第 16 部

OSI ディレクトリサービスと ODA

第 1 章

1994 年度の活動概要

1.1 ISODE WG と ISODE パッケージ

WIDE ISODE ワーキンググループ (ISODE WG) は、ネットワークプロトコルの共存と移行に関する技術の蓄積と研究を目的としており、現在は TCP/IP と OSI を対象とした研究・実験を行なっている。

ISODE WG はその研究を行うに当たって、OSI プロトコルのプラットホームとして ISODE パッケージを使用している。ISODE は既存の TCP/IP 上に OSI トランスポート層を提供するものであり、OSI セッション層、OSI プレゼンテーション層、OSI アプリケーション層を構成する。

現在の ISODE パッケージは、ISODE Consortium (IC) によって製品化の作業が行われており、OSI ディレクトリサービス、MHS (メッセージハンドリングシステム)、SNMP、FTAM が含まれている。最新のバージョンは IC 2.2 (ISODE 12.0) である。

IC 版の ISODE パッケージは IC 参加組織にのみ配布されている。WIDE プロジェクトは、最新の OSI 技術を元に研究をするために、1993 年度から IC に参加し、IC 版の ISODE パッケージを使用しているが、IC に参加していない組織にその成果を還元するため、フリーソフトウェアである ISODE 8.0 も使用している。

1.2 研究・実験内容

1994 年度は、昨年度に引き続いての OSI ディレクトリサービスの研究と、ODA についての研究を行った。具体的な実験・研究内容は以下の通りである。

- 昨年度検討した OSI ディレクトリサービスにおける国際化の仕様に基づいて、その実装を行った。
- ディレクトリサービスを用いた情報検索の仕組みとして WWW の情報検索機構を研究し実装した。
- Transport Service Bridge による OSI ディレクトリの運用実験を行った。

- セキュリティを強化した ODA のユーザエージェントの研究を行った。

第 2 章

国際化 OSI ディレクトリサービス (X.500) の実装

2.1 はじめに

広域分散環境において、ディレクトリサービスは重要なアプリケーションサービスである。基本的に、ディレクトリサービスはネットワークアプリケーションが必要とする各種の情報を格納しておくものであるが、それ以外にも様々な情報を格納し提供できる。中には人が直接利用する情報もある。そのため、各種の情報を各国の言語で格納できるようにする事は、ディレクトリサービスを利用する上で必須と言える。

WIDE ISODE ワーキンググループでは、今まで OSI ディレクトリサービスを用いて各種の実験・研究を行ってきた。その中にはディレクトリサービスの日本語化の試みもあった [110]。ただし、この日本語化は不完全な実装であったため、より本格的な実装が望まれていた。その後本格的な実装を検討するにあたって、日本語だけではなく、多国語を同時に取り扱う事ができる方がより望ましい事から、国際化 (多国語化) の仕様を検討した [111]。

2.2 国際化の方針

OSI ディレクトリサービスを国際化するには、次の事ができるようにしなければならない。

1. 多国語の属性値を入力し、保存し、表示し、検索が行える。
2. 属性名を指定された言語で表示できる。

1 については、OSI ディレクトリを構成する DSA (Directory System Agent) と DUA (Directory User Agent) の両方について、多国語の文字を取り扱えるようにする事を意味する。一方、2 は、DUA において、LOCALE 等で表示すべき言語を指定し、その言語で表現された属性名を表示できるようにすればよい。

DSA と DUA の多国語文字の取り扱いについては、以前の仕様 [111] の検討に従い、その方針を以下のようにする。

- 属性値として多国語を使用できるのは、その属性文法として T.61 テレックス文字列を使用しているものとする。

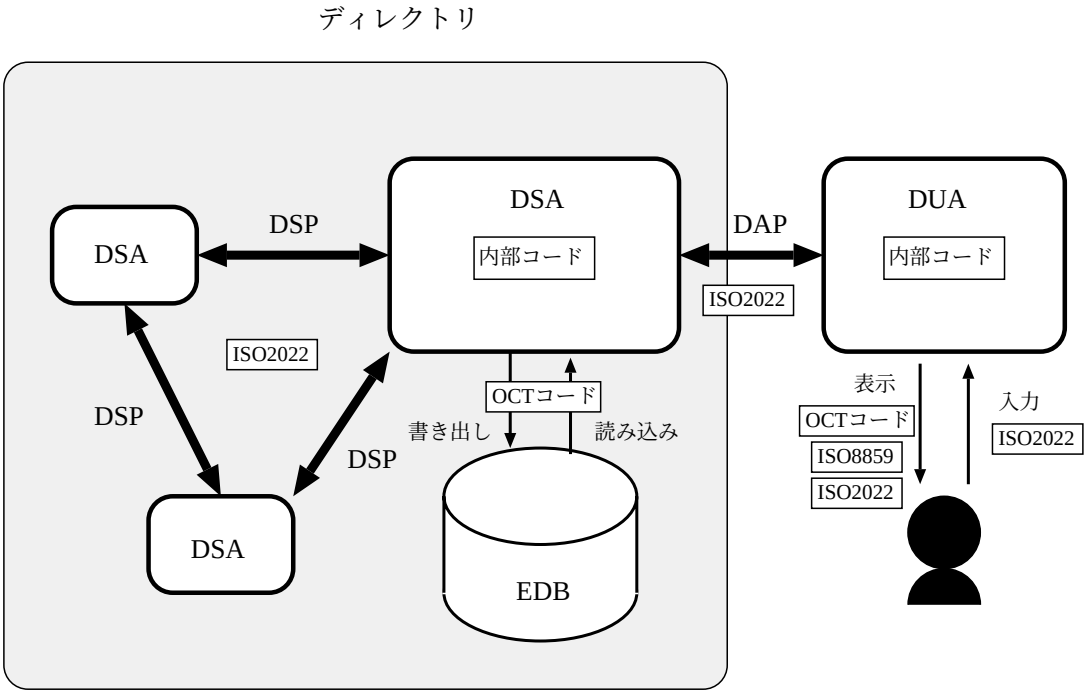

図 2.1: ディレクトリと文字コード

- DSA, DUA 間の通信用の文字コードとして T.61 テレックス文字列内で ISO2022 を取り扱えるように拡張した物を使用する。
- DSA, DUA 内の内部表現形式として、当面 Mule の内部表現形式 [112] を使用する。
- DUA への入力や DUA による表示は ISO2022 の 7 ビット環境用の符号化方式を使用する。
- DSA が保持する情報のファイル上の文字コードについては支障がない限り変更しない。

これにより、国際化 OSI ディレクトリで使用される文字コードは図 2.1 のようになる。

2.3 国際化 OSI ディレクトリサービスの実装

この方針に従い、ISODE コンソーシアム版の ISODE パッケージに含まれる QUIPU パッケージを使用して、国際化 OSI ディレクトリサービスの実装を行った。今回の実装は、DSA および DUA で、属性値に多国語を使用できるようにするためだけで、属性名の多国語での表示については、実装を行っていない。

2.3.1　実装対象ソフトウェア

使用した ISODE パッケージのバージョンは IC-R2.0v3 (ISODE 11.0 version 3) である。このバージョンでは、内部文字コードとして 8 ビットコードをサポートしており、ISO8859-1 によるヨーロッパ言語のサポートが既に実装されている。

ただし、ISO8859-1 による表示を行うためには、DUA 側でその指定を行わなければならない。実際には、DUA の起動時の初期設定ファイルである dsaptailor において`ch_set iso8859-1` という文字コードセットの指定を行う。この指定を行わないと、“`Espa\c4na`” のように該当部分が `\xx` という形で表示される事となる。

国際化された QUIPU においても、この方式を採り入れる事とした。これにより、多国語対応になっていない DUA において、多国語の属性値をそのまま表示する事によるトラブルを防止する事ができる。今回の実装では dsaptailor 中の `ch_set` の値として `iso2022` を追加し、この指定がされている場合にのみ ISO2022 による多国語を表示するようにした。

2.3.2　実装内容

QUIPU を多国語対応にするために、以下の改造を行った。

- QUIPU のライブラリ内の T.61 文字列の外部表現形式と内部表現形式の変換部分において、ISO2022 とその内部表現形式の変換を行うようにした。
- QUIPU のライブラリ内の T.61 文字列の表示部分において、`ch_set` の値が `iso2022` の場合にそのまま表示するようにした。

QUIPU の DSA と DUA は同じライブラリを使用しているため、この改造でどちらも多国語文字列を扱う事ができるようになった。国際化 DUA (dish) による日本語の属性値の表示例を図 2.2 に示す。

2.4　考察

2.4.1　サポート文字コードの問題

今回の実装では、入出力の際の文字コードとして ISO2022 の 7 ビット表現形式のみをサポートするようにしてあるため、日本語 EUC やシフト JIS、韓国語 EUC などの文字コード環境下では利用しにくい。

これについては、dsaptailor 等の DUA の初期設定ファイルで、各言語固有の表現形式を指定できるようにする事で解決できるであろう。

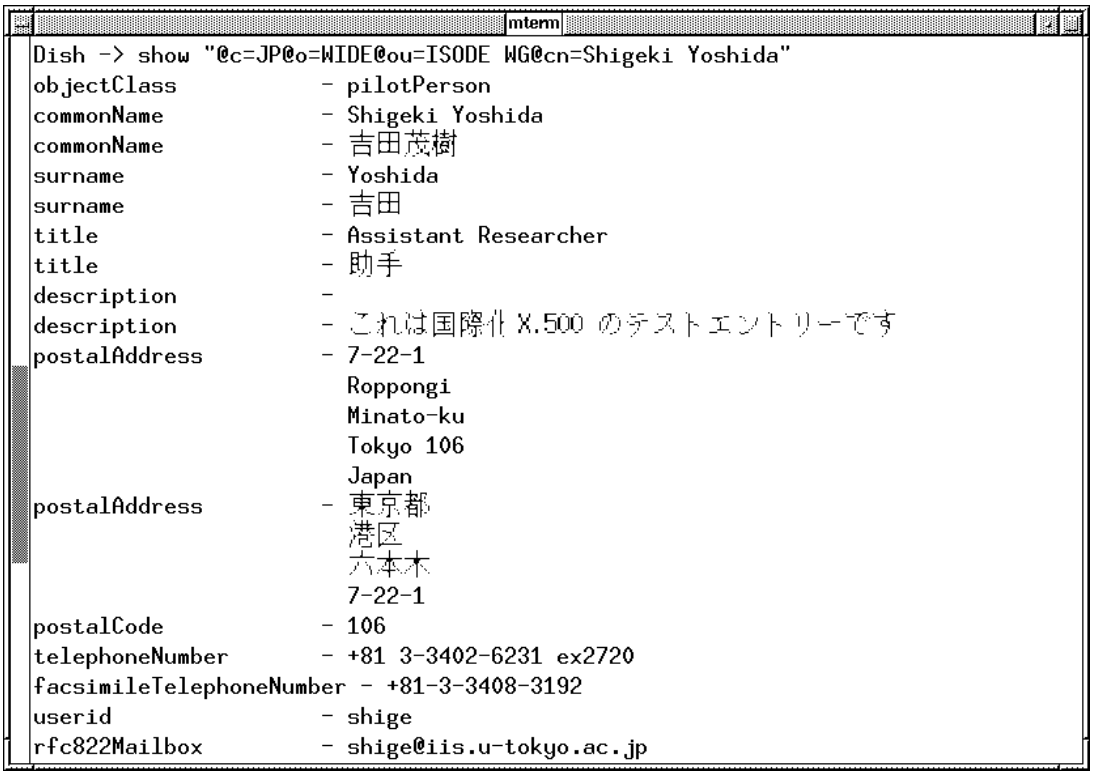

図 2.2: 国際化 DUA (dish) の表示例

2.4.2 サポート属性の問題

今回の実装では、T.61 文字列を選択できる属性文法を使用している属性のみで多国語の文字が利用できるようになっている。そのため、その属性文法が T.61 文字列を選択できない属性では多国語を扱う事ができない。本来は属性文法で多国語文字列を選択できるようになっているべきである。

これについては、OSI ディレクトリサービスの 1992 年版の規格で、新たに ISO10646 を選択できる属性文法が定義されているため、今後の検討課題としたい。

2.4.3 属性値の種類

図 2.2 では、同じ属性を英語と日本語で用意してあるが、必ずしもこのようにする必要はない。しかし、ディレクトリサービスが世界中から利用される事を考えると、各国の言語で情報を提供する以外に、共通言語としての英語の情報も提供するのが望ましいであろう。

さらに、可能であれば同じ情報を様々な言語で提供できるとよりよい。その場合、DUA 側では表示言語を指定して、その言語の情報があればそれを、なければ英語の情報を表示できるようになっていると、表示が効率的になるであろう。これを、属性名の言語の指定

と組み合わせれば、国際化 DUA をベースにして特定の言語対応にする事ができる。

2.4.4　DUA の問題

ディレクトリに多国語の文字コードで表現された情報が格納されている場合、それを全て表示するためには、多国語対応の DUA (または端末) が必要となる。dish のように文字端末用の DUA であれば、多国語の文字コードを同時に扱う事ができる Mule や mterm などを利用する事で、その環境を用意する事ができる。

しかし、実際には前述のように特定の言語で提供されている情報のみが必要である場合が多いであろう。その時にはその言語だけを表示できる DUA または端末があればよい。ただし、その DUA または端末で他の言語が表示された時でも、正常に動作し続けなければならない。

2.4.5　属性値入力時の問題

属性値は DUA を通じて入力するが、QUIPU の dish では一時ファイルを作成し、それをエディタで編集する事で行う。この一時ファイルの作成と解釈は、DSA が情報を保持する EDB (Entry Data Block) ファイルの作成と解釈と同じライブラリを使っている。そのため、一旦情報を書き込んで再度編集すると、図 2.3 のような EDB のファイル内の形式と同じものを編集する事となる。

これについては、EDB ファイル内も ISO2022 形式で情報を保存するようにすれば解決する。

2.5　まとめと今後の予定

OSI ディレクトリサービスを国際化し、多くの属性で多国語の属性値を扱う事ができるようになった。

今後は今回実装をしていない以下の国際化を引続き行なう。

- EDB ファイルと DUA 用ファイルの切り分け
- 多国語による検索。
- DUA による属性名の多国語表示。

その後、国際化していない OSI ディレクトリサービスとの混在環境での運用についての実験を行う予定である。

```
cn=Shigeki Yoshida
objectClass= top & person & pilotObject & newPilotPerson & quipuObject & pilotPerson
cn= Shigeki Yoshida &\
{T.61}\1b\24\405HEDLP<y\1b(B
sn= Yoshida &\
{T.61}\1b\24\405HED\1b(B
title= Assistant Researcher &\
{T.61}\1b\24\40=u<j\1b(B
description= {T.61}\1b\24\40\243\24l\24O9q:]2=\1b(J X.500 \1b\24\40\24N\25F\259\25H\
\25(\25s\25H\25j!<\24G\249\1b(B
description= {T.61}\1b\24AOBCfJGJd3vHk1‘Bk!\23\1b(B
postalAddress= 7-22-1 $ Roppongi $ Minato-ku $ Tokyo 106 $ Japan
postalAddress= {T.61}\1b\24\40El5˜ET\1b(B $ {T.61}\1b\24\409A6h\1b(B $ {T.61}\1b\24\
\40O;K\\LZ\1b(B $ 7-22-1
postalCode= 106
telephoneNumber= +81 3-3402-6231 ex2720
facsimileTelephoneNumber= +81-3-3408-3192
userPassword= {CRYPT}NJOHOBMG
uid= shige
mail= shige\40iis.u-tokyo.ac.jp
```

図 2.3: EDB ファイル内の多国語表現

第 3 章

X.500 を用いた WWW の検索機構の構築

3.1 はじめに

ハイパーテキスト型の情報提供システムである WWW (World Wide Web) [113] は、近年非常な勢いでその利用者が増えている。これに伴い、WWW によって提供される情報も膨大なものとなっている。

しかし、現在の実装ではネットワークに点在する情報は管理者が手でつながりを記述し、利用者はそのつながりを見つけ出すことでしかたどることはできない。そして断片的な情報から目的の情報を検索し見つけ出す事は非常に困難である。

3.2 情報検索の実装例

WWW の情報を検索する仕組みとしては以下のような方法が考案されている。

3.2.1 Virtual Library

様々な WWW の情報をあつめ、図書館のような分類を行いそれぞれのサーバへのリンクを用意し、検索・閲覧が可能になっている。

多くの場合、ネットワーク上に無数に存在する WWW サーバを人が探したり情報提供を受ける事で、それらを人の手でまとめ一ヶ所で集中管理し、検索を可能にしている。

3.2.2 WWW Robot

Robot は WWW 等から得られる情報とポインタをどのように探索し検索するかを記述し、人が手で情報検索を行わずに情報を得る事を可能にしている。

3.2.3 これらの問題点

情報検索の実装例は他にもあるが、いくつかの欠点があげられる。

- Library 上と、実際のサーバ上との情報に矛盾が生じる可能性がある。
- いつまで Library の管理や提供がされるかの保証がない。
- リンクの先の情報が必ずあるとは限らない。
- 検索のために次々に WWW サーバにアクセスしていくことは負荷が大きく、ループなども起きやすい。

3.3 設計と実装

本研究では前述の問題点を考慮し X.500 を使った情報検索機構の設計と実装を行った。

3.3.1 X.500 の概要

X.500 (OSI ディレクトリサービス) [114] のモデルの特徴を以下にあげる。

- ディレクトリへのアクセスは DUA (Directory User Agent) が 1 つ以上の DSA (Directory System Agent) と通信することで行われる。
- ディレクトリの情報の単位である「エントリ」は属性の集合となっており、各属性は 1 つの型と 1 つ以上の値から構成されている。
- エントリは木構造を用いて関連づけられている。
- 情報木は世界的に一意に見える。

3.3.2 方針

WWW サーバから提供される情報の単位である「ページ」は HTML (Hyper Text Markup Language)[115] で記述されている。HTML は HREF=“http://www.wide.ad.jp/” 等と記述することで他の様々な情報の参照やつながりの記述を可能にしている。この参照するページとされるページの間には強い関係があることが多い。この参照されているつながりを X.500 上の木構造に当てはめることで効率の良い検索が可能となる。

また、HREF が他のサーバを示すような場合は X.500 上に新たな属性を追加し、その属性値をもって他のエントリを参照するようなグラフ構造を作成する。

また、情報の分類に対する要求として、以下の 2 点があげられる。

1. 世界中の情報を整理したもの。
   Virtual Library 的な分類。

2. 個人的興味のあるものを集めたもの。
   Mosaic の hotlist 的な分類。個人的な利用。

上の 2 点は規模や構築する方法は違うかもしれないが、情報の検索機構から考えると同質にみる事が可能である。この 2 つの要求を実現する事は、WWW の情報木を X.500 上に構築するときに、その木を構築するエントリの位置を変える事で可能である。

3.3.3　情報木の構築法

情報木上では、一つの WWW のページに一つのエントリを対応させる。基本的に URL (Uniform Resource Locators)[116] から得るディレクトリの木構造をそのまま X.500 上の木構造に反映させる。しかし、他のサーバの情報を参照しているような場合には、その下に木構造を作るのではなく、別のエントリからそのサーバの木構造を作成し、属性値としてその対応するエントリを参照するようにする。

3.3.4　属性の定義

HTML で記述された情報を X.500 のエントリに反映するために、エントリの属性として以下のものを追加した。

- WWW のページの URL を X.500 上のエントリの CommonName(cn) の属性値とした。
- ページの TITLE を surname(sn) の属性値とした。
- ページから提供されるキーワードとなりうる情報、<A〜/A>などを登録する属性として、htmlkeyword を定義した。
- 別のエントリの情報を参照する属性として、htmlref を定義した。
- 実際の URL そのものを表す属性として、url を定義した。
- HTML で記述されたページを登録するエントリとして html を定義した。

実際に定義されたエントリの表示を図 3.1に示す。不必要なデータは省略している。また、html はファイルとして登録しているので、16 進でダンプされた結果になっている。また、X.500 の情報木上で別のエントリを指すときのモデルを 3.2に示す。

3.3.5　検索アルゴリズム

WWW に登録されている情報の検索には、X.500 上の検索システムを利用する。

あらかじめ X.500 上のいくつかのエントリの特徴が分かっていればそのエントリから検索を始めることで、目的にあった検索が出来る。また、X.500 上に構築された WWW の情

```
Dish -> list
5  organizationalUnitName='smiley.cs.uec.ac.jp/pdl/'
6  commonName='smiley.cs.uec.ac.jp/Index.html'
Dish -> showentry 6
objectClass           - newPilotPerson & quipuObject & html
commonName            - '//smiley.cs.uec.ac.jp/Index.html'
surname               - PDL Home Page
htmlkeyword           - PDL Members (In Japanese)
htmlkeyword           - WIDE Project
htmlkeyword           - shuji@cs.uec.ac.jp
url                   - 'http://smiley.cs.uec.ac.jp/Index.html'
html                  - [6] '3c484541443e0a203c5449544c453e50....
             :
             :
Dish -> list 6
7  commonName='smiley.cs.uec.ac.jp/pdl/Index.html'
8  commonName='smiley.cs.uec.ac.jp/pdl/kei.html'
             :
             :
```

図 3.1: 実際のエントリの表示例

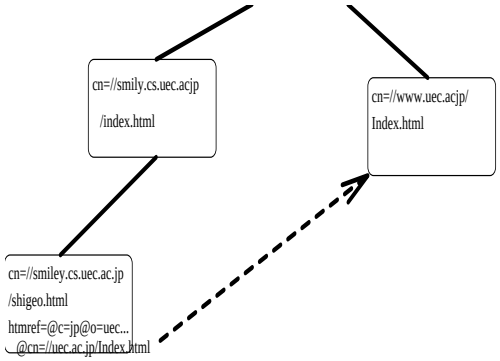

図 3.2: 別のエントリの情報を参照した例

報木全体を検索可能とするエントリから検索を行うことで、非常に効率は悪いが木構造のある部分全体への検索が可能になる。

検索自体には WWW サーバへのアクセスは無いため WWW サーバやネットワークに対する負荷はなくなる。また、検索に対するループなどは X.500 上のエントリが一意に定義されるので解決出来る。

3.3.6　実装

以上の設計に基づいて、情報木を作成するプログラムと X.500 の DSA にアクセスし検索を行う DUA プログラム・ライブラリを実装した。

情報木作成プログラム

情報木作成プログラムは Perl 言語で記述をした。プログラム構造的には WWW Robot と同質なものになっており、WWW サーバにアクセスし X.500 の情報に変換を行って情報を次々に更新する。

実際の運用に関しては cron などで定期的に行えば実際のサーバの情報との矛盾が生じることは少ないと考えられる。

DUA

DUA が DSA にアクセスする時には DAP (Directory Access Protocol) または LDAP (Lightweight Directory Access Protocol) を用いて情報検索を行う。

WWW の情報木を検索するために、エントリ中の属性で定義されている参照部分を考慮した検索プロトコルを作成した。これは、エントリの特定の属性を検索し、属性値が定義されているならばそこで定義されているエントリをさらに検索する。

また、Mosaic などのアプリケーションに容易に組み込めるようなライブラリを作成する必要がある。今回は簡易版のライブラリを作成した。

3.3.7　問題点

X.500 上に情報を構築するために、結局は WWW Robot と同様なプログラムで情報木を作成している。これは、WWW サーバやネットワークに負荷をかけている。しかし、ユーザが個別にアクセスするのとは異なり、共有して使う情報木を作成するためなので、負荷は軽いと予測している。また、WWW サーバ自体に情報木を作成する機構を組み込むことでこの負荷を無くすことも可能であると考えている。

検索対象の位置が不明の場合は、WWW の情報木全体を検索する事になり、かなりの時間がかかることが考えられる。

また、情報のつながりは HTML で記述されたつながりをそのまま反映しているため、WWW サーバによってはあまり関連性の強くない情報が記述されていることもある。関連性が強い弱いをキーワードの比較などによって考慮し、関連性の薄いものはつながりを切ったり、関連性の強いものは新たなつながりを作成する機能が必要である。

3.4 広域への運用課題

現在、WWW の情報木の管理は電気通信大学の DSA 一台でおこなっており、実際の情報木の大きさも小規模なものとなっている。これを実際に広域で運用する場合、複数の DSA で管理を行うことになる。その場合に、それぞれの DSA で管理する情報木の構成と協調を考慮しなくてはならない。

- 情報検索木自体を X.500 のどこに作成するか
- 各組織で管理をしている情報木の相互参照を効率よく管理するにはどうするか

3.5 まとめと今後の予定

X.500 を WWW の情報検索機構として運用するための情報木の定義と実装を行った。それの検索が出来るプロトコルの実装を行った。

今後は現在実現していない以下の事を進めて行く予定である。

- これらの検索システムを実際のアプリケーションに組み込み運用する
- 広域で分散して管理する場合の問題点を考慮した情報木の設計と実装を行う
- これらの検索システムの有効性を評価する

第 4 章

Transport Service Bridge による OSI ディレクトリの運用

4.1 はじめに

現在の OSI ディレクトリサービス (X.500)[117] の全国実験では、DSA(Directory System Agent) は Internet に直接アクセス可能なホストに置くことを前提としている。さらに広範囲な運用にあたっては、ネットワーク層で直接接続可能でなければいけないという点は大きな制約である。

1993 年度は、DSA を Internet と直接通信ができないホストに設置する試みとして、Relay DSA による方法の実験を行った。しかし、Relay DSA による方法では以下の様な問題が存在した。

- 連鎖によるオペレーションの制限
- DSA 自身の情報の更新

これらは、ある程度運用によって回避が可能であったが、非常に強い制約であった。

そこで、今回はアプリケーション・サービスではなく、下位のトランスポート・サービスでの中継の方法による解決を試み、良好な結果を得たのでここに報告する。

4.2 OSI ネットワークと Transport Service Bridge

4.2.1 OSI のネットワーク層の問題点

IP や XNS 等ではネットワーク層はコネクションレスの、単一のサービスが規定されている。この単一のネットワーク・サービスによって、インターネットワーク[1]に接続した 2 つのノード間の通信ができている。

一方、OSI では次の 2 つのサービスの両方が規定されている。

[1]固有名詞としての Internet ではなく、一般に複数のネットワークの相互接続された形態

- コネクション指向のサービス
- コネクションレスのサービス

このどちらか一方が使用されている場合は問題ないが、両者が混在した場合の食い違いを解決するための方法はネットワーク層のサービスとしては用意されていない。すなわちネットワーク層の目的である、インターネットワーク上の任意の 2 つのノード間の通信を提供できないことになる。これは、OSI のネットワーク・サービスは当初コネクション指向のものだけで、後からコネクションレスのサービスを単純に追加しただけの、標準を制定するときの妥協によるものであるためである。

OSI のネットワーク層の問題点は、ネットワーク層より上位の層で中継することによって解決できる。これは、アプリケーションからの透過性から、すぐ上のトランスポート層で中継が一般的で Transport Service Bridge と呼ばれる。

4.2.2 OSI コミュニティ

Transport Service Bridge により、異なるネットワーク・サービス間での通信が可能となる。一方、アプリケーション層において通信相手はプレゼンテーション・アドレスで指定する。アプリケーション・サービスが通信できるためには、プレゼンテーション・アドレスに含まれるトランスポート層以下のアドレスを共通に扱えなければならない。

この目的を達成するための方法が RFC1277[118] で提案されており、TCP/IP、X.25、CLNP などの様々なネットワークのアドレスを表現できる。この方式では、同一のトランスポート層やネットワーク層を使用し、相互に通信が可能なアドレスには共通な識別部分を割り当てる。これにより、直接通信できるネットワークかどうかが判別できる。

ISODE ではこの識別部分を OSI コミュニティと呼んでいる。同じ TCP/IP であっても、firewall などのため直接通信できないプライベート・ネットワークは異なるネットワーク・サービスとして、新しい OSI コミュニティとすることもできる。

4.3 ISODE の tsbridge コマンド

ISODE の tsbridge コマンドは Transport Service Bridge を実装したものである。
これは、TCP/IP、X.25、CLNP+TP4 といったネットワーク層で直接通信ができない、異なるネットワークに接続したホスト上で中継を行う。単純に特定のトランスポート・アドレスへのアクセスを別のトランスポート・アドレスに中継する機能と、任意のトランスポート・アドレスに中継する機能がある。

4.3.1　特定トランスポート・アドレスへの中継

単純な動作のモードでアクセスを受け付けるトランスポート・アドレスと、中継先のトランスポート・アドレスを指定する。この方式は透過モードとも呼ばれる。すなわち、tsbridge

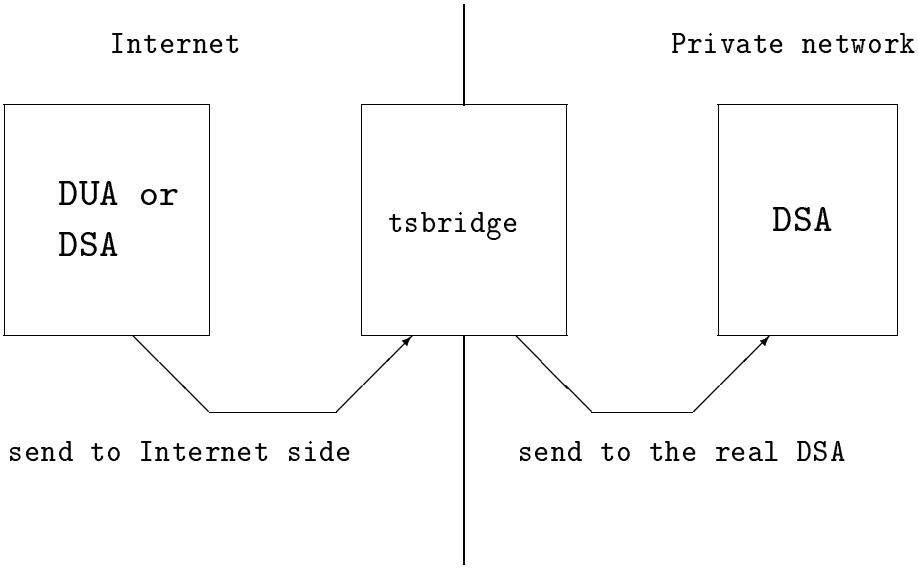

図 4.1: tsbridge の透過モード

の両側とも特に tsbridge の存在を意識しないで通信が可能である。

今回は、Internet からプライベート・ネットワークの DSA へのアクセスを中継するために使用している。

4.3.2　任意のトランスポート・アドレスへの中継

トランスポート・アドレスは、ネットワーク・アドレスとセレクタと呼ばれるトランスポート層での識別子から構成されている。TCP のポートが 16 ビットの符号無し整数と決められているのと異なり、セレクタには任意のデータ[2]を格納できる。すなわち、セレクタに他のコミュニティのトランスポート・アドレスをエンコードすることも可能である。これを利用して、tsbridge の存在を前提とした、別の OSI コミュニティのトランスポート・アドレスとの通信が可能となる。

この方式では、tsbridge を利用したトランスポート・コネクションを開始する側が、tsbridge を前提とした宛先トランスポート・アドレスの指定を行う必要があるが、他の OSI コミュ

[2]実際は実装規約で決められる。

ニティの任意のトランスポート・アドレスに接続することが可能である。これを一般的なモードと呼ぶ。

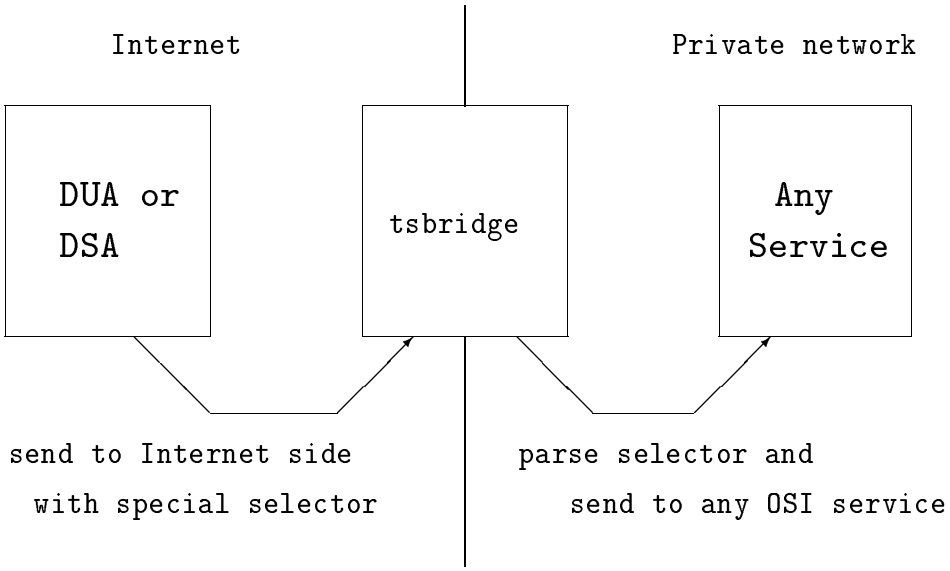

図 4.2: tsbridge の一般的なモード

次のような場合を考える。

- OSI コミュニティA にノード x がある。
- OSI コミュニティB にノード y がある。
- OSI コミュニティA と OSI コミュニティB の両方に tsbridge、 t が接続している。

x が t を使って、 y と通信する流れは、以下の通りである。

1. x は t への TSDU 中の宛先のトランスポート・アドレスのセレクタ部に、y のトランスポート・アドレスに埋め込み、 t に送る。
2. x から TSDU を受け取った t は、次のように y 宛 TSDU を作成する。
   (a) 受け取った TSDU の宛先アドレスのセレクタから、y の宛先アドレスを取り出して y 宛 TSDU の宛先アドレスとする。
   (b) 受け取った TSDU の送信元アドレスを、y 宛 TSDU の送信元アドレスのセレクタ部に埋め込む。

(c) 受け取った TSDU のデータを、 y 宛 TSDU のデータにコピーする。

3. 作成した TSDU を y に送る。

y が t から受け取った TSDU の送信元アドレスには、既に x のアドレスがセレクタ部に含まれた形となっている。従って、 y は単純に t に向けて x 宛の TSDU を送れば、t 逆の処理をが行って x に TSDU が届く。

4.4　設定

4.4.1　実験環境

FXIS は富士ゼロックスと共通の社内 TCP/IP ネットワークを構成していて、firewall を介して WIDE Internet と接続している。Internet に対するプライベートな TCP/IP ネットワークに富士ゼロックスは129.249 の IP アドレスを、FXIS は131.221 の IP アドレスを使用している。このプライベートな TCP/IP ネットワーク内の通信に特に制限はないが、129.249.88 のサブネットだけは WIDE Internet に接続していて、プライベートな TCP/IP ネットワークと直接の通信はできない。ここで、実験の環境は以下の様に簡単なものである。

- c=JP@cn=Leaf Fish は IP アドレスが 131.221.64.1 のホストで動作している。
- 富士ゼロックスと WIDE Internet の両方にアクセス可能な fxwidegw.fujixerox.co.jp で tsbridge を動作させた。このホストは、129.249.88.2 と129.249.38.235 のアドレスを持っている。

4.4.2　OSI コミュニティの設定

OSI コミュニティは、1993 年度の Relay DSA の実験のときに設定したものから、変わりはない。これを表 4.1に示す。

表 4.1: 設定した OSI コミュニティ

OSI コミュニティ	対応するネットワーク・アドレス
Internet	TELEX+00728722+RFC-1006+03+
FujiXeroxTCP	TELEX+07247983+RFC-1006+01+

ここのアドレスの表現は RFC1287[119] に基づいたものである。

4.4.3 tsbridge の立ち上げ

tsbridge は引数に設定ファイルを指定しなければならない。設定ファイルには、中継を行うアドレスの情報を記述する。今回の実験で使用した設定ファイルを図 4.3に示す。最初

```
Internet=129.249.88.2+17103 FujiXeroxTCP=131.221.64.1+17003 -s
FujiXeroxTCP=129.249.38.123+17101
```

図 4.3: tsbridge の設定ファイル

の行は、特定のトランスポート・アドレスへの中継の設定である。第 1 フィールドはコネクションを受け付けるアドレスで、第 2 フィールドは tsbridge からコネクションを行うアドレスである。第 3 フィールド以降は動作を成業するオプションである。

この例では、

1. `Internet=129.249.88.217103+`に来た接続要求を受け付け、
2. `FujiXeroxTCP=131.221.64.117003+`に接続する。

という形で中継を行う。なお、TCP/IP がトランスポート・スタックの場合に、OSI コミュニティ名の後は=に続いて IP アドレスまたはホスト名、+に続いて TCP のポート番号である。

2 行目は、任意のトランスポート・アドレスへの中継を行うための設定である。中継の要求を受け付けるトランスポート・アドレスを記述する。この例では、

1. `Internet=129.249.88.217101+`に来た接続要求を受け付け、
2. 宛先のトランスポート・アドレスのセレクタに含まれている、任意の OSI コミュニティのトランスポート・アドレスへ接続する。

という形で中継を行う。コミュニティ`FujiXeroxTCP` のアドレスだけ書いているため、プライベートな TCP/IP ネットワークから接続要求に基づいた Internet への中継だけを行う。両方向の中継を行うためには、次の様に形で設定を行えば良い。

```
FujiXeroxTCP=129.249.38.123+17101|Internet=129.249.88.2+17209
```

4.4.4 `isotailor` の設定

4.4.2の、任意のトランスポート・アドレスへの中継を行うモードを tsbridge に接続要求を出す側の設定が必要である。これは、実行環境の設定ファイルである`isotailor` で行う。まず始めに、新しい OSI コミュニティを設定している場合の`isotailor` の例を図 4.4に示す。

tsbridge を利用するためには、これを図 4.5の様に変更する。

変更点は以下の通りである。

```
ts_stacks: tcp
ts_interim: FujiXeroxTCP
ts_communities: FujiXeroxTCP
default_tcp_community: FujiXeroxTCP
```

図 4.4: OSI コミュニティを設定した isotailor

```
ts_stacks: tcp
ts_interim: FujiXeroxTCP
ts_communities: FujiXeroxTCP Internet
default_tcp_community: FujiXeroxTCP
tsb_communities: Internet FujiXeroxTCP=129.249.38.123+17101
```

図 4.5: OSI コミュニティを設定した isotailor

- `ts_communities` に、ISODE にデフォルトで定義されている OSI コミュニティである `Internet` を、使用するトランスポート・スタックとして追加した。
- `tsb_communities` で tsbridge を経由させるコミュニティと、tsbridge のアドレスを指定した。

この `isotailor` を設定した OSI コミュニティ `FujiXeroxTCP` のホスト上では、OSI コミュニティ `Internet` に tsbridge を経由して接続できるようになる。

4.4.5　DSA の情報の修正

OSI 上のサービスにアクセスするときは、ディレクトリ・サービスからサービスを受け付けているプレゼンテーション・アドレスを取得し、接続を試みる。

プライベートな TCP/IP ネットワークから Internet にある OSI サービスを利用する場合は 4.4.4の様に、プライベートな TCP/IP ネットワーク側で設定を加えることで可能となる。しかし、逆に Internet から tsbridge を経由して、プライベートな TCP/IP ネットワークにある OSI サービスをアクセスする場合に同様な方法を取ると、すべての OSI コミュニティの情報を Internet 中で共用しなければならない。これは非常に困難であるし、セキュリティの面からも問題である。

また、サーバはディレクトリの情報に基づいて、コネクションを受け付けるプレゼンテーション・アドレスを決定するが、プライベートな TCP/IP ネットワーク上のサービスの場合は、どのようにプレゼンテーション・アドレスをディレクトリに登録するかが問題となる。

- tsbridge が接続を受け付けている Internet のアドレスを登録すると、サーバの動作しているホストと別のホストのアドレスとなってしまう。このため、サーバはコネクションを受け付けるためのアドレスを用意できなくなってしまう。
- サーバの動作しているホストのアドレスで登録すると、ディレクトリから検索した結果は、Internet からアクセスできないアドレスとなってしまう。

ISODE ではディレクトリの`presentationAddress` 属性に加えて、`listenAddress` 属性を用意することで解決している。`listenAddress` 属性は QUIPU で独自に定義したディレクトリの属性の 1 つであり、以下の様に利用する。

1. サーバは提供するサービスに対応するエントリに`listenAddress` 属性があると、これをコネクションを受け付けるアドレスとして使用する。
2. `listenAddress` 属性がない場合は`presentationAddress` 属性を使用する。

このようにして、サーバのプログラムが接続を受け付けるアドレスと、ディレクトリに登録するアドレスを分離している。

以上のサービスは DSA の場合も全く同様である。プライベートな TCP/IP ネットワーク上の DSA である`c=JP@cn=Leaf Fish` に以下の様な変更を行った。

- `listenAddress` 属性を、従来の`presentationAddress` 属性の値で追加した。
- `presentationAddress` 属性に、tsbridge の受け付け用のアドレスを追加した。

これらの属性の値を図 4.6に示す。この図中では、本来は 1 行とすべき記述を複数行で表示している。

```
presentationAddress='0101'H/
                TELEX+00728722+RFC-1006+03+129.249.88.2+17103|
                TELEX+07247983+RFC-1006+01+131.221.64.1+17003
listenAddress= '0101'H/TELEX+07247983+RFC-1006+01+131.221.64.1+17003
```

図 4.6: 変更した DSA の属性

4.5 結果

4.4で述べた変更を行い、`c=JP@cn=Leaf Fish` の運用を行った。結果は、以下の通りである。

- 1993 年度の Relay DSA の実験で問題となった運用上の問題はすべて解消した。

- Relay DSA の連鎖によるディレクトリのオペレーションと比べると、はるかに良好な性能が得られた。

後者については、以下の原因が考えられる。

1. Relay DSA を動作させているマシンが元々性能が高いは言えない、Sun-4/110 であり、その上他の様々なサービスのプログラムも動作させている状況であったこと。
2. 単純なデータのコピーしか行わない tsbridge に比べると、DSA の連鎖によるディレクトリのオペレーションが負荷の重い処理であること。

4.6　結論と課題

Internet と直接通信のできないホストの DSA を、tsbridge を用いて接続する実験を行った。tsbridge による手法は有効な方法であることの検証ができた。

課題としてはセキュリティの面での強化が考えられる。以下に問題と思われる事項は以下の通りである。

- tsbridge は接続を受けた元のアドレスをチェックする機能はない。
- 一般的なモードでは中継先の任意の TCP ポートにアクセスできる。

TCP/IP 上の OSI のサービスを利用する、RFC1006[120] に基づいた方式でないといけないという制約はあるものの、「一般的なモード」の利用には注意が必要と思われる。

第 5 章

Secured ODA Document User Agent

5.1 Introduction

The Office Document Architecture (ODA) [121] standard was developed to interchange documents in an open systems environment. A document is transmitted through a network or with a floppy disk in the form of an encoded octet stream called the Office Document Interchange Format (ODIF). With regard to security, a security addendum [122] to the standard has been published, which gives a specification how a document in the form of the *secured* ODIF should be transferred with security properties such as confidentiality, authenticity and integrity.

University College London (UCL) has been active in both the ODA and the security area. In the context of the Piloting ODA (PODA) series of projects [123], UCL has developed an ODA/Slate compound document editor and the resulting system has been incorporated into X.400 and SMTP message systems [124]. UCL has also taken in part of the PASSWORD Project [125] and developed an OSI security toolkit and various secured applications including X.400 and Privacy Enhanced Mail (PEM) [80] User Agents (UAs). The OSI security toolkit based on the X.509 Authentication Framework [126], OSISEC [127], contains encryption libraries and tools for managers operating Certification Authorities (CAs). The applications based on OSISEC have been implemented and deployed in the PASSWORD Project.

Two ODA utilities have been developed during the PASSWORD Project; while DOCSEC [128] secures a document as a whole, PDOCSEC secures each part of a document individually, and both utilities are integrated with the Slate UA.

In this paper the authors describe security requirements for a document UA in section 2, outline the ODA security in section 3, and describe the security toolkit and the secured document UA in section 4 and 5. Some aspects raised by our work are discussed in section 6, comparison of the ODA security with PEM is given in section 7. Finally section 8 concludes the paper.

5.2 Requirements for Secure Document UA

While only security as a whole document is enough in the case of the interchange between two parties, securing parts of a document is required when the document is interchanged in a group. For example, digital signatures of parts of the document may be demanded since some parts may be written by one author and other parts by another. Encipherment of parts of the document may be required because an author wants to restrict access to the parts to some members of the group. For these reasons,

- confidentiality, integrity and authentication services of parts of a document

are necessary. In addition, the following facilities may be desirable for cooperative authoring:

- addition of a signature to the part which has been already signed by other members (circulation)
- addition/modification/deletion of a part to/of/from an existing document while keeping signatures of other parts valid
- signing content of a part of a document with some other information such as time, location, signer's role

Security services of a whole document can be implemented with DOCSEC [128] or Public-Key Cryptography Standards (PKCS) #7 messages syntax [129]. DOCSEC supports the three security services of a whole document and an arbitrary octet string. RSA Data Security, Inc. has defined PKCS#7, a cryptographic message syntax which supports the same services as well as other syntax such as enciphered content with or without encryption keys. Both have been implemented on OSISEC as filters; there are two programs one of which encodes plain data into secured (enciphered and/or signed) data and the other decodes the secured data into the original plain data.

For the sake of security services of parts of a document, the authors have adopted the ODA security addendum. This addendum supports the above security services of parts of a document, i.e., parts of the document profile and the document body.

Security functions must be implemented to support as many existing document UAs as possible, and the security information which users concern should be displayed, or if necessary, input in a user-friendly manner.

5.3 ODA and Security

5.3.1 ODA Concepts

The ODA standard [121] describes an abstract view of an office document and a document processing model as well as an interchange format of a document.

A document consists of a document profile, generic structures (logical and layout object classes), specific structures (logical and layout objects), styles (layout and presentation styles) and content portions. These components give two views of the document; a logical structure which represents a logical view of the document such that a letter header consists of a date, an addressee and a subject, and a layout structure which represents a layout view of the document such that a letter header page consists of a logo frame and a date frame.

The ODA standard defines three kinds of document form: a processable form with logical structures created after an editing process, a formatted form with layout structures produced by a formatted process, and a formatted processable form with both logical and layout structures produced by a formatted process. An imaging process takes a formatted or formatted processable form and fabricates a final document. All these document forms are encoded into the ODIF in which all components are represented as sets of attributes and interchanged through various media such as message or file transfer services.

5.3.2 Security of ODA Document

In the security addendum [122] to the ODA standard, two concepts of document security are introduced; securing a whole document and parts of a document. While securing a whole document is out of scope of the addendum, it is suggested that the security policy of the domain to which the originator belongs may specify how the whole document should be handled as a single unit. One method of realisation is to use a secured transport mechanism such as a secured X.400 or a secured FTAM. The second concept is securing of parts of a document, i.e., any parts of the document profile or the document body.

The addendum does not specify any particular algorithm or scheme, but it provides the means to protect the parts of the document. The security features defined in the addendum are confidentiality, integrity, authenticity and non-repudiation of origin; each is described below.

1. **Confidentiality**: This property ensures that semantic knowledge of a specified part of a document is made available or disclosed only to authorized receivers, entities or processes. Confidentiality is achieved by encipherment of the part; the originator

enciphers the part of the document and only privileged recipients can decipher and read it.

2. **Integrity**: This property ensures that a specified part of a document has not been changed or destroyed in an unauthorised manner. The privileged recipients can verify that the part has not been altered since the originator sealed it. The certainty provided by this property is limited to a detection of change; the replacement of the whole sealed part and the seal itself cannot be recognized. Nevertheless, by sealing the whole document, tampering with its part can be recognized; moreover recipients may well choose not to trust a document which has not been sealed.

3. **Authenticity**: This property ensures that the privileged recipients can verify the source of a specified part of a document as claimed. Authentication is accomplished when the integrity seal can be determined as having been made by the claimed source.

4. **Non-Repudiation of Origin**: This property ensures that it cannot be denied that a particular source is the originator of a specified part of a document. Non-repudiation of origin can also be achieved by adding a seal to the part.

5.4 OSISEC security toolkit

The current security services in open systems use both symmetric and asymmetric encryption techniques called "digital envelope". For the sake of the confidentiality service, content is enciphered using a randomly generated symmetric key and the key is encrypted using the recipient's asymmetric public key. In order to achieve the authentication, integrity and non-repudiation of origin services, the message digest (checksum) of content is encrypted using the originator's asymmetric private key.

The recipient needs the originator's public key to verify authenticity and integrity of the content. In the X.509 Authentication Framework, a reliable public key can be got from a certificate sealed by a trusted third party called a Certification Authority (CA). While a CA only issues certificates for its users, it is possible for a CA to sign a certificate which contains another CA's public key.

In the PASSWORD Project [125], three kinds of CAs have been established: a single top level CA, policy CAs certified by the top level CA, organizational CAs certified by the policy CAs. The organizational CAs issue certificates for users. All applications are required to attach a complete forward certification path to the sealed content; certificates of the signer, the organizational CA which issued the signer's certificate, and the policy CA which issued the certificate of the organizational CA. The recipient who has the trusted public key of the root CA can verify the seal with the public keys contained in the certificates.

The OSI security toolkit, OSISEC [127], includes libraries which implement encryption algorithms such as RSA encryption [130], DSS, message digest algorithm MD2/5 [131], and tools for managers operating CAs. Applications based on OSISEC have been implemented and deployed in the PASSWORD Project; they are a secured directory service (DISH/DE directory user agents and QUIPU directory system agent), PEM and secured X.400 UAs [125].

5.5 Implementation of Document UA

5.5.1 Slate Multimedia UA

The Slate editor [132] can handle multimedia information, i.e., text, raster-graphics, geometric-graphics, audio and spreadsheet; it is configurable to work with external message handling systems. UCL has developed Slate/ODA converters which support the FOD26 Document Application Profile (DAP) and has integrated the editor with X.400 and SMTP message handling systems [124]; neither audio nor spreadsheet are supported because they have not yet been included in the ODA standard.

5.5.2 PDOCSEC: ODIF Security Filter

There are many ways to implement a secure document UA; one is to change a non-secure document editor in order to support the security services, another is to implement a filter which encodes an ordinary (plain) ODIF stream to a secured ODIF stream and decodes the secured ODIF stream to the plain ODIF stream, and the filter must be integrated with the non-secure document editors.

The authors have adopted the second approach for two reasons: first a document editor is normally very complex and hard to change; secondly, since many document editors support ODIF input and output or have filters which transform the original format into the ODIF and vice versa, the filter procedure is applicable to all such document editors.

The UCL implementation of the filter is called PDOCSEC; it consists of two programs, namely an encoder and a decoder. The encoder requires information about identities of parts of a document to be sealed and/or enciphered, privileged recipient(s), sealing location, date and time. It enciphers and generates seals of specified parts of the document and produces a secured ODIF stream. The decoder verifies and/or deciphers parts of the document of which the associated privileged recipient is the user, and produces a plain ODIF stream. The decoder also outputs information on which parts of the document are passed or failed during the verification and the decipherment.

5.5.3 Integration of PDOCSEC with Slate UA

In order to support security services on parts of a document, the Slate/ODA [124] converters have been changed to support handling security information.

For sending a secured document, a user attaches a tag to a part of the document on the screen by selecting a menu, and the tag containing a security service type and privileged recipients is displayed on the screen. The editor stores the tag into the output Slate file as well as the document content. The security information is placed before or near a Slate object (text, graphics, etc.) in the Slate file. The Slate-ODA filter converts the Slate file into a plain ODIF stream and produces a file enclosing security information which consists of records of a service type, privileged recipients and an object or class identifier (OCID) indicating the part. The PDOCSEC encoder reads this file and the plain ODIF stream, and produces a secured ODIF stream, which is sent to a message transfer system.

The mechanism to produce the secured ODIF stream is almost same as the "digital envelope". For confidentiality of a part of a document, a randomly generated DES-CBC [133] key and initialization vector pair is used to encrypt the content and the pair is encrypted with a privilege recipient's RSA [130] public key. For the purpose of authentication, integrity and non-repudiation of origin, the message digest of the `Sealed Information` attribute including the message digest of the part, the originator's `Personal Name`, signing place, date and time, is encrypted with the originator's RSA private key. The `Sealed Information` is discussed further in the next section.

On receipt of a secured document, first the secured ODIF stream is converted into a plain ODIF stream with the PDOCSEC decoder. The decoder writes security information which consists of records of an OCID, a security service type, privileged recipients, a verification and decipher result and optionally an originator to an external file. The ODA-Slate filter reads this file and the plain ODIF stream, and converts into a Slate file in which the security information is included in the tag form. Note that the Slate editor does not know the OCIDs at all.

A secured ODIF stream is just an octet string; it can be transferred by any message systems such as X.400, MIME [134] + SMTP, but care must be taken in handling of the whole document. Deletion of seal information is undetectable, thereby some process must be applied to the whole exported data stream.

There are three options to send a secured ODIF stream; DOCSEC + X.400, PKCS#7 + X.400 and MIME + PEM + SMTP. Both DOCSEC and PKCS#7 support confidentiality and integrity of a whole ODA document and the secured whole document can be sent through a X.400 system in the form of a bilaterally or externally defined body part. MIME supports the ODA application subtype and the standard specifies how an ODIF stream should be encoded into printable characters according to the base64 encoding rule. A

MIME message containing ODA can be enclosed in a PEM body according to the PEM-MIME integration working draft [135]. Currently the third method MIME + PEM + SMTP has been implemented.

5.6 Discussion

In general, OSI standards are defined without prior implementations; as a result ambiguities and contradictions with other standards are often introduced. This section outlines certain such problems which the authors have uncovered as well as ones encountered in the integration of PDOCSEC and the Slate editor.

5.6.1 Sealed Information

A seal is calculated from the `Sealed Information` attribute which consists of a message digest of content of a part, seal time, location and originator's `Personal Name`, and other information cannot be included into the seal. However, a security policy might require to include the following information in the seal:

- **authenticated time**: Time information authenticated by a third-party or a time-stamp server. The time specified in the addendum is claimed by the originator.
- **role name**: The `Personal Name` includes only `title` information as far as information of a person's attribute. However, it is desirable to include the originator's role and e-mail address, the department name to which the originator belongs, etc.

In order to include such information in the seal, it might be better to change the type to a sequence of attributes such as the authenticated attribute defined in PKCS#7[129].

5.6.2 Message Digest

The security addendum specifies that the `Sealed Information` attribute should include a plain message digest of a part of a document. However, when the part is enciphered the message digest of the part should be enciphered to prevent an adversary from deducing the content. This is because the adversary can determine which of a list of candidate content (e.g., "yes" or "no") is the actual content by comparing the message digests of them to one in the attribute.

The current implementation decrypts the message digest using the DES-CBC algorithm, and the key information encrypted with a privileged recipient's RSA public key is enclosed in a sub-parameter of the attribute. As a consequence only the privileged recipient knows the actual message digest value.

5.6.3 Personal Name Type

The PDOCSEC system adopts the RSA encryption algorithm to seal content and to encrypt a content encryption key which is used to encrypt the content; a public key is retrieved from the X.509 certificate issued from a CA. The X.509 Authentication Framework [126] defines `Distinguished Name` to designate a user. On the contrary the ODA standard adopts `Personal Name` as a name of a user which includes a surname, a givenname, initials and a title. A user in a small community such as an organization could be identified with the information, and a certificate or a trusted public key can be retrieval from a local cache or a directory service. However, it is impossible in a large scale community.

For the purpose of solving the problem, a new parameter `name` which type is `Distinguished Name` is added to the `Personal Name` type.

A privileged recipient can get an originator's certificate via the directory service with the parameter to verify its seal, and an originator can get a privileged recipient's public key for encipherment. However, there is still a problem because the directory service is not yet so ubiquitous that every recipient can get certificates from it.

5.6.4 Certification Path

One solution to get an originator's certificate without directory access is to send a certificate path with the sealed information, as explained in section 5.4. Some applications use this method; a PEM message may enclose `Originator Certificate` and `Issuer Certificate` fields, and a X.400 message has a `certificates` attribute in the envelope for this purpose. However, there is no attribute which corresponds to the above attributes in the addendum.

The `Certification Path` type is imported from the X.509 standard in order to include a complete forward certification path, which is send with the seal information. A privileged recipient having the top level CA's key can get the originator's public key and verify the seal. This is useful when a document is archived for a long term, because it is necessary to preserve only the top level CA's public key and not necessary to store other information such as public keys of users and CAs.

But there still remains a problem; if the security policy of the privileged recipient's domain enforces a check of the originator's certification revocation list [126], the privileged recipient must access the list, which could be stored with the seal information as well as the certification path.

5.6.5 Method Information

The `Method Information` type, which identifies an algorithm used for generation of a message digest, a seal, etc., consists of two parameters; the object identifier and the character string describing the algorithm. However, this definition is inappropriate when the algorithm requires parameters. In addition the type is incompatible with the `Algorithm Identifier` type defined in the X.509 [126] standard which consists of two parameters, the object identifier identifying the algorithm and the required parameter to the algorithm.

5.6.6 DES-CBC Key and IV

The current implementation utilizes the DES-CBC algorithm [133] with a randomly generated session key and initialization vector pair for the encipherment of each part of a document. Before the encryption of the pair, it must be encoded in accordance with some method. A typical method is that only the session key is encrypted while the initialization vector is handled as the parameter of the algorithm. However, the current PDOCSEC stores the pair in the following ASN.1 type, encodes and encrypts for the cause that the ODA's `Method Information` type does not have the parameter field. Since a different system may encode in another way, an agreement is necessary to establish interoperability.

```
DES-CBC-Key-Information  ::= SEQUENCE {
  deskey  [0] IMPLICIT OCTET STRING,
  vector  [1] IMPLICIT OCTET STRING }
```

5.6.7 Integration with Slate UA

Two of four requirements described in section 5.2 have been realised. However, the last two requirements are not realised: the addition and deletion of parts while keeping signatures of other parts valid, and the signing with additional information. In the following the reasons for their absence and proposed solutions are described.

Addition and Deletion of Parts

Addition of a new part and deletion of one are not implemented because OCIDs are changed from originals and the change makes seals invalid. Supposing a new paragraph is added between two paragraphs which OCIDs are "3 0 1" and "3 0 2" respectively, the Slate-ODA filter assigns "3 0 2" to the new paragraph and "3 0 3" to the old second paragraph and hence the seals containing objects which OCID is greater than "3 0 2" become invalid. It is same in the case of deletion of a part from a signed document. This is because the

Slate editor does not know OCIDs at all as mentioned in section 5.5.3 and the Slate/ODA converter assigns a new OCID. However, this is not a particular problem of the Slate editor and the Slate/ODA converter; it is a common problem to the existing editors which uses converters to transform the ODIF to the editor original format and vice versa. There are a few solutions:

1. Generate message digests of bodyparts without OCIDs
2. Keep mapping information between newly assigned OCIDs and old OCIDs (the identifiers when the original message digest was calculated) in a parameter, and use the old OCIDs during the verification time

While the first solution violates the addendum, the second one requires a new ASN.1 type to interchange the mapping information.

Signing of Additional Information

Signing time and location can be sealed with content because the addendum includes these parameters in the `Sealed Information`. On the contrary other information, such as authenticated time, is not supported because the `Sealed Information` type is not extensible as mentioned in section 5.6.1.

5.7 Comparison with PEM

Both the ODA security and PEM provide end-to-end security services, that is, authentication, integrity, non-repudiation of origin and confidentiality, and their mechanisms are basically same. In the following differences between them are described.

5.7.1 Security Infrastructure

PEM adopts a hierarchical CA structures which root is the Internet Policy Registration Authority (IPRA) [136]. Security policies are defined by each policy CA, which is certified by the root CA. However, the ODA addendum does not specify anything about key managements nor security policies.

5.7.2 Encryption Algorithms

PEM defines specific algorithms for message encryption and signing; DES in CBC mode, MD2 (MD5) and RSA encryption algorithms are used when an asymmetric key management is adopted. On the contrary, the ODA security does not specify any algorithms.

5.7.3 Whole message versus parts of document

While PEM only supports securing a whole message, the ODA security supports securing parts of a document. However, the emerging new PEM, which is integrated with MIME, will support securing parts of a message and convey not only ascii text but also content of arbitrary types.

5.7.4 Multiple Signatures

PEM does not support multiple signatures; in a header of a PEM message multiple Originator Id fields may appear when different originator oriented interchange key components are used in order to prepare the message so as to be suitable for processing by different recipients who require different cryptographic mechanisms. On the other hand, the ODA security supports multiple signatures basically in order to seal by multiple originators.

While ODA signers can seal a part of a document with sealing time and place, PEM does not prepare to sign with such information.

5.7.5 User Name

While PEM adopts the X.509 Authentication Framework and uses `Distinguished Name` as a user name, the ODA standard adopts `Personal Name` to designate a user, which is incompatible with the `Distinguished Name` described in the previous section.

5.8 Conclusion

This paper has described how a specific editor can be augmented to process secured ODA documents and uncovered a number of weaknesses in the security addendum. While the described system here can be considered only as an early prototype, it provides a powerful set of functions; for example, the functionality which it introduces can be very important for its introduction of secure EDI.

The authors are implementing a Wide Area Information Server which authenticates a client, authorizes client's access with various information such as user's name, role, security label, network location and access time, and transmits a document with the protection described in the paper. The authorization scheme is based on a capability scheme which uses the combination of Privilege Attribute Certificate (PAC) and Control Attribute Package (CAP), and useful for document distribution in an organizational network and a world wide network.

5.9 Acknowledgement

The authors wish to express thanks to Peter Williams for his advice and his implementation of the OSISEC toolkit.

第 6 章

今後の活動

ネットワーク上では様々な情報が公開され提供されている。特に WWW や Gopher の普及に伴いその量は爆発的に増えている。しかしそれらの情報を効率よく探す仕組みはいくつかのものが提案されているが、十分なものとは言えない。

今年度の研究として WWW の情報を検索する機構を模索したが、問題点が多く残っている。次年度は、OSI ディレクトリサービスによる情報検索の応用研究として、ネットワーク上で各種の方法によって公開されている大量の情報を検索する事ができる機構を研究する予定である。

この研究の項目としては以下の 3 つが考えられる。

1. 「情報 index」の収集機構
   OSI ディレクトリを使い、ネットワーク上に存在する全ての情報の index をもれなく収集する。

2. 「情報 index」の整理機構
   収集した情報を検索しやすい形で整理する。

3. 「情報 index」の検索機構
   OSI ディレクトリサービスを利用して、情報を欲している人にとって最適な場所から最適な物を取得できるようにする。