Draft EN 301 099-1 V1.1.1 (1997-10)

European Standard (Telecommunications series)

Telecommunications Security; Trusted Third Parties (TTP); Specification for TTP services; Part 1: Key management and key escrow/recovery

European Telecommunications Standards Institute

Reference

DEN/SEC-003001-1 (a6c90ico.PDF)

Keywords

ISDN, multimedia, security

ETSI Secretariat

Postal address

F-06921 Sophia Antipolis Cedex - FRANCE

Office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

X.400

c= fr; a=atlas; p=etsi; s=secretariat

Internet

secretariat@etsi.fr
http://www.etsi.fr

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 1997.
All rights reserved.

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETR 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available **free of charge** from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.fr/ipr).

Pursuant to the ETSI Interim IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETR 314 (or the updates on http://www.etsi.fr/ipr) which are, or may be, or may become, essential to the present document.

Foreword

This European Standard (telecommunications series) has been produced by ETSI Technical Committee Telecommunications Security (SEC), and is now submitted for the ETSI standards One-step Approval Procedure (OAP).

The present document is part 1 of a multi-part EN covering the specification for Trusted Third Parties (TTP) services, as identified below:

Part 1: "Key management and key escrow/recovery";

Part 2: "Key management without key escrow/recovery";

Part 3: "Authentication and access control services";

Part 4: "Non-repudation services".

Proposed national transposition dates	
Date of latest announcement of this EN (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

Introduction

Achieving the appropriate level of user confidence in the application of IT systems for processing and communicating information is closely related to the need for practical and appropriate technical and legal controls to protect this information. Users across a wide range of industrial sectors and user communities need to have confidence that such systems can be relied upon to support their business obligations and commitments, and to provide a level of trust in the protection of their information.

There is a need to facilitate the growing importance and development of electronic commerce, the European Information Infrastructure (EII) and the Global Information Infrastructure (GII) by the introduction of suitable measures to safeguard the integrity and confidentiality of electronic information. The provision of Trusted Third Party (TTP) services to satisfy this user need, and the requirement to be compliant with national legislation, is of major importance to establish the appropriate level of user assurance.

TTP services can be considered as value-added communication services available to users that need to enhance the trust in the services used. By signing up to a licensed or accredited TTP, the user can communicate securely with every user of every TTP with whom his TTP has an agreement (or shares a common root in the case of a hierarchical infrastructure of TTPs). Therefore, TTPs could be able to offer value with regard to integrity and confidentiality of the electronic information being carried by these communications. A TTP has been defined by ISO/IEC as a security authority or its agent, trusted by users with respect to security-related activities, e.g. to support the use of digital signatures and confidentiality services.

Standardization of TTPs is a large and complex task covering a range of different, but related, services and supporting infrastructure. Given that it is not possible to do everything at once, it is necessary to decide upon a logical starting point and adopt a structured set of evolutionary steps leading to the required final position. Two alternative approaches stand out - structure the standards by service, or by infrastructure. It was decided here to focus first upon infrastructure then to discuss the services that can be realized within this infrastructure.

The principle motivation for this approach is that it relates more closely to the way in which a service provider would actually carry out a commercially phased roll out. The infrastructure would be put in place to support a set of fundamental services. Further services could then be designed, tested and rolled out, using the infrastructure as a framework and the fundamental service set as independent building blocks, whilst adding value. The infrastructure provides the framework for a set of interoperable services for global trading and electronic commerce.

Accordingly, it was decided to focus attention upon a TTP key management standard based on an infrastructural approach, in particular using PKI components on the grounds that it would underpin a good number of all TTP services.

The specification defined in this standard covers the TTP services of key management and key escrow/recovery to support user requirements for confidentiality and integrity related services, and is based on publicly available specifications, standards and definitions. These user requirements are as identified in EG 201 057 [1], which includes the requirements for lawful access in accordance with that given in ETR 331 [2] and the International User Requirements given in the Resolution of the Council of the European Union on the Lawful Interception of telecommunications [17].

It is the intention that any implementation conforming to this specification should not infringe any known patents or other intellectual or industrial property rights.

1 Scope

The present document gives a functional and interface specification for the following TTP services:

- key management; and
- key escrow/recovery.

The present document covers services for both symmetric and asymmetric cryptographic systems. The present document can be used to support communications and storage based applications having confidentiality and/or authenticity requirements.

All functions and services specified in the present document are optional but if any of them are used then they should be implemented in the way described in the present document.

The present document is Part 1 of a set of TTP specifications covering the requirements presented in the EG 201 057 [1].

2 References

References may be made to:

a) specific versions of publications (identified by date of publication, edition number, version number, etc.), in which case, subsequent revisions to the referenced document do not apply; or

b) all versions up to and including the identified version (identified by "up to and including" before the version identity); or

c) all versions subsequent to and including the identified version (identified by "onwards" following the version identity); or

d) publications without mention of a specific version, in which case the latest version applies.

A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

2.1 Normative references

[1] EG 201 057, V1.1.2: "Telecommunications Security; Trusted Third Parties (TTP); Requirements for TTP services".

[2] ETR 331: "Security Techniques Advisory Group (STAG); Definition of user requirements for lawful interception of telecommunications; Requirements of the law enforcement agencies".

2.2 Informative references

[3] IETF Internet-Draft (July 1997): "Internet Public Key Infrastructure Part I: X.509 Certificate and CRL Profile".

[4] IETF Internet-Draft (September 1997): "Internet Public Key Infrastructure Operational Protocols - LDAP".

[5] IETF Internet Draft (September 1997): "Internet Public Key Infrastructure Certificate Management Protocols".

NOTE: Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts shadow directory: nic.nordu.net (FTP path: ftp://ftp.nordu.net/internet-drafts/1id-abstracts.txt).

[6] ISO/IEC JTC1/SC27 N1360 (May 1996): "Guidelines for the use and management of Trusted Third Party Services - Part 2 : Technical Aspects" (Working Draft ISO/IEC 14516-2).

[7] ISO 7498-2: "Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture".

[8] ISO/IEC 10181-2: "Information technology - Open Systems Interconnection - Security frameworks for open systems: Authentication framework".

[9] ISO/IEC 10646-1: "Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane"

[10] ISO/IEC 11770-1: "Information technology - Security techniques - Key Management - Part 1: Framework".

[11] ISO/IEC 11770-2: "Information technology - Security techniques - Key management - Part 2: Mechanisms using symmetric techniques".

[12] ISO/IEC 11770-3: "Information technology - Security techniques - Key Management - Part 3: Mechanisms using asymmetric techniques".

[13] ISO/IEC 9798-1: "Information technology - Security techniques - Entity authentication - Part 1: General".

[14] ITU-T Recommendation X.509 | ISO/IEC 9594-8: "Information Technology - Open Systems Interconnection - The Directory: Authentication Framework".

[15] JMW scheme, Jefferies N, Mitchell C and Walker M: "A proposed architecture for Trusted Third Party Services", In Dawson E and Golic J (Eds.), Lecture Notes in Computer Science 1029, Cryptography; Policy and Algorithms Conference pp 98-104, Springer Verlag, 1996.

[16] Diffie W and Hellman M: "New directions in cryptography", IEE Transactions on Information Theory, 22:644-655, 1976.

[17] Resolution of the Council of the European Union: "International Requirements on the Lawful Interception of telecommunications", 17 January 1995.

3 Abbreviations and definitions

3.1 Abbreviations

AE	Authorized Entity
A-I	Application Interface
BER	Basic Encoding Rules
CA	Certification Authority
CRL	Certificate Revocation List
DN	Distinguished Name
I&A	Identification and Authentication
IT	Information Technology
LEA	Law Enforcement Agency
KER-I	Key Escrow/Recovery Interface
MAC	Message Authentication Code
PIN	Personal Identification Number
PKI	Public Key Infrastructure
T-I	Inter-TTP Interface

TTP	Trusted Third Party
U-I	User Interface

3.2 Definitions

access control: The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner (ISO 7498-2 [7]).

asymmetric cryptographic technique: A cryptographic technique that uses two related transformations, a public transformation (defined by a public key) and a private transformation (defined by a private key). The two transformations have the property that, given the public transformation, it is computationally infeasible to derive the private transformation (ISO/IEC 11770-1 [10]).

NOTE 1: A system based on asymmetric cryptographic techniques can either be an encipherment system, a signature system, a combined encipherment and signature system, or a key management system. With asymmetric cryptosystems there are four elementary transformations: sign and verify for signature schemes, encipher and decipher for encipherment systems. The signature and decipherment transformation are kept private by the owning entity, whereas the corresponding verification and encipherment transformation are published.

NOTE 2: If the same cryptosystem is used for different services, then keys used for confidentiality services should not be used for integrity services.

authentication: The provision of assurance of the claimed identity of an entity (ISO/IEC 10181-2 [8]).

certificate: An attribute certificate, public key certificate or privilege attribute certificate.

Certification Authority (CA): An authority (e.g. a TTP) trusted by one or more users to create and assign certificates. Optionally the certification authority may create the user's keys (ISO/IEC 14516-2 [6]).

Certificate Revocation List (CRL): A list of certificates which are no longer valid. A CRL is generated and distributed by a TTP. (ISO/IEC 14516-2 [6]).

digital signature: Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery, e.g. by the recipient (ISO 7498-2 [7]).

key distribution: This is a set of procedures to provide key management information objects securely to authorized entities (ISO/IEC 11770-1 [10]).

key escrow/recovery system: A key escrow/recovery system is an encryption system with a backup decryption capability that allows authorized persons (users, officers of an organization or law enforcement authorities) under certain prescribed conditions, to decrypt ciphertext with the help of key escrow/recovery information supplied by one or more trusted parties who hold special data recovery keys.

key management: The administration and use of the generation, registration, certification, deregistration, distribution, installation, storage, archiving, revocation, derivation and destruction of keying material in accordance with a security policy (ISO/IEC 11770-1 [10]).

Law Enforcement Agency (LEA): An organization authorized by a lawful authorization, based on a national law, to receive the results of telecommunication interceptions (ETR 331 [2]).

lawful authorization: Permission granted to an LEA under certain conditions to intercept specified telecommunications and requiring co-operation for a network operator or service provider. Typically this refers to a warrant or order issued by a lawfully authorized body (ETR 331 [2]).

lawful access: The action by some entity of gaining access to certain stored information which it is legally entitled to obtain under criminal, civil, private or business law.

lawful interception: The action (based on the law), performed by a network operator or service provider, of making available certain information and providing that information to a Law Enforcement Monitoring Facility (ETR 331 [2]).

private key: That key of an entity's asymmetric key pair which should only be used by that entity (ISO/IEC 9798-1 [13], ISO/IEC 11770-1 [10] and ISO/IEC 11770-3 [12]).

public key: That key of an entity's asymmetric key pair which can be made public (ISO/IEC 9798-1 [13], ISO/IEC 11770-1 [10] and ISO/IEC 11770-3 [12]).

public key certificate: Public key information of an entity signed by the certification authority and thereby rendered unforgeable (ISO/IEC 9798-1 [13], ISO/IEC 11770-1 [10] and ISO/IEC 11770-3 [12]).

PKI registration authority: An entity which acts as a go between for a subject requiring its key to be certified and the certification authority.

secret key: A key used with symmetric cryptographic techniques and usable only by a set of specified entities (ISO/IEC 11770-1 [10] and ISO/IEC 11770-3 [12]).

symmetric cryptographic technique: A cryptographic technique that uses the same secret key for both the originator's and the recipient's transformation. Without knowledge of the secret key, it is computationally infeasible to compute either the originator's or the recipient's transformation (ISO/IEC 9798-1 [13] and ISO/IEC 11770-1 [10]).

4 Architecture for TTP services

4.1 Design requirements

The following requirements for key management and key escrow/recovery services to be offered by a TTP are used as the basis for this standard. These requirements have been derived from clause A.1 of EG 201 057 [1] and these are dependent on the services a TTP might provide.

User benefits and secure communications

Any service may be offered to users on a commercial basis. Any key management service offered by a TTP should be capable of enabling the users of this TTP to communicate securely with every user of every TTP with whom their TTP has an agreement.

Protection of data in storage

The services offered by a TTP may support the protection of data in storage, and if so. it should include:

- the distribution of cryptographic keys that can be used to protect the confidentiality and authenticity of data in storage;
- the ability for lawful access to cryptographic keys to be used for protecting the confidentiality of data in storage; and
- the ability for user/business access to cryptographic keys to be used for protecting data in storage.

National and international operation

Any key management or key escrow/recovery service offered by a TTP should allow national and international operation. The services offered should comply with the different laws and regulations of the relevant countries.

Electronic communications

The TTP key management and key escrow/recovery services should not restrict the form of electronic communication which is used.

Security techniques

The key management and key escrow/recovery services offered by a TTP should be based on well known and publicly available security techniques and specifications. In addition, this should support a variety of encryption algorithms, both in hardware and software.

Laws and regulations

Any TTP key management and key escrow/recovery services should comply with the different relevant laws and regulations of participating countries, concerning data protection, lawful interception, and lawful access as well as the use, export and sale of cryptographic products and services. Some countries may mandate lawful interception while some others may not.

Protection of communications and access to communications

- the distribution of cryptographic keys that can be used to protect the confidentiality and authenticity of data being communicated;
- lawful interception: In case that lawful authorization is given, a TTP should provide information to enable access to a user's incoming and outgoing communications without prior knowledge of other parties involved (unless national laws so require). In the case that the TTP has encrypted the data, the TTP shall provide the data en clair;
- lawful access: In the case that lawful authorization is given and a user has encrypted and stored the data, a TTP may provide key escrow/recovery information to support the lawful access to stored data. In the case that the TTP has encrypted the data, the TTP shall provide the data en clair.

Protection against false evidence

Those with lawful authorization to use key management and key escrow/recovery services to enable access to information should not be able to fabricate false evidence.

Updating keys

The key management and key escrow/recovery services offered should, as far as possible, enable users to update their keys according to the requirements of their own security policies.

Detecting abuse

The key management and key escrow/recovery services should incorporate procedures to ensure that any attempted abuse by users of these services can be detected.

Time Limits

Key escrow/recovery services should incorporate technical and/or procedural means to enforce time limits for access to the appropriate decryption keys.

Communications with TTPs

The key management and key escrow/recovery services should work in a way that users should not have to communicate with TTPs other than their own.

On-line communications

The key management and key escrow/recovery services should work in a way that on-line communication between TTPs should not be required.

Interfaces and protocols

The interfaces and protocols used to support key management and key escrow/recovery services should include the required security functions to protect the interactions among the entities involved and to support secure interworking.

4.2 Services

4.2.1 Key management services

TTP key management services can include provision of:

- generation and updating of keys;
- public-key certification;
- distribution of keys;
- distribution of public-key certificates;

- key archiving facilities;
- key storage/retrieval facilities;
- key/certificate revocation facilities.

These services can be used to support cryptographic based security services e.g. confidentiality services or authentication and integrity services based on the use of digital signatures.

4.2.2 Key escrow/recovery services

A TTP may be required to provide a backup decryption capability that allows authorized persons, under certain prescribed conditions, to decrypt ciphertext with the help of key escrow/recovery information supplied by one or more trusted parties. A TTP might combine the roles of a key distribution agent for its clients, and a supplier of user keys (under warrant) to an interception agency. In addition, key escrow/recovery may provide the user with a form of disaster recovery where lost or corrupted cryptographic keys can be retrieved.

4.3 Functions and operations

4.3.1 Functions

4.3.1.1 User registration

Certain information regarding the user needs to be stored by the TTP. It may be necessary for the user to present physical evidence of identity to the TTP. User registration may involve the generation of symmetric or asymmetric keys to be used for authentication or encrypting communications between the TTP and user.

4.3.1.2 Key generation

This may involve the generation of symmetric or asymmetric keys, or both for a range of users in different commercial environments.

4.3.1.3 Key distribution

A TTP may be required to distribute keys (generated either by some external entity or by the TTP itself) for subsequent use by entities, e.g. business users, service providers, network operators, or other TTPs.

4.3.1.4 Public-key certification

A TTP can be used to protect the integrity of published information. Typical information that may be protected might include public keys for asymmetric cryptography and privilege attributes for access control management. The information is published in the form of a certificate which contains the information itself together with other essential information such as the ID of the owner, the certificate's expiry date and the ID of the TTP that issued it. To provide integrity protection the certificate has a digital signature appended, which is computed on the contents of the certificate using the TTP's private certification key. Provided that all participants have an authentic copy of the TTP's public certification key, they can check the integrity of the issued certificates at any time by checking the validity of the signature on the certificate.

4.3.1.5 Key archiving

A TTP may be responsible for archiving cryptographic keys that it has generated or distributed. In addition, a TTP may be requested to archive keys generated by a user.

4.3.1.6 Key storage/retrieval

A TTP may be required to provide facilities for the storage and retrieval of keys.

4.3.1.7 Key/certificate revocation

Keys and certificates may become invalid after a certain point in time, e.g. if the key has been compromized or if the security policy states that all keys are changed after a specific time.

Certificates generally contain an expiry date after which they no longer guarantee the authenticity of the certified information. This feature is provided as a safeguard against cryptanalysis. Before the expiry date the certificate shall be revoked by the TTP and a new one generated and issued. The TTP may also have to revoke certificates before the expiry date as a result of alarms created by alert management functions. This means that a list of revoked certificates needs to be published, so that users can check the validity of certificates before using them.

It is the responsibility of users to check current revocation lists in order to guarantee the validity of a certificate. The Certificate Revocation List (CRL) shall be maintained by the TTP. It shall include a date and time that the list has been updated so that any entity requiring it can be sure that it is up-to-date (the CRL shall also have its own signature, so that its integrity can be checked).

4.3.1.8 Claimant and verifier functions

In its role as an in-line authentication server, a TTP may act as a claimant or a verifier in the authentication scheme. Therefore it shall be able to provide the functionality for taking on these roles.

4.3.1.9 Time stamping functions

A TTP may be required to certify that it has affixed a signature to a data item at a particular time. This is an important function in non-repudiation services.

4.3.2 Operations

The following types of internal operation are required to support the functions identified in subclause 4.2.1:

NOTE 1: It is not intended that all TTPs should provide all these operations. The selection of functions to be supported by a TTP determine which types of operation it needs to be capable of performing.

O1 Cryptographic key generation:

- secret keys for symmetric algorithms;
- public/private key pairs for asymmetric algorithms;

O2 Certificate generation and updating:

- assembling information to be certified into an appropriate form and then adding the TTPs signature;

O3 Cryptographic computation:

- encipherment/decipherment for symmetric algorithms;
- encipherment/decipherment for asymmetric algorithms;
- signature generation/verification;
- Message Authentication Code (MAC) and other (symmetric) cryptographic check function calculation;
- hash function computation;
- generation of challenge/response for authentication;
- secret sharing calculations;

O4 Cryptographic key storage:

- private signature key(s) for the TTP;
- secret keys agreed with other TTPs;

- secret keys agreed with users;
- public keys of other TTPs;
- public keys of users;

O5 User information storage:

- registration information;
- key escrow/recovery information for a user;
- account/credit information for user;
- policy information for user;
- access control information;

O6 Event information storage:

- information to support audit, and alert management functions;

O7 Time stamp generation:

- requires the TTP to maintain an accurate real-time clock to produce the time stamps.

NOTE 2: This is a limited aspect of what could be offered by a TTP time service. Parts 2 and 3 of the present document will specify more aspects of a time service.

4.4 Interfaces

4.4.1 External interfaces

A number of interfaces are needed to support the working and interoperability of a TTP, and the users and applications it serves. The following types of interfaces are required to support the functions specified in clause 5:

- user interface (U-I);
- key escrow/recovery interface (KER-I);
- inter-TTP interface (T-I).

The three interfaces can be logically split into two service interfaces (user interface and a key escrow/recovery interface for authorized entities AEs such as LEAs) over which the TTP offers its services, and a supportive interface (inter-TTP) which are used to support the services offered by the TTP.

In addition, there may be an application interface (A-I), as shown in the diagram below, which can be used to support other communications and services, e.g. security manager services and directory services. This interface is outside the scope of the present document, therefore it is not covered in the specification described in clauses 5 and 6.

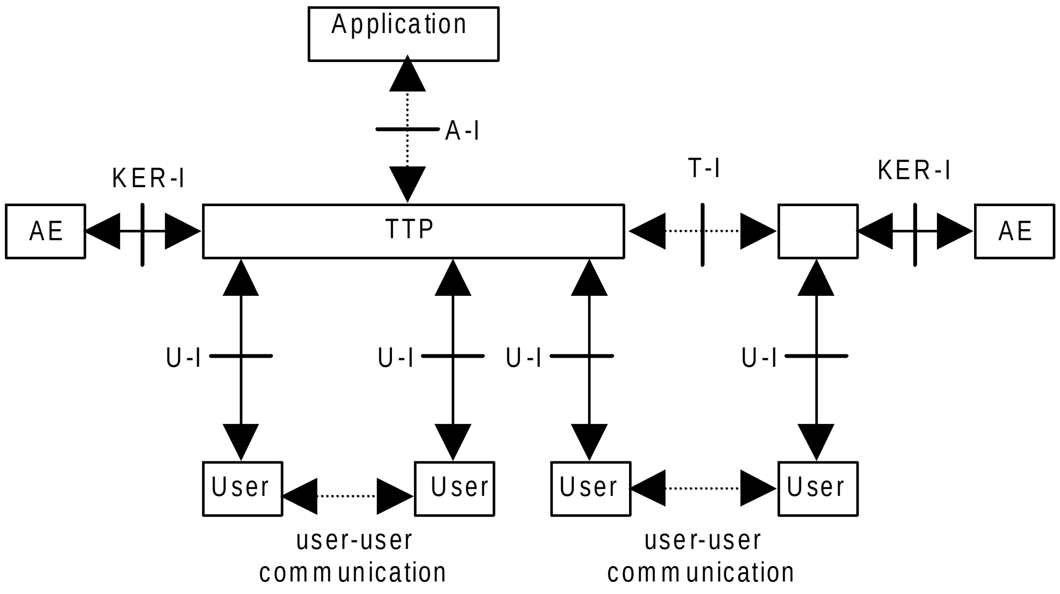

Figure 1: General interface configuration

4.4.2 Internal interfaces

The specific internal interfaces of a TTP server are implementation dependent. However, there are a number of generic interface standards such as those being developed by the Open Group which could be used as the basis for specific implementations.

The advantage of using such generic interface standards is that they aid interoperability and interworking of systems.

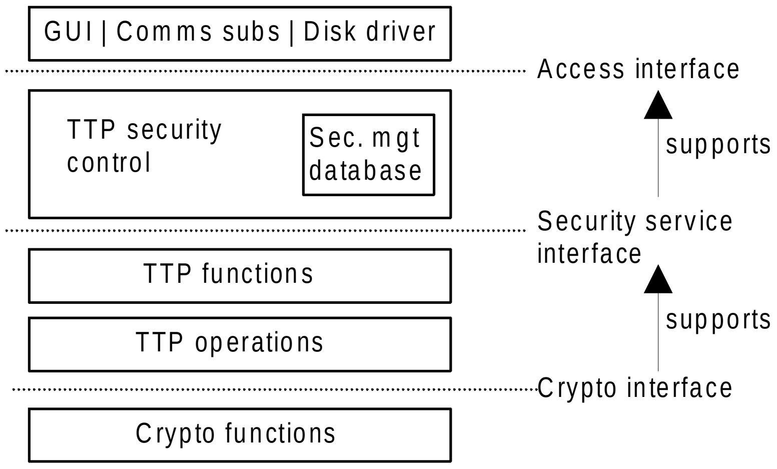

Figure 2: TTP internal interfaces

Figure 2 illustrates a generic interface architecture for a TTP server. This is introduced here for completeness of the TTP architecture and as an indication of the type of internal interfaces that will be required.

The first layer is the external communications layer. This includes the functionality required in order to allow the TTP to interact with external entities. Typically, the external communications layer includes a graphical user interface and some kind of communications interface which allows the TTP to interact with its users.

The second layer is the TTP security control layer consisting of two functional blocks: a security control block (which responds to the user's security requirements by managing a group of TTP security functions and preparing security related information using the TTP database) and a TTP database block (which securely stores three types of information (i) cryptographic keys, (ii) user information and (iii) event information).

The third layer is the TTP function and operations layer consisting of two functional blocks: TTP functions and TTP operations. This layer provides security services to the second layer across a TTP security service interface.

The fourth layer is the cryptographic function layer and provides cryptographic services to the third layer across a cryptographic interface.

4.5 Matching functions to operations and external interfaces

4.5.1 Key generation

TTPs may be required to generate keys for a variety of entities, including:

- individual users;
- network operators;
- service providers;
- TTP's own use; and
- to be shared with other TTPs.

These keys may either be asymmetric (public/private) key pairs or symmetric secret keys. When a TTP is required to generate asymmetric key pairs, care needs to be taken in the specification and design of the key generation software/hardware.

It potentially involves the following internal operations and external interfaces:

O1	cryptographic key generation;
O3	cryptographic computation;
U-I	User Interface.

4.5.2 Key distribution (on-line)

Keys for subsequent on-line distribution may be obtained by the TTP in one of four ways:

- via the user interface (e.g. for receiving user-generated keys);
- via the inter-TTP interface;
- by using the TTP's own cryptographic key generation facility; or
- from the TTP's cryptographic key storage facility.

The (on-line) key distribution process will clearly involve use of a network application. Given that this will involve keys being sent across potentially insecure network components, the cryptographic computation and cryptographic key storage facilities will almost certainly be required to protect and certify the keys being distributed.

A user information storage capability may be required to enable the TTP to work out what types of key to generate for a user, and how to charge the user for services rendered.

It will be necessary for the TTP to record its actions for audit.

Thus this TTP function will potentially involve the following internal operations and external interfaces:

O1	cryptographic key generation;
O2	certificate generation;
O3	cryptographic computation;
O4	cryptographic storage;

O5	user information storage;
O6	event information storage;
U-I	User Interface;
T-I	inter-TTP Interface.

4.5.3 Key archiving

The key archiving process will involve three main stages:

- the initiation of the archiving process for a key or keys;
- obtaining the keys to archive; and
- the archiving process itself.

The archiving process may be triggered in a number of ways, for example by the receipt of an archiving request from a user via the User Interface (U-I).

Keys for archiving may be obtained from:

- a user or other entity via the U-I;
- the TTP's user information storage facility (where user keys are stored); or
- the TTP's own cryptographic key storage (in the case where the TTP's own keys are to be archived).

The archiving process itself may involve the following processes:

- adding a time-stamp to the key information to be archived, generated using the time-stamp generation facility;
- operating cryptographically on the key information to be archived, e.g. signing it, using the cryptographic computation and cryptographic key storage facilities;
- adding other user information to the key information to be archived, obtained from the user information storage facility; and
- storing the archive information, as part of the event information storage facility.

Thus this TTP function will potentially involve the following internal operations and external interfaces:

O3	cryptographic computation;
O4	cryptographic storage;
O5	user information storage;
O6	event information storage;
O7	time-stamp generation;
U-I	User Interface;
T-I	inter-TTP interface

4.5.4 Certification functions

An "obvious" TTP function which is necessary to support the use of digital signatures is *Public Key Certificate Generation*. A widely accepted format for such certificates is specified in the 1993 version of ITU-T Recommendation X.509 [14] (known as X.509 version 2). A revised version of this specification (known as X.509 version 3, 1997) currently has draft amendment (DAM) status. The TTP responsible for generating certificates is often referred to as a *Certification Authority (CA)*. This TTP will need to have its own digital signature key pair.

The generation of a user public key certificate requires the following main steps:

1) the user identity needs to be verified;

2) the TTP needs to be supplied with a public signature verification key for the user. This may be part of a key pair generated in advance by the user, in which case the user will pass the TTP its public key, or it may be part of a key pair generated by the TTP for the user, in which case the TTP also needs to pass both parts of the key pair to the user;

3) the TTP will need to pass its public signature verification key to the user;

4) the TTP can then generate the *user certificate*, which will be a string containing the user name, user verification key, certificate expiry date, and other information, all signed using the TTP's private signature key;

5) the user certificate will then be passed to the user, as well as possibly being distributed by other means (e.g. via an X.500 directory).

Thus the certification function will potentially involve the following internal operations and external interfaces:

O2	certificate generation;
O3	cryptographic computation;
O4	cryptographic key storage;
O5	user information storage;
O7	time-stamp generation;
U-I	User Interface.

4.5.5 Revocation functions

A TTP will be responsible for regularly generating and distributing *Certificate Revocation Lists (CRLs)*, containing a list of certificates which, although not expired, are no longer valid for some reason (for example, the user private key may have been compromised). Deciding whether or not to revoke a certificate may require invoking the User information storage facility. The CRL will be a signed structure containing a time-stamp, and thus generating a CRL will require use of the Cryptographic computation, Cryptographic key storage and Time-stamp generation facilities. The CRL may be provided to a directory using the Directory interface, or may be sent directly to users using the User interface. The entire revocation function can be managed using the Security manager interface.

This function also includes the capability for authority revocation.

It will potentially involve the following internal operations and external interfaces:

O3	cryptographic computation;
O4	cryptographic key storage;
O5	user information storage;
O7	time-stamp generation;
U-I	User Interface;
T-It	inter TTP Interface.

4.5.6 Time stamping functions

For the purposes of this Part of the TTP standard the time-stamping function will involve three main stages:

- the receipt of information to be time-stamped;

- the addition of a time-stamp and a signature; and

- the transmission of the signed, time-stamped information back to the requester.

The information to be time-stamped can be received for example via the U-I (the same will apply to the transmission of "time-stamped" data). The time-stamping process itself will obviously involve use of the Time-stamp generation facility, as well as the Cryptographic computation and Cryptographic key storage facilities to generate the signature. Finally, the Certificate generation facility will be required to generate the certificate data structure for the time stamp.

Thus this TTP function will potentially involve the following internal operations and external interfaces:

O2	certificate generation;
O3	cryptographic computation;
O4	cryptographic storage;
O7	time-stamp generation;
U-I	User Interface.

5 Service and functional specifications

This clause provides a specification of key management and key escrow/recovery services and is in two parts. Subclause 5.1 specifies a key management service for confidentiality, incorporating key escrow/recovery, which uses asymmetric techniques to generate keys for symmetric cryptosystems. Subclause 5.2 specifies key management services for integrity, including the certification of public keys. It should be noted that the services specified in subclause 5.2 are distinct from those specified in subclause 5.1. A TTP may choose to offer only key management services for integrity; only key management services for confidentiality, or both key management services for integrity and key management services for confidentiality.

All messages used in this subclause are defined in clause 6.

5.1 Key management and key escrow/recovery services for confidentiality

5.1.1 Architecture

The key management and key escrow/recovery services specified in this subclause are derived from an architecture for TTP services based on the JMW scheme [15]. The architecture assumes administrative domains associated with TTPs which are responsible for registering users and supplying them with keying material for confidentiality services.

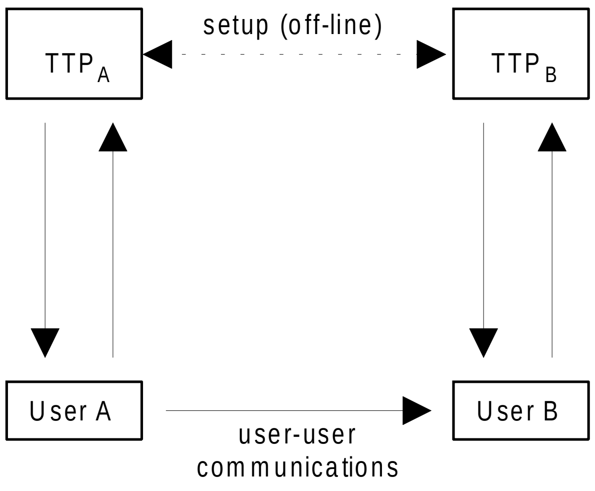

Figure 3: TTP architecture

In this architecture, a user A that requires a key in order to encrypt a transmission to another user B needs to obtain its "private send key", a certificate on the corresponding "public send key" and the "public receive key" of the receiving user from its TTP. To construct the key in order to decrypt the transmission, user B needs to obtain its "private receive key" and the means to verify the certificate on the "public send key" of user A from its TTP. The "secret send keys" are known to the user's TTP only, while the "secret receive keys" are known to both users" TTPs and typically are calculated from public information (e.g. the user's name) and a secret key set-up off-line between the two TTPs.

The key shared between the two users may be constructed using the Diffie-Hellman key agreement protocol [16]. The sending user constructs the key using its "private send key" and the "public receive key", while the receiving user constructs the key using its "private receive key" and the "public send key". Recovery of the shared secret key is possible from either the sending user's TTP or the receiving user's TTP. The sending user's TTP can calculate the key from its knowledge of either the "private send key" or the "private receive key", while the receiving user's TTP can calculate the key from its knowledge of the "private receive key".

5.1.2 Initialization

5.1.2.1 TTP to TTP initialization

Before users of one TTP can establish keys to secure exchanges with users of another TTP, keys and other information for interoperability between the TTP user domains may need to be established between the TTPs. This function may be supported by exchange mechanisms different from that used for user interactions with the TTP (e.g. physical exchange between TTP managers) and so need not necessarily be supported by on line exchange protocols.

TTP to TTP initialization involves the use of two messages:

- TTP-TTP initialization request message (**InitTTPTTPReq**);
- TTP-TTP initialization response message (**InitTTPTTPRep**).

5.1.2.2 User initialization

Before a user accesses these key management and key escrow/recovery services it is necessary to obtain information about the functions supported by the TTP and also obtain a copy of keys needed to inter-operate securely with the TTP. This function may be supported by exchange mechanisms different from that used for further interactions with the TTP (e.g. traditional - i.e. non electronic - postal services) and so need not be supported by any on line exchange protocols.

5.1.3 Key generation

A TTP might be involved in the generation or reconstruction of a key for a symmetric cryptosystem by a user for one of two purposes:

1) to encrypt a transmission to another entity;
2) to decrypt a transmission received from another entity.

The following specification uses asymmetric techniques to agree a key for a symmetric cryptosystem. In these circumstances, there are four components to the agreed key:

- the encrypting user's private component, denoted **EncPriComp**;
- the encrypting user's public component, denoted **EncPubComp**;
- the decrypting user's private component, denoted **DecPriComp**;
- the decrypting user's public component, denoted **DecPubComp**.

The encrypting user's private component should be generated by either the encrypting user or its TTP and may be random. The decrypting user's private component needs to be generated by both the encrypting user's TTP and the decrypting user's TTP, using some function agreed between the TTPs, to permit off-line working.

The encrypting and decrypting user's public components are mathematically derived from their respective private components. The agreed key can be constructed either from **EncPriComp** together with **DecPubComp**, or from **EncPubComp** together with **DecPriComp**.

5.1.3.1 Key generation for encryption

When a user requires a key for a symmetric cryptosystem in order to encrypt a transmission to another entity, the following steps occur:

1) the user sends an Encryption Information request message to its TTP (**EncInfoReq**), requesting any of: information required to generate the key a certificate for its public component of the key, and other optional information;

2) the TTP responds by sending an Encryption Information Response message (**EncInfoRep**) to the user, containing the requested data;

3) using the contents of the **EncInfoRep** message, the user generates the agreed key. This key may be used either as a session key, or as a key-encrypting key;

4) to any encrypted transmission to the other entity, the user attaches the certificate for its public component of the key, and optionally other information.

After the initial pass through all these steps to generate a key the user may not need to go through steps 1 and 2 again for further key generation. Whether the user decides to do this will depend on the specific security policy and application they are using.

The user has two options:

1) to generate its own private component and corresponding public component of the agreed key, **EncPriComp** and **EncPubComp** respectively;

2) to request that the TTP generates **EncPriComp** and **EncPubComp**.

User Generated Private Component

If the user generates its own components of the agreed key (or already holds them) then the **EncInfoReq** message it sends to its TTP may request either a certificate for its own public component, or the intended recipient's public component, or both. If the intended recipient's public component is requested then the **EncInfoReq** message shall contain the identity of the intended recipient and the identity of the TTP to which the intended recipient subscribes. The user may also submit its private component to the TTP for archiving; (in some systems the TTP might be required to archive information, in which case this would be mandatory). If the intended recipient subscribes to more than one TTP, then the user may choose which of those TTP's identities is given in the **EncInfoReq** message. If a certificate for the user's own public component is requested then the **EncInfoReq** message shall contain **EncPubComp**, unless a scheme is used where the TTP knows the method used to generate **EncPubComp** and can also generate it itself.

Upon receipt of the **EncInfoReq** message, the TTP generates the intended recipient's public component **DecPubComp** (if requested), a certificate for **EncPubComp** (if requested), and optionally a certificate for **DecPubComp**. It may also archive any information necessary to re-generate these three pieces of information at a later date. It then responds to the user with an **EncInfoRep** message containing whichever were requested of **DecPubComp** (possibly within a certificate) and the certificate for **EncPubComp**.

TTP Generated Private Component

If the user wants its TTP to generate its components of the agreed key, then the **EncInfoReq** message it sends to its TTP will act as a request for its private and public components (with the public component in a certificate). The **EncInfoReq** message may also request the intended recipient's public component. If the intended recipient's public component is requested then the **EncInfoReq** message shall contain the identity of the intended recipient and the identity of the TTP to which the intended recipient subscribes. It may also request that the TTP archives information necessary to reconstruct the user's components at a future date (in some systems the TTP might be required to archive information).

Upon receipt of the **EncInfoReq** message, the TTP generates the encrypting user's private component **EncPriComp**, the corresponding **EncPubComp**, and a certificate for **EncPubComp**. It also generates the intended recipient's corresponding public component **DecPubComp** (if requested) and optionally a certificate for **DecPubComp**. It may archive any information necessary to re-generate these pieces of information at a later date. It then responds with an **EncInfoRep** message containing **EncPriComp**, the certificate for **EncPubComp** and, if requested, **DecPubComp** (possibly within a certificate).

Keys for Multiple Recipients

If the encrypting user wishes to generate agreed keys with several other entities, one agreed key for each other entity, then the encrypting user's private component (and hence public component) could be the same for each key. In this case the **EncInfoReq** message that the user sends to its TTP contains a sequence of identities of intended recipients in place of the single identity described above. The corresponding **EncInfoRep** message then contains a sequence of decrypting user's public components in place of the single decrypting user's public component described above.

5.1.3.2 Key generation for decryption

Any encrypted transmission received by a user from another entity has a certificate attached to it containing the sender's public component **EncPubComp** for the agreed key required to decrypt the transmission. To decrypt the transmission, the user requires its private component **DecPriComp** for the agreed key. This is then combined with **EncPubComp** to yield the agreed key.

The certificate attached to the transmission contains the identity of the sender's TTP and the time and date that the certificate was created and optionally other information. Using these, the recipient may check to see if it already has a copy of the decrypting user's private component, **DecPriComp**, valid for that time for transmissions received from any user's of the sender's TTP. If so, or if the user already has sufficient information to generate **DecPriComp**, no communication with its TTP is necessary. Otherwise, the user shall communicate with its TTP to obtain the necessary **DecPriComp**, or sufficient information to generate it. The user may also request from its TTP a copy of the signature-verification key of the sender's (encrypting user's) TTP in order to validate the certificate attached to the received transmission.

When a user requires a decrypting user's private component in order to decrypt a transmission received from another entity, the following steps occur:

1) the user sends a Decryption Information Request message to its TTP (**DecInfoReq**). This requests the relevant private component (or sufficient information to generate it) and the signature-verification key of the encrypting user's TTP (if required);

2) the TTP responds by sending a Decryption Information Response message (**DecInfoRep**) to the user, containing the requested data.

The **DecInfoReq** message sent from the user to its TTP contains the identity of the TTP to which the sender of the transmission belongs (this is obtained from the certificate, attached to the transmission, for the sender's public component), and a field which specifies whether the signature-verification key of the sender's TTP is required.

Upon receipt of the **DecInfoReq** message, the TTP generates sufficient information for the user to be able to generate its private component, **DecPriComp**. It then responds with a **DecInfoRep** message containing this information (**DecPriCompInfo**), the time period for which the private component is "valid" (if **DecPriComp** itself is provided) and the signature-verification key of the sender's TTP (if requested in the **DecInfoReq** message). Note that **DecPriCompInfo** could be either **DecPriComp** itself, or information which the user can use in conjunction with other information (such as time) to generate **DecPriComp**.

Upon receipt of the **DecInfoRep** message, the user is in possession of **EncPubComp** and **DecPriComp** and so can generate the agreed key required to decrypt the received transmission. The user may choose to store the contents of the **DecInfoRep** message for later use.

5.1.3.3 Establishing keys split between TTPs

Two users may wish to establish a key for a symmetric cryptosystem which cannot be reconstructed by any one TTP (or possibly even several TTPs). This process can be viewed as the two users simply establishing several keys, using different TTPs for each key, and then combining the established keys to produce one agreed key. This agreed key is effectively "split" between TTPs (the mechanism of combining the keys can be agreed by the users; one example is producing their X-ORed product).

The mechanisms described in subclause 5.1.3 for establishing a key for a symmetric cryptosystem allow for the establishment of a key "split" between TTPs. The encrypting user can obtain several encrypting user's components from different TTPs with which the user is registered. Similarly, the encrypting user can ask one of its TTPs for several decrypting user's public components corresponding to different TTPs with which the intended recipient is registered. Thus, the encrypting user can initiate the establishment of a key "split" between several TTPs.

5.1.3.4 Key distribution

When two entities wish to agree a key for a symmetric cryptosystem, they each request information necessary to generate the key from their own TTP. Each TTP generates the necessary information and communicates it to its user as described in subclause 5.1.3.

Thus keys for symmetric cryptosystems are not actually distributed on-line, the only distribution of information is in the form of a TTP sending to one of its users a **EncInfoRep** or **DecInfoRep** message.

5.1.4 Key revocation

The revocation of a confidentiality key agreed between two parties can be presented as the revocation of either the encrypting user's private component (**EncPriComp**) or the decrypting user's private component (**DecPriComp**).

A revocation process will need to take place for an encrypting user's private component (**EncPriComp**) if the encrypting user's TTP (denoted here as TTPSEND) requires the component to be revoked. In this case, TTPSEND sends a Encryption component Revocation Announcement message (**EncRevAnn)** to the encrypting user. This message contains the encrypting user's public component which corresponds with the private component to be revoked, and the time from which the private component is to be revoked. The **EncRevAnn** message may be forwarded by the encrypting user to other entities with whom keys have been agreed using the revoked **EncPriComp** (and corresponding **EncPubComp**) if required. Clearly, the **EncRevAnn** message shall be protected for integrity (see subclause 6.1.3).

A decrypting user's private component (**DecPriComp**) is used to agree keys for all transmissions from all users of a particular TTP (denoted here as TTPSEND) within a certain period of time. The revocation of a given **DecPriComp** might be initiated by any one of: TTPSEND; the decrypting user's TTP, or the decrypting user.

5.1.4.1 Revocation initiated by TTPSEND

When a TTP (denoted here as the "revoking" TTP) needs to revoke a **DecPriComp** belonging to a given user (denoted here as the "decrypting" user) of another TTP, the following steps occur:

1) the revoking TTP sends a Decryption component Revocation Announcement message (**DecRevAnn**) to the decrypting user's TTP. This message contains the identity of the revoking TTP, the identity of the decrypting user and the time from which the **DecPriComp** is to be revoked;

2) the revoking TTP checks to see from which of its users it has received a **EncInfoReq** message containing the identity of the decrypting user. It then forwards the **DecRevAnn** message to all those from which it has received such a message within a certain time period;

3) upon receipt of the **DecRevAnn** message from the revoking TTP, the decrypting user's TTP forwards the **DecRevAnn** message to the decrypting user;

4) both the revoking TTP and the decrypting user's TTP update the method of generating components for the decrypting user according to some pre-agreed arrangement.

5.1.4.2 Revocation initiated by decrypting user's TTP

When a TTP (denoted here as the "revoking" TTP) needs to revoke a **DecPriComp** belonging to one of its own users (denoted here as the "decrypting" user) and corresponding to all transmissions from users of a TTP denoted TTPSEND (which could in fact also be the revoking TTP), the following steps occur:

1) the revoking TTP sends a **DecRevAnn** announcement to both the decrypting user and TTPSEND. This message contains the identity of the decrypting user, the identity of TTPSEND, and the time from which the **DecPriComp** is to be revoked;

2) upon receipt of the **DecRevAnn** message, TTPSEND checks to see from which of its users it has received a **EncInfoReq** message containing the identity of the decrypting user. It then forwards the **DecRevAnn** message to all those from which it has received such a message within a certain time period;

3) both TTPSEND and the decrypting user's TTP update the method of generating components for the decrypting user according to some pre-agreed arrangement.

5.1.4.3 Revocation initiated by decrypting user

When the decrypting user (denoted here as the "revoking" user) needs to revoke a **DecPriComp** belonging to itself and corresponding with all transmissions from a TTP denoted TTPSEND (which could be its own TTP), the following steps occur:

1) the revoking user sends a Decryption component Revocation Request (**DecRevReq**) message (**DecRevReq**) to its TTP. This message contains the identity of TTPSEND and the time from which the **DecPriComp** is to be revoked;

2) the revoking user's TTP sends a **DecRevAnn** message to TTPSEND. This announcement message contains the identity of the revoking user and the time from which the **DecPriComp** is to be revoked;

3) the revoking user's TTP sends a Decryption component Revocation Response message (**DecRevRep**) to the user, confirming that all keys for which the decrypting user's private component is **DecPriComp** have been revoked;

4) upon receipt of the **DecRevAnn** message, TTPSEND checks to see from which of its users it has received a **EncInfoReq** message containing the identity of the revoking user. It then forwards the **DecRevAnn** message to all those from which it has received such a message within a certain time period;

5) both the revoking user's TTP and TTPSEND update the method of generating decrypting user's components for the revoking user according to some pre-agreed arrangement.

5.1.5 Key archival

When requesting a key for a symmetric cryptosystem, a user can specify whether his private component of the key should be archived by the TTP. The user can specify this option in the **archiveOptions** field of the **EncInfoReq** message. If the user generates the private component himself, then **archiveOptions** allows him to send the private component to the TTP for archiving. However, if the TTP generates the private component, then **archiveOptions** allows the user to specify whether or not the private component should be archived by the TTP.

5.1.6 Key retrieval

There are two options for key retrieval depending on whether the user requests the key, or a private component which may be used to generate the key. A user needs to authenticate itself to the TTP and the authorization, to retrieve the required keys, needs to be checked.

NOTE: In some cases the TTP may only be permitted to release the key, rather than a private component used to generate the key.

5.1.6.1 Retrieval of a private component

Retrieval of a private component used to generate the key will involve the following steps:

1) the requesting user gathers information on the public component associated with the required private component. This information is placed in the **CertTemplate** field of the private component retrieval request message **PriComKeyRetReq**;

2) the **PriComKeyRetReq** message is transported to the TTP;

3) the responding TTP processes **PriComKeyRetReq.** If the request is successful, the private component used to generate the key is placed in the **CertificateKeyPair** field of the response message **PriComKeyRetRep**;

4) the **PriComKeyRetRep** message is transported to the requesting user.

Retrieval of the private component will only be successful if it has been archived previously, and if the TTP is permitted to release it to the requesting user. The status field **PKIStatusInfo** in the response message will indicate the success or failure of this process.

5.1.6.2 Retrieval of the key

Retrieval of the key itself will involve the following steps:

1) the requesting user gathers information on the appropriate public component associated with the required key. This information is placed in the **CertTemplate** field of the key retrieval request message **KeyRetReq**;

2) the **KeyRetReq** message is transported to the TTP;

3) the responding TTP processes **KeyRetReq**. If the request is successful, the key is placed in the response message **KeyRetRep**;

4) the **KeyRetRep** message is transported to the requesting user.

Key retrieval will only be successful if the private component used to generate the key has been archived, and if the TTP is permitted to release the corresponding key to the requesting user. The status field **PKIStatusInfo** in the response message will indicate the success or failure of this process.

5.2 Key management for integrity and related security services

The TTP user for the functions described below may either be a subject requiring TTP support for certification and related function or a "PKI Registration Authority" (RA) which acts as a go between for the subject and certification authority.

5.2.1 Initialization

5.2.1.1 TTP to TTP initialization

Before users of one TTP can use the public keys of another user of another TTP trust between these TTPs needs to be established. This may be achieved by using a cross certificate between these TTPs or by means of a trusted certification path. This function may be supported by exchange mechanisms different from that used for user interactions with the TTP (e.g. physical exchange between TTP managers) and so need not necessarily be supported by on line exchange protocols.

Cross certification involves the use of three messages:

- cross certification request message (**CrossCertReq**);

- cross certification response message (**CrossCertRep**);

- PKI Confirm message (**PKIConfirm**).

5.2.1.2 User initialization

Before using key management services offered by a TTP, a user may need to obtain information about the functions supported by the TTP along with keys needed to communicate securely with the TTP and TTP public-key needed to verify certificates. The identity of the user may also need to be verified by the TTP. This initialization procedure may involve off-line and even non-electronic communications (e.g. using the postal service) and need not be supported by any on-line exchange protocols.

In addition, two other messages are involved to exchange public-key information:

- PKI Information Request (**PKIInfoReq**);

- PKI Information Response (**PKIInfoRep**).

When carrying out the registration / process a user may indicate that the request is its first certification request by using the following message instead of the CertReq and **CertRep** messages as described in subclause 5.2.3:

- initial registration / certification request (**InitReq**);
- initial registration / certification request (**InitRep**).

5.2.2 Key generation

The public / private key pair used in an asymmetric system may be generated by either the user, or the TTP. Wherever the key is generated this shall be done in a manner which can be trusted to properly generate the keys according to the algorithm used and maintain the integrity of both keys and the confidentiality of the private keys.

NOTE: The use of other trusted parties in the certification process (for example an agent acting on behalf of the user) is not currently addressed in this specification, except for the case where an agent fully represents the user for the purposes of certification.

5.2.2.1 User generated

If the user generates the public/private key pair, the public key is passed to the TTP as part of the registration/certification process (see subclause 5.2.3). This is included in a blank field of **FullCertTemplate** of the **CertReq** message.

5.2.2.2 TTP generation

If the TTP generates the public/private key pair, this pair is passed to the user as part of the registration / certification process (see subclause 2.3). The private key is passed within the private key field of **CertKeyPair** within **CertRep**. The public key is passed within its certificate; the certificate field of **CertKeyPair** within **CertRep** is contained in **CertRep**Content.

5.2.2.3 Key update

If a certificate has already been issued for the user and the user decides that a new certified key is required the following messages may be used instead of the **CertReq** and **CertRep** messages to request a new certificate as described in 5.2.3, thereby indicating that it is an update to certified key requested earlier.

- registration / certification request following key update (**KeyUpdReq**);
- registration / certification response following key update (**KeyUpdRer**).

The key is generated and carried in these messages as part of the registration / certification process as described above.

NOTE: In the case where the user wishes to extend the life of an existing certified key then the certificate needs revalidating.

5.2.2.4 TTP key update announcement

When a TTP updates its own public/private key pair the following messages may be used to announce this event to users to the TTP.

- announce TTP key update (**TTPKeyUpdAnn**)

5.2.3 Key certification

A user registers with a TTP and requests a certificate using the following messages:

- registration / certification request (**CertReq**);
- registration / certification response (**CertRep**).

An initial registration certification request may optionally be carried using the following messages:

- initial registration / certification request (**InitReq**);
- initial registration / certification response (**InitRep**).

When the user identifies the need for a new key pair and associated certificate certification / registration requests may optionally be carried using the following messages:

- registration / certification request following key update (**KeyUpdReq**);
- registration / certification response following key update (**KeyUpdRep**).

The user includes in the request any information which it requires to be placed in the certificate including its name. Certain information in the request, such as expiry time may differ from that requested by the user.

In order to complete the registration /certification process further exchanges may be necessary to attest the validity the user's name and other information to be included in or implied by the certificate.

If the key pair is generated by the user then the public key may be passed to the TTP in the request. If the key pair is generated by the TTP then the encrypted private key may be passed to the user in the response.

If key pair is generated by the user and it is for digital signatures then **POPOSigningKey** field may be included in the request to prove ownership of the private key.

If required the user may send the following message to demonstrate acceptance of the certificate.

- confirm (**PKIConfirm**).

5.2.4 Certificate announcement

Having completed the registration / certification process, including if necessary receipt of a **PKIConfirm** message, the TTP makes the certificate available to other parties. This may be achieved by a range of mechanisms including:

a) placing the certificate in a repository such as a directory service or Web server;

b) passing the certificate to other users who are known to require the certificate (e.g. a members of a known user community)

Alternatively the user may make its certificate available to other parties by including the certificate with any protected data.

A TTP may send certificates to users using the following message:

- certificate announce (**CertAnn**).

The exchanges required to place certificates in a repository will depend on the form of repository used.

If a party requiring a certificate has not already been provided it by a certificate announce message or along with the protected data then the certificate may be obtained from a repository such a directory service or web server.

The exchanges required to obtain a certificate from a repository will depend on the form of repository used.

5.2.5 Key distribution

For key distribution in the asymmetric case, the private key needs to be treated differently from the public key.

Private key

For the private key, a differentiation needs to be made as to who generates the key pair. If users generate the key pairs themselves, no distribution of the private key is necessary. Nevertheless, the user might want or need to send some key escrow/recovery information to the TTP (see subclause 5.2.2.1). If the key pair is not generated by the users (i.e. it is generated by the TTP), the private key needs to be distributed to the user in a secure way.

NOTE: Information about the distribution of private keys can be found in ISO/IEC 11770-2 [11], which introduces different methods for key distribution.

Public key

The public keys are distributed together with the certificates assigned to them. The assignment process involving key generation differs slightly from the process without key generation, but in both cases the public key and the corresponding certificate have to be transported to the user in some secure and authentic way. Associated messages are those concerning the announcement and retrieval of certificates (see subclause 5.2.4).

5.2.6 Key/certificate revocation

Certificate revocation should take place whenever there is a suspect that something went wrong with the keys (private key is compromised), or if changes relevant to the certificate associated with the public key occur (change of name, termination of employment in an organization, etc.). If a user (or any other authorized person) wants to revoke the public key and the certificate associated with it, a request **RevReq** is sent to the TTP, which answers with a **RevRep**.

If a TTP has revoked, or is about to revoke, the certificate requested, it may issue an announcement of this event (**RevAnn**). Especially, this might be the case if the request for revocation was not issued by the user but by somebody else.

In combination with the revocation activities, the entries in the CRL change. To give notice of these changes, the TTP issues an announcement of the new CRL (**CRLAnn**). This information should be sent to all users concerned.

If somebody wants now to retrieve the certificate already revoked (**CertRevReq**), then the answer given by the TTP (**CertRevRep**) should contain the information that the certificate asked for has been revoked. Further information about the retrieval of certificates can be found in subclause 5.2.3.

6 Interface specification

The specification defined in this subclause is derived from the functional specification given in clause 5.

6.1 General message structure

This clause describes the general data structure required for the messages going across the various interfaces which are listed in subclause 6.2. All messages which might be used by TTPs in combination with key management use the following structure:

```
        TTPMessage ::= SEQUENCE {
        header          TTPHeader,
        body                TTPBody,
        protection      [0] TTPProtection               OPTIONAL,
        extraCerts      [1] SEQUENCE OF Certificate     OPTIONAL
}
```

The TTP message header, body and protection are discussed in detail in subclauses 6.1.1 to 6.13. The extra certificates field **extraCerts** can contain certificates which may be useful to the recipient. For example, this can be used by a TTP to present an end entity with certificates which it needs to verify its own new certificate (if the TTP that issued the end entity's certificate is not a root for the end entity).

It should also be noted that this field does not necessarily contain a certification path - the recipient may have to sort, select from, or otherwise process the extra certificates in order to use them.

6.1.1 TTP message header

All TTP messages require some header information for addressing and transaction identification. Some of this information will also be present in a transport-specific envelope; however, if the message is protected then this information is also protected (i.e. we make no assumption about secure transport).

The following data structure is used to contain this information and is followed by a brief explanation:

```
TTPHeader ::= SEQUENCE {
    pvno                        INTEGER     { version1 (0) },
    sender                      GeneralName,
    recipient                   GeneralName,
    messageTime          [0]     GeneralizedTime         OPTIONAL,
    protectionAlg        [1]     AlgorithmIdentifier     OPTIONAL,
    senderKID            [2]     KeyIdentifier           OPTIONAL,
    recipKID             [3]     KeyIdentifier           OPTIONAL,
    transactionID        [4]     OCTET STRING            OPTIONAL,
    senderNonce          [5]     OCTET STRING            OPTIONAL,
    recipNonce           [6]     OCTET STRING            OPTIONAL,
    freeText             [7]     TTPFreeText             OPTIONAL
}
```

The **pvno field** is fixed for this version of the TTP standard. The **sender field** contains the name of the sender of the **TTPMessage**. This name (in conjunction with senderKID, if supplied) should be usable to verify the protection on the message.

If nothing about the sender is known to the sending entity (e.g. Distinguished Name (DA), e-mail name, IP address, etc.), then the "sender" field shall contain a "NULL" value; that is, the SEQUENCE OF relative distinguished names is of zero length. In such a case the senderKID field shall hold an identifier (i.e. a reference number) which indicates to the receiver the appropriate shared secret information to use to verify the message.

The **recipient field** contains the name of the recipient of the **TTPMessage**. This name (in conjunction with recipKID, if supplied) should be usable to verify the protection on the message. The **messageTime field** contains the time at which the sender created the message (used when sender believes that the transport will be "suitable", i.e. that the time will still be meaningful upon receipt). This may be useful to allow end entities to correct their local time to be consistent with the time on a central system. The **protectionAlg field** specifies the algorithm used to protect the message. If no protection bits are supplied (**TTPProtection** is optional) then this field shall be omitted; if protection bits are supplied then this field shall be supplied.

The fields senderKID and **recipKID** are usable to indicate which keys have been used to protect the message. The **transactionID field** within the message header is required so that the recipient of a response message can correlate this with a previously issued request. For example, in the case of an RA there may be many requests outstanding at a given moment.

The **senderNonce** and **recipNonce fields** protect the **TTPMessage** against replay attacks. Nonces are used to provide replay protection, senderNonce is inserted by the creator of this message; recipNonce is a nonce previously inserted in a related message by the intended recipient of this message. The **freeText field** may be used to send a human-readable message to the recipient. The structure of this field is:

```
TTPFreeText ::= CHOICE {
    iA5String       [0] IA5String,
    bMPString       [1] BMPString
}
    --NOTE:     The text included here would ideally be in the
    --          preferred language of the recipient
```

6.1.2 TTP message body

The **TTPBody** contains message-specific information. The content of these messages are described in subclauses 6.3 and 6.4.

```
TTPBody ::= CHOICE {
    -- message-specific body elements
    ir          [0]     InitReqContent,
    ip          [1]     InitRepContent,
    cr          [2]     CertReqContent,
    cp          [3]     CertRepContent,
    kur         [7]     KeyUpdReqContent,
    kup         [8]     KeyUpdRepContent,
    rr          [11]    RevReqContent,
    rp          [12]    RevRepContent,
    ccr         [13]    CrossCertReqContent,
    ccp         [14]    CrossCertRepContent,
    cann        [16]    CertAnnContent,
    rann        [17]    RevAnnContent,
    crlann      [18]    CRLAnnContent,
    conf        [19]    PKIConfirmContent,
    nested      [20]    NestedMessageContent,
    infor       [21]    PKIInfoReqContent,
    infop       [22]    PKIInfoRepContent,
    error       [23]    ErrorMsgContent,
    tkuann      [200]   TTPKeyUpdAnnContent,
    enr         [201]   EncInfoReqContent,
    enp         [202]   EncInfoRepContent,
    decr        [203]   DecInfoReqContent,
    decp        [204]   DecInfoRepContent,
    derr        [205]   DecRevReqContent,
    derp        [206]   DecRevRepContent,
    enann       [207]   EncRevAnnContent,
    decann      [208]   DecRevAnnContent,
    prckr       [209]   PriCompKeyRetReqContent,
    prckp       [210]   PriCompKeyRetRepContent,
    krrr        [211]   KeyRetReqContent,
    krrp        [212]   KeyRetRepContent,
    itr         [213]   InitTTPTTPReqContent,
    itp         [214]   InitTTPTTPRepContent
}
```

6.1.3 TTP message protection

Some TTP messages will need to be protected for integrity or confidentiality reasons or for both.

For example, when protection is applied for integrity TTPProtection contains bits which protect the TTP message. In this case the following structure is used:

```
    TTPProtection ::= BIT STRING
```

The input to the calculation of the protectionBits is the Basic Encoding Rules (BER) encoding of the following data structure:

```
ProtectedPart ::= SEQUENCE {
        header      TTPHeader,
        body            TTPBody
}
```

Depending on the circumstances the **TTPProtection** bits may contain a MAC or signature.

For confidentiality, the TTP message will be some encrypted value.

Multiple layers of protection can be accomplished using the following nested message:

```
NestedMessageContent :: = ANY
    -- This will be a TTPMessage.
```

The present document does not specify the mechanisms and protocols to be used as these will be dependent on a number of implementation factors including the underlying transport protocol being used.

Further details will be included in the next edition of the present document.

6.2 Interfaces and messages

Table 1 shows the types of message that are related to each of the TTP interfaces.

Table 1

INTERFACE	SERVICE: Key Management and Key Escrow/Recovery for Confidentiality Services		Key Management for Integrity and Related Security Services	
U-I	EncInfoReq EncInfoRep DecInfoReq DecInfoRep DecRevReq DecRevRep EncRevAnn DecRevAnn PriCompKeyRetReq KeyRetReq PriCompKeyRetRep KeyRetRep	5.1.3.1 / 6.3.1 5.1.3.1 / 6.3.2 5.1.3.2 / 6.3.3 5.1.3.2 / 6.3.4 5.1.4 / 6.3.5 5.1.4 / 6.3.6 5.1.4 / 6.3.7 5.1.4 / 6.3.8 5.1.7 / 6.3.9 5.1.7 / 6.3.9 5.1.7 / 6.3.10 5.1.7 / 6.3.10	PKIInfoReq PKIInfoRep InitReq InitRep CertReq **CertRep** CertRetReq CertRetRep KeyUpdReq KeyUpdRep TTPKeyUpdAnn CertAnn RevAnn RevReq RevRep CRLAnn	5.2.1.2 / 6.4.16 5.2.1.2 / 6.4.17 5.2.1.2 / 6.4.1 5.2.1.2 / 6.4.2 5.2.3 / 6.4.3 5.2.3 / 6.4.4 5.2.6 5.2.6 5.2.2.3 / 6.4.5 5.2.2.3 / 6.4.6 5.2.2.4 / 6.4.11 5.2.4 / 6.4.12 5.2.6 / 6.4.13 5.2.6 / 6.4.7 5.2.6 / 6.4.8 5.2.6 / 6.4.14
KER-I	EncRevAnn DecRevAnn PriCompKeyRetReq KeyRetReq PriCompKeyRetRep KeyRetRep	5.1.4 / 6.3.7 5.1.4 / 6.3.8 5.1.7 / 6.3.9 5.1.7 / 6.3.9 5.1.7 / 6.3.10 5.1.7 / 6.3.10		
T-I	InitTTPTTPReq InitTTPTTPRep EncRevAnn DecRevAnn	5.1.2 5.1.2 5.1.4 / 6.3.7 5.1.4 / 6.3.8	CrossCertReq Cross**CertRep** PKIConfirm CertAnn	5.2.1.1 / 6.4.9 5.2.1.1 / 6.4.10 5.2.3 / 6.4.15 5.2.4 / 6.4.12

6.3 Messages for confidentiality services

6.3.1 Encryption information request

An encryption information request message (**EncInfoReq**) contains an **EncInfoReqContent** data structure which specifies the information required and provides the TTP with additional information it may require. The name of the requester is present in the header structure.

```
EncInfoReqContent ::= SEQUENCE {
    reqOwnCompCert      BOOLEAN,
    -- 'TRUE' if this message is to request a certificate for the user's own public
    -- component, 'FALSE' otherwise

    reqRecPubCompCert   BOOLEAN,
    -- 'TRUE' if this message is to request public receive components for intended
    -- recipients, 'FALSE' otherwise

    recipIds            SEQUENCE OF Recip   OPTIONAL,
    -- present if reqPubCompCert is 'TRUE'

    ownPriComp          BOOLEAN             OPTIONAL,
    -- present if reqOwnCompCert is 'TRUE', this field specifies whether the
    -- encrypting user's EncPriComp is to be generated by the user ('TRUE') or the
    -- TTP ('FALSE').

    archiveOptions      PKIArchiveOptions   OPTIONAL,
    -- present if reqOwnCompCert is 'TRUE'

    encPubComp          EncPubComp          OPTIONAL,
    -- present if both reqOwnCompCert and ownPriComp are 'TRUE'

    status              PKIStatusInfo,
    -- contains failure information if key generation was not possible

    otherInfo           Extensions          OPTIONAL
}

EncPubComp :: = BIT STRING

Recip ::= SEQUENCE {
    idRecip UserId,
    idRecipTtp  TtpId
}

PKIArchiveOptions ::= CHOICE {
    encryptedPrivKey        [0] EncryptedValue,
    -- the actual value of the private component

    keyGenParameters        [1] KeyGenParameters,
    -- not used in this context

    archiveRemGenPrivKey    [2] BOOLEAN
    -- set to TRUE if sender wishes the TTP to archive the private
    -- component which the TTP generates in response to
    -- this request; set to FALSE if no archival is desired.
}
```

6.3.2 Encryption information response

An encryption information response message (**EncInfoRep**) is always in reponse to an **EncInfoReq** message. It contains an **EncInfoRepContent** data structure which specifies the information required.

```
EncInfoRepContent ::= SEQUENCE {
    certEncPubComp  Certificate             OPTIONAL,
    -- present if reqOwnCompCert in the corresponding EncInfoReq message
    -- was 'TRUE'

    encPriComp      EncPriComp              OPTIONAL,
    -- present if both reqOwnCompCert was 'TRUE' and ownPriComp was
    -- 'FALSE' in the corresponding EncInfoReq message

    recipInfo       SEQUENCE OF RecInfo     OPTIONAL,
    -- present if reqRecPubCompCert in the corresponding EncInfoReq message
    -- was 'TRUE'

    status          PKIStatusInfo,
    -- contains failure information if key generation was not possible

    otherInfo       Extensions              OPTIONAL
}

EncPriComp :: = EncryptedValue

RecInfo ::= SEQUENCE {
    idRecip UserId,
    --  corresponds with one of the idRecip entries in the corresponding
    --  EncInfoReqContent

    decPubComp      DecPubComp              OPTIONAL,
    certDecPubComp  Certificate             OPTIONAL
}
```

Depending on the system adopted, one of either **decPubComp** or **certDecPubComp** will be present.

```
DecPubComp :: = BIT STRING
```

6.3.3 Decryption information request

An decryption information request message (**DecInfoReq**) contains an **DecInfoReqContent** data structure which specifies the information required. The name of the requester is present in the header structure.

```
DecInfoReqContent ::= SEQUENCE {
    sendTtpId       TtpId,
    sigVerKeyReq    BOOLEAN,
    otherInfo       Extensions      OPTIONAL
}
```

The sigVerKeyReq field defines whether the user requires the signature verification key of the sending user's TTP ('TRUE') or not ('FALSE').

6.3.4 Decryption information response

A decryption information response message (**DecInfoRep**) is always in reponse to an **DecInfoReq** message. It contains a **DecInfoRepContent** data structure which specifies the information required.

```
DecInfoRepContent ::= SEQUENCE {
    decPriComp      DecPriComp,
    valTimePeriod   ValTimePeriod,
    sigVerKey       SigVerKey           OPTIONAL,
    otherInfo       Extensions          OPTIONAL
}
```

```
DecPriComp :: = EncryptedValue
```

The field **SigVerKey** is present if and only if the field **SigVerKeyReq** in the corresponding **DecInfoReqContent** was 'TRUE'.

```
ValTimePeriod ::= SEQUENCE {
    startTime       Time,
    endTime         Time
}
```

```
Time :: = BIT STRING
```

6.3.5 Decryption key revocation request

A decryption component revocation request message (**DecRevReq**) contains a **DecRevReqContent** data structure which contains the information necessary for the receiving TTP to initiate the revocation of the component.

```
DecRevReqContent :: = SEQUENCE {
    sendTtpId       TtpId,
    badTime         Time
}
```

```
TtpId :: = BIT STRING
```

6.3.6 Decryption key revocation response

A decryption component revocation response message (**DecRevRep**) is always in response to a **DecRevReq** message. It contains a data structure which contains information confirming that a given decrypting user's private component (and corresponding public component) has been revoked.

```
DecRevRepContent ::= SEQUENCE {
    sendTtpId       TtpId,
    badTime         Time,
    revOk           BOOLEAN
}
```

6.3.7 Encryption key revocation announcement

An encryption key revocation announcement (**EncRevAnn**) is sent by a TTP to one of its users. The user may then forward the message to other users.

```
EncRevAnnContent ::= CHOICE {
    encPubComp              [0] EncPubComp
    keyIdentifier           [1] KeyIdentifier
    --- import definition from X.509 v3

    certificateSerialNumber [2] CertificateSerialNumber
    --- import definition from X.509 v3
}
```

6.3.8 Decryption key revocation announcement

A decryption component revocation announcement (**DecRevAnn**) is initially sent out by a TTP to another TTP. Both TTPs forwards the message to those users who they decide need to receive it.

```
DecRevAnnContent ::= SEQUENCE {
    compUserId      UserId,
    sendTtpId       TtpId,
    badTime         Time
    identifier      Identifier          OPTIONAL
}

Identifier ::= CHOICE {
    keyIdentifier               [1] KeyIdentifier
    certificateSerialNumber     [2] CertificateSerialNumber
 }
```

6.3.9 Key retrieval request

There are two separate types of key retrieval request depending on whether or not the requesting user wants to retrieve a private component used to generate the key (**PriComRetReq**), or the key itself (**KeyRetReq**). In both cases the name of the requester is present in the header structure.

```
PriComRetReqContent ::= InitReqContent

KeyRetReqContent ::= InitReqContent
```

6.3.10 Key retrieval response

A private component retrieval response message (**PriComRetRep**) is always generated in response to a **PriComRetReq** message. It contains a **PriComRetRepContent** data structure which specifies the information required.

Similarly, a key retrieval response message (**KeyRetRep**) is always generated in response to a **KeyRetReq** message. It contains a **KeyRetRepContent** data structure which specifies the information required.

```
PriComRetRepContent ::= SEQUENCE {
    status      PKIStatusInfo
    keyPair     SEQUENCE OF     CertificateKeyPair  OPTIONAL
}

KeyRetRepContent ::= SEQUENCE {
    status      PKIStatusInfo
    pubComp     PubComp
    key         Key
}
```

6.4 Messages for integrity and related security services

The standard supports the following messages as defined in: Internet Public Key Infrastructure Certificate Management Protocol, Internet Draft, September 1997 [5]. These messages are current as at the date of this Internet draft.

6.4.1 Initialization request

An initialization request message (**InitReq**) contains an **InitReqContent** data structure which specifies the requested certificate(s). Typically, **SubjectPublicKeyInfo**, **KeyId**, and Validity are the template fields which may be supplied for each certificate requested.

```
InitReqContent ::= SEQUENCE {
    protocolEncKey      [0] SubjectPublicKeyInfo    OPTIONAL,
    fullCertTemplates   FullCertTemplates
}
```

6.4.2 Initialization response

An initialization response message (**InitRep**) contains an **InitRepContent** data structure which has for each certificate requested a **PKIStatusInfo** field, a subject certificate, and possibly a private key (normally encrypted with a session key, which is itself encrypted with the **protocolEncKey**).

```
InitRepContent ::= CertRepContent
```

6.4.3 Registration/certification request

A Registration/Certification request message (**CertReq**) contains a **CertReqContent** data structure which specifies the requested certificate.

```
CertReqContent ::= CHOICE {
    fullCertTemplates       [0] FullCertTemplates,
    pkcs10CertReqContent    [1] PKCS10CertReqContent
}
```

6.4.4 Registration/certification response

A registration response message (**CertRep**) contains a **CertRepContent** data structure which has a TTP public key, a status value and optionally failure information, a subject certificate, and an encrypted private key.

```
CertRepContent ::= SEQUENCE {
    caPub                   [1] Certificate             OPTIONAL,
    response                    SEQUENCE OF CertResponse
}

CertResponse ::= SEQUENCE {
    certReqId                   INTEGER,
    -- to match this response with corresponding request

    status                      PKIStatusInfo,
    certifiedKeyPair            CertifiedKeyPair        OPTIONAL
}

CertifiedKeyPair ::= SEQUENCE {
    certificate             [0] Certificate             OPTIONAL,
    encryptedCert           [1] EncryptedValue          OPTIONAL,
    privateKey              [2] EncryptedValue          OPTIONAL,
    publicationInfo         [3] PKIPublicationInfo      OPTIONAL
}
```

Only one of the failInfo (in PKIStatusInfo) and certificate fields should be present in CertResponse (depending on the status). For some status values (e.g. waiting) neither of the optional fields will be present.

The CertifiedKeyPair structure shall contain either a Certificate or an EncryptedCert, and an optional EncryptedPrivateKey (i.e. not both a Certificate and EncryptedCert).

Given an EncryptedCert and the relevant decryption key the certificate may be obtained. The purpose of this is to allow a TTP to return the value of a certificate, but with the constraint that only the intended recipient can obtain the actual certificate. The benefit of this approach is that a TTP may reply with a certificate even in the absence of a proof that the requester is the end entity which can use the relevant private key (note that the proof is not obtained until the PKIConfirm message is received by the TTP). Thus the TTP will not have to revoke that certificate in the event that something goes wrong.

6.4.5 Key update request content

For key update requests the following syntax is used. Typically, **SubjectPublicKeyInfo**, KeyId, and Validity are the template fields which may be supplied for each key to be updated.

```
KeyUpdReqContent ::= SEQUENCE {
    protocolEncKey          [0] SubjectPublicKeyInfo    OPTIONAL,
    fullCertTemplates       [1] FullCertTemplates       OPTIONAL
}
```

6.4.6 Key update response content

For key update responses the syntax used is identical to the initialization response.

```
KeyUpdRepContent ::= InitRepContent
```

6.4.7 Revocation request content

When requesting revocation of a certificate (or several certificates) the following data structure is used. The name of the requester is present in the **PKIHeader** structure.

```
RevReqContent ::= SEQUENCE OF RevDetails

RevDetails ::= SEQUENCE {
    certDetails         CertTemplate,
    -- allows requester to specify as much as they can about
    -- the cert. for which revocation is requested
    -- (e.g. for cases in which serialNumber is not available)

    revocationReason    ReasonFlags,
    -- from the DAM, so that TTP knows which Dist. point to use

    badSinceDate        GeneralizedTime     OPTIONAL,
    -- indicates best knowledge of sender

    crlEntryDetails     Extensions
    -- requested crlEntryExtensions
}
```

6.4.8 Revocation response content

The response to the **RevReqconten**t message. If produced, this is sent to the requester of the revocation (a separate revocation announcement message may be sent to the subject of the certificate for which revocation was requested).

```
RevRepContent ::= SEQUENCE {
    status              PKIStatusInfo,
    revCerts            [0] SEQUENCE OF CertId          OPTIONAL,
    -- identifies the certs for which revocation was requested

    crls                [1] SEQUENCE OF CertificateList OPTIONAL
    -- the resulting CRLs (there may be more than one)
}
```

6.4.9 Cross certification request content

Cross certification requests use the same syntax as for normal certification requests with the restriction that the key should have been generated by the requesting TTP and should not be sent to the responding TTP.

```
CrossCertReqContent ::= CertReqContent
```

6.4.10 Cross certification response content

Cross certification responses use the same syntax as for normal certification responses with the restriction that no encrypted private key can be sent.

```
CrossCertRepContent ::= CertRepContent
```

6.4.11 TTP key update announcement content

When a TTP updates its own key pair the following data structure may be used to announce this event.

```
TTPKeyUpdAnnContent ::= SEQUENCE {
    oldWithNew              Certificate,
    -- old pub signed with new priv
    newWithOld              Certificate,
    -- new pub signed with old priv
    newWithNew              Certificate
    -- new pub signed with new priv
}
```

6.4.12 Certificate announcement

This data structure may be used to announce the existence of certificates.

NOTE: This structure (and the **CertAnn** message itself) is intended to be used for those cases (if any) where there is no pre-existing method for publication of certificates; it is not intended to be used where, for example, X.500 is the method for publication of certificates.

```
CertAnnContent ::= Certificate
```

6.4.13 Revocation announcement

When a TTP has revoked, or is about to revoke, a particular certificate it may issue an announcement of this (possibly upcoming) event.

```
RevAnnContent ::= SEQUENCE {
    status                  PKIStatus,
    certId                  CertId,
    willBeRevokedAt         GeneralizedTime,
    badSinceDate            GeneralizedTime,
    crlDetails              Extensions          OPTIONAL
    -- extra CRL details (e.g., crl number, reason, location, etc.)
}
```

A TTP may use such an announcement to warn (or notify) a subject that its certificate is about to be (or has been) revoked. This would typically be used where the request for revocation did not come from the subject concerned.

The willBeRevokedAt field contains the time at which a new entry will be added to the relevant CRLs.

6.4.14 CRL announcement

When a TTP issues a new CRL (or set of CRLs) the following data structure may be used to announce this event.

```
CRLAnnContent ::= SEQUENCE OF CertificateList
```

6.4.15 PKI confirmation content

This data structure is used in three-way protocols as the final PKIMessage. Its content is the same in all cases.

NOTE: In fact, there is no content since the PKIHeader carries all the required information.

```
PKIConfirmContent ::= NULL
```

6.4.16 PKI information request content

```
InfoTypeAndValue::= SEQUENCE {
    infoType        OBJECT IDENTIFIER
    infoValue       ANY DEFINED BY infoType     OPTIONAL
}
    --Example InfoTypeAndValue contents include, but are not limited to:
    -- {TTPProtEncCert      =   {xx}, Certificate}
    -- {SignKeyPairTypes    =   {xx}, SEQUENCE OF AlgorithmIdentifier}
    -- {EncKeyPairTypes     =   {xx}, SEQUENCE OF AlgorithmIdentifier}
    -- {PreferredSymmAlg    =   {xx}, TTPKeyUpdAnnContent}
    -- {TTPKeyUpdateInfo    =   {xx}, TTPKeyUpdtAnnContent}
    -- {CurrentCRL          =   {xx}, CertificateList}

PKIInfoReqContent ::= SET OF InfoTypeAndValue
    -- The OPTIONAL infoValue parameter of InfoTypeAndValue is unused.
    -- The TTP is free to ignore any contained OBJ IDs that it does not recognize.
    -- The empty set indicates that the TTP may send any/all information that it wishes
```

6.4.17 PKI information response content

PKIInfoRepContent ::= SET OF InfoTypeAndValue

```
    -- The end entity is free to ignore any contained OBJ IDs that it does not recognize.
```

6.4.18 Error message content

```
ErrorMsgContent ::= SEQUENCE {
    pKIStatusInfo       PKIStatusInfo,
    errorCode           INTEGER                             OPTIONAL,
    -- implementation-specific error codes

    errorDetails        CHOICE { IA5String, BMPString }     OPTIONAL
    -- implementation-specific error details
}
```

6.5 Common message structures

6.5.1 Public key certificate

The public key certificate used for confidentiality and integrity shall be to the syntax:

```
CertTemplate ::= SEQUENCE {
    version     [0] Version                 OPTIONAL,
    -- used to ask for a particular syntax version

    serial      [1] INTEGER                 OPTIONAL,
    -- used to ask for a particular serial number

    signingAlg  [2] AlgorithmIdentifier     OPTIONAL,
    -- used to ask the TTP to use this alg. for signing the cert

    subject     [3] Name                    OPTIONAL,
    validity    [4] OptionalValidity        OPTIONAL,
    issuer      [5] Name                    OPTIONAL,
    publicKey   [6] SubjectPublicKeyInfo    OPTIONAL,
    issuerUID   [7] UniqueIdentifier        OPTIONAL,
    subjectUID  [8] UniqueIdentifier        OPTIONAL,
    extensions  [9] Extensions              OPTIONAL
    -- the extensions which the requester would like in the cert.
}
```

The fields and extension fields shall be used as defined in ITU-T Recommendation X.509 | ISO/IEC 9594-8 [14] with the following specific requirements:

a) TTPs shall encode validity dates through the year 2049 as UTCTime; validity dates in 2050 or later shall be encoded as GeneralizedTime;

b) it is recommended that the authority key identifier is included;

c) key usage:

- the key usage extension field for end user certificates for confidentiality as described in subclause 5.1 when carrying **EncPubComp** shall have the **encipherOnly** and **keyAgreement** bits set;

- the key usage extension field for end user certificates for confidentiality, as described in subclause 5.1, when carrying **DecPubComp** shall have the **decipherOnly** and **keyAgreement** bits set;

- the key usage extension field for end user certificates for integrity as described in subclause 5.2 shall include **digitalSignature** bits set (unless the key is used exclusively for non-repudiation);

- the key usage extension field for TTP certificates shall include **keyCertSign** bit set;

- it is recommended that the key usage extension field is critical;

NOTE: It is recommended that use of this field is enforced by the standard to facilitate the clear division between equipment supporting confidentiality and equipment.

d) it is recommended that certificates also include a CerticatePolicy field to explicitly indicate the domain of applicability of the certificate.

   This specification defines two policy qualifiers types for use by certificate policy writers and certificate issuers at their own discretion. The qualifier types are the CPS Pointer qualifier, and the User Notice qualifier.

   The CPS Pointer qualifier contains a pointer to a Certification Practice Statement (CPS) published by the TTP. The pointer is in the form of a URI.

   The User Notice qualifier contains a text string that is to be displayed to a certificate user (including subscribers and relying parties) prior to the use of the certificate. The text string may be an VisibleString (visible characters from International Alphabet 5 plus space) or a BMPString - a subset of the ISO/IEC 10646-1 [9] multiple octet coded character set. A TTP may invoke a procedure that requires that the certficate user acknowledge that the applicable terms and conditions have been disclosed or accepted;

```
id-ce-certificatePolicies OBJECT IDENTIFIER ::=  { id-ce 32 }
```

```
    certificatePolicies ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

    PolicyInformation ::= SEQUENCE {
        policyIdentifier        CertPolicyId,
        policyQualifiers        SEQUENCE SIZE (1..MAX) OF
        PolicyQualifierInfo                             OPTIONAL
}

    CertPolicyId ::= OBJECT IDENTIFIER

PolicyQualifierInfo ::= SEQUENCE {
        policyQualifierId       PolicyQualifierId,
        qualifier               ANY DEFINED BY policyQualifierId
}

    -- policyQualifierIds for Internet policy qualifiers

    id-pkix-cps                 OBJECT IDENTIFIER ::=  { pkix 4 }
    id-pkix-unotice             OBJECT IDENTIFIER ::=  { pkix 5 }

    PolicyQualifierId ::= ENUMERATED { id-pkix-cps, id-pkix-unotice }

Qualifier ::= CHOICE {
        cPSuri                  CPSuri,
        userNotice              UserNotice
}

CPSuri ::= IA5String

UserNotice ::= CHOICE {
        visibleString           VisibleString,
        bmpString               BMPString
}
```

e) certificates shall include the Basic Constraints field with the TTP flag set for TTP certificates and false or not present for end user certificates.

6.5.2 Certificate revocation list

The certificate revocation list shall be to the syntax::

CertificateList as defined in ITU-T Rec. X509 | ISO.IEC 9594-8.

The fields and extension fields shall be used as defined in ITU-T Recommendation X.509 | ISO/IEC 9594-8 [14] with the following specific requirements:

a) TTPs that issue CRLs shall encode **thisUpdate** and **nextUpdate** as UTCTime for dates through the year 2049 as UTCTime. TTPs conforming to this profile that issue CRLs shall encode **thisUpdate** and **nextUpdate** as GeneralizedTime for dates in the year 2050 or later;

b) the CRL Number shall be included in all CRLs;

c) it is recommended CRLs include reason codes whenever this information is available.

6.5.3 Encrypted values

When encrypted values are sent in TTP messages the following data structure is used:

```
EncryptedValue ::= SEQUENCE {
    encValue                BIT STRING,
    -- the encrypted value itself

    intendedAlg     [0] AlgorithmIdentifier     OPTIONAL,
    -- the intended algorithm for which the value will be used

    symmAlg         [1] AlgorithmIdentifier     OPTIONAL,
    -- the symmetric algorithm used to encrypt the value

    encSymmKey      [2] BIT STRING              OPTIONAL,
    -- the (encrypted) symmetric key used to encrypt the value

    keyAlg          [3] AlgorithmIdentifier     OPTIONAL
    -- algorithm used to encrypt the symmetric key
}
```

```
OptionalValidity ::= SEQUENCE {
    notBefore       [0] UTCTime                OPTIONAL,
    notAfter        [1] UTCTime                OPTIONAL
}
```

6.5.4 Status codes and failure information for PKI messages

All response messages will include some status information. The following values are defined.

```
PKIStatus ::= INTEGER {
    granted                    (0),
    -- you got exactly what you asked for

    grantedWithMods            (1),
    -- you got something like what you asked for; the requester is responsible
    -- for ascertaining the differences

    rejection                  (2),
    -- you don't get it, more information elsewhere in the message

    waiting                    (3),
    -- the request body part has not yet been processed, expect to hear more later

    revocationWarning          (4),
    -- this message contains a warning that a revocation is imminent

    revocationNotification     (5),
    -- notification that a revocation has occurred

    keyUpdateWarning           (6)
    -- update already done for the oldCertId specified in
    -- FullCertTemplate
}
```

Responders may use the following syntax to provide more information about failure cases.

```
PKIFailureInfo :: = BIT STRING {
    -- since we can fail in more than one way!
    -- More codes may be added in the future if/when required.

    badAlg  (0),
    -- unrecognized or unsupported Algorithm Identifier

    badMessageCheck     (1),
    -- integrity check failed (e.g., signature did not verify)

    badRequest  (2),
    -- transaction not permitted or supported

    badTime     (3),
    -- messageTime was not sufficiently close to the system time, as defined by
    -- local policy

    badCertId   (4),
    -- no certificate could be found matching the provided criteria

    badDataFormat   (5),
    -- the data submitted has the wrong format

    wrongAuthority  (6),
    -- the authority indicated in the request is different from the
    -- one creating the response token

    incorrectData   (7),
    -- the requester's data is incorrect (for notary services)

    missingTimeStamp (8)
    -- when the timestamp is missing but should be there (by policy)
}

PKIStatusInfo ::= SEQUENCE {
    status          PKIStatus,
    statusString        PKIFreeText         OPTIONAL,
    failInfo            PKIFailureInfo      OPTIONAL
}
```

6.5.5 Certificate identification

In order to identify particular certificates the following data structure is used.

```
CertId ::= SEQUENCE {
    issuer              GeneralName,
    serialNumber        INTEGER
}
```

6.5.6 Archival options

Requesters may indicate that they wish the PKI to archive a private key value using the following structure:

```
PKIArchiveOptions ::= CHOICE {
    encryptedPrivKey            [0] EncryptedValue,
    -- the actual value of the private key

    keyGenParameters            [1] KeyGenParameters,
    -- parameters which allow the private key to be re-generated

    archiveRemGenPrivKey        [2] BOOLEAN
    -- set to TRUE if sender wishes receiver to archive the private
    -- key of a key pair which the receiver generates in response to
    -- this request; set to FALSE if no archival is desired.
}

KeyGenParameters ::= OCTET STRING
```

6.5.7 Full request template

The following structure groups together the fields which may be sent as part of a certification request:

```
FullCertTemplate ::= SEQUENCE {
    certReqId              INTEGER,
    -- to match this request with corresponding response
    -- (note:  needs to be unique over all FullCertReqs in this message)

    certTemplate           CertTemplate,
    popoPrivKeyVerified    BOOLEAN DEFAULT FALSE,
    popoSigningKey         [0] POPOSigningKey              OPTIONAL,
    archiveOptions         [1] PKIArchiveOptions           OPTIONAL,
    publicationInfo        [2] PKIPublicationInfo          OPTIONAL,
    oldCertId              [3] CertId                      OPTIONAL
    -- id. of cert. which is being updated by this one
}
```

If the certification request is for a signing key pair (i.e. a request for a verification certificate), then the proof of possession of the private signing key is demonstrated through use of the **POPOSigningKey** structure.

```
POPOSigningKey ::= SEQUENCE {
    poposkInput            POPOSKInput,
    alg                    AlgorithmIdentifier,
    signature              BIT STRING
    -- the signature (using "alg") on the DER-encoded
    -- value of poposkInput
}

POPOSKInput ::= CHOICE {
    popoSigningKeyInput     [0] POPOSigningKeyInput,
    certificationRequestInfo    CertificationRequestInfo
    -- imported from [PKCS10] (note that if this choice is used,
    -- POPOSigningKey is simply a standard PKCS #10 request; this
    -- allows a bare PKCS #10 request to be augmented with other
    -- desired information in the FullCertTemplate before being
    -- sent to the TTP)
}
```

```
POPOSigningKeyInput ::= SEQUENCE {
    authInfo            CHOICE {
        sender                  [0] GeneralName,
        -- from PKIHeader (used only if an authenticated identity
        -- has been established for the sender (e.g., a DN from a
        -- previously-issued and currently-valid certificate)

        publicKeyMAC            [1] BIT STRING
        -- used if no authenticated GeneralName currently exists for
        -- the sender; publicKeyMAC contains a password-based MAC
        -- (using the protectionAlg AlgId from PKIHeader) on the
        -- DER-encoded value of publicKey
    },
    publicKey               SubjectPublicKeyInfo
    -- from CertTemplate
}
```

On the other hand, if the certification request is for an encryption key pair (i.e. a request for an encryption certificate), then there are four cases related to proof of possession of the private decryption key:

- for the TTP generated case (see subclause 5.2.2.3) there is no need for POP;
- by the inclusion of the private key (encrypted) in the **FullCertTemplate** (in the **PKIArchivalOptions** structure);
- by having the TTP return not the certificate, but an encrypted certificate (i.e. the certificate encrypted under a randomly-generated symmetric key, and the symmetric key encrypted under the public key for which the certification request is being made). The end entity proves knowledge of the private decryption key to the TTP by MACing the **PKIConfirm** message using a key derived from this symmetric key;

NOTE: If several **FullCertTemplates** are included in the **PKIMessage**, then the TTP uses a different symmetric key for each **FullCertTemplate** and the MAC uses a key derived from the concatenation of all these keys.

- by having the end entity engage in a challenge-response protocol (using the messages **POPODecKeyChallContent** and **POPODecKeyRespContent**) between the **CertReq** and **CertRep** messages.

Annex A (informative): TTP scenarios, configurations and information flows

The following are example scenarios of the use of a TTP to support different applications. These scenarios are independent of the particular mechanism that might be used to implement the TTP service.

A.1 End-to-end encryption with key escrow/recovery

A.1.1 Scenario

One of the applications of a TTP will be to provide confidentiality services in such a way as to satisfy lawful interception requirements. In this role a network of TTPs will offer services to users enabling them to send or receive encrypted messages to each other. The TTPs will also provide services to law enforcement agencies enabling them to decrypt lawfully intercepted communications of users. In this model a user will be registered with a TTP in the network. Once registered, the user will be able to receive services from his TTP, which enable him to send or receive encrypted messages to any other user who is registered with another TTP in the same network. In order to aid interoperability and to recognize the need to provide such services across a broad range of communication networks and user environments, it is important that TTP developers create applications that offer standardized services and interfaces.

In a telecommunications environment, network operators, or any other entity, may run their own TTP, which will offer end-to-end confidentiality services to users across the standardized U-I. In order to set up an encrypted communications channel with a user in another network, the TTPs may need to communicate across a standardized inter-T-I. Assuming that the networks are operated in two different countries, then the law enforcement agencies in each country will be able to intercept the communications from the TTP under their jurisdiction via the standardized KER-I.

A.1.2 Configuration

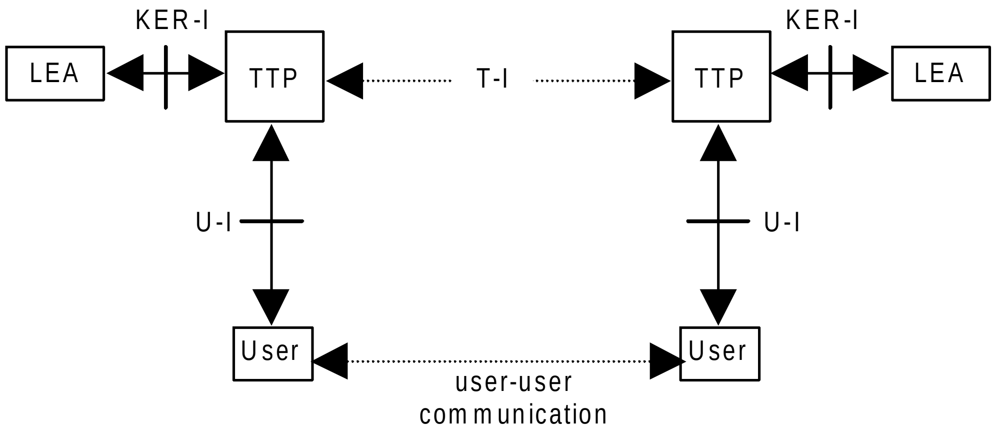

Figure 4: General end-to-end encryption configuration with key escrow

A.1.3 Information flow (external interfaces)

In figure 4 the entities involved can pass messages across the three standardized interfaces. The content and structure of these messages will depend on the mechanism used.

Table 4 contains those messages necessary to support key management and key escrow/recovery services for end-to-end encryption. These messages are defined in clauses 5 and 6.

The U-I provides the means by which the user can interact with the TTP in order to request and receive TTP key management and key escrow/recovery services. This interface carries the user-request and TTP-response messages to facilitate the provision of these services.

The inter T-I provides a means by which TTPs can exchange, if necessary appropriate user information.

Finally, the KER-I provides the means by which a law enforcement agency can request and receive from a TTP the relevant decryption key to enable encrypted communications, lawfully intercepted, to be decrypted.

Table A.1

Interface	Process	Request message	Response message
U-I	Encryption key generation	**EncInfoReq**	**EncInfoRep**
U-I	Decryption key generation and signature verification key	**DecInfoReq**	**DecInfoRep**
U-I	Revoke key	**DecRevReq**	**DecRevRep**
U-I	Announcement of revoked key	**EncRevAnn** **DecRevAnn**	
T-I	User initialiation	**InitTTPTTPReq**	**InitTTPTTPRep**
T-I	Announcement of revoked key	**EncRevAnn** **DecRevAnn**	
KER-I	Decryption key generation and signature verification key	**DecInfoReq**	**DecInfoRep**
KER-I	Revoke key	**DecRevReq**	**DecRevRep**
KER-I	Announcement of revoked key	**EncRevAnn** **DecRevAnn**	

A.1.4 Information flow (internal interfaces)

The TTP will handle service requests across three separate interfaces (T-I, KER-I and U-I). Requests will arrive at the standardized TTP internal access interface via some external communication layer (e.g. communications subsystem, GUI, floppy disk).

At the standardized access interface the request will be processed and a response issued for delivery back to the originator, via the external communications layer.

In order to process the request, the TTP security control will need to access TTP security services, via the TTP internal security service interface.

The TTP services layer will implement the high level security functionality: access to cryptographic functions and algorithms will be via a TTP internal cryptographic interface.

A.2 Support for key escrow/recovery of encrypted data in storage

A.2.1 Scenario

One of the services a TTP may provide is to facilitate the secure storage of data and to provide lawful access to stored data whenever lawful authorization is given. This scenario is restricted to the interaction between a user and his/her TTP and does not consider communication with other users or with other TTPs.

The communication between the user and the TTP will take place via a standardized user interface (U-I). In case lawful authorization is given, access to the data in storage will be provided to the Authorized Entity (AE) requesting it by the TTP via a standardized KER-I.

A.2.2 Configuration

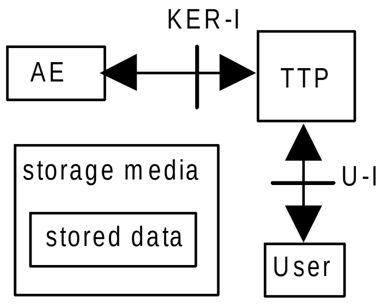

Figure 5: General configuration for stored encrypted data

A.2.3 Information flow (external interfaces)

In figure 5 the entities involved can pass messages across the two standardized interfaces. The content and structure of these messages will depend on the mechanism used for the encryption of the data.

In this scenario, it is assumed that the user encrypts the data and stores it in some storage medium which can be accessed by the user themselves or by any entity having lawful authorization (AE) (see figure 5).

There are a number of cases where access to the decryption key may be required. For example, in the case where the user has lost or destroyed the key, and a back-up is required. Another case might be where someone is lawfully authorized to carry out some legal duty and needs access to the encrypted data e.g. in dealing with a bankruptcy or a civil liability suit, dealing with company accounts or dealing the administration of a deceased person's estate.

The messages necessary to support key management and key escrow/recovery services for encrypted data in storage are defined in clauses 5 and 6.

The U-I provides the means by which the user can interact with the TTP in order to request and receive TTP key management and key escrow/recovery services. This interface carries the user-request and TTP-response messages to facilitate the provision of these services.

Finally, the KER-I provides the means by which a law enforcement agency can request and receive from a TTP the relevant decryption key to enable encrypted data in storage to be decrypted.

Table A.2

Interface	Process	Request message	Response message
U-I	Encryption key generation	**EncInfoReq**	**EncInfoRep**
U-I	Decryption key generation and signature verification key	**DecInfoReq**	**DecInfoRep**
U-I	Revoke key	**DecRevReq**	**DecRevRep**
U-I	Announcement of revoked key	**EncRevAnn** **DecRevAnn**	
KER-I	Decryption key generation and signature verification key	**DecInfoReq**	**DecInfoRep**
KER-I	Revoke key	**DecRevReq**	**DecRevRep**
KER-I	Announcement of revoked key	**EncRevAnn** **DecRevAnn**	

A.3 Support for authentication using digital signatures

A.3.1 Scenario

One service a TTP might want to provide is the support for authentication using digital signatures. This can be done for two different purposes:

- the authentication used for access control, i.e. authentication of an entity (like a person or a computer) authenticating itself to another entity (i.e. a server); or
- the authentication of the data origin (in this case, integrity of the data and non-repudiation are provided as well).

In both cases, digital signatures can be used to implement these services; the use of the digital signature by the entity which wants to authenticate itself or the data origin, and the verification of the digital signature by the recipient of the signature.

For the following, it will be assumed that there are several users (entities) registered to one or more TTPs in a network. Once registered, the users will be able to receive services from their TTPs which enable them to either authenticate themselves to other users[1] or to check the authentication of another user, as long as the other users are registered to other TTPs in the same network. It is also assumed that each two TTPs should either certify each other or should be both certified by a certification authority, which is supplying the certificates to the TTPs.

The standardized interfaces involved are:

U-I:	user interface for the communication between a user and his TTP;
T-I:	interface for the communication between two TTPs.

A.3.2 Configuration

According to the arrangements of the various TTPs, there are different configurations possible. One is the hierarchical solution, in which any two TTPs which want to interact for authentication purposes (and maybe for other purposes as well) are certified by a certification authority, which is shown in figure 6.

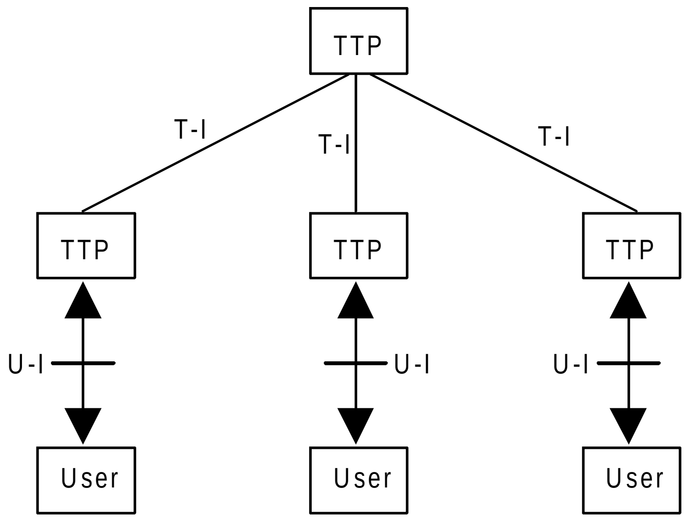

Figure 6: General hierarchical configuration for TTPs

Another possibility is that any two TTPs which want to interact for authentication purposes cross-certifying each other

1 Here and in the following, authentication is considered as a one-to-many relationship, which is also shown in the configuration.

Any of these mixed forms can be reduced to a combination of the hierarchical and the cross-certificate form and the cross-certificate form can be considered as a special form of the hierarchical model (if two TTPs are cross-certifying each other and user B wants to verify the authentication of user A, then the TTP A (the TTP of user A) acts as a TTP for TTP B (the TTP of user B)).

A.3.3 Information flow (external interfaces)

In figure 6, the entities involved are communicating via the two standardized interfaces. For the purpose of authentication (user A wants to authenticate himself to a user B which wants to verify this authentication), the following main steps in the communication will occur (for the following, we assume that the users are registered at the respective TTPs and that TTP A acts as a TTP; we also assume that TTP A has a digital signature key pair). User A wants to digitally sign a document:

- user A requests TTP A, TA , to generate a public key certificate;
- user A's identity is verified;
- to generate the user's public key certificate, TA needs to have the public key of the user; [this key can be generated by the user (in this case it needs to be transferred to TA) or it might be generated by TA; in this case, the key pair needs to be transferred to the user; in any case, it has to be ensured that the assignment of the key pair to the user is done in an authentic way];
- TA creates the user certificate and signs it with its private key (optionally, a time stamping service can be provided);
- TA passes its public key and the user certificate to user A via the user interface U-I.

Table A.3

Interface	Process	Request message	Response message
U-I	Key generation (user)	KerArchReq	KerArchRep
U-I	Key update (TTP)	KeyUpdReq	KeyUpdRep
U-I	Key certification	CertReq	**CertRep**
U-I	Confirmation	PKIConf	
U-I	Key revocation	RevReq	RevRep
U-I	Certificate retrieval	CertRetReq	CertRetRep
U-I	TTP announcement	CertAnn, RevAnn, KeyUpdAnn & CRLAnn	
T-I	TTP Initialization	InitTTPTTPReq	InitTTPTTPRep
T-I	Cross certification	CrossCertReq	Cross**CertRep**

A.3.4 Information flow (internal interfaces)

The TTP will handle service requests across the two separate external interfaces described above (U-I and T-I). Requests will arrive at the standardized TTP internal access interface via some external communications layer (e.g. communications subsystem, GUI, floppy disc). At the standardized access interface the request will be processed, and a response is given via the external communications layer.

In order to process the request, the TTP security control will need to access TTP security services via the TTP internal security service interface. Access to cryptographic functions and algorithms will be via a TTP internal cryptographic interface. The generic interface architecture showing the internal interfaces required is described in subclause 4.4.2 of the present document.

Annex B (informative): Operational and management procedures

B.1 Internal management procedures

B.1.1 Procedures for key generation

The TTP will need procedures for the secure generation of it's own keys. Of particular concern will be protection of it's own secret key as the security of the TTP relies on it. If the TTP also generates key pairs for users, procedures and software will be required for this.

The procedures will depend on the reason for updating the key. For example, in the case of a scheduled update (the normal case) the old key may be used to authenticate the new key, where as in the case of a breach the new key should be distributed using another, secure channel.

B.1.2 Periodic TTP key update and distribution

Procedures are required to enable the TTP to update and re-publish it's own public key certificate in line with it's own security policy, on a regular scheduled basis or as a consequence of some breach.

B.1.3 Management of critical system security parameters

Procedures for generating, issuing, storing and deletion of all passwords, Personal Identification Number (PINs), keys, security tokens, id's, needs to be drawn up in accordance with established security policies. These should set out who can do what, including when and how they can do it.

B.1.4 Procedures to generate certificate

Generation of User Public Key Certificates: the TTP needs to have the software required to format and sign digital certificates. A standard needs to be chosen for the certificate structure (e.g. ITU-T Recommendation X.509 [14], v1, v2 or v3, including decisions on usage of optional extensions), an appropriate naming scheme needs to be chosen (X.500 name, e-mail address, Social Security number etc.), digital signature algorithms needs t be selected, and appropriate validity periods for certificates needs to be established.

B.1.5 Maintenance of a certificate revocation list

Procedures are needed for the day to day maintenance of the CRL including insertion, and deletion of items in the list. Issues to be addressed concern where and how often the list is to be published, which distribution points will be used, and whether, for example, to have separate lists for expired certificates and breached certificates.

B.1.6 Procedures relating to certificate archival/recovery

Certificate Recovery: The TTP may retain a record of all certificates issued to enable users to recover lost certificates. This activity may be delegated to a third party (e.g. a directory manager).

Certificate Archival: The TTP may provide an archive of expired certificates for use in resolving disputes involving old digitally-signed documents (non-repudiation).

B.1.7 Disaster recovery procedures

This will include regular backup and archive, procedures as well as provision for resilience and redundancy in the underlying platforms. To be considered are provision of warm or cold standby servers, regular simulation of disaster events and testing of cutover procedures.

B.1.8 Replacement procedures

In case of security breaches, there have to be procedures for replacement of e.g. existing procedures, keys, algorithms, and personnel. These should cause minimal disruption of service.

B.1.9 Logical access

Operational personnel for the TTP will need login access to the systems. Usual procedures will be required for the issuing of usernames and passwords, together with a policy for allocation of appropriate access privileges, and a mechanism for identification, authentication and authorization at login time.

B.1.10 Physical access

Usual procedures in case of natural disasters should be put in place. Access to the installation housing any such systems will have to be controlled appropriately.

B.1.11 Configuration management

Procedures are required to ensure the system is set up correctly and cannot be tampered with. Any updates and modifications of the system and the network connection need to be introduced in a controlled manner and in such a way as to ensure the system is always in a consistent and recoverable state.

B.1.12 Accountability and audit

In order to make users and TTP staff accountable for their actions, audit logs of security critical activities need to be kept, such as, creation/deletion of users, issue and revocation of certificates, adding hardware and software, adding or modifying network and modem connections. Procedures are required for the analysis of such audit logs particularly in relation to security critical events and to ensure follow up actions are carried out.

B.1.13 Training and support

Personnel involved in running the systems underlying the functioning of the TTP, be they operational, or customer facing people, will need specific training in the running of the TTP.

B.1.14 Monitoring and reporting, management information

Regular reports should be provided on the status and activities of the TTP in a form suitable for managers of the service to monitor and improve the service.

B.2 User - TTP management procedures

B.2.1 Identification procedures

Procedures for user identification checks prior to certificate issue need to be in place. Before certifying a user's public key, the TTP needs to ensure that the user's claimed identity is correct. This is readily achieved in an established user group where details of each user are already known. For a generic TTP service, the registration process should involve the checking of personal credentials (e.g. birth certificate, passport, driving licence) and may require a face-to-face meeting. In addition, the TTP may need to *prove* that the public key belongs to the named individual.

B.2.2 Claimed closed user group membership check

Procedures to carry out registration checks in the case of claimed membership of a closed user group. Here the emphasis is not so much upon the identity of the applicant as upon some attribute of that applicant, such as being an employee of a particular company or membership of a specific professional association or community of interest.

B.2.3 Secret key distribution media

In the event of the TTP carrying out the generation of key pairs on behalf of end users, procedures will be required for the distribution process. This will include generation of the keys, initialization and personalization of the distribution media, dispatch and logging of such media, procedures to cater for returned or faulty media. A critical aspect will be the security of such media, in terms of a trusted supplier and a trusted means of delivery.

B.2.4 Distribution of critical security parameters

Procedures are required for the establishment of a secure channel for the communication of critical parameters (e.g. PIN, passwords). This could be off-line, out of band, or encrypted under a strong authentication process such as a one-time password with challenge response.

B.2.5 Publication of user certificates

Various means are possible including some form of public directory. This would necessitate procedures for record maintenance such as create, update, delete.

B.2.6 Publication of the TTP public key

The TTP needs to have some procedure for publishing its own public key certificate. (The TTP's public key is required for validating the digital signatures on the user certificates.) The certificate may be published in a directory, on a WWW site etc., distributed to users, or embedded in the client software.

B.2.7 Procedure for key compromise actions

Procedures are required to permit users to notify the TTP in the event of a suspected breach. Consequent actions would include the revocation of an existing certificate, publication of the revocation, issue of a new certificate, investigation of the breach, update of existing procedures and security policy in light of any findings.

B.2.8 Procedures for certificate revocation

The TTP needs tot have a mechanism for revoking certificates before they reach the end of their stated validity periods (e.g. to handle security breaches, cater for people leaving the user group, etc.). The most commonly used mechanism is the CRL. The CRL needs to be updated regularly and either published (e.g. using a directory or WWW site) or distributed to users. The revocation mechanism needs to be accompanied by a published policy outlining the conditions under which revocation will occur.

B.2.9 Periodic certificate renewal and certificate distribution

Distribution of User Public Key Certificates: The TTP needs tot have a means of sending the certificate to the user (a secure channel is not required).

Certificate Update Process: The TTP needs to have a process for renewing user certificates when they expire.

B.2.10 Procedures relating to the specific TTP service

The TTP may be a CA or may offer some other value added service such as time stamping or notarisation. Each such service will have a set of specific operational procedures which needs to be defined. For example, for a notarisation service the procedures will relate to running a server which takes as input some document or hash, appends its own digital signature and returns it to the sender. This particular interaction will usually happen automatically, and the operational procedures for running such a service will concern such matters as delivery and configuration, day to day start-up and shutdown, interaction with users, and ongoing maintenance.

NOTE: A user might go via an agent to acquire the services of the TTP. In this case the agent would act as a representative of the TTP towards the user and the same relevant operational and management procedures as described in this annex will apply.

B.2.11 Working practices

Also a TTP should issue a set of working practices to its users which outlines how the TTP will operate in a trusted manner.

B.2.12 Customer-facing "front office"

This will concern operational procedures relating to dealing with telephone calls, personal visits, answering e–mail requests, responding to fax and written requests, provision of Helpdesk facilities.

History

Document history	
V.1.1.1	One-step Approval Procedure OAP 9809: 1997-10-31 to 1998-02-27