# The DCE Security Service

A security protocol consisting of encryption keys, authentication credentials, tickets, and user passwords is used to provide secure transmission of information between two transacting parties in a DCE client/server enterprise.

**by Frédéric Gittler and Anne C. Hopkins**

The Open Software Foundation's Distributed Computing Environment (DCE) is a collection of integrated services that support the distribution of applications on multiple machines across a network. In most cases, networks are inherently insecure because it is possible for someone to listen to traffic or behave as an impostor. Without countermeasures this threat could prohibit the distribution of business applications.

The DCE security service described in this article provides a set of security mechanisms that can be easily used by a distributed application to remove the security vulnerabilities mentioned above.

The security functionality provided by the DCE security service includes:
- Identification and authentication of users to verify that they are who they claim to be
- Authorization for applications to decide if a user can access an operation or object
- Secure data communications to protect the data communication of an application against tampering or eavesdropping.

## Security Services

The DCE security service, with additional new services and facilities, is based on the Kerberos system.[1] The Kerberos system performs authentication of users and servers based on cryptographic keys so that communicating parties can trust the identity of the other. DCE augments Kerberos with a way to transfer additional security attributes (beyond just identity) to a server which may choose to perform access control on those attributes. The DCE communication protocol contains support for protected communications that relies on crytographic session keys provided by Kerberos.

Fig. 1 shows the environment in which the DCE security service operates, and the services provided on the DCE security server.

**Registry.** Every DCE security service user is known as a principal. Interactive (human) users, systems (computers), and application servers (processes) are all principals. Each principal shares a secret key† with the DCE security server. The secret key for interactive users is derived from the user password. This security model relies on the fact that a particular key is known only to the principal and the DCE security service.

† As in the Kerberos system, keys are used for encrypting and decrypting data transferred in a network transaction, and are known only to the DCE security server and the parties involved in the transaction.

The registry service is the manager of the central registry database which contains the principal's name, universal unique identifier (UUID), secret key, UNIX® account attributes, and other attributes of the principals. These attributes include the extended registry attributes (ERA), which may be defined and instantiated by an administrator.

Like other DCE services, access to the registry service is based on the use of remote procedure calls (RPCs). The registry's operation is secure because it uses a protected RPC for all of its transactions. Extended registry attributes are covered in more detail later in this article.

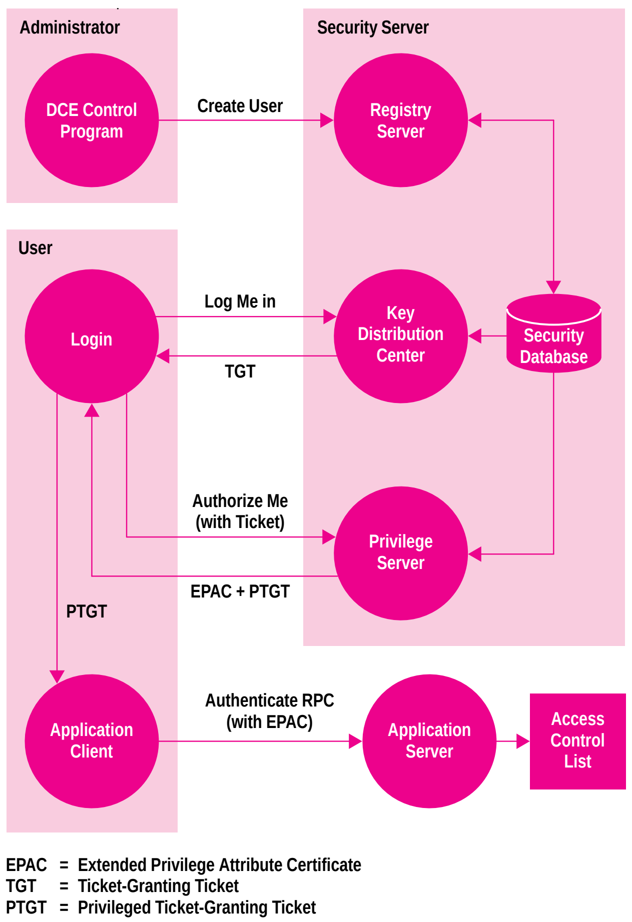


**Fig. 1.** The components of the DCE security server in relation to the other components typically found in a distributed environment.

## Glossary

The following are some of the terminology and associated acronyms frequently used in this article.

**Extended Privilege Attribute Certificate (EPAC).** A credential provided by the DCE privilege service containing user and group identities and attribute-value pairs. This information is used by an application server to make authorization decisions.

**Extended Registry Attribute (ERA).** A mechanism in which attribute-value pairs are associated with principals. The information in these attribute-value pairs may be used to deny or grant an authorization request.

**Principal.** An entity such as a user, an application, or a system whose identity can be authenticated.

**Service Ticket.** A credential used by an application client to authenticate itself to an application server.

**Ticket-Granting Ticket (TGT).** A credential that indicates that a user has been authenticated and is therefore eligible to get tickets to other services.

**Identification and Authentication.** The first interaction between a user and the DCE security service is the login sequence when the identity of a user is authenticated by a secret key. The result of this authentication is a ticket-granting ticket (TGT) containing the user principal's credentials. The TGT indicates that the user has been authenticated. It is used, as its name implies, to obtain tickets to other services. The life span of a TGT is limited to ensure that the user represented by the credentials is the user currently using the system and that the user's credentials are up-to-date.

The user and group identity and the extended registry attributes are not part of the TGT issued by the authentication service. The privilege service supports an additional authorization by providing user and group identities and attributes in the form of an extended privilege attribute certificate (EPAC). During a login sequence, after the TGT is obtained, the run-time DCE security service makes a request to the privilege server to issue a privilege TGT. This ticket is a combination of the TGT and a seal of the EPAC.

The privilege TGT is stored in the user's environment and is used by the secure communication mechanisms to obtain a service ticket from the authentication service. The service ticket is used by the communication mechanisms to perform mutual authentication between the application client and the application server.

In each of these exchanges, secret session keys, which are known only to the DCE security service server, are generated for a particular session between the client and server. The DCE security run-time environment, RPC, and GSS (Generic Security Service)[2] API use these keys for data encryption or integrity protection generation in any network communication during a particular session. A brief description of the GSS API is given later in this article.

**Authorization.** DCE security provides application servers with multiple options for authorization. A server can choose to grant access to a user based on one of the following three models.

- Name-based authorization. The simplest but least scalable way of doing authorization is to compare the name of the remote principal with the names stored in an application-specific database. This method is called name-based authorization and is available when using the DCE secure communication mechanisms.
- Privilege-based authorization with access control lists. DCE servers can choose to protect their resources with access control lists (ACLs). An ACL contains entries that describe the particular permissions granted to various principals. An ACL entry may specify an individual user (principal) name, a group name that implies several principals, or "other" to indicate any principal not already matching a user or group entry. Users, groups, and others from a foreign cell may also be specified in an ACL entry.

  When a server receives a remote request, it asks the authenticated RPC run-time environment for the caller's EPAC. The EPAC contains the caller's principal and group identities, which are compared against the ACL to determine if access is granted. If the caller's principal identity matches the principal in an ACL entry, and if that ACL entry contains the required permissions, then access is granted. If there is no match on the principal, but one of the caller's groups matches a group ACL entry, then the permissions in the group entry apply.

  The DCE library includes facilities to manage ACLs and perform authorization checks based on ACLs. ACLS are described in the article on page 49.
- Other authorization. Other authorization mechanisms are made possible by the ERA facility. A server can use the value of any given attribute in a user's EPAC to decide whether it should service or deny any given request.

### Secure Data Communication

DCE provides the remote procedure call (RPC) communication mechanism as one of its core services. The DCE security service is designed to support protected RPC communication.

Not all distributed applications in a DCE environment will use RPC. Most client/server applications in existence today are message-based, and changing them to use the RPC paradigm is expensive and time-consuming. It is also not practical for certain applications to use RPC. These applications nonetheless require security. For this reason the DCE security service now supports the Generic Security Service API (GSS API), which allows an application to authenticate itself to a remote party and secure data for transmission over an arbitrary communication mechanism.

Four basic levels of protection are available with either RPC or GSS API:

- No protection. The DCE security service does not mediate or participate in the connection.
- Authentication. The user of the client application is authenticated to the server.
- Data integrity. A cryptographic checksum is included with the data transmitted. The DCE security service guarantees the data received is identical to the data transmitted.
- Data privacy. The data is transmitted in an encrypted form and is therefore private to the sender and the receiver. United States export regulations limit the availability of this level of protection outside of the United States and Canada.

The higher protection levels include the protections offered by the lower levels.

## Security beyond DCE

Logically, two login sequences are required: the login to the system and the login to DCE. Entries in the DCE security service registry contain all the attributes associated with a UNIX account. These entries can be used instead of the traditional /etc/passwd file or NIS† database as the source of information for the UNIX system login. The HP-UX* operating system integrates the system and DCE login sequences into an integrated login facility, which is described in the article on page 34.

The DCE security service can be used as the core security service for the enterprise because it features an extensible registry through the ERA facility. Products from HP and other manufacturers licensing DCE from the Open Software Foundation (OSF) will undoubtedly use the extended registry attribute facility either to provide other integrated login facilities or to synchronize the DCE security service registry with other user databases.

The secure data communication mechanisms described above can be used by system vendors to secure the standard network communication protocols, such as the file transfer protocol (ftp).

## Security Mechanisms

The mechanisms used by the DCE security service to provide secure data communication are a combination of key distribution, data encryption, and data hash tables. The purpose of this section is to give more details about these mechanisms. Some details have been omitted for brevity and readability. More formal and complete descriptions of the algorithms can be found in the references indicated below.

**Data Encryption.** The DCE security service uses the Data Encryption Standard (DES)[3] algorithm to protect the data it transmits. This algorithm is used by both RPC and the GSS API to protect user data and guarantee its integrity. DES requires that the two parties exchanging information share a secret key, which is only known to the two parties. This key is 64 bits long and has 56 bits of data and 8 bits of parity.

DES encrypts plain text in blocks of 64 bits. The encryption is obtained by the iteration of a basic operation which combines permutation of bits for both key and data with exclusive-OR operations. The result of the encryption is a block of cipher text in which each bit depends on all the bits of the key and the plain text. Decryption of the cipher text involves the inverse of the same basic operation. The party receiving the cipher text and performing the decryption has a copy of the key used for encryption.

The DCE security service uses DES in cipher-block-chaining mode in which plain text blocks are exclusive-ORed with the previous cipher text block before being encrypted. The DCE security service also uses confounder data, which is a dummy block of random data placed before the application data. Confounder data is used to prevent guessing by correlation between blocks of encrypted data. The same block of plain text can result in two completely different blocks of data once encrypted with the same key because of the fact that the confounder data will be different. These two techniques render security attacks particularly difficult because each block of cipher text depends on the previous cipher block and some random data.

**One-Way Hash.** The DCE security service uses the message digest 5 (MD5) algorithm[4] coupled with the DES encryption algorithm to guarantee the integrity of the data being transmitted and verify the success of the decryption operations. MD5 produces a 128-bit signature (also called a message digest) that represents the data being transmitted. This message digest is obtained by processing the data in blocks of 512 bits. The algorithm is driven by a fixed table containing 64 operations. It uses four 32-bit variables and involves rotation, exclusive-OR, OR, negation, AND, and addition operations on these variables and the 16 32-bit segments contained in each block. Like all one-way hash functions, MD5 is designed to be easy to compute and difficult to break (i.e., derive plain text from a given hash). DCE uses CCITT-32 CRC,[5] a checksum algorithm, to verify data integrity in certain cases.

**Keys.** The DCE security service uses two types of keys: long-term principal secret keys and conversation or session keys.

- Principal keys. The DCE authentication protocol (described below) requires that the DCE security server and the principal requesting authentication share a secret key. For a machine or process principal, this key is stored in a file and is protected by the local operating system protection mechanisms. In the case of a human principal, the secret key is derived from the user's password by a one-way hash function.[1] All the principal keys are stored in the DCE registry.
- Conversation or session keys. Conversation and session keys are used to encrypt the data and checksums exchanged between the application client , the application server, and the DCE security server. The designs of the DCE and Kerberos security mechanisms avoid the need to establish a long-term secret key for each pair of communicating principals by creating short-lived session keys and communicating them securely to each principal engaging in a data exchange. In addition, session keys reduce the vulnerability of long-term principal keys because the latter are used less often and therefore are less susceptible to offline attacks.

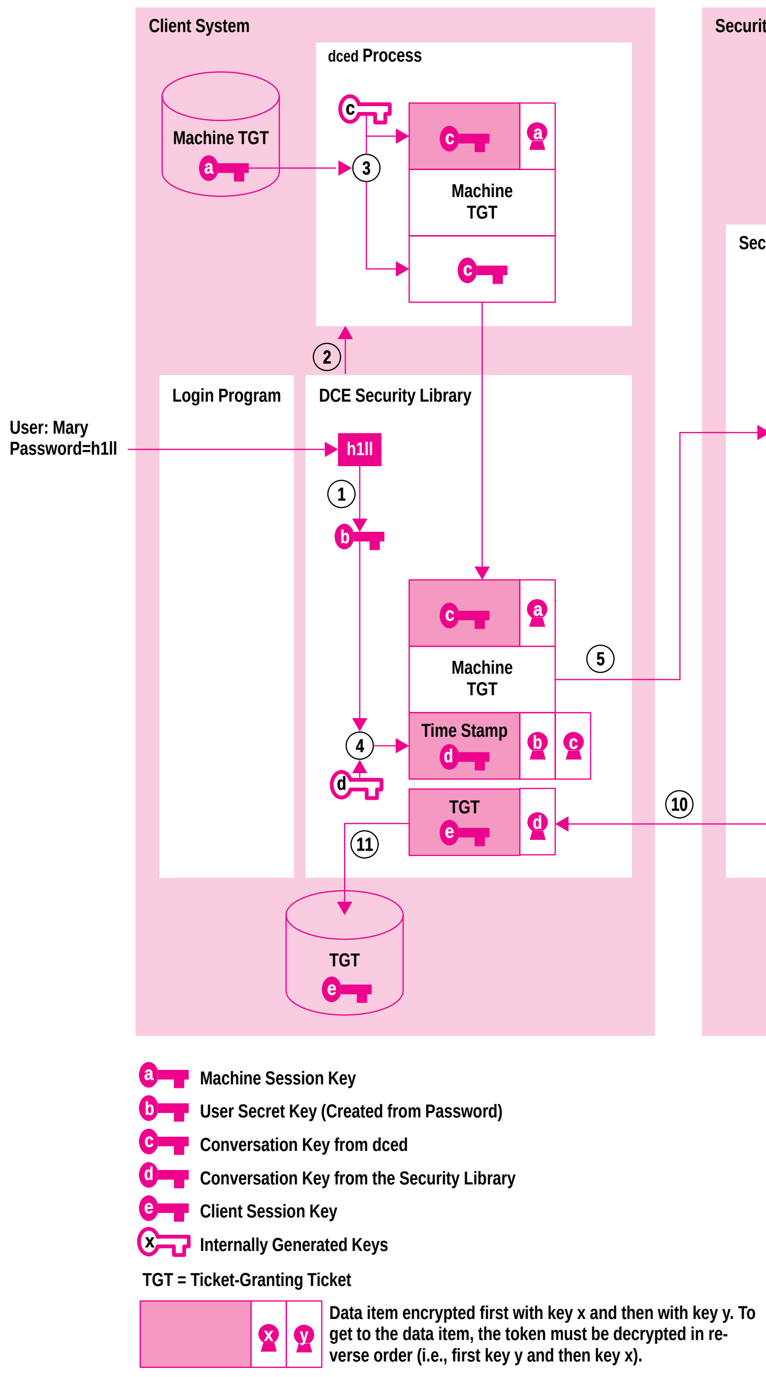
The conversation and session keys are generated as random numbers by the DCE security service and are not reused. These keys have typical lifetimes measured in minutes. Session keys are keys communicated to principals in tickets, whereas conversation keys are established dynamically by the RPC run-time environment to protect the data transmission. Session keys are used in the establishment of communication keys.

## Authentication Protocol

A simplified illustration of the authentication protocol is shown in Fig. 2. The circled numbers in this section correspond to the circled numbers in Fig. 2.

At the start of a user login sequence the computer establishes a session with the DCE security service. The user's password is transformed into a secret key ①. The client system has a file containing a machine TGT and a machine session key. Knowledge about the machine session key and

†NIS, or Network Information Services, is a product from Sun Microsystems.



Client System
dced Process
Machine TGT
Machine TGT
Login Program
DCE Security Library
User: Mary
Password=h1ll
h1ll
Machine TGT
Time Stamp
TGT
TGT
Security Server System
Registry
Mary
Security Server
Machine TGT
Time Stamp
Time Stamp
TGT
Machine Session Key
User Secret Key (Created from Password)
Conversation Key from dced
Conversation Key from the Security Library
Client Session Key
Internally Generated Keys
TGT = Ticket-Granting Ticket
Data item encrypted first with key x and then with key y. To get to the data item, the token must be decrypted in reverse order (i.e., first key y and then key x).

**Fig. 2.** Creation of a ticket-granting ticket (TGT) via the authentication protocol..

the user secret key is shared between the client system and the DCE server system (see keys a and b in Fig. 2).

The protocol used for authentication is known as the DCE third-party preauthentication protocol. The protocol starts with the DCE security library requesting, on behalf of a login utility, a conversation key and a machine TGT from the DCE daemon, dced ②. Dced provides the first conversation key and the machine TGT along with a copy of the conversation key encrypted with the machine session key ③. The security library then generates a token containing a time stamp and a second conversation key. The library encrypts that token twice: once with the key derived from the user password and once with the first conversation key ④. This encrypted token is passed to the DCE security server along with the machine TGT and the encrypted conversation key received from the dced process ⑤.

Upon receipt of the token and other items, the DCE security server decrypts the first conversation key using the machine session key ⑥. It then decrypts the token containing the time stamp and the second conversation key using the first conversation key ⑦. Next, the token is decrypted using the user's secret key stored in the registry database ⑧. If the time stamp is within acceptable limits, the DCE security server creates a token containing a TGT and a client session key ⑨. The security server passes the token back to the client encrypted with the second conversation key ⑩. The client decrypts the token, validates its content, and stores the TGT and the client session key in the login context for use in future requests for service tickets ⑪.

At this point the user and the DCE security server are mutually authenticated. Note that the user's secret key was never sent (in plain or ciphered format) to the DCE security server. Proof that the user knows the correct password is verified by the fact that the time stamp is successfully decoded by the DCE security server.

**Privilege Service.** The TGT described above does not contain the information necessary for the advanced authorization mechanisms such as groups and ERAs. The privilege service provides this information by creating an EPAC and a privilege TGT, which contains the TGT and a seal (checksum) of the EPAC.

When an authenticated RPC is attempted and a valid privilege TGT is not available, the privilege service is contacted by the security library. First the library obtains a service ticket for the privilege service in a manner similar to what is described below, but using a TGT instead of the privilege TGT.

The privilege service then prepares the extended privilege attribute certificate, creates the privilege TGT, and communicates it back to the client. Application servers will be able to request the EPAC through the RPC run-time environment.

**Secure Communication.** The authentication and key exchange protocol needed to establish a secure communication channel between a client and its associated server is transparent to the application. The RPC and GSS API facilities and the DCE security service library cooperate in establishing a secure communication channel.

Fig. 3 is a simplified† representation of the sequence of events for establishing a protected RPC communication channel, assuming a valid privilege TGT has already been established by the privilege service as described above. The circled numbers in Fig. 3 correspond to the circled numbers in this section. First, the application client makes a request to the application server by calling an RPC stub ①.

Since the application client needs a service ticket to authenticate itself to the application server, the security library generates a request to get a ticket and a conversation key from the security server. This results in the creation of a token containing the request for the ticket and the privilege TGT encrypted by the client session key learned during the login sequence. The token is sent to the key distribution center (KDC) which is in the security server ②.

The KDC decrypts and validates the request and then generates a conversation key for use between the application client and the application server. It encrypts the conversation key and the authentication information (in the service ticket) with the secret key it shares with the application server ③. It attaches another copy of the conversation key to the service ticket and encrypts the whole structure with the client session key ④. This token is then sent to the application client system ⑤, which decrypts it and learns the conversation key ⑥.

RPC then encrypts the RPC request with the conversation key ⑦ and sends it to the application server. The application server learns the conversation key and checks the client's authenticity. To accomplish this, the application server sends a challenge, which is just a random number ⑧. The client receives this challenge and replies by sending a token containing the encrypted challenge and the encrypted service ticket and conversation key obtained from the security server ⑨. The server decrypts the ticket and obtains the client privileges and the conversation key ⑩. It decrypts the challenge with this conversation key ⑪, and if it matches what is sent, the authenticity of the client is assumed. It then proceeds to decrypt the request from the client ⑫. The client and server now share a secret conversation key.

## Additional Functionality

### Extended Registry Attributes

The DCE registry contains principal account data in a well-defined format (i.e., a static schema). Every account record contains the same number and types of data fields, all targeted to meet the requirements of either DCE security or UNIX platform security. To support integration with other platforms and security systems, the DCE registry needed a way to store non-DCE or non-UNIX security data for principals. To meet this need, the DCE registry was augmented with a dynamic schema facility called the extended registry attribute (ERA) facility, which supports the definition of new types of data fields called attribute types and the assignment of specific values for those attribute types to principals and other registry objects like groups and organizations.

In the ERA schema, administrators define new attribute types by specifying a unique attribute name (e.g., X.500_Distinguished_Name), the appropriate data type (e.g., string), the type of registry object (e.g., principal) that supports attributes of this type, and other related information. Once the attribute type has been defined in the schema, an administrator can attach an instance of that attribute type to any registry object that supports it. For example, an attribute instance whose type is X.500_Distinguished_Name and whose value is /C=US/o=HP/OU=OSSD/G=JOE/S=KING could be attached to the principal Joe.†† From then on applications that require knowledge of Joe's X.500 distinguished name could query the registry for that attribute type on the principal Joe.

In some cases, attribute values of a certain type are more appropriately created and maintained outside of the DCE registry. These could include attributes that are already maintained in a preexisting legacy database or attributes whose values differ depending on discriminating factors such as time of day or operation to be invoked. The ERA trigger facility supports cases such as these by providing an automatic trigger (or callout) to a remote trigger server that maintains the attributes of interest. For example, if the registry receives a

† In particular, the conversation key is established in more steps than shown, and the protocol implements caching so as not to require all steps to be executed every time.

†† See the article on page 23 for an explanation of the fields in this string.

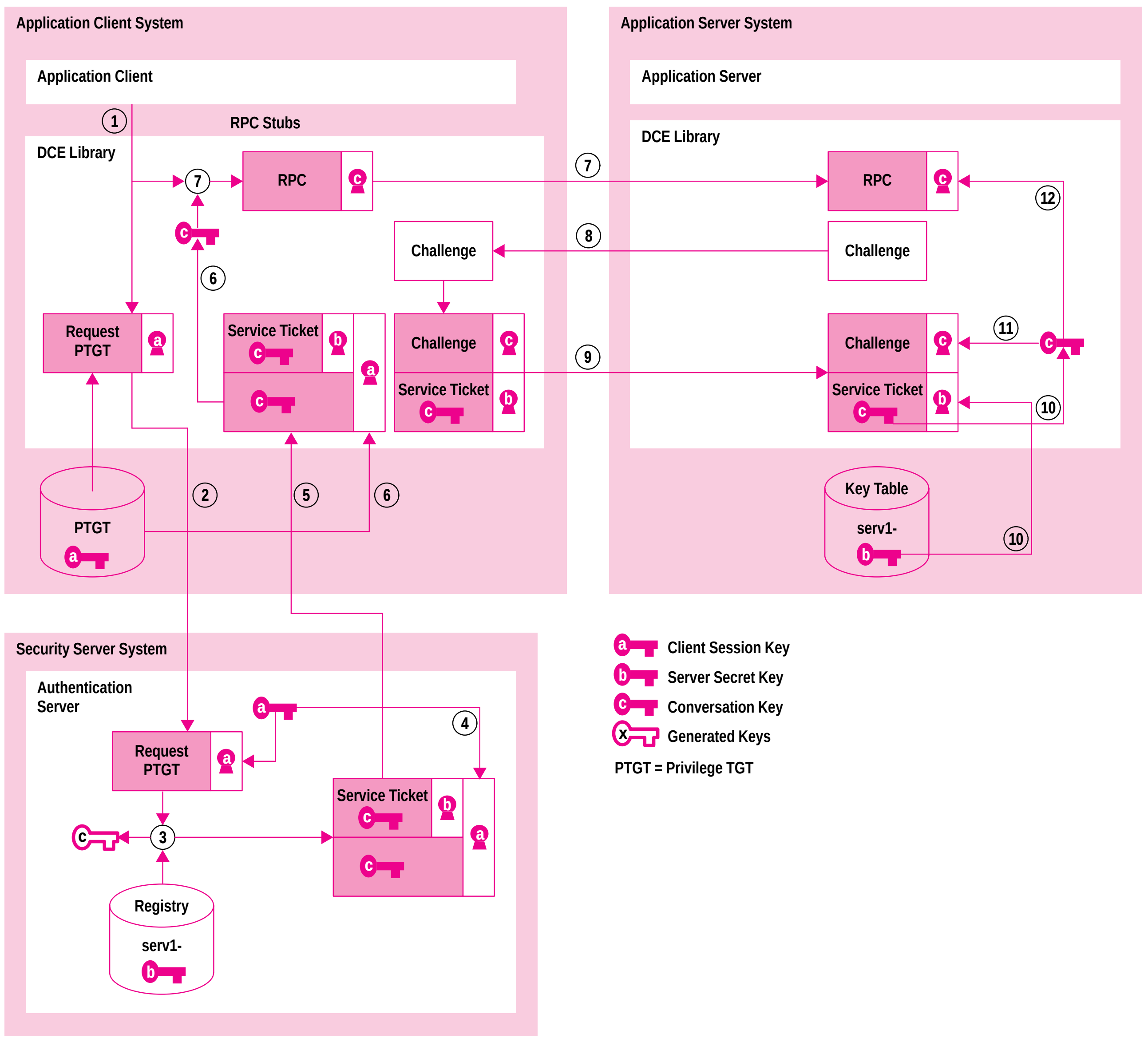


**Fig. 3.** Setting up a secure communication between a client and server.

query for a particular attribute type that is marked as a trigger, the registry forwards the query to a preconfigured trigger server. The server will return the appropriate attribute value to the registry, which will then respond to the original query with this value. A query for a trigger attribute may include input data required by the trigger server to determine the appropriate attribute value to return. Trigger servers are not provided as part of the DCE package; they are provided by third-party integrators of security systems. The ERA trigger facility provides the rules, interfaces, and mechanisms for integrating trigger servers with the DCE security service.

Some application servers need to make decisions, especially authorization decisions, based on the calling principal's attribute values. The DCE privilege service supports this by providing a way for applications to request that specific attributes be included in a principal's EPAC. As described earlier in the "Identification and Authentication" section, the RPC run-time environment supports queries for obtaining the calling principal's EPAC. This enables application servers to base decisions on the caller's attribute values and the identity and groupset information in the EPAC.

## Delegation

In a distributed environment, an application server processing a client request may have to make a request on its own to another server to complete the client request. We will call the application server with the request an intermediate server. The identity reported by the intermediate server to the server it contacts can be either its own identity or the identity of the client that made the original request. This latter case is called delegation because the intermediate server acts as a delegate of the client. A delegation chain is built as intermediate servers call other intermediate servers.[6]

For delegation to be possible, the client has to enable this feature explicitly. Two types of delegation are available:

- Traced delegation in which the identity and privileges of each intermediary are kept and can be used for access control
- Impersonation in which only the originator's identity and privileges are carried in the extended privilege attribute certificate.

ACLs have been extended to support delegation, making it possible to grant access based not only on the originator of the request, but also on the intermediaries. This allows administrators to grant access to servers acting as delegates on behalf of particular originators without granting access to the same servers operating on their own behalf.

**Compatibility with Kerberos**

The authentication service provided in the DCE security is derived from Kerberos version 5.[1] The protocol used between a client and server using the DCE security service is the native Kerberos protocol and has been adapted for RPC transport.

DCE security supports Kerberos version 5 clients (e.g., a telnet, or a terminal server that uses Kerberos version 5). This removes the need to manage a separate Kerberos realm because DCE security supports the registration and authentication of Kerberos principals.

DCE security also provides an API that can be used to promote Kerberos credentials that have been forwarded to a DCE client into full DCE credentials. Full DCE credentials represent an authenticated DCE principal, thereby enabling use of DCE services.

**Auditing**

DCE offers an auditing service that is part of DCE security. The DCE security and time services use auditing to record security-relevant events like account creation, ticket granting, and system time changes.

DCE auditing is controlled by the DCE control program, with which DCE administrators can select the events to audit and control the operation of the audit subsystem.

**Authenticated RPC**

The DCE remote procedure call (RPC) facility is described in more detail in the article on page 6. The RPC facility is integrated with the DCE security service and is referred to as the authenticated RPC run-time environment.

When an application client wants to make a protected remote call, it calls the authenticated RPC run-time environment to select:

- The authentication service, which can be either no authentication or secret key authentication
- The protection level, which specifies whether authentication should occur only at the beginning of an RPC session or at each message or packet and whether message data should be integrity or confidentially protected
- The authorization service, which can be name-based, in which case only the name of the caller is known to the server, or privilege-based, in which case all the privileges of the client, in the form of an EPAC, are made available to the server for authorization.

The application developer can trade off the resources consumed by an application with the level of security required.

**Generic Security Service API**

The GSS API improves application portability by reducing security-mechanism-specific code. It also provides transport independence since the data protection is not tied to a particular communication mechanism (e.g., DCE RPC). GSS API calls are used to authenticate and establish a security context between communicating peers and to protect blocks of data cryptographically for transmission between them. The data protection includes data origin certification, integrity, and optionally, confidentiality.

The GSS API supports many different underlying security mechanisms. The GSS API implementation provided with DCE supports both the DCE and the Kerberos version 5 mechanisms.

**Security Run-Time Environment**

Applications can access security functions directly through the security library, which is part of the DCE library on the HP-UX operating system. The security library provides APIs to make access decisions based on ACLs, manage key tables, query and update registry data, login and establish credentials, and so on.

System administrators and users can use a series of commands to administer the security service or manage their local security resources such as credentials, ACLs, or key tables. Most of the administrative commands are part of the DCE control program.

**Multicell Configurations**

In large enterprise networks, it is often impractical or undesirable to configure a single cell. For this reason, DCE features intercell communication mechanisms. See the article on page 6 for a brief description of cells.

The DCE security service is an actor in this intercell environment. Through a mechanism of key exchange, a relationship of trust can be established between two cells. When an application client wants to communicate with a server in a foreign cell, it must obtain a service ticket for that server. To do so, the DCE security service automatically generates a foreign privilege TGT, which contains the privilege information about the principal (application client) in its local cell encrypted using the foreign cell's keys. This key, shared between the two cells, is used to authenticate and secure this protocol. The DCE security service then proceeds to get a service ticket to the foreign server by contacting the foreign authentication service as it would do for the local cell by using the foreign privilege TGT instead of the privilege TGT used in the example given earlier.

ACLs support the intercell operations by allowing foreign users, groups, and others to be granted permissions.

**High Availability**

The DCE security service is an essential piece of the distributed computing environment. Thus, the security service must stay operational around the clock even when systems are down or network connections are unavailable, which could happen frequently in wide area network environments.

For this purpose, the DCE security service features a server replication mechanism. The master replica is the only one that can accept requests for updates such as password changes or account modifications. These modifications are sent securely to slave replicas, which contain a duplicate image of the registry database, but support only query, not update operations. The use of slave replicas improves performance in busy environments since additional DCE security servers are available to process queries and requests for secure communication.

The DCE security service administrative commands allow the role of master to be moved between replicas. In case the machine hosting the master is not available for some time, the administrator can force a slave to become the master.

In the rare case in which no network connection is available to reach a DCE security server, the DCE security login client will use a local cache of credentials that have been granted recently to perform authentication. However, the credentials usually cannot be used to obtain service tickets.

**System Security Requirements**

The use of the DCE security service alone does not guarantee a secure distributed computing environment. The security service relies on protection features offered by the local operating system to store its data and credentials.

The systems hosting a DCE security server must be protected from unauthorized access. They should be placed in a secure area, such as a locked room, and be given the highest security considerations. In particular, certain network services should be disabled and a limited number of users should be given access. This security is required because the DCE security server holds the keys to all the principals in the enterprise.

The systems hosting the application servers should also be managed with care, mainly to protect the enterprise data, which is often not protected by the DCE security service.

Application clients do not need such stringent management guidelines. On multiuser systems, the user environment should be partitioned so that one user cannot steal the credentials of another active user, which could be done by reading the other user's credential files.

The DCE security service does not guarantee that there are no undetected intruders in the system. It offers no protection if the program used for login has been modified to steal the password, saving it for future retrieval by an intruder.

If a system other than one hosting a DCE security server is compromised, only the application servers residing on that system and the users who performed a login on that system during the period of compromise are affected. The overall distributed computing environment protected by the DCE security service is not affected. This is because the keys are known only by the owner (server, machine, or application) and the DCE security servers, and they are never communicated to a third party.

**Acknowledgments**

The DCE security service was developed by Hewlett-Packard for the Open Software Foundation as part of the Distributed Computing Environment project. The DCE security service finds its roots in the Domain operating system, the Network Computing System (NCS), and the Athena project at the Massachusetts Institute of Technology, which created Kerberos. Over the years, many different engineers and managers, too numerous to recognize by name, have worked on the product. The DCE security service continues to evolve as new services are added through the prestructured technology program under the auspices of the OSF.

The authors would like to thank Sue Kline, Mike Kong, Sean Mullan, Joe Pato, Bill Sommerfeld, and Rob Stanzel for their contributions to this article.

**References**

1. J. Kohl, et al., *The Kerberos Network Authentication Service*, Version 5, RFC-1510, September 1993.
2. *Generic Security Service API*, Preliminary Specification P308, X/Open Company Ltd., January 1994.
3. *Data Encryption Standard*, NBS FIPS PUB 46-1, National Bureau of Standards, U. S. Department of Commerce, January 1988.
4. R. Rivest, *The MD5 Message Digest Algorithm*, RFC-1321, April 1992.
5. *Error-Correcting Procedures for DCEs Using Asynchronous-to-Synchronous Conversions*, Recommendation V.42, CCITT, 1988.
6. M. Erdos and J. Pato, "Extending the OSF DCE Authorization System to Support Practical Delegation," *Proceedings, Privacy and Security Research Group Workshop on Network and Distributed System Security,* February 1993.