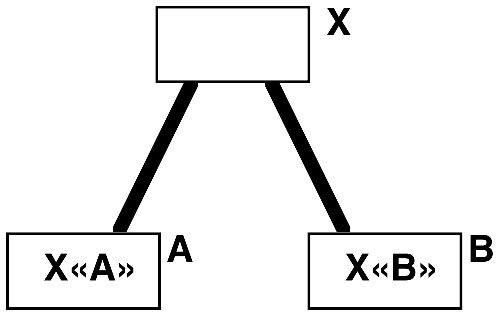
In a system such as this, every principal publishes a private key certificate. A principal's certificate contains the principal's identity and a private key that is valid for the life of the certificate. The certificate is encrypted using the Authentication Server's master key. These certificates can be published in world readable directories, just like public key certificates.

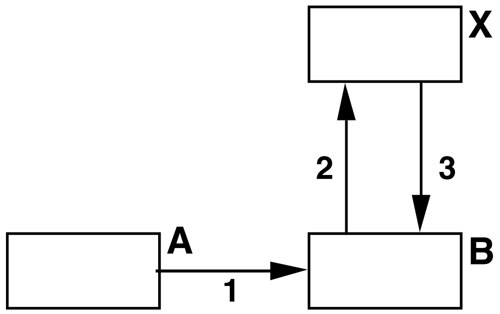
Let's assume that X wants to send a message to Y. X still encrypts that message using its private key (as in the PKCS, although in this case, there is no corresponding public key to decrypt the message). This is the same private key that is in X's certificate.

When Y receives the message from X, Y must retrieve X's private key certificate and then send the encrypted message, X's private key certificate, and its own private key certificate to the Authentication Server. The Authentication Server will decrypt X's certificate, retrieve X's private key and decrypt the message, and then reencrypt the message with Y's private key which it obtained from Y's private key certificate. The Authentication Server then sends the message back to Y. Y can now read the message. The reply from the authentication Server also contains information that assures Y that the message really did come from X.

Note that this type of service is not very efficient as every message must be sent to the Authentication Server for translation, verification, and reencryption.

Private Key Certificates





Mutual Trust

One of the main problems with certification paths is that ultimately, you require a single trusted point to start with so that all other CAs and public keys can be verified. Without that single point on trust, the recursive nature of certification paths has no terminating state.

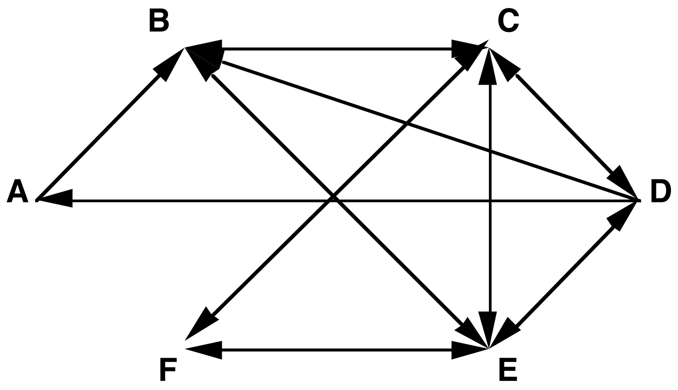
In Harn and Yang's paper [HY92], they discuss a number of techniques in which a group of mutually distrusting members may collectively sign a message, and have that message verified and trusted by a third party. Mostly, Harn and Yang suggest a new PKCS in which there are multiple secret keys (one for each member), and only one group public key.

One other technique which has promise is to replace the single point of trust with a mutually certifying group of (m) members. The public key for at least one of those members must be well known. Each group member issues a certificate for every other group member, certifying their belief of that member's public key. To determine if a member is issuing valid certificates that may be trusted, and that you have the correct public key for that member (including the well known public key), a *quorum* (n) of member certificates may be obtained and verified, one from each member certifying a chosen member. In this way, there must be at least (n) rogue CAs working as group members for a successful attack to be executed. If a greater degree of trust is required, the quorum can be increased, up to and including all members of the group (m). Note that if insufficient agreement is attained within the quorum (at least m - n members disagree on the correct public key being sought), then that key cannot be trusted. Once a valid public key is obtained, this can then be treated as a single trusted point and the certificates issued by that CA may be trusted as being valid.

Note that a sufficient number of rogue CAs (even only one, if the quorum is m) may cause a denial of service attack. However, this should be relatively easy to detect and rectify (be excluding that rogue CA from group membership). The rogue CAs will in general be in the minority of group members that disagree.

It is up to each group member to satisfy themselves in some manner that a potential group member is who and what they claim to be. If the potential member cannot provide sufficient credentials to satisfy at least (n) existing members of the group, the they will be unable to participate as a single trusted source, and hence any certificates issued by them will not be trusted.

Mutual Trust



Example - require quorum of 4

A has 1
B has 4
C has 4
D has 2
E has 4
F has 2

Therefore, we believe the certificates issued by B, C, and E for users outside the quorum group

Conclusion

The use of a PKCS for private communications and digital signatures is examined. To correctly do this requires that a user's public key be obtained in a trusted manner. Certificates are one mechanism for doing this. Certificates are issued by a Certification Authority. However, to verify the certificate, the CA's public key must be determined in a trusted manner.

Thus the problem of determining public key information becomes a recursive one, forming a certification path. Terminating the certification path requires a single point of trust. This single point of trust may be alleviated through the use of a group of CAs, providing a quorum of consensus from which a single point of trusted may be determined.

Conclusion

PKCS

Digital Signature

Obtaining Public Keys

Certificates

Certification Authorities

Certification Paths

Optimisations

Private Key Certificates

Mutual Trust

Bibliography

[DS90] D. Davis and R. Swick, Network Security via Private-Key Certificates, MIT Project Athena, 1990.

[HY92] L. Harn and S. Yang, Group-Oriented Undeniable Signature Schemes Without the Assistance of a Mutually Trusted Party, Proceedings of Auscrypt '92, Gold Coast, Australia, 1992.

[ISO7498] ISO 7498-2:1987, Information processing Systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture.

[SPX] Joseph J. Tardo and Kannan Alagappan, SPX: Global Authentication Using Public Key Certificates, (in preparation for IEEE), 1992.

[X.208] CCITT Recommendation X.208 (1992) | ISO/IEC 8824-2:1992, Information Technology - Open Systems Interconnection - Abstract Syntax Notation One (ASN.1): Information Object Specification.

[X.501] CCITT Recommendation X.501 (1992) | ISO/IEC 9594-2:1992, Information Technology - Open Systems Interconnection - The Directory: Information Model.

[X.509] CCITT Recommendation X.509 (1992) | ISO/IEC 9594-8:1992, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework.