# Architecture for Public-Key Infrastructure

## STATUS OF THIS MEMO

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

To learn the current status of any Internet-Draft, please check the 1id-abstracts.txt listing contained in the Internet-Drafts Shadow Directories on ds.internic.net, nic.nordu.net, ftp.isi.edu, or munnari.oz.au.

Comments and suggestions on this document are encouraged. Of particular interest are interfaces and protocols which may have been omitted and specifications which are believed to be suitable bases for standardization of APKI interfaces and protocols. Comments on this document should be sent to the APKI WG discussion list:

pki-tg@opengroup.org

## ABSTRACT

This document describes Requirements and an Architecture for Public-Key Infrastructure components, identifies which elements of the architecture should (in the opinion of the authors) be standardized, and identifies candidate interface and protocol specifications which might serve as base documents for the standardization effort.

ACKNOWLEDGMENTS

Members of the APKI working group contributing substantially to this document include:

Anne Anderson (HP), Charles Blauner (JP Morgan), Belinda Fairthorne (ICL), Warwick Ford, Robert Jueneman (Novell), Ellen McDermott (Open Market), Howard Melman (OSF), Denis Pinkas (Groupe Bull), Walt Tuvell (OSF), and John Wray (Digital Equipment Corporation).

Many other working group members contributed important insights during conversations of the material in this document.

CONTENTS

1. Requirements ..... 4
1.1 Baseline Requirements ..... 4
1.2 The Importance of Architecture ..... 10
1.3 Document Overview ..... 14

2. Overview of the PKI Architecture ..... 15

3. Public-Key Infrastructure Components ..... 16
3.1 Crypto Primitive Components ..... 17
3.2 Crypto Service Components ..... 19
3.3 Long-Term Key Services Components ..... 22
3.4 Protocol Security Services Components ..... 29
3.5 Secure Protocol Components ..... 33
3.6 System Security Enabling Components ..... 35
3.7 Security Policy Services Components ..... 36
3.8 Supporting Services Components ..... 37

4. Hardware Security Devices in the Architecture ..... 37

5. Relationship to Other Standards and Architectures .. 39
5.1 ISO 7498-2 ..... 39
5.2 IETF IPKI Drafts ..... 39
5.3 X/Open XDSF ..... 39
5.4 ECMA TR-46 ..... 39
5.5 RSA PKCS Standards ..... 39

6. Example Applications of the Architecture ..... 40
6.1 OSF DCE 1.1 ..... 40
6.2 SESAME ..... 40
6.3 Nortel Entrust ..... 40
6.4 OMG CORBA ..... 40
6.5 Lotus Notes ..... 40
6.6 Novell Netware ..... 40

7. Glossary ..... 40

8. Security Considerations ..... 40

9. References ..... 40

# 1. Requirements

The following requirements have been collected by the Open Group (OSF - X/Open) Security Program Group Public Key Infrastructure (PKI) Task Group (TG), with the participation of the following organizations:

Barclays Bank, Shell International, Sweden Post, UK Ministry of Defense, BCTEL, US DISA, The Open Group, Telecom Finland Ltd., Pacific Gas & Electric, Electronic Data Systems, Jet Propulsion Laboratory, Boeing, Information & Support Group, Harris Corporation, ICL, Lockheed Martin, Guide International, J P Morgan, IBM, Bellcore, Nortel, HP, NIST, SUN, Siemens Nixdorf, Dynasoft, SCO, Bull, NCR, US NSA, Digital Equipment Corporation, Amdahl, OpenVision, Citicorp, Fujitsu-ICL, Mitre.

The Open Group PKI TG continues to refine and extend these requirements; comments should be sent by electronic mail to OGsecurity@opengroup.org.

## 1.1 Baseline Requirements for a Global PKI and PK Services

An interoperable global PKI is required to provide privacy and digital signature services in support of international commerce, balancing the legitimate needs of commerce, governments and privacy of citizens. The global PKI must support multiple governance policy models within a single global PKI framework, and must enable the enforcement of all existing governance policy mandates.

### 1.1.1 Required Services

A. Establishment of domains of trust and governance
B. Confidentiality (sealing)
C. Integrity and authentication (signing)
D. Non-repudiation
E. End-to-end monitoring, reporting and auditing of PKI services

### 1.1.2 Required Functionality and Characteristics

A. Key life-cycle management

The actual life cycle of a key depends on whether it is used for confidentiality or signature purposes. Key life-cycle facilities to be supported are:

1. Key recovery facilities

The PKI shall provide for key recovery. Key recovery facilities shall provide the following functionality:

i. Use of key recovery facilities implies acceptance of a mandatory policy for the protection and recovery of keys. The policy defines how the keys are to be protected and under what conditions and to whom a key will be made available. The mandatory aspect of policy arises as the operations of a key recovery facility may be regulated by legislation or procedures required under commercial contracts for liability management.

ii. Only key recovery enabled systems shall be usable within a PKI.

iii. A key recovery facility shall be unconditionally trusted and be liable to uphold the stated policy with redress for loss arising from failures to uphold policy through contractual liability and penalties.

iv. A key recovery center shall be able to verify the legitimacy of a key submitted to it for storage.

v. A user of a key recovery repository shall be able to verify that it is an authorized repository.

vi. The PKI shall provide for coordination between the management of public and private keys in PKI and in data recovery centers. Note that public and private key parts do not have the same life cycle and key parts may be archived.

vii. The PKI shall support aging, revocation, and repudiation of keys.

viii. The PKI shall support discretionary key fragmentation between key recovery facilities.

2. Key Generation facility

The method of key generation shall be discretionary, subject to commercial decision and business requirement. Selection of key quality, uniqueness, secrecy and recoverability of keys must be left to the discretion of the organization generating the keys (and any governance authorities to which it is subject).

3. Key Distribution, Revocation, Suspension, Repudiation and Archive

The PKI must support the following functionality:

i. Facilities for the distribution of keys to appropriate storage devices and directories.

ii. Ability of a certification authority to revoke certificates for individual keys under the terms of the applicable policy.

iii. Ability of a certification authority to suspend and reactivate certificates for individual keys under the terms of the applicable policy.

iv. Ability of a certification authority to force delivery of revocation, suspension, and reactivation notices.

v. Facilities to enable a user to repudiate his public key under the terms of the applicable policy.

vi. Facilities to enable a user to suspend and reactivate his public key under the terms of the applicable policy.

vii. Facilities to enable the user and subscriber to retrieve revocation, suspension, and reactivation notices.

viii. Facilities to enable the user and subscriber to determine the status (e.g., revoked or suspended) of a specific certificate.

ix. Facilities to enable the archive and subsequent retrieval of certificates in support of the retrieval and verification of long term information in accordance with governance policy.

4. Warranted retrieval

The PKI must enable the following warranted retrieval scenarios:

i. Law enforcement retrieval (subject to policy conditions)

ii. Corporate agency retrieval (subject to policy and authorizations)

iii. Individual retrieval (subject to policy and authorizations)

The following functionality is required in support of warranted retrieval:

i. An electronic vehicle for the delivery of a notarized electronic warrant, to support the automation of key retrieval under due process (this must be able to take advantage of existing legal agreements)

ii. A permanent, non-repudiable and independently verifiable record of key retrieval operations must be maintained.

Note that warranted retrieval policy includes policy regarding disclosure or non-disclosure of key retrieval to owner of the retrieved key.

B. Distributed Certificate Management Structure

The PKI must provide distributed Certificate Management functionality, driven by the requirements of the transaction or business domain. The following

Certificate Management function must be provided by the PKI:

1. Policing and policy enforcement (governance model), including the following:

   i. Policy creation and maintenance. The policies include those covering key generation, key recovery, key distribution, revocation, suspension, repudiation, archive and warranted retrieval .
   ii. Ability to register a key and the binding between the key and a name.
   iii. Ability to query which keys are bound to a name
   iv. Policies (for services built on PKI access control) must not be required to be based on individual identity.
   v. Certification of the binding between a public key and a directory name shall be mandatory
   vi. Certification of the binding between additional attributes and a directory name shall be discretionary
   vii. Auditing and support for the monitoring of policy compliance is required

2. Concurrent support of multiple policies
3. exchange of certificates.
4. Support for continuance of service in the event of transfer of certificate services from one certification authority to another.
5. Certificate authority policy mapping services to establish cross certification between CAs.
6. Support for arbitration to determine acceptability of certificates in the event of multiple conflicting certification paths.
7. Support for separation of the certification authority and repository functions in accordance with the governance policy. changes to certificate repositories must be transactional (e.g., two-phase commits).

C. Security of the PKI

The PKI itself must be secure. In particular, the PKI must:

1. Protect the confidentiality, integrity and availability of the PKI services, for example key generation, key distribution, and key storage.
2. Provide strong non-repudiation services for actions of certificate services.
3. Prevent PKI services themselves from repudiating their own actions.
4. Prevent users and subscribers from repudiating their own actions.

D. Time service

A universal, networked time service must be available for time stamping.

E. Interoperability

PKI elements provided by different vendors must interoperate. In support of interoperability, PKI elements must:

1. support international standards for certificates and associated data
2. support international standards for certificate services
3. support internationalization of all certificates and associated data
4. support internationalization of all certificate services

### 1.1.3 Known Issues

For interoperability there is a dependency upon the definition of standard application program interfaces to and protocols between the component services of the Public Key Infrastructure.

Work is required to define and agree profiles of option fields in certificates.

### 1.1.4 Recommendations

Adopt X.509 version 3 as a basis for certificates in the development of the PKI.

Adopt and adapt existing standards and protocols wherever possible, invent new standards or protocols only as a last resort.

## 1.2 The Importance of Architecture

The APKI working group feels that a robust, flexible, standard, open Public-Key Infrastructure Architecture is critical to the success of secure systems based on Public-Key technology. This section explains why.

### 1.2.1 What is Architecture?

The architecture of a software system is the set of interfaces through which its functions are accessed, and the set of protocols through which it communicates with other systems.

The remainder of this section discusses the importance of standardizing the interfaces and protocols which comprise the Public-Key Infrastructure software Architecture.

### 1.2.2 Interfaces

The following figure illustrates a system on which three security products have been installed.

In the figure:

- Product 1 includes a protocol and all the security functionality needed to protect data flowing over that protocol. Only the secure protocol's interface is exposed; the underlying security functionality is not available to other applications.
- Product 2 also includes a protocol and its requisite security functionality, but it exposes the data protection functionality through a public interface so that other applications can use it. It does not permit direct access to cryptographic functionality.
- Product 3 is a hardware cryptographic adapter; it

```
          Product 1                   Product 2
      +-------------------+      +-------------------+
      |+-----------------+|      |+-----------------+|
      || Secure          ||      || Secure          ||
      || Protocol        ||      || Protocol        ||
      |+-----------------+|      |+-----------------+|
      |+-----------------+|      |+-----------------+|
      || Security        ||      || Security        ||
      || State Mgmt &    ||      || State Mgmt &    ||
      || Data Protection ||      || Data Protection ||
      |+-----------------+|      |+-----------------+|
      |+-----------------+|      |+-----------------+|
      || Crypto          ||      || Crypto          ||
      || Software        ||      || Software        ||
      |+-----------------+|      |+-----------------+|
      +-------------------+      +-------------------+

                  +-----------------+
         Product 3 | Crypto Hardware |
                  +-----------------+
```

comes with a software driver permitting access by applications to its cryptographic functionality.

This configuration has several bad characteristics:

- Because neither product 1 nor product 2 accesses cryptographic functionality through a standard interface, neither can use the cryptographic adapter. Furthermore, because both product 1 and product 2 embed cryptographic functionality without exposing an interface through which it can be accessed, neither can use the other's cryptographic software. The end result is that three different cryptographic subsystems (two software and one hardware) must be installed on the system, even if all three products use the same cryptographic algorithms!
- Because product 1 and product 2 embed cryptographic functionality rather than accessing a separate cryptographic subsystem through a published interface, they will not be deployable (without code changes) in countries whose regulatory environment restricts or forbids use of the cryptographic functions they embed.

This example illustrates some of the benefits of standard

interfaces; these include:

- Replaceability of services (e.g. cryptography) without change to exploiting applications
- Elimination of duplicate service implementations in configurations in which multiple applications require the same kind of service
- Reduced programmer training costs (programmers need learn only one standard interface for a service rather than learning the proprietary interfaces of multiple products providing the same service)
- Reduced application porting complexity (code exploiting services through standard interfaces need not be changed, or requires only minimal changes, when porting from one platform supporting the standard interface to another such platform)

### 1.2.3 Protocols

The following figure illustrates two certificate-management products.

In the figure:

- Product 1 communicates key requests to the Certification Authority (CA) via electronic mail, and receives keys and certificates from the CA via email.
- Product 2 communicates key requests to the CA using a proprietary protocol and retrieves keys from a directory service using the LDAP protocol.

A configuration including both products would have several bad characteristics:

- Neither product's CA could accept key requests from the other product's clients.
- Applications using product 1 clients and wishing to advertise their certificates in the directory service would require installation of a separate directory-access product.
- Applications using product 1 clients and wishing to retrieve partners' certificates from the directory service would require installation of a separate directory-access product.

```
  Product 1
+---------------------------------+      +-----------+
|               +-------+   +-+  |      | +-+ +----+|
| Application --|Key    |---|e|--|------|-|e|-|    ||
|     ^         |Request|   |m|  |      | |m| |    ||
|     |         +-------+   |a|  |      | |a| | CA ||
|+----------+               |i|  |      | |i| |    ||
||User Key  |---------------|l|--|------|-|l|-|    ||
||Management|               +-+  |      | +-+ +----+|
|+----------+                    |      |           |
+---------------------------------+      +-----------+

  Product 2
+-------------------------+             +-----------+
|               +-------+ |             |     +----+|
| Application --|Key    |-|-------------|-----|    ||
|     ^         |Request| |             | +-+ |    ||
|     |         +-------+ |             | |L| | CA ||
|+----------+   +------+  | +---------+ | |D| |    ||
||User Key  |---| LDAP |--|-|Directory|-|-|A|-|    ||
||Management|   |      |  | +---------+ | |P| +----+|
|+----------+   +------+  |             | +-+       |
+-------------------------+             +-----------+
```

This example illustrates the benefit of standard protocols:

- Applications supporting standard protocols can interoperate, even if produced by different providers.

### 1.2.4 Profiles

Many of the services in the Public-Key Infrastructure Architecture can be implemented using a variety of different mechanisms and protocols (e.g. data privacy protection can be implemented using a variety of different cryptographic algorithms). This variety of mechanisms and protocols has arisen in part because different environments impose different security requirements.

Multiplicity of mechanisms means that different providers' implementations of the PKI Architecture will not necessarily interoperate - even though they support the standard interfaces and a selection of the standard

protocols.

A profile defines the set of mechanisms and protocols which should be used in a particular environment. The mechanisms and protocols comprising a profile are usually chosen on the basis of their strength against the attacks which are common in the environment supported by the profile. Profiling has the following advantages:

- Systems conforming to an environment's profile will interoperate.
- Systems conforming to an environment's profile will be well-protected against that environment's risks.
- Profiling helps to assure that mechanisms in use work together appropriately and securely.

### 1.2.5 Negotiation

Some profiles will allow multiple mechanisms and protocols in order to support different qualities of protection, or to accommodate a fragmented security product market. In these environments, it is desirable to provide a negotiation meta-protocol which allows communicating partners to determine:

- which mechanisms and protocols they both (or all) share
- which mechanism and protocol, among the shared set, best supports the desired quality of protection.

It is important to note that negotiation does not always require an on-line dialog between the negotiating entities.

## 1.3 Document Overview

Section 2 presents the high-level structure of the PKI Architecture by grouping the architecture's components into broad functional categories.

Section 3:

- enumerates the components in each of the Architecture's functional categories
- describes the functionality of each component and

lists existing specifications which could serve as candidate standards for each component's interfaces and protocols (To be considered a "candidate" for purposes of the public-key infrastructure architecture, an interface or protocol must: (1) be described by a publicly-available specification, and (2) support a significant fraction of the functionality of the PKI component for which it is proposed as a candidate. It is assumed that the candidate interface and protocol specifications identified in this document will serve as base documents for open standardization processes, which will produce finalized PKI component interface and protocol specifications.)
- identifies where negotiation facilities are required to deal with the probable existence of a multiplicity of security mechanisms
- enumerates important public-key-related protocols and discusses the need for environment-specific profiles

Section 4 discusses the use of hardware security devices in the architecture.

Appendices to the document provide:

- examples illustrating application of the architecture to existing secure systems,
- the relationship of this document to existing security architecture standards
- a glossary of terms related to security and public-key cryptography
- an annotated list of references

## 2. Overview of the PKI Architecture

The PKI architecture components are grouped into the following broad functional categories:

A. System Security Enabling Services provide the functionality which allows a user's or other principal's identity to be established and associated with his actions in the system.
B. Crypto Primitives and Services provide the cryptographic functions on which public-key security

is based (including secret-key primitives such as DES).

C. Long-term Key Services permit users and other principals to manage their own long-term keys and certificates and to retrieve and check the validity of other principals' certificates

D. Protocol Security Services provide security functionality (data origin authentication, data integrity protection, data privacy protection, nonrepudiation) suitable for use by implementors of security-aware applications such as secure protocols.

E. Secure Protocols provide secure inter-application communications for security-unaware and "mildly" security-aware applications.

F. Security Policy Services provide the policy-related information which must be carried in secure protocols to enable access control, and provide access-control checking facilities to security-aware applications which must enforce policy.

G. Supporting Services provide functionality which is required for secure operation, but is not directly involved in security policy enforcement.

The following figure illustrates the PKI architecture.

Section 3 describes each of these categories in more detail (listing the components in each category), and identifies interfaces and protocols which may be candidate bases for standardization of each component. Appendix B uses the figure above to illustrate how a number of existing security technologies fit into the architecture.

Note that while the architecture described in this document could be implemented on insecure operating system platforms, implementors of the architecture must insure that keys, security context data, and policy data are appropriately protected in such environments.

## 3. Public-Key Infrastructure Components

Each of this section's subsections describes one of the Architecture's categories in detail, enumerating its components and describing component functions,

```
 +-------------------------------------------------------+
 |                                                       |
 |                  [Applications]                       |
 |                                                       |
 +-------------------------------------------------------+
 |          |                                |            |
 |          |       Secure Protocols         |  Security  |
 |          |                                |   Policy   |
 |  System  +--------------------------------+  Services  |
 | Security |                                |            |
 | Enabling |  Protocol Security Services    +------------+
 | Services |                                |            |
 |          +--------------------------------+            |
 |          |                                |            |
 |          |    Long-Term Key Services      |            |
 |          |                                |            |
 +----------+--------------------------------+ Supporting |
            |                                |  Services  |
            |    Cryptographic Services      |            |
            |                                |            |
            +................................+            |
            |                                |            |
            |   Cryptographic Primitives     |            |
            |                                |            |
            +--------------------------------+------------+
```

interfaces, and protocols.

### 3.1 Crypto Primitive Components

The figure below illustrates the Crypto Primitive Components:

Note that the architecture's cryptographic primitives may be provided by hardware (e.g. smartcards or cryptographic modules) or by software.

#### 3.1.1 Function

These components provide access to low-level cryptographic primitives such as key generation, hash function application to a data buffer, encryption of a data buffer using secret-key or public-key algorithms, decryption of a data buffer using secret-key or public-key algorithms, etc....

```
+------------+ +--------------+ +--------------+ +------------+
|   Random   | |      Key     | |   Secret     | |            |
|   Number   | |  Generation  | |   Sharing    | |  Hashing   |
| Generation | |              | |              | |            |
+------------+ +--------------+ +--------------+ +------------+

+------------+ +--------------+ +--------------+
|   Keyed    | |  Symmetric   | |  Asymmetric  |
|  Hashing   | | (secret-key) | | (public-key) |
|            | |    Crypto    | |    Crypto    |
|            | |  Primitives  | |  Primitives  |
+------------+ +--------------+ +--------------+
```

### 3.1.2 Protocols

Cryptographic primitives are typically called locally; it is not anticipated that any cryptographic primitive protocols will be defined.

### 3.1.3 Interfaces

Candidate interfaces for access to cryptographic primitives include:

- The RSA BSafe library interface
- The X/Open GCS-API
- The Microsoft CryptoAPI 1.0

Other interfaces which may support some or all of the cryptographic primitive function include

- Fortezza
- IBM CCA

Standardization of these interfaces would be of interest to developers of cryptographic service modules and to providers of cryptographic primitive modules. Standardization of an interface for access to cryptographic primitives would facilitate "pluggable" implementations of cryptographic services. The consensus of the APKI working group, however, is that cryptographic functionality will ordinarily be used through the cryptographic service interfaces rather than through the cryptographic primitive interfaces. Therefore, standardization of cryptographic primitive interfaces is

not viewed as essential.

### 3.1.4 Profiles

Most cryptographic modules provide support for multiple primitives. Many primitives are subject to legal restrictions on deployment (including both intellectual property encumbrances and national and international regulatory constraints on export, import, and deployment).

Cryptographic primitive profiles will have to be developed for PKI environments of interest (including, for example, the Internet, OMG CORBA, OSF DCE, Financial, etc.).

### 3.1.5 Negotiation

Cryptographic primitives are ordinarily used only by the implementors of cryptographic services. Negotiation should be used to establish which cryptographic service(s) are to be used, rather than to establish what primitives should be used. Ordinarily this negotiation will be done at a higher level than that of the cryptographic primitives and services themselves. No protocol for negotiating cryptographic primitives should be required.

## 3.2 Crypto Service Components

The figure below illustrates the Cryptographic Service Components:

```
+-----------+ +------------+ +------------+ +------------+
|  Crypto   | |    Key     | |     Key    | |    Key     |
|  Context  | |  Import &  | | Derivation | | Agreement  |
| Management| |   Export   | |            | |            |
+-----------+ +------------+ +------------+ +------------+

+-----------+ +------------+ +------------+ +------------+
|    Key    | |    Data    | |    Data    | |    Data    |
|   Usage   | |  Integrity | |   Privacy  | |  Signature |
|  Control  | |            | |            | |            |
+-----------+ +------------+ +------------+ +------------+
```

### 3.2.1 Function

These components provide access to cryptographic services such as data integrity and privacy protection ("data" here might be a file, a message, an i/o stream, etc...), key import and export, digital signature, keyed hash, etc....

Cryptographic Context Management provides the facilities through which applications initialize the cryptographic subsystem, activate keys for encryption and decryption, and clean up the state of the cryptographic subsystem after use.

Key usage controls permit control over a variety of aspects of key use, including how many times a key may be used; for what purposes it may be used (e.g. for signature only, for privacy only, for both signature and privacy, etc...), and so on.

Key derivation services permit generation of cryptographic-quality keys from non-key values such as passwords.

Crypto services are built on crypto primitives. A crypto service may support multiple implementations, each of which uses a different crypto primitive.

Descriptions of a few DES-based services will illustrate the difference between primitives and services; note that these are only examples:

- DEA is a crypto primitive which uses a 56-bit key and an initialization vector to transform a 64-bit plaintext into a 64-bit ciphertext.
- Data privacy is a crypto service. DES-CBC is an implementation of the cryptographic data privacy service which uses a 56-bit key, an initialization vector, and the DEA primitive to transform a plaintext of arbitrary length into a ciphertext of the same length subject to some rules defined by a "mode of operation". The rules describe how to "pad" plaintexts to a multiple of 64 bits and whether and how to induce dependencies among 64-bit blocks of the ciperhtext by feeding ciperhtext material from

previous rounds of the encryption process into the current round.

- Data integrity is a crypto service. DES-CBC-MAC is an implementation of the data integrity service which uses the DEA primitive to generate a message authentication code given a 56-bit key, an initialization vector, and a plaintext of arbitrary length.

### 3.2.2 Protocols

Cryptographic services are typically called locally; it is not anticipated that any cryptographic service protocols will be standardized.

### 3.2.3 Interfaces

Candidate interfaces for cryptographic services include:

- X/Open GCS-API
- Microsoft CryptoAPI 1.0
- SESAME CSF API

Other interfaces which may support some or all of the cryptographic primitive function include

- Cryptoki
- RSA BSAFE

Standardization of these interfaces would be of interest to developers of long-term-key service and protocol security service modules and to providers of cryptographic service modules. The APKI working group feels that it is important to standardize a single interface for cryptographic services.

### 3.2.4 Profiles

Most cryptographic modules provide support for multiple services. Many crypto services are subject to legal restrictions on deployment (including both intellectual property encumbrances and national and international regulatory constraints on export, import, and deployment).

Cryptographic service profiles will have to be developed for PKI environments of interest (including, for example, the Internet, OMG CORBA, OSF DCE, Financial, etc.). These profiles will have to be developed with international deployment issues in mind. Each profile should be expressed in terms of the parameters used to select cryptographic services (and implementations of cryptographic services -- often called "mechanisms") through the cryptographic service interface (see the next section for more information on service and mechanism selection).

Profiles will need to specify, in addition to mechanism information, the data formats which each service can accept and return.

### 3.2.5 Negotiation

Negotiation of cryptographic services to be used by secure protocols and other security-aware applications is generally done at level higher than that of the cryptographic services themselves. The cryptographic service interface therefore must allow selection among available cryptographic services, and among available implementations of a single service, but it need not support negotiation.

## 3.3 Long-Term Key Services Components

The following figure illustrates the Long-Term Key Services Components; each component is described in more detail below.

```
+-----------+ +------------+ +------------+ +------------+
|    Key    | |    Key     | |    Key     | |   Virtual  |
| Lifecycle | |   Escrow   | |  Recovery  | |  Smartcard |
| Management| |            | |            | |  Service   |
+-----------+ +------------+ +------------+ +------------+

+-----------+ +------------+
|Certificate| | Public-Key |
| Management| | Delivery & |
|           | |Verification|
+-----------+ +------------+
```

### 3.3.1 Function

Key Lifecycle Management

This component provides including key revocation, key repudiation, key expiration, and related services.

Key Escrow and Key Recovery

These components permit secure storage of keys for later recovery under policy control.

Virtual Smartcard Service

The Virtual Smartcard Service Component permits users and other principals to store long-term personal security information (including private keys, certificates, and other information) in protected storage, to activate personal keys for use via an authentication procedure, and to use those keys for encryption, decryption, and signature activities.

The following figure illustrates the structure of this component.

```
+------------------------+
|                        |
| Virtual Smartcard Svc. |
|                        |
+----------+--+----------+
| Hardware |  | Software |
| Security |  | Personal |
|  Token   |  | Security |
|          |  |  Model   |
+----------+  +----------+
```

Certificate Management

The Certificate Management component allows users, administrators and other principals to request certification of public keys and revocation of previously certified keys. It may optionally generate key pairs and provide key-pair recovery services. There are four Certificate Management sub-components:

The Local Registration Authority provides interfaces for requesting generation of key-pairs and corresponding certificates, requesting certification of existing public keys, and requesting revocation of existing certificates.

The Certification Authority Agent (CA Agent) provides interfaces for certifying existing public keys, generating and returning key pairs and corresponding certificates, revoking existing certificates. The CA Agent implements these interfaces via calls to a Certification Authority (CA).

The Certification Authority certifies public keys (returning the generated certificate) and generates certificate revocation lists. In some configurations it will be "off-line".

The Publication Authority provides interfaces through which CAs and CA Agents can place certificates and CRLs into public repositories or transmit them directly to requestors.

Public-Key Delivery and Verification

This component allows a program to retrieve any principal's certificate, verify its validity, and extract the principal's certified public key from the certificate.

The following figure illustrates the structure and interrelationships of the Certificate Management and Public-Key Delivery and Verification components and sub-components.

### 3.3.2 Protocols

Virtual Smartcard Service

When the Virtual Smartcard Service component is used for retrieval of user private keys, two models exist. One model (exemplified by PGP and Lotus Notes) manages private keys primarily on the client principal's machine (either in a software personal security module, or in a security token or other device external to the principal's workstation). In this model, no protocols

```
+-----------+---------------++=====+==============++
|           |  Public       ||     |     Local    ||
|           |  Key          ||     | Registration ||      double
border
| Virtual   |  Delivery     ||     |   Authority  ||    / encloses
| Smartcard |  and          ||     +--------------++   /  Certificate
| Service   |  Verification ||         CA         ||  /   Management
|           |               ||        Agent       || <    Component
|           |               ++---------+----------++
|           |               ||         |          ||
|           +-------++======++         |          ||
|                   ||                 |          ||
|                   ||   Publication   |    CA    ||
|                   ||    Authority    |          ||
+-------------------++=================+==========++
```

are required for User/Principle Personal Security Info Management, since all operations are client-local.

The second model (exemplified by Novell NetWare) manages private keys at a central server and distributes them to client principals using a secure protocol. In this model, the client/server protocol for retrieval of private keys needs to be supported by the software personal security module subcomponent of the Virtual Smartcard Service component, as illustrated in the figure below (the dotted arrow in the figure represents the protocol):

```
      +-----------------------+
      |User/Principal         |
      |Personal Security Info |
      |Management Interface   |
      +--------------+--------+     +----------+
                     |Software|     |Remote    |
                     |Personal|     |Private   |
                     |Security|---->|Key       |
                     |Module  |     |Repository|
                     +--------+     +----------+
```

The APKI working group does not view standardization of this protocol to be essential.

Certificate Management

Protocols must be defined to permit creation, revocation,

and update of certificates. The diagram below illustrates Certificate Management protocols which might be standardized; each arrow in the diagram represents a protocol.

```
                              Certificate Management
                    +-----------------------------------------+
+--------------+    | +------------+            +-----------+ |
| Application  |    | |   Local    |            |Certificate| |
|(or Smartcard)|---->|Registration|----------->| Authority | |
|              |    | | Authority  |            |   Agent   | |
+--------------+    | +------------+            +-----------+ |
        |           |                            /      ^     |
        |           +-------------+             /       |     |
        |                         |            v        |     |
  +------------+                  | +-----------+       |     |
  | Public-Key |                  | |Publication|       |     |
  | Delivery & |<.................| Authority |       |     |
  |Verification|                  | |           |       |     |
  |            |-----+            | +-----------+       |     |
  +------------+     |            +---/-------+         |     |
        |            v               v        |         v     |
  +------------+    +---------------+         | +-----------+ |
  |  Virtual   |    | Repository &  |         | |Certificate| |
  | Smartcard  |    | Subscription  |         | | Authority | |
  |  Services  |    |   Services    |         | |           | |
  +------------+    +---------------+         | +-----------+ |
                                              +---------------+
```

Note that implementations may choose to assign the responsibility for generation of private keys (through use of the key generation facilities of the PKI architecture) to the CA, the LRA, or the User Workstation or Smartcard; additional protocols will be required to transmit the private key to the User Workstation or Smartcard if it is not generated there in the first place.

The APKI working group feels that the following protocols should be standardized at a minimum:

- User Workstation or Smartcard to Certificate Management component
- Local Registration Authority to CA Agent

A candidate protocol specification including these protocols as well as a protocol for the Publication Authority to Public-Key Delivery protocol exists as IETF draft RFC ietf-pkix-ipki-part3-01.txt

Public-Key Delivery and Verification

Protocols must be defined to transport certificates from the repositories in which they reside to the requester's machine. In the diagram, these protocols are represented by the arrows from the Publication Authority to the Public-Key Delivery and Verification component. The APKI working group feels that these protocols should be standardized. At least LDAP, email, and HTTP versions of these protocols should be defined.

A candidate protocol specification is in preparation and will be published as IETF draft RFC ietf-pkix-ipki-part2-01.txt

### 3.3.3 Interfaces

Virtual Smartcard Service

Candidate interfaces for this component include:

- PSM (HP Submission to OSF)
- SESAME CSF API

Other interfaces which may support some or all of the Virtual Smartcard Service functionality include:

- Microsoft Wallet

The APKI working group feels that the Virtual Smartcard Service interface should be standardized.

Additionally, the APKI working group feels that the interface through which software communicates with Hardware Security Tokens should be standardized. A candidate interface for this functionality is:

- RSA PKCS-11

Public-Key Delivery and Verification

Candidate interfaces for this component include:

- CMS-API (Nortel)
- SESAME PKM-API
- NSA CM-API
- Nortel CMS-API

Other interfaces which may support some or all of the Public-Key Delivery and Verification function include

- Microsoft CryptoAPI version 2.0
- Intel CDSA

The APKI working group feels that the Public-Key Delivery and Verification interface should be standardized.

Certificate Management

Candidate interfaces for this component include:

- Nortel CMS-API
- SESAME PKM API
- OSF RFC 80 API

Other interfaces which may support some or all of the Certificate Management function include

- Microsoft CryptoAPI version 2.0
- Intel CDSA

The APKI working group feels that the following interfaces should be standardized at a minimum:

- CA Agent
- Local Registration Authority

Specification of the Publication Authority interface would also be useful to providers of repositories and communications protocols who wish to make their products available as certificate and CRL transmission media; a standard Publication Authority interface would allow them to provide Publication Authority services without requiring changes to CA Agent code.

### 3.3.4 Profiles

It is anticipated that multiple CAs will exist in typical PKI environments; individual servers may require the use of certificates with specific properties (signing CA, supported extensions, name format, etc...) Profiles for certificate format, contents, extensions, and policy will be needed for PKI environments of interest, including the Internet, Financial Industry, Credit Card Industry (for use with SET), Government, and Healthcare Industry environments.

A draft profile (for the Internet PKI environment) for certificate format, contents, and extensions exists as IETF draft RFC ietf-pkix-ipki-part1-01.txt. A draft policy profile for the Internet PKI environment is in preparation and will be published as IETF draft RFC ietf-pkix-ipki-part4-01.txt.

### 3.3.5 Negotiation

It is not anticipated that any of the Long-Term Key Services components will require negotiation protocols. The Certificate Management interfaces will need to provide a mechanism through which callers can identify which CA should issue certificates and CRLs requested through its interface, in case more than one CA is available.

The Virtual Smartcard Service interface will need to support selection of user/principal certificates for environments in which users have more than one certificate.

## 3.4 Protocol Security Services Components

Protocol security services are divided into two fundamental classes:

- Session-Oriented: security services which require exploiting entities to maintain security state information associated with protocol exchanges.
- Store & Forward: security services which encapsulate all required security state information inside the protected message tokens they generate; these services

do not require exploiting entities to maintain security state information. Nonrepudiation services are necessarily store-and-forward services, because they must allow for "protection" of the nonrepudiability of a transaction after it has been completed and its state information destroyed. Nonrepudiation services are depicted separately from other store-and-forward protocol security services because, unlike store-and-forward data privacy and integrity services, use of Nonrepudiation services usually requires explicit user action.

The following figure illustrates the Protocol Security Services Components.

```
+--------+    +----------+    +---------------+
|Session-|    | Store &  |    |Non-Repudiation|
|Oriented|    | Forward  |    |Services       |
|Services|    | Services |    +---------------+
+--------+    +----------+
```

### 3.4.1 Function

These components provide security services appropriate for use by designers of protocol stacks. Specifically, these components:

- Provide security mechanism and quality-of-protection negotiation protocols for use by communication partners needing to agree on a common security regime
- Manage security state information (if any) needed by protocol partners wishing to set up and maintain secure associations
- Encapsulate data origin authentication, data protection, and credential and privilege transport transparently within a single service (Crypto Services, by contrast, typically provide only data protection)
- Apply security mechanisms based on administered policy information

### 3.4.2 Protocols

Session-Oriented Protocol Security Services

A wide variety of protocol security services can be used to provide security for session-oriented protocols; examples which are described in existing or proposed Internet standards include the SPKM (which is Public-Key based), Kerberos (which is Secret-Key based), and SESAME (which has Public-Key, Secret-Key, and hybrid variants). Some of these services define their own protocols for run-time access to on-line security servers of a variety of types. All of them define formats for protected message tokens to be transported by their callers.

Store & Forward Protocol Security Services

Only a few protocol security services suitable for protection of store & forward protocol messages have been defined. The IDUP and SESAME services are proposed for Internet standardization. Both of these services define formats for protected message tokens to be transported by their callers.

Notary and Non-Repudiation Services.

These services must define formats for Non-Repudiation evidence tokens to be transmitted along with notarized data, and protocols implementing non-repudiable delivery and non-repudiable receipt.

The APKI working group feels that multiple protocol security services will continue to be required to meet the needs of diverse environments. No single standard for Session-Oriented, Store-and-Forward, or Nonrepudiation Protocol Security Services is proposed, therefore. The Protocol Security Services component interfaces will need to provide negotiation (for environments in which more than one service is available), and Protocol Security Service profiles will have to be established for PKI environments of interest.

### 3.4.3 Interfaces

The APKI working group feels that all of the Protocol Security Services interfaces should be standardized.

The structure of the Protocol Security Services is illustrated by the following figure.

```
+----------+ +-----------+ +----------------+ +---------------+
|Privilege,| |Mechanism  | |Session-Oriented| |Store & Forward|
|Delegation| |Negotiation| |   Protection   | |   Protection, |
|Management| |           | |                | |Non-Repudiation|
+----------+ +-----------+ +----------------+ +---------------+
       |              |         |        |           |
       |           +------------------+  |           |
       |           |Context Management|  |           |
       |           +------------------+  |           |
       |                                 |           |
       |                     +--------------------------------+
       |                     |       Token Processing         |
       |                     +--------------------------------+
       |                      |          |           |       |
   +----------------------------+ +-----------+ +----------+ |
   |    PAC Generation, Use,    | |Key        | |Dialog Key| |
   |  Protection, Verification, | |Acquisition| |Generation| |
   |        Delegation          | |Negotiation| |          | |
   +----------------------------+ +-----------+ +----------+ |
                                        |           |       |
                                   +-------------------------+
                                   |    Mechanism Provider   |
                                   |         Interface       |
                                   +-------------------------+
                                       |      |       |
                                    +----+ +----+ +------+
                                    |Kerb| |SPKM| |Sesame|
                                    +----+ +----+ +------+
```

Session-Oriented Protocol Security Services

The preferred interface for these services is GSS-API (IETF RFC 1508).

Store & Forward Protocol Security Services

The preferred interface for these services is IDUP-GSS-API (IETF CAT draft ietf-cat-idup-gss-api-04.txt).

Non-Repudiation Services

The preferred interface for these services is IDUP-GSS-API (IETF CAT draft ietf-cat-idup-gss-api-04.txt).

In addition to these interfaces, the APKI working group

feels that interfaces for Protection Mechanism Negotiation and Privilege and Delegation Management should be standardized. The preferred interfaces for these services are draft-ietf-cat-gss-nego and draft-ietf-cat-xgss, respectively.

Other interfaces which may support some or all of the Protocol Security Services functionality include:

- Microsoft SSPI
- OMG CORBA Security
- TIPEM
- SHTTP

### 3.4.4 Profiles

GSS-API and IDUP-GSS-API are capable of supporting multiple security mechanisms; each API also allows selection of a wide range of qualities of data protection (e.g. strength of supported privacy protection, delegation mode, etc...) for each supported security mechanism.

Profiles will have to be developed to describe the set of preferred mechanisms and data protection quality parameters for PKI environments of interest. The APKI working group is not aware of a draft profile in this area.

### 3.4.5 Negotiation

Because they will be deployed in environments which require and provide multiple data protection mechanisms, the Protocol Security Services interfaces will need to support negotiation (of both protection mechanisms to be used and Quality of Protection to be applied).

A negotiation mechanism for GSS-API has been proposed and is described in IETF draft ietf-cat-gss-nego-02.txt.

## 3.5 Secure Protocol Components

There are many kinds of secure protocols. Three important categories of secure protocols are:

- Connection-oriented peer-to-peer: These protocols allow exactly two partners, each of which must be on-line, to communicate securely.
- Connectionless peer-to-peer: These protocols allow exactly two partners, one or both of which may be off-line for some portion of the time interval during which messages are transmitted, to communicate securely.
- Connectionless multicast: These protocols allow one entity to communicate simultaneously and securely with several partners. Any or all entities may be off-line for some portion of the time interval during which messages are transmitted.

The following figure illustrates the Secure Protocol Components.

```
+-------------------+ +-------------------+ +-----------------+
|Connection-Oriented| |  Connectionless   | |    Multicast    |
|   Peer-to-Peer    | |   Peer-to-Peer    | |                 |
+-------------------+ +-------------------+ +-----------------+
```

### 3.5.1 Function

Secure protocols provide protected data transfer between communicating partners without requiring any calls to security services. Applications using secure protocols may have to specify a desired quality of protection before initiating a secure protocol exchange.

### 3.5.2 Protocols

Examples of secure protocols include:

- Connection-oriented peer-to-peer: Secure RPC, SSL, SHTTP, OMG SECIOP
- Connectionless peer-to-peer: IPSec, secure e-mail
- Connectionless multicast: Secure e-mail

### 3.5.3 Interfaces

Each secure protocol typically has its own interface.

### 3.5.4 Profiles

It is not yet clear whether profiles will be established for which Web transaction security protocols (e.g. SHTTP, HTTP-over-GSSAPI, etc...) should be used in which contexts.

### 3.5.5 Negotiation

The APKI working group feels that negotiation of secure protocols is outside the scope of the Public-Key (or even Security) Infrastructure effort.

## 3.6 System Security Enabling Components

The following figure illustrates the System Security Enabling Components.

```
+---------+ +--------------+ +------------+
|         | |  Credential  | |  Security  |
|  Logon  | | Acquisition  | |   Context  |
|         | |              | | Management |
+---------+ +--------------+ +------------+
```

### 3.6.1 Function

System functions (for example, Operating System functions) are needed to support user logon, user credential acquisition, and association of security state information with user processes and threads. For example, once a user has acquired credentials by authenticating himself to a smartcard, that user's processes should be able to use the smartcard interface to sign data using a private key stored on the smartcard. This will only be possible (and secure) if the system has maintained security state information associating the user's processes with the handle returned when the user authenticated himself to the smartcard.

It is not anticipated that the Internet Public-Key infrastructure will define any interfaces, protocols, profiles, or negotiation mechanisms in the area of System Security Enabling Services.

### 3.7 Security Policy Services Components

The following figure illustrates the Security Policy Service Components.

```
+----------+ +----------+
| Privilege| |  Access  |
|Services &| |  Control |
|Delegation| |          |
+----------+ +----------+
|User Info |
|Management|
|(Registry)|
+----------+
```

#### 3.7.1 Function

Security Policy Services manage information about users' (and other principals') privileges and resource access control policies, and make access control decisions based on that information.

#### 3.7.2 Protocols

Formats for privilege attribute tokens to be transported within secure protocols will need to be standardized. The most prominent existing privilege attribute format definitions today are those defined by ANSI X9, OSF DCE, SESAME, and the OMG CORBASEC standard. Privileges could be carried in X.509v3 certificate extensions, or in separate privilege attribute tokens.

#### 3.7.3 Interfaces

It is not anticipated that the Internet Public-Key Infrastructure will define interfaces to privilege attribute services or access control services.

#### 3.7.4 Profiles

Interoperation of systems in differing security management domains will require standardization of privilege attribute types and of the semantics of values of those types. No proposed standard profile for privilege attributes exists today.

### 3.7.5 Negotiation

<<TBD>>

## 3.8 Supporting Services Components

The following figure lists the Supporting Services Components.

```
+----------+ +---------+ +------------+
| Security | |  Time   | | Directory &|
| Auditing | | Service | |Subscription|
| Service  | |         | |  Services  |
+----------+ +---------+ +------------+
```

### 3.8.1 Function

These components provide functions required by the security services or required for secure operation of a networked system; however they do not enforce security policies.

### 3.8.2 Protocols

<<TBD>>

### 3.8.3 Interfaces

<<TBD>>

### 3.8.4 Profiles

<<TBD>>

### 3.8.5 Negotiation

<<Not germane to this document?>>

# 4. Hardware Security Devices in the Architecture

The architecture is intended to support at least two kinds of hardware security devices:

- Security Tokens

This class of devices includes smartcards, memory cards, time-synchronized tokens, and challenge-response tokens. These devices may provide crypto primitives and services, Virtual Smartcard services, and authentication functions.

Smartcards are assumed by the architecture to provide Virtual Smartcard Services. They will also frequently also provide at least the "Key Activation" and "Signing" components of Crypto Services; they may also provide other Crypto Services.

Memory cards provide only storage; Virtual Smartcard services involving state maintenance (e.g. key activation) or cryptography will have to be provided by the memory card's software drivers. The figure below illustrates how smartcards and memory cards can be used to support the Virtual Smartcard services:

```
     +-------------------------------------+
     |     Virtual Smartcard Interface     |
     +-------------------------------------+
       |           |                 |
       |   +---------------+  +-----------+
       |   |Memory Card    |  |           |
       |   |Personal Crypto|  |           |
       |   |Module         |  | Software  |
       |   +---------------+  | Personal  |
       |         |        |   | Security  |
     +-----------------+  |   |  Module   |
     | Security Card   |  |   |           |
     | Access Interface|  |   |           |
     +-----------------+  |   |           |
         |         |      |   |           |
     +-------+  +------+  |   |           |
     | Smart |  |Memory|  |   +-----------+
     | Card  |  | Card |  |          |
     +- - - -+  +------+ +----------------+
     | Crypto|           | Cryptographic  |
     | Funcs |           |   Services     |
     +-------+           +----------------+
```

Time-synchronized and challenge-response tokens provide only authentication functionality, and will

typically be integrated into the architecture through modifications to the System Security Enabling Services (particularly the "Logon" and "Obtain Credentials" components of those services).

- Cryptographic Modules

  This class of devices includes chipsets, bus-connected cryptographic adaptors, and remote cryptographic servers providing crypto primitives and services, but not providing user authentication functions.

  Cryptographic modules are assumed by the architecture to provide the full range of Crypto Services (and they may provide direct access to some Crypto Primitives for the convenience of designers of new Crypto Services).

## 5. Relationship to Other Standards and Architectures

### 5.1 ISO 7498-2

<<TBD>>

### 5.2 IETF IPKI Drafts

This document anticipates adoption (mutatis mutandis) of the four IPKI drafts as Internet RFCs, and seeks to define the larger architectural context into which those drafts fit.

### 5.3 X/Open XDSF

<<TBD>>

### 5.4 ECMA TR-46

<<TBD>>

### 5.5 RSA PKCS Standards

This architecture assumes support for the following PCKS standards: Cryptographic message syntax; signature

formats (PCKS-7) Smartcard function access (PKCS-11)

## 6. Example Applications of the Architecture

<<This section TBD by various people>>

### 6.1 OSF DCE 1.1

### 6.2 SESAME

### 6.3 Nortel Entrust

### 6.4 OMG CORBA

### 6.5 Lotus Notes

### 6.6 Novell Netware

## 7. Glossary

<<TBD>> Notes:

1. We use "confidentiality" and "privacy" interchangeably.
2. "Secret-key cryptography" is used to mean cryptography using a symmetric-key algorithm; "public-key" cryptography has the usual meaning; "private key" is used only to describe the private (secret) half of a key-pair generated for use with a public-key cryptographic system.

## 8. Security Considerations

Security issues are discussed throughout this document.

## 9. References

[1] draft-ietf-pkix-ipki-part1-00.txt.

This document describes profiles for use of X.509 certificates and certificate revocation lists (CRLs) and their respective extension fields in the Internet

environment

[2] draft-ietf-pkix-ipki-part3-00.txt.

This document describes protocols for certificate management in the Internet environment

[3] Internet RFC 1508.

This document describes the GSS-API interface, which provides integrity and privacy services for session-oriented messages

[4] draft-ietf-wts-gssapi-00.txt.

This document describes how to use GSS-API to protect Web transactions (HTTP protocol exchanges, in particular)

[5] draft-ietf-cat-idup-gss-04.txt.

This document describes the IDUP-GSS-API interface, which provides integrity and privacy services for store-and-forward messages, and non-repudiation services.

[6] draft-ietf-cat-spkmgss-06.txt.

This document describes how to use the SPKM protocol under a GSS-API interface

[7] draft-ietf-cat-sesamemech-00.txt.

This document describes the use of the SESAME protocols under a GSS-API interface.

[8] draft-ietf-cat-snego-01.txt.

This document describes a proposed mechanism negotiation preamble protocol for use by protocol partners wishing to use GSS-API to establish a secure association.

[9] X/Open Distributed Security Framework.

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

[11] ECMA TR/46 "Security in Open Systems: A Security Framework".

[12] The SSL Protocol v3.

Describes version 3 of the SSL protocol; available from Netscape Web site

AUTHOR'S ADDRESS
Bob Blakley
IBM
Mail Stop 9132
11400 Burnet Road
Austin, TX 78758 USA

Phone: +1 512 838 8133
FAX : +1 512 838 0156

email: blakley@vnet.ibm.com