Open Software Foundation **M. Warner (Telstra)**
Request For Comments: 85.0
October 1995

IMPROVED PUBLIC KEY LOGIN PROTOCOLS FOR DCE

1. INTRODUCTION

A number of protocols for ‘‘public-key’’ based login to DCE have been defined in OSF-RFC 68.1 [RFC 68.1]. These protocols have a number of problems, including the resulting complexity of the login process as well as security weaknesses. Furthermore, two of the proposed protocols fail to make use of public key cryptography for authentication.

This RFC proposes alternate protocols to RFC 68.1 for public key based user authentication in DCE. It is assumed that the user is in possession of his private key, either by virtue of a smart card, or a floppy disk. The use of public key cryptography is not considered useful if the private keys are stored in a central repository and distributed protected only by passwords. Mechanisms for distributing secret keys protected by passwords suffer from all the same limitations of Kerberos, and only reduce the performance of the system, without adding to the security. Until an organisation is willing to adopt smart cards (or other similar security tokens) there is little to be gained in migrating to public key technology.

Three protocols are defined in this RFC. The first defines interactions between the smart card and the workstation, which would be used to log in to the workstation. The second protocol defines the interaction between the workstation and the Kerberos KDC which provides the workstation with a TGT. The TGT is used to gain access to Kerberos based servers in the normal manner. Both protocols are based on the abstract protocols defined in X.509 [X.509].

The second protocol (workstation to KDC) is a specialisation of a more general protocol which is suitable for authentication to arbitrary servers which have their own public/private key-pair. This will ensure a smooth migration path to a purely public-key authentication system, which does not require a secure, on-line key distribution centre. The totally off-line system is defined in the third set of protocols.

The protocols are independent of the actual cryptographic algorithms; however, they assume the existence of a digital signature algorithm and a public key encryption algorithm which is used to ensure the confidentiality of symmetric session keys. Examples of suitable public key algorithms which can be used for both digital signatures and encryption are RSA [RSA] and the elliptic curve cryptosystem described in [DEMY].

2. MOTIVATION

Current DCE security is based on symmetric key cryptography. This requires the use of an on-line key distribution service. This service, which is based on Kerberos, must be both highly available (since authentication cannot occur while the key distribution centre is unavailable), and also highly secure (since an attack on the key distribution centre will provide an attacker with access to all system resources).

The use of asymmetric or public key cryptography can eliminate the need for a secure, on-line key distribution service, while at the same time decreasing the management overhead associated with very large scale systems (including multi-cell systems). So far, only public key based login protocols have been considered in DCE. The full benefits of using public key cryptography will be realised when authentication between clients and servers occurs directly, without the assistance of an on-line key distribution service. Current proposals for public key based login [RFC 68.1] have a number of problems, and are incompatible with this ultimate goal.

The public key based protocols for user authentication described in this document provide a migration path toward a fully public key based security system for DCE. This will minimise the number of incompatible systems which will be deployed and which will require on-going support. This cost of this support cannot be over estimated. Additional advantages of the protocols described here are greater security, and compatibility with a number of important standards.

3. GOALS AND NON-GOALS

This proposal aims to provide:

(a) Strong authentication of users to the local workstation.

(b) Secure acquisition of TGT from KDC.

(c) Elimination of passwords (except to activate smart cards).

(d) Maximal use of existing standards.

(e) A smooth migration path to an authentication system which does not rely on on-line servers.

It does not address:

(a) Distribution of private keys to users without smart cards.

(b) Integrity and confidentiality of DCE wire protocols above the existing DES based system.

(c) Multiple certification authorities (although this is not excluded).

4. TERMINOLOGY

User	The human user (and associated smart card).
Client	An application running on the user's workstation. The login process is an example of a client.
`Cx`	Certificate of X.
`K`	Symmetric (session) key.
`{M}Px`	Message M encrypted with X's public key.
`{M}Sx`	Message M signed by X.
`Nx`	A nonce (e.g., random number) generated by X.
`Tx`	A time stamp generated by X.
`KDC`	The Kerberos Key Distribution Centre.[1]
`TGT`	A Kerberos Ticket Granting Ticket.
`Ktgt`	The session key to be used with a TGT.
`Tcs`	A ticket for client C to use a server S.
`Kcs`	A symmetric session key for use between C and S.

5. DEFICIENCIES OF RFC 68.1

Three different login protocols are defined in RFC 68.1. This fact alone, implies the need for complex signalling and negotiation to determine which protocol is being used. This will increase the size and complexity of the client code.

The provision of multiple protocols, particularly of different strengths, also introduces the potential to exploit weaker protocols when strong authentication should be used. Prevention of these potential attacks requires additional management effort, and demands greater security knowledge from the system administrators.

The protocols included in RFC 68.1 are incomplete as specified, making detailed analysis of them difficult. In particular, claimed identities are not included in any of the protocols. If correct, this would imply that the KDC would have to deduce the client's identity, by performing an exhaustive search until it found a key which successfully

1. No distinction is made here between the Authentication Service and the Ticket Granting Service for reasons of clarity.

decrypted the request. More detailed comments on the individual protocols defined in RFC 68.1 are given below.

RFC 68.1 does not eliminate problems associated with an on-line key distribution centre. In particular, if the KDC is compromised, an attacker will have access to all resources in the system. Thus claims that the attacker must mount a dictionary attack on each individual account are misleading. Only the adoption of a fully off-line system (as provided by the protocols in section 6.3 of this RFC) will eliminate this threat. While it may not be possible to provide this functionality in DCE 1.2, the protocols used for obtaining a TGT should be compatible with this long term goal.

The protocols of RFC 68.1 are intended to provide a migration path toward a robust and useful public key based login system. However, the protocols defined are incompatible with a fully public key system. Adoption of these protocols will result in the potential deployment of multiple different (incompatible) systems. All of these protocols will have to be supported for an indefinite period of time to ensure interworking with other systems. These stop gap solutions will provide on-going burdens in terms of code-size (in both clients and the security service), protocol complexity and management complexity, the cost of which cannot be over estimated. They should be avoided in OSF releases unless they are addressing major deficiencies.

5.1. PADATA-ENC-THIRD-PARTY Protocol

The first protocol option basically distributes the user's private key protected by a password. It should be noted that one of the acknowledged goals of this protocol was to ''not require the login module to have prior access to the principal's private key''. This statement indicates that this protocol is not providing public key based authentication.

Authentication is based on a user's knowledge of the password, and thus provides no increase in security over the existing Kerberos based system. In addition, since an on-line repository of private keys is required, it does not eliminate a single point of attack or a single point of failure. A compromised KDC will reveal password protected private keys, which would have to be changed to recover. The cost of recovery is therefore not reduced.

In short this protocol provides no significant benefits over continued use of Kerberos. It will add considerable complexity to the login process, increase the size of client code, add to the management task, and increase the problems associated with single sign-on. Adoption of this ''solution'' will seriously undermine the credibility of security in DCE.

5.2. PADATA-ENC-PUBLIC-KEY-KDC Protocol

The second protocol also does not use the private key for authentication. Authentication is on the basis of knowledge of a password. The use of the KDC's public key to encrypt the authenticator in the first message does help protect against password guessing attacks on H2, but the response of the private key serves absolutely no purpose whatsoever, since the TGT could be encrypted using SK. The protocol is therefore inefficient both in terms of computation and message lengths. The only advantage this protocol offers over standard Kerberos is protection against password guessing. The centralised private key repository remains a single point of attack and failure for the whole system.

Again this protocol adds complexity without providing significant advantages over the existing Kerberos system. The same deployment costs will be associated with this protocol as with the preceding one. This protocol should not be implemented.

5.3. PADATA-ENC-PRIVATE-KEY Protocol

The third protocol assumes that the private key is available at the principal. It is the only protocol that actually makes use of public key cryptography for authentication.

The protocol described is seriously flawed. The authenticator signed by the principal does not include the identity of the intended recipient. While this may be adequate in situations where public key based authentication is only between principals and the KDC, such an environment cannot be assumed, and also limits future use of public key authentication.

To illustrate the threat, consider an environment in which DCE and a secure electronic mail application exist, both using similar protocols for authentication (i.e., signed time stamps without the intended recipient). An attacker who receives an authenticated electronic mail message from a principal will receive a time stamp signed by that principal. This could then be used to construct the first message of the PADATA-ENC-PRIVATE-KEY protocol, and the attacker could successfully log in with the principal's identity.

The design of the protocol also requires that the symmetric session key be generated by the client. Good key generation is difficult without dedicated hardware. While it may be economical to provide this hardware at servers (and in particular the KDC) it may be expensive to provide such hardware at every client machine. The key generation process is also more vulnerable to attack at the client workstation. For these reasons it is preferable for the server to generate the symmetric session key. This will make the use of secure (truly random) key generation more feasible.

Protocols which could be used instead of the PADATA-ENC-PRIVATE-KEY protocol are described in this RFC.

6. FUNCTIONAL SPECIFICATION

This section defines abstract protocols suitable for authentication in DCE. These include:

(a) Protocols for authenticating end users at login.

(b) Protocols for obtaining a Kerberos TGT.

(c) Protocols for establishing a secure session between clients and servers without the use of an on line authentication or key distribution service.

All protocols make use of public key cryptography and smart cards. No assumptions are made about the security of the underlying communications mechanisms. In particular these protocols do not make use of the existing authenticated RPC of DCE. The protocol messages include sufficient security as described.

A single certification authority is assumed within the cell. Certificate chaining is not discussed although it is not precluded. Delegation, as described here, will form a natural extension to certificate chaining.

While it may be impractical to provide a fully public key system initially, it is nevertheless important to consider these protocols when designing the login and TGT establishment protocols.

6.1. User Login

The concept of ''logging in'' to a distributed system is an abstract one. A user will log in to a workstation within the system, and will be able to issue requests to individual servers within the system. The distinction between logging in to a workstation and logging in to the distributed system that exists in Kerberos based systems is less useful in a purely public key environment, since clients may establish secure sessions with servers without the aid of an on-line intermediary.

Logging in to the workstation is probably outside the scope of DCE. Nevertheless, the protocols used to do this must be considered to ensure that the functionality required of the smart card is similar to that required by DCE. In addition, the objective of providing a single sign-on may be facilitated by defining a common protocol within OSF.

To log in to the workstation, the user should insert a smart card into the reader. The workstation will then prompt for a PIN, which will be entered into the card, card reader or workstation depending on the facilities available. When correctly entered the smart card will be activated. This will allow the workstation to interrogate the smart card, asking for the user's identity. The workstation will then request the user's certificate from the directory service (the smart card may provide a copy of the certificate along with the user's identity; however, the workstation may choose to ignore this certificate). The certificate will be checked using the certification authority's public key which is kept local to the workstation (perhaps in Read Only Memory, or on a local disk).

An X.509 style authentication exchange is then performed between workstation and the smart card. The workstation itself need not be an identifiable principal in the system, in which case it would not sign the challenge it issues to the smart card. Unless a secure bootstrapping procedure is provided for workstations there is little point in authenticating the workstation to the user. Secure bootstrapping is outside the scope of DCE.

6.1.1. Smart card to workstation protocol

The user authentication protocol may be summarised as follows:

```
W-->U:          PIN
```

Initially the smart card must be activated by entering the PIN. The workstation prompts the user for the PIN, which is entered via the keyboard (or the smart card reader if it has a keypad).

```
U-->W:          U,Cu
```

The card responds with the user's identity `U` and the user's certificate `Cu`.

The workstation must obtain the user's public key to successfully authenticate him/her. The user's certificate can be used to extract the public key, after validating the certificate with the local certification authority's public key. If the local policy requires, the certificate can be retrieved directly from the directory service or the DCE registry, to prevent the use of revoked certificates.

```
W-->U:          W,Nw
```

The workstation generates a challenge `Nw`, which it sends along with its identity `W`. The identity may be a null value if workstations cannot be authenticated.

```
U-->W:          {Nw,W,Nu}Su
```

The smart card responds by signing the workstation's nonce and identity with the user's secret key. It is desirable that the smart card includes a nonce `Nu`. This can either be a sequence number or a random number, depending on the capabilities of the smart card. Inclusion of the card generated nonce prevents the authentication exchange from being used to get the smart card to sign arbitrary data.

The workstation verifies the signature on the response using the user's public key. If the signature is valid, the user has been successfully identified, and is allowed into the system.

6.1.2. Delegation keys

The concept of delegation is introduced here, although many of the details require further study. Delegation is not required in the first instance; particularly when public key cryptography is only being used to acquire a TGT. It is included to demonstrate how it could be used in the future.

To eliminate the need for the smart card to participate in subsequent authentication interactions, a new private-public key pair can be generated after the user has been authenticated. The smart card must sign a delegation certificate, allowing the holder of the key pair (i.e., the workstation) to act on his behalf for a limited period of time.

The format of a delegation certificate should be similar to a user's public key certificate as generated by the certification authority. Thus delegation certificates will be included as the final element of the certificate chain used to establish the user's public key. Servers may impose their own policies when evaluating delegation certificates. In particular they may choose not to accept delegation certificates if the length of time for which the certificates are valid is too long.

It may be desirable to limit the privileges of the delegation to a subset of the user's normal privileges. This could be done by optionally including an EPAC in the certificate. This EPAC must be a restriction of the user's normal EPAC.

Of the information required to generate a delegation certificate, the commencement time and public key must be supplied by the workstation (assuming the card cannot generate keys and does not have direct access to a time service). If possible, the expiry time should be calculated by the card from the commencement time to limit the time period for which a corrupt workstation could be delegated to.

6.2. TGT Acquisition

The protocol described here replaces the protocols defined in RFC 68.1. The protocol assumes that the client has access to the private key of the user. Access could be via smart card operations to sign challenges etc., or could be stored in a local file (e.g., on a floppy disk). There is no point in distributing private keys from a central repository, since this uses passwords to authenticate the user, rather than public key cryptography. If private keys are not available at the client, then Kerberos should be used.

The process of TGT acquisition requires an X.509 authentication exchange with the KDC. This exchange should take the same form as an authentication to an arbitrary server (as defined in section 6.3), and will result in the mutual authentication of both client and server, as well as the establishment of a shared symmetric session key. This symmetric key should be generated by the server, which is more likely to be able to generate keys securely.

Public keys (in the form of certificates) may be obtained from the directory service, or the DCE registry. In general it will be acceptable for the server to return its certificate to the client in the authentication response. Frequently used certificates may be cached, without special precautions.[2]

A two-message protocol is used to acquire a TGT. This protocol relies, in part, on time stamps to guarantee the freshness of messages. There is no reason to adopt a challenge-response mechanism since the subsequent Kerberos protocols rely on time stamps. Nevertheless, since the session key is encrypted with the public key of the client, successful use of the TGT implies the ability to decrypt this session key, and therefore possession of the user's private key.

```
C-->KDC:        C,{Tc,KDC,Nc}Sc,Cc
```

The client process signs a message including a time stamp, a nonce and the identity of the KDC. This is sent to the KDC along with the client's claimed identity and certificate.

The KDC may choose to ignore the certificate and retrieve it directly from a directory or the DCE Registry. Alternatively, the KDC may check revocation lists to establish the validity of the certificate.

```
KDC-->C:        KDC,{Tkdc,C,Nc,{Ktgt}Pc}Skdc,Ckdc,TGT
```

The KDC checks the time stamp and signature. If the time stamp is sufficiently current (and the nonce new) it responds with a signed message including a time stamp, the client's nonce, the client's identity, and the symmetric key to be used with the TGT. This key is encrypted using the client's public key. The KDC includes its certificate in the response. The TGT is returned unencrypted since it is subsequently used in the clear by the client.

By checking the signature on this response, the client can be assured that the reply is from the KDC. The session key can only be decrypted by the legitimate client who possesses the corresponding private key. The TGT and associated session key are then used as normal.

6.3. Off Line Client-Server Authentication

This section defines two protocols for establishing a secure context between an arbitrary client and server, without the aid of an on-line key distribution service. It is not expected that this functionality will be implemented in DCE 1.2; however, the intention of this RFC is to provide a migration path toward totally off-line authentication. Therefore, these protocols are included to allow readers to see the long term strategy.

Two protocols are defined, one based on the use of time stamps, and the other based on pure challenge-response techniques. While the definition of two separate protocols does require additional overhead in terms of code size and negotiation, it is justified by the other advantages. The two-message protocol is identical to the TGT acquisition protocol, and fits well with the RPC message flow. While the three-message protocol requires an additional message, it does not rely on the existence of a secure time service. The management overhead associated with setting up and maintaining the time service is considerable. This will be even more problematic in an environment consisting of large numbers of cells. If DCE is to be used in very large scale environments (such as the Internet), reliance on synchronised clocks might carry a very significant cost penalty.

6.3.1. Three-message protocol

In the following protocol description, the client (`C`) is assumed to possess the server's public key, and has access to his private key.

```
C-->S:          C,{S,Nc}Sc,Cc
```

2. Where impersonation of servers is considered a threat, and where the server's key may be subject to attack, the certificates should be checked to ensure they have not been revoked.

The client initiates the exchange by sending a nonce bound to the server's identity with the client's signature. The client's claimed identity and certificate are also included.

```
S-->C:          S,{C,Nc,Ns,{Kcs}Pc}Ss,Cs,Tcs
```

The server responds with its own nonce which again is bound to the user's identity, and the user's initial challenge with the server's signature. Also included, is a session key for later use. This session key is encrypted with the public key of the client. This message authenticates the server to the client.

In both this and the next protocol, the session key can be accompanied by a Kerberos-like ticket for using the server. This ticket (`Tcs`) would be generated by the server itself, and contain the session key, client identity and a time stamp, all encrypted with a key known only by the server. The client would present this ticket with subsequent communications, thereby eliminating the need for the server to maintain a mapping of clients to session keys.

```
C-->S:          C,{S,Ns}Sc
```

The final message in the three-message authentication is a signed response from the client including the server's nonce and identity. This message completes the mutual authentication by verifying the client's identity to the server.

To reduce communications overheads, the final message can be attached to the RPC call from the client to the server.

6.3.2. Two-message protocol

The two-message protocol reduces communication overheads, but relies on a synchronised clock between the client and server. The support of a secure, synchronised time service may be expensive in many situations.

```
C-->S:          C,{Tc,S,Nc}Sc,Cc
```

This message is identical to the first message of the three-message protocol, except for the addition of a time stamp. The nonce is included to prevent replay attacks within the life time of the time stamp. This message is identical in form to the first message of the TGT acquisition protocol.

```
S-->C:          S,{Ts,C,Nc,{Kcs}Pc}Ss,Cs,Tcs
```

The server responds with a time stamp, and a session key encrypted with the client's public key. This information is bound to the client identity by the server's signature. This completes the mutual authentication.

Again, this message is identical to the corresponding message in the TGT acquisition protocol. The session key can be accompanied by a Kerberos-like ticket for using the server.

7. SMART CARD FUNCTIONALITY

This section defines the minimal functionality required of the smart card. More sophisticated cards could provide even greater security, by taking an active part in the authentication protocols rather than simply signing data provided by the workstation. In the absence of any such on-card intelligence it is advisable to use separate key pairs for authentication and digital signatures, in order to prevent attackers providing the card with dummy challenges which have been generated by hashing a message.

To prevent the exposure of the private key outside the smart card, all signing and decryption operations must be performed on the card. The smart card must therefore provide sufficient processing power and memory to sign and decrypt messages (for algorithms such as RSA, the decryption and signing functions are identical). In general only a single block will be decrypted or signed per operation.

Storage will be required for the private key(s) and the user's certificate(s). As indicated above, separate key pairs should be used for authentication (and key exchange) and for general digital signatures. While general digital signatures are not described in this RFC, smart cards should accommodate this facility to support future applications.

Assuming 512 bit (64 byte) private keys, certificates of around 500 bytes, and separate authentication and digital signature keys and certificates, the amount of non-volatile memory needed is around 1200 bytes. The key files are likely to be larger if various techniques (such as the Chinese remainder theorem) are used to speed up computations. This may be offset by storing common elements of the authentication and digital signature certificates (such as the issuer name, subject name, certificate signature algorithm, validity time, and certificate version number) in one file, rather than duplicating them in both the certificates.

Cards which meet these requirements are widely available. An example is the Philips DX card [PHIL] which performs 512 bit modular exponentiation, and provides 2032 bytes of EEPROM.

8. ALTERNATIVES TO SMART CARDS

In some circumstances the use of smart cards will not be necessary or justified. Typically, this will occur when the workstation from which the user accesses DCE servers is in a physically secure environment. In such cases, it may be possible to store the user's private key on a local disk. The private key should be encrypted with a password in such cases. In such a system, the login process must prompt the user for a password and decrypt the private key.

An alternative model would be to have the user's private key stored on a floppy disk. Again passwords should be used to encrypt the key. This solution is only suited to environments where the workstation is trusted to behave correctly. In particular, it must not copy the private key onto a hard disk. Again, this solution is only really suited to a physically secure workstation environment.

9. DATA STRUCTURES

The messages used in the authentication exchanges between clients and servers take the form of bind tokens. In the two-message protocol these bind tokens should conform to the structure defined in X.511 [X.511]. Where a three-message authentication protocol is used, or where negotiation of session security mechanisms is required, the appropriate fields should be added. There is no current international standard which defines such a bind token. The Simple Public Key Mechanism described in an Internet Draft [ADAM] defines a suitable token; however, this is not compatible with the two way X.511 token format, and is itself not yet an accepted standard. Work within ISO/JTC1/SC21 is likely to define a suitable token format.

In the absence of any emerging standard, the following ASN.1 definition of a bind token could be adopted:

```
Token ::= SIGNED { SEQUENCE {
    algorithm         [0] AlgorithmIdentifier
    name              [1] DistinguishedName,
    time              [2] UTCTime,
    random            [3] BIT STRING,
    response          [4] BIT STRING OPTIONAL,
    bindIntAlgorithm  [5] SEQUENCE OF AlgorithmIdentifier OPTIONAL,
    bindIntKeyInfo    [6] BindKeyInfo OPTIONAL,
    bindConfAlgorithm [7] SEQUENCE OF AlgorithmIdentifier OPTIONAL,
    bindConfKeyInfo   [8] BindKeyInfo OPTIONAL } }

BindKeyInfo ::= ENCRYPTED { BIT STRING }
```

This definition is a straight forward extension of the X.511 definition, adding only the optional fields. It allows the use of distinct keys for integrity and confidentiality, and provides a mechanism for negotiating the integrity and confidentiality algorithms used.

10. API'S

It should be possible to insulate applications from the level of smart card functionality, not to mention the choice of card reader, through the definition of appropriate interfaces. The level at which standard interfaces are defined will determine the level at which interworking is possible, and the degree of openness provided. There are two suitable levels at which interfaces may be defined. Each has different advantages and lead to different levels of interworking. Note that these interfaces are not mutually exclusive, and may be provided in a layered manner.

Standardisation at the smart card command level (defining the full syntax and semantics of data passed to the smart card) allows user mobility, since a user's smart card will be compatible with any login facility. This does, however, limit the functionality of the smart cards to that defined in the standard, and prevents the use of more sophisticated (and hence more secure) smart cards where available. The login facility would be customised for a particular card reader.

Standardisation at the functional level (e.g., defining an API which includes functions such as sign and decrypt) will insulate the applications from actual smart card commands. This will allow the introduction of different smart cards with different capabilities. Libraries for the appropriate smart card (and smart card reader) would have to be provided.

The definition of these two API's requires further work.

11. ACKNOWLEDGEMENTS

The author would like to thank Edward Zuk and Gary Gaskell for their comments.

The permission of the Director, Telstra Research Laboratories, to publish this RFC is hereby acknowledged.

REFERENCES

[RFC 68.1] Anderson, A, and Wray, J., ''DCE 1.2 Public Key Login Functional Specification'', Open Software Foundation, Request For Comments 68.1, February 1995.

[X.509] ITU-T, ''Recommendation X.509 — The Directory: Authentication Framework'', 1993.

[RSA] RSA Encryption Standard, Version 1.5, RSA Data Security Inc., November 1993.

[DEMY] Demytko, N., ''An Elliptic Curve Analogue of RSA'', Proceedings of Eurocrypt '93, Springer-Verlag, 1994.

[PHIL] Philips, ''DX Card Reference Manual'', December 1992.

[X.511] ITU-T, ''Recommendation X.511 — The Directory: Abstract Service Definition'', 1993.

[ADAM] Adams, C., ''The Simple Public-Key GSS-API Mechanism (SPKM)'', Internet Draft, draft-ietf-cat-spkmgss-02.txt, March 1995.

AUTHOR'S ADDRESS

Michael Warner
Telstra Research Laboratories
770 Blackburn Rd
Clayton, Vic, 3168
Australia

Internet email: m.warner@trl.oz.au
Telephone: +61-3-9253-6720
Facsimile: +61-3-9253-6352