Achieving PKI Interoperability 2003

Results of the JKST-IWG Interoperability project

Japan PKI Forum
Korea PKI Forum
PKI Forum Singapore
Chinese Taipei PKI Forum

July 7 2003

Acknowledgements

We wish to thank the following personals for the valuable contributions to this document. This report is collective efforts and achievement with the individuals that volunteered their valuable hours to produce this report.

Contributing Writers

Japan:
Masaki, SHIMAOKA (Secom)
Yuji, AIKAWA (Secom)
Ako, MISHIO (NEC)
Shun, NAGAKURA (Fujitsu)
Satoshi, TAKEMOTO (Hitachi)
Masahiko, FURUYA (Hitachi)
Hitoshi, SHIMONOSONO (Hitachi)
Kiyoshi, WATANABE (Hitachi)

Korea:
JaeIl LEE, Korea Information Security Agency (KISA)
Yong LEE, KISA
Seok-Lae LEE, KISA
MoonBo SHIM, KISA
HwaSoo SHIN, KISA
InKyung JEON, KISA
BoSung HWANG, KISA
SangHwan PARK, KISA

Singapore:
Wei Qiang - PKI Forum Singapore
Pradip Shetty - PKI Forum Singapore
Thio Fu Wang - PKI Forum Singapore
Tan Chee Young - PKI Forum Singapore

Chinese Taipei:
Dr. Wen Cheng WANG, ChungHwa Telecom Lab (CHT)
Dr. Chung Min OU, ChungHwa Telecom Lab (CHT)
Leon CHEN, Taiwan-CA.com Inc. (TWCA)
Blues LIN, Taiwan-CA.com Inc. (TWCA)
Sam HWANG, Taiwan-CA.com Inc. (TWCA)
Li Hsuan Liang, NII Enterprise Promotion Association (NIIEPA)

Editorial Review Team

IWG Technical WG

Table of Contents

Appendix:

Executive Summary

2002 JKST-IWG Interoperability project, which ran between May 2002 and March 2003, is continuous and collaborative efforts of four Asia PKI Forum members – Japan PKI Forum, Korea PKI Forum, PKI Forum Singapore as well as Chinese Taipei PKI Forum.

The main objective defined in 2002 JKST-IWG Interoperability project is to promote PKI interoperability by adopting common PKI technical specifications that support each party's operations. Moreover, the practical experiment experiences gained in the project can be a reference model for other Asia PKI Forum members when intending to establish trust relationships with PKI domains. There are three themes included in the 2002 JKST-IWG Interoperability Project and they were implemented into 3 experiments:

- CA-CA Interoperability Experiment with Cross Certification / Cross Recognition models;
- Path Processing Experiment intending to Resolve the certificate path processing issues of repository by clarifying the path processing logic described in RFC3280;
- PKCS#11 Experiment tempting to approach PKI application interoperability using a commonly defined API (application interface).

The document of "Results of the JKST-IWG Interoperability project" consolidates all the test results of the aforementioned three experiments. More importantly, it contains the recommended technical specifications and the lessons learnt, which are valuable for CA operators, VA (validation authority) and application developers when dealing with relevant interoperability matters. In this document, each one of the three experiments is described in details, including topics such as overviews, assumptions, test environments, test plans, results and recommendations.

In addition to this overall project result document, other four documents were developed as appendixes, which are:

- Appendix 1. IWG Recommended Profiles
- Appendix 2. CA-CA Interoperability Interface Specification for experiment
- Appendix 3. Certificate Path Processing Implementation Guideline
- Appendix 4. Certificate Path Processing Testing Guideline

The project execution experiences and learned lessons have also proved that technical interoperability issues are complicated but can be resolved without enormous difficulties, however the negligence of certain trivial details in the process can become really distressful.

Finally, all the testing participants in the project have concluded that business application interoperability is one of the most critical issues at the present moment. It has also been determined by the project participants to fuel more efforts in the future, as the next step, initiating business application oriented experiments to truly foster the fulfillment of Asia PKI Forum goal as "to promote the establishment of interoperable PKIs throughout the Region and the realization of borderless and seamless e-commerce".

1 Background

1.1 The Breath of JKST IWG

The Internet economy depends heavily on the security and validity of information, such as online transactions will not occur if trading partners do not trust each other in the virtual business process and credit card numbers will not be passed to the vendor without encryption procedure.

Along the wave, the term PKI (Public Key Infrastructure) will soon become an important piece of the Internet commerce as it promises secure transactions in terms of confidentiality and integrity protection, and provides trust infrastructures to make non-repudiation and identity authentication possible over the Internet.

As the inherent nature of globalization in Internet commerce arena, the developments of PKI tend to get beyond the nation borders. The recent global PKI initiatives in various regions, such as certification framework and digital signature legislations, are shaping the international PKI at the state levels, and could potentially bring greater economic impacts across the regions in varying degrees.

Like all the other Internet related technology evolvements, the uncontrolled proliferation of PKI related terminologies, products, services, legal enforcements and so on have caused confusions: PKI systems do not interoperate, products or services offered are not based upon common specifications, and the bewildering array of different levels of security assurance has created more uncertainties.

In order to promote a global PKI, the blurred PKI condition should be eliminated to ensure that heterogeneous PKI domains and practices are able to interoperate with each other. Further, to develop a mutually agreed inter-working PKI framework at the regional and subsequently, international levels. Asia PKI Forum was formed in June 2000[1] with the mission to resolve the cross-border PKI interoperability obstacles from the technical, legal and business operational perspectives.

Initially, KSJ (Korea, Singapore, Japan) Interoperability working group (IWG) of ASIA PKI Forum was formed in 2001 with a focus on "CA to CA Interoperability", an experiment was carried out and closed by Q1 2002[2] . In mid 2002, Chinese Taipei PKI Forum joined the IWG activity and the working group is renamed as JKST (Japan, Korea, Singapore, Chinese Taipei) Interoperability working group of ASIA PKI Forum (JKST IWG).

New initiatives have been explored by the four parties thereafter, which include PKI Application Interface, Path Processing Interoperability and expansions of the previous experiment with a focus to facilitate the developments of PKI interoperable structures and PKI applications.

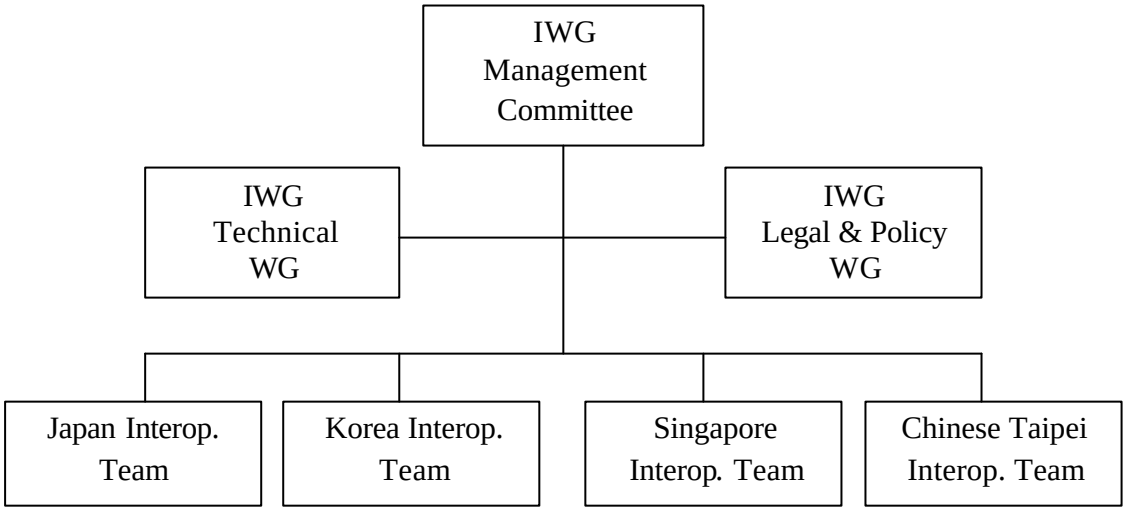
[1] Refer to http://www.asiapkiforum.org for more information about Asia PKI Forum

[2] Refer to http://www.japanpkiform.org for the JKS IWG CA to CA Interoperability Experiment results

1.2 The IWG Organization

1.2.1 IWG Structure

The figure shows the structure of the IWGorganization. The IWG Management Committee coordinates the IWG activities including the MOU signing and regular meeting, and reviews and authorizes the progress. IWG Technical and Legal and Policy WGs set up the working items, prepare the documents, and solve the problems.



As IWG community to support this activity, the three Mailing Lists are used. The Management ML discusses the managerial and general issues on the project. The Technical ML and the Policy & Legal ML address the technical and legal aspects of the project respectively.

1.2.2 IWG Participants

Korea	Singapore	Japan	Chinese Taipei
- Korea Information Security Agency - Korea Information Certificate Authority - Korea PKI Forum	PKI Forum Singapore	Japan PKI Forum	Chinese Taipei PKI Forum

2 Target Audiences

The report is prepared for wide ranges of audiences and it highlights both the results and recommendations in the 2002 JKST IWG experiment activities.

Though the report is for anyone who wants to learn more about the way JKST IWG approached to resolve the PKI Interoperability technical issues and to explore the derivative findings. The report is especially useful for CA operators, VA developers, application developers and security professionals as the project is divided into 3 sub experiments with three specific themes.

Each one of the experiments is described and concluded individually in Chapter 4 of the report. For readers' convenience, following provides an audience index for a quick start:

Audience	
Business Executives	Chapter 0 Executive Summary & Chapter 3 Project Overview
CA operators	Chapter 4.1 CA-CA Experiment
VA developers	Chapter 4.1 CA-CA Experiment & Chapter 4.3 PPIG and PPTG Experiment
Application developers	Chapter 4.1 CA-CA Experiment & Chapter 4.2 PKCS#11 interoperability

3 Project Overview

3.1 Project Objectives and Scope

The objective of the 2002 JKST IWG project is to achieve PKI interoperability in Asia region using recommended PKI specifications that are capable of supporting each participating country's operating conditions. The project also aims at offering a reference model to allow future participants from Asia PKI Forum members when establishing cross border PKI interoperability relationships within the heterogeneous PKI domains.

The project is continuous and collaborative efforts of the participating members during 2001 and 2002. It has moved to the second phase after Japan, Korea and Singapore teams concluded their CA-CA Interoperability Experiment results in May 2002 as the first phase. The project second phase began when Chinese Taipei entered the partnerships of IWG, and it can be regarded as an expansion of the previous phase project. The project second phase contains 3 themes:

- Extending "CA-CA Interoperability Experiment" to other country/region
- Resolving the certificate path processing issues of repository
- Approaching the PKI application interoperability

Respectively, the 3 themes refer to the 3 experiments, which are summarized as below:

3.1.1 CA - CA Interoperability Experiment

There are many identified technical issues regarding establishing common infrastructures for critical PKI components in different domains. The CA - CA Interoperability Experiment intends to explore issues when implementing the CA-CA interoperability, especially those related to the international context, by setting up different trust models and deploying common component infrastructures between project participating parties' simulated CAs.

Throughout the experiment, JKST IWG has identified a common set of technical agreements to achieve the reliable infrastructure where secure business transactions are conducted. The common sets of technical agreements were documented in the form of a profile, describing the key infrastructure interface and the certificate profile, which are along with the widely accepted X.509 and IETF standards. Tests were implemented with different business applications in order to evaluate the feasibility of the infrastructures as well as issues encountered in the experiment. The profile was consolidated by the JKST IWG and was inserted in the Appendix section of the report. The profile was submitted to Asia PKI Forum and is currently under evaluations to become Asia PKI Forum recommendation references.

3.1.2 Path Processing Experiment

The objective of the Path Processing Experiment is to clarify the certificate path processing logic described in RFC 3280. Test environments for Relying Party (RP) applications are also created for the purpose.

In order to have commonly agreeable criteria for checking and verifying the certificate path processing logic in PKI applications, two guidelines were developed by JKST IWG: Certificate Path Processing Implementation Guideline and Certificate Path Processing Testing Guideline. In addition to defining the inter-domains (inter-connections) PKI environment specifications where the RP applications validate the certificate, the guidelines also provide the optimized test cases for RP applications.

3.1.3 PKCS#11 Application Interoperability Experiment

The current e-business trends show that more and more web-based applications are evolving while cryptographic components are designed in the possibly plug-in module manner at the user side. As a result, API developments and standardizations are critical to facilitate the PKI application deployments. The objective of PKCS#11 Application Interoperability Experiment is to achieve the compatibility between the implementation of cryptographic functions of different vendors based on the commonly agreed API, making the PKI enable applications to be portable in different environments.

The experiment includes three phases: 1) the documentation, which is "PKCS#11 IWG Conformance Profile " as the interface between web applications and PKCS#11 library; 2) the development of the application wrapper; and 3) the actual experiment using PKCS#11 library and web applications.

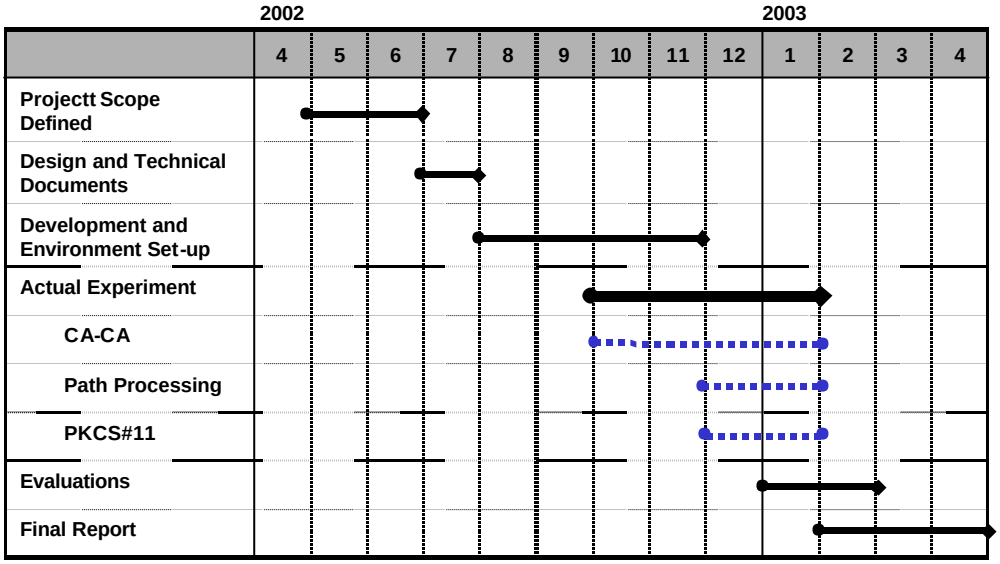
In the experiment, each participating party prepared its own PKCS#11 library and web-based application. After the full set-up of the environment, each participating party verified the application interface with each other using its respective applications and PKCS#11 library.

3.2 Project Participants

Korea	Singapore	Japan	Chinese Taipei
- Korea Information Security Agency - Korea Information Certificate Authority - Korea PKI Forum	PKI Forum Singapore	Japan PKI Forum	- ChungHwa Telecom - Taiwan CA Corp. (abbrev. TWCA) - NII Enterprise Promotion Association

3.3 Project Timescale & Milestones

Timescale of 2002 IWG JKST Project



The project has started at the end of the April in 2002 and ended at the end of the March in 2003. Three different experiments in the project were successfully conducted in parallel.

3.4 Deliverables

Deliverable Number[3]	Deliverable Name	Type	Public Availability
CC – 01	Project Plan	Presentation	Restricted Circulation
CC – 02	IWG Interface Specification	Document	Downloadable
CC – 03	Japan Test Results	Presentation	Restricted Circulation
CC – 04	CT TWCA Test Results	Presentation	Restricted Circulation
CC – 05	CT CHT Test Results	Presentation	Restricted Circulation
PPI/TG – 01	Data Generation Sheet	Document	Downloadable
PPI/TG – 02	Path Processing Implementation Guideline	Document	Downloadable
PPI/TG – 03	Path Processing Testing Guideline	Document	Downloadable
PPI/TG – 04	Path Processing Data Generation Specification	Document	Downloadable
PPI/TG – 05	Japan PPI/TG Test Results	Presentation	Restricted Circulation

[3] CC (CA - CA Interoperability Experiment); PPI/TG (PPIG and PPTG Experiment); P11 (PKCS#11 Application Interoperability Experiment)

PPI/TG – 06	Korea PPI/TG Test Results	Presentation	Restricted Circulation
PPI/TG – 07	Chinese Taipei PPI/TG Test Results	Presentation	Restricted Circulation
P11 – 01	PKCS#11 IWG Conformance Profile	Document	Downloadable
P11 – 02	PKCS#11 Profile Proposal	Presentation	Restricted Circulation
P11 – 03	PKCS#11 Test Template	Document	Downloadable
P11 – 04	Japan PKCS#11 Test Results	Presentation	Restricted Circulation
P11 – 05	Korea PKCS#11 Test Results	Presentation	Restricted Circulation
P11 – 06	Singapore PKCS#11 Test Results	Presentation	Restricted Circulation
P11 – 07	Chinese Taipei PKCS#11 Test Results	Presentation	Restricted Circulation

4 Experiment

4.1 CA-CA Interoperability Experiment

In order to facilitate a reliable and open infrastructure where secure online transactions can be conducted across borders, a variety of cross-border PKI initiatives internationally in fashions of closed domain, open domain based on the domestic policy or consolidation have been developed.

The CA-CA Interoperability Experiment of IWG intends to explore issues when implementing the CA-CA interoperability, especially those related to the international context, by setting up different trust models, deploying common component infrastructures and business applications between project participating parties' simulated CAs.

To perform the experiment, a four-phase approach was taken by the experiment teams as described below:

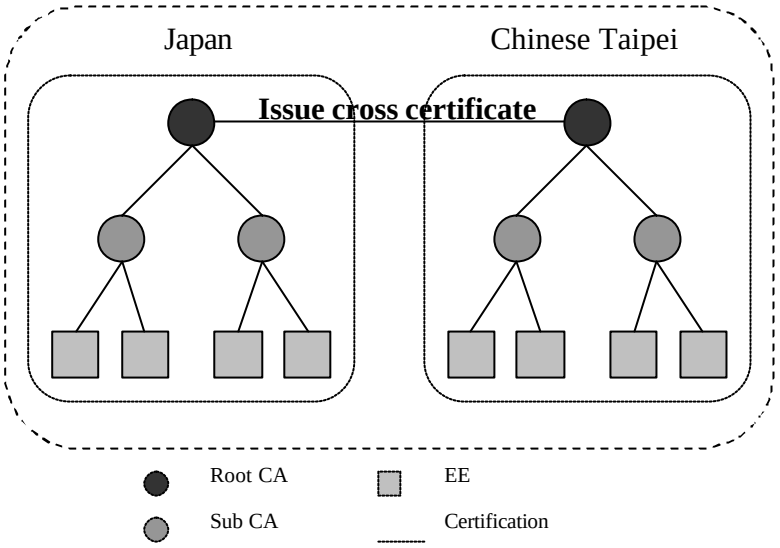
Phase	Description
Analysis and Design	Discussing about the trust model and application software selection, identifying component interfaces and technical issues, and preparing a technical profile for certificate and CRL, and repository information.
Development	Developing application software and setting up the simulated CA systems, repositories and other PKI components
Experiment	Planning the test scenarios and performing the actual experiment with the application software
Evaluation	Evaluating and sharing the experiment results and offering recommendations learnt

4.1.1 Assumptions: Experiment Model

4.1.1.1 Defining CA-CA Trust Model in the Experiment

The CA-CA Experiment employed two fundamental trust models: Cross Certification (CC) and Cross Recognition (CR). In general, the Cross Certification model and its variants, such as Bridge CA model, are commonly deployed in the government PKI domains in Asia region, while the Cross Recognition model and its variants, such as trust list solution employed in the web browser, are often adopted by the commercial PKI initiatives. The following two charts illustrate two different trust models in the experiment:

4.1.1.1.1 Cross Certification

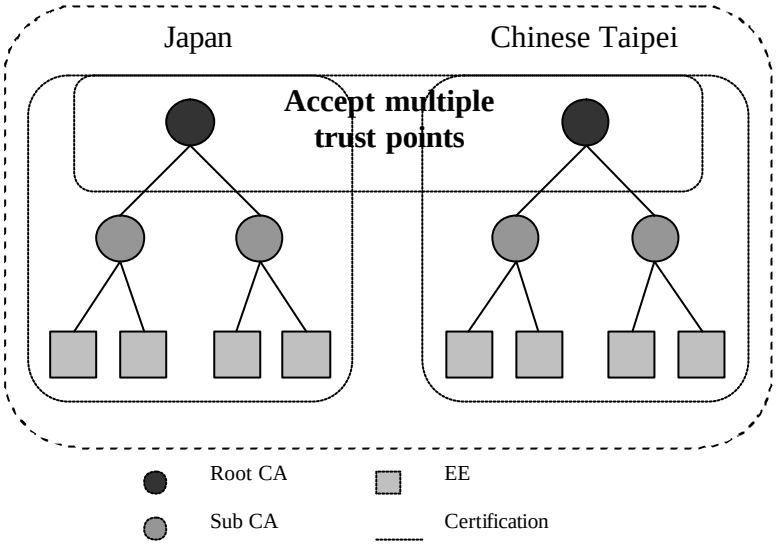


The concept of Cross Certification has arisen in the PKI environment to deal with precisely this need for forming trust relationships between formerly unrelated PKI installations [4] by issuing a cross certificate and provide a certificate path for validating the certificate chains. The X.509 specification defines a cross certificate as "A Certification Authority may be the subject of a certificate issued by another Certification Authority. In this way, the certificate is called a cross certificate."

It is well understood that Cross Certification is the base of Mesh trust model, Bridge CA model, and Accreditation certification model and the proliferation of Cross Certification relationship establishment will simply increase the number of the cross certificates causing a more and more complexity for certificate path constructions. However, this trust model is a fundamental and core technology to explore the issues for the multiple domain environments.

[4] "Understanding PKI, Concepts, Standards, and Deployment Considerations", by Carlisle Adams and Steve Lloyd, 1999

4.1.1.1.2 Cross Recognition



The APEC E-Security Task Group defines cross-recognition as

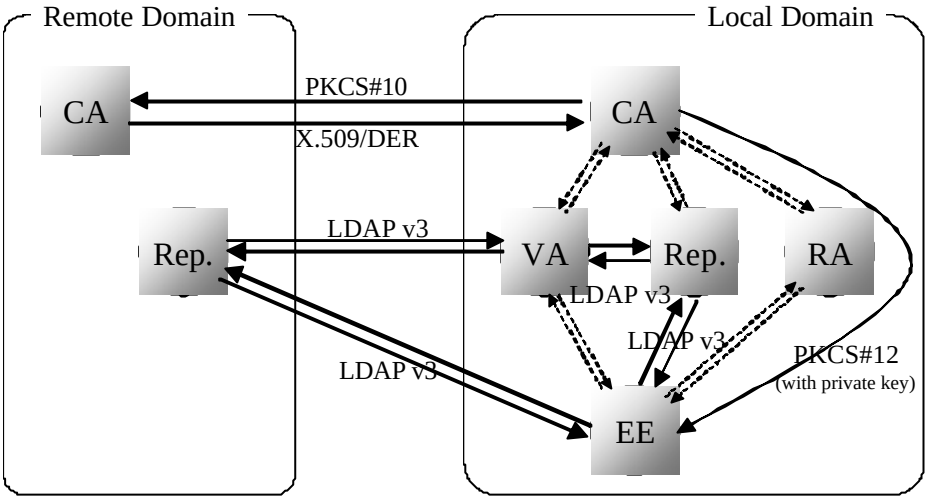
" An interoperability arrangement in which a relying party in one PKI domain can use authority information in another PKI domain to authenticate a subject in the other PKI domain, and vice-versa." [5]

In other words, the Cross Recognition model involves issues of the acceptance of multiple trust points. Relying Parties in the local community can validate and process a received subject certificates issued by an external CA. To accomplish this goal in the experiment, trust list, a data structure stored in the application with a root public key of the recognized CA, was employed.

[5] See the definition of APEC TEL document at http://www.apectelwg.org/apecdata/telwg/eaTG/eatf06.html

4.1.1.2 Defining Component Interfaces

There is a minimum set of PKI component interfaces to be agreed upon between the different domains. Below summarizes the component interfaces between two domains in the experiment. The solid line shows the interfaces to other domains and the dotted line is out of scope in this experiment.



For the purpose of cross certificate request in the experiment, Certification Request Syntax Standard PKCS #10 v1.0 was adopted without using extension fields. As for the cross certificate response, since there is no formal standardized protocol or extra data structure to support the function, a simple certificate DER format was is defined in order to exchange the cross certificates.

For the certificate path construction, it was designed in the experiment that each domain set up a LDAP server using LDAP v3 protocol. The referral function was decided as "must-have" in the respective LDAP servers in order to allow a CA to obtain necessary data from the repository of the other PKI domain. For the certificate distribution purpose in the experiment, Personal Information Exchange Syntax PKCS#12 format with private key was adopted.

In the experiment, each CA generated certificates and private keys for its own respective end entities. End entities may use their certificates and private keys to conduct transactions in the applications of the other trust domain.

Finally, cross certification operation procedures, such as the POP (proof of possession) and the fingerprint verification methods, are out of scope in the experiment. For details, read the appendix of "CA-CA Interoperability Interface Specification for experiment Version 1.0".

4.1.1.3 Defining Application Model

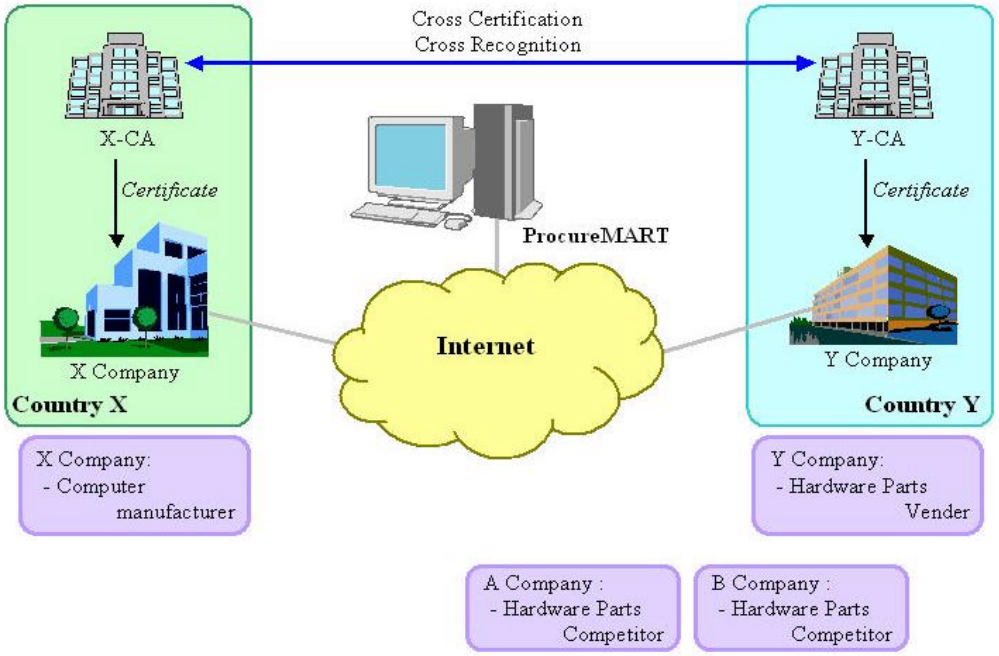
The chapter defines the business application models of the experiment. In the experiment, there were no strict requirements specified as long as the applications conforming to the designated

interface specifications. It was also suggested that the project participants could develop any application at their preferences or upon the actual business targets and scenarios.

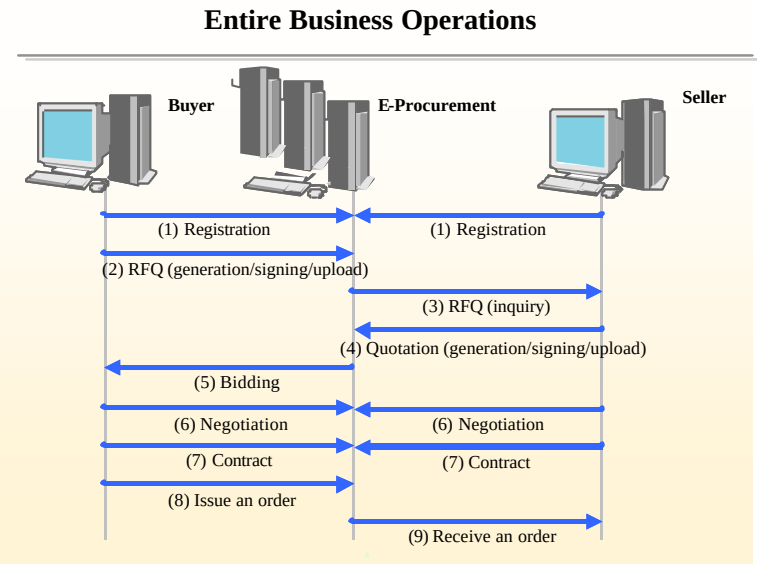
4.1.1.3.1 Japan Application Model

For this experiment, Japan testing team adopted an e-procurement system as the business application. The scenario of the application is "business-to-business" (B2B) procurement transactions conducted over the Internet using the predefined system, and the scenario can be illustrated as the chart below:

X-CA offers PKI service to Company X and issues certificates to the employees in Company X; meanwhile Y-CA offers PKI service to Company Y and issues certificates to the employees in Company Y. It is assumed that Company Y is located in a foreign country. In the experiment, the 2 CAs are capable to adopt both Cross Certification and Cross Recognition trust relationships.



The business flow of the e-procurement system is depicted as the chart below:



Step 1. User registration
Step 2. Generation of Estimate Expense Sheet / Signing Signature / Upload（buyer）
Step 3. Reference to Estimate Expense Sheet / Signature Validation（seller）
Step 4. Generation of Reply Sheet / Signing Signature / Upload（seller）
Step 5. Reference to Reply Sheet / Signature Validation / Estimation（buyer）（Negotiation）* not applicable in this experiment.
Step 6. Generation of Contract Document / Signing Signature / Sending（buyer）
Step 7. Reception of Contract Document / Signature Validation / Signing Signature / Reply（seller）
Step 8. Generation of Purchase Order / Signing Signature / Sending（buyer）
Step 9. Reception of Purchase Order / Signature Validation / Acceptance Inspection（seller）

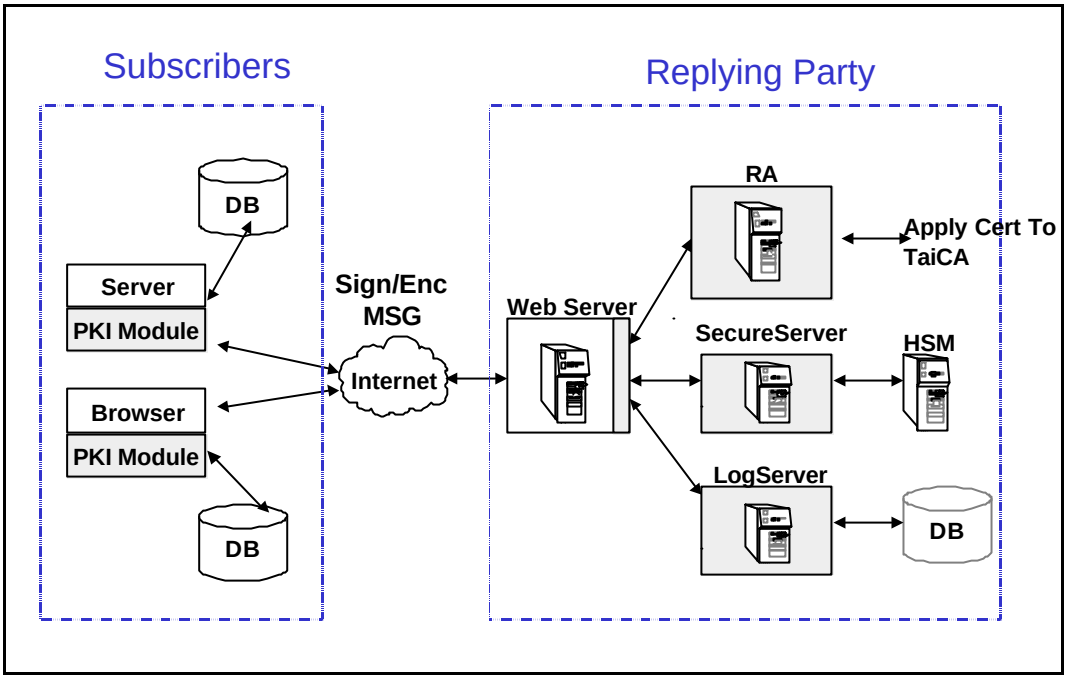
In the experiment, End-Entities (sellers and buyers) performed the aforementioned nine business steps to complete a transaction.

4.1.1.3.2 Chinese Taipei Application Model

Following describes the two different application models which were adopted by Chinese Taipei team in the experiment:

4.1.1.3.2.1 Payment SignOn Application Model

“ Payment SignOn” application was brought into the experiment by TWCA, as a real business case, it has 6 features: issuing X509 v3 based certificates, adopting dual-key system (one for signing and the other for encryption), supporting certificate status checking using CRL or OCSP, supporting PKCS#7 or XML format signatures, storing key-pairs in CAPI compatible token or IC cards. The Payment-SingOn architecture is shown as the chart below:



In the Relying Party side of the architecture, RA server is responsible for certificate application, renewal, revocation, suspension, query, downloads and registration SecureServer takes duties of generating receipts and conducting verifications, while LogServer registers all the transaction and receipt records.
The transaction flow in the AP model contains 8 steps:

Step 1. Subscriber to sign for transaction data;
Step 2. Subscriber to store transaction data in the local Database and send the transaction data to Replying Party Servers.
Step 3. Replying Party Log Server to register the received transaction data
Step 4. Replying Party Secure Server to proceed transaction verification
Step 5. Replying Party Secure Server to generate receipt
Step 6. Replying Party Log Server to register receipt data and send the receipt to Subscriber
Step 7. Subscriber to store receipt in the local Database
Step 8. Subscriber to proceed the receipt verification

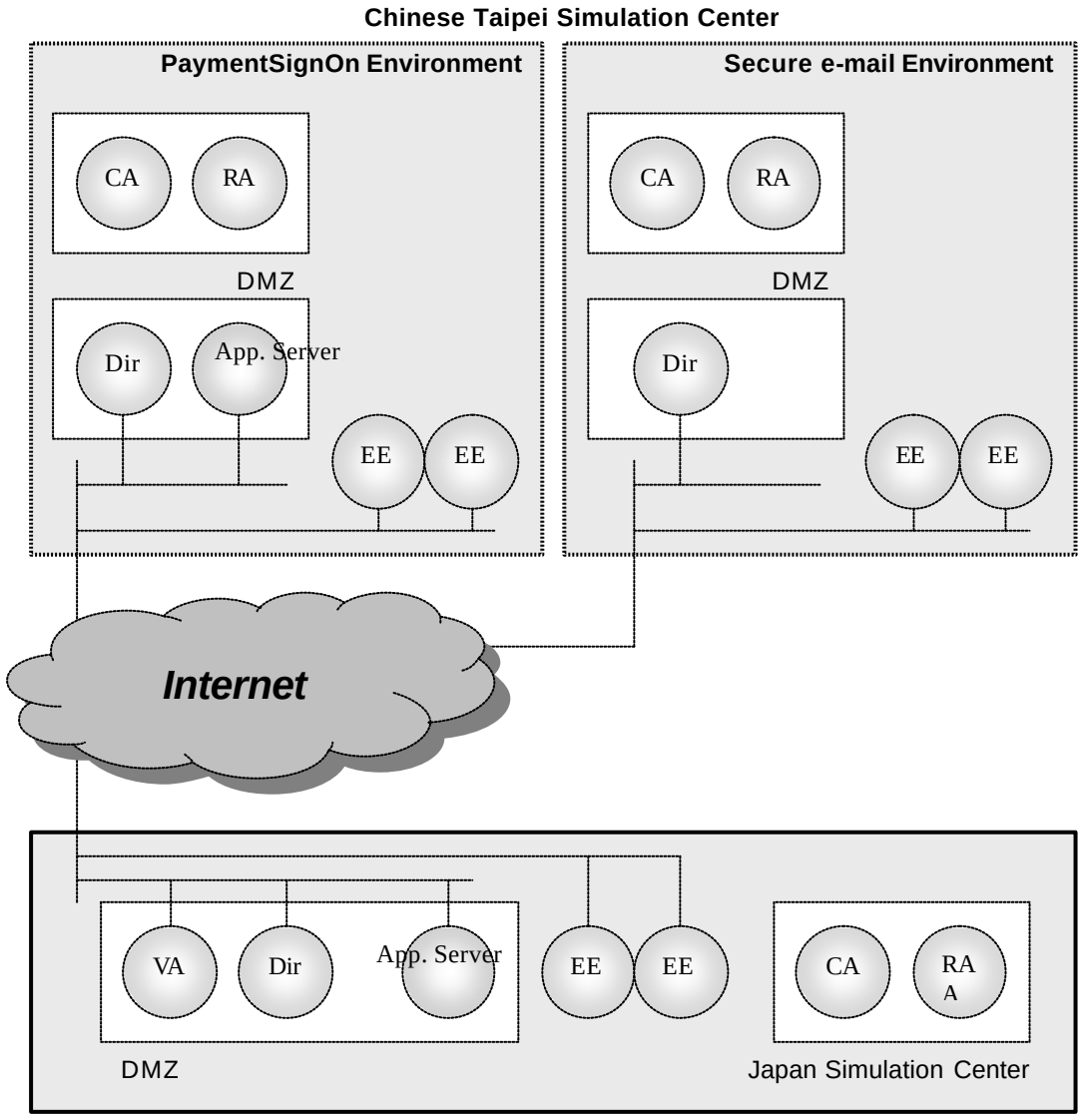
4.1.1.3.2.2 Secure Email Application Model

" Secure Email" application was included in the experiment with the on-the-shelf email product "Microsoft Outlook Express 6".

4.1.2 Test Environments

4.1.2.1 Overall Test Environment

The overall test environment chart below illustrates the network connectivity and the locations of the PKI components in each respective simulation center. The components defined in a simulation center are: Certification Authority (CA), Registration Authority (RA), directory and application.



4.1.2.2 Japan Test Environment

Component overview

Japan CA:

- JCSI (Japan Certification Services, Inc.) simulated CA, which is internally cross certified with the node of the Japan Bridge CA architecture, now cross certified with the TWCA and ChungHwa Telecom CAs in this experient. The CA stores the information in the Directory server.

Japan Client and Server:

- E-Procurement clients and server (developed by Fujitsu) with web browsers. The End Entity is located outside the DMZ of the Japan simulation center in order to simulate the client accesses from outside.

Japan VA:

- Certificate Validation Server, which constructs and validate the certificate path on behalf of the End Entity.

The table below summarizes the specifications of the Japan experiment environment:

Item	Context
Base Specification	JCSI AccreditedSign Public Service Type 2, Japan GPKI specifications
Certificate profile	X.509 (97) v3, RFC3280
Certificate encoding format	DER
CRL profile	X.509 (97) v2, RFC3280
CRL encoding format	DER
Cross-Cert request format	PKCS#10
Cross-Cert response format	X509/DER
The method of sending fingerprint	FAX/email
POP (proof of possession)	Verification of digital signature on certificate request format
Storage device	HSM
EE Certificate response format	PKCS#12 (Private-key included)
Repository access protocol (e.g., LDAPv2, LDAPv3, DAP)	LDAPv3
Certificate path validation method	RFC2459 (son of) and RFC 3280
Certificate validation entity	EE and VA

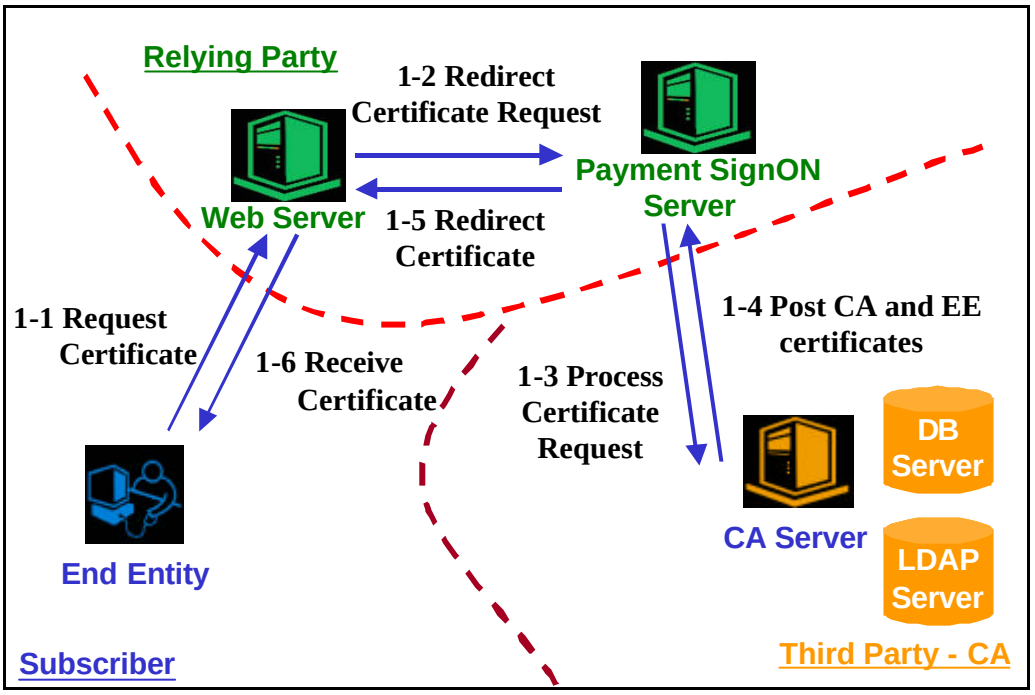
The experiment applications utilized in each component are listed in the following table.

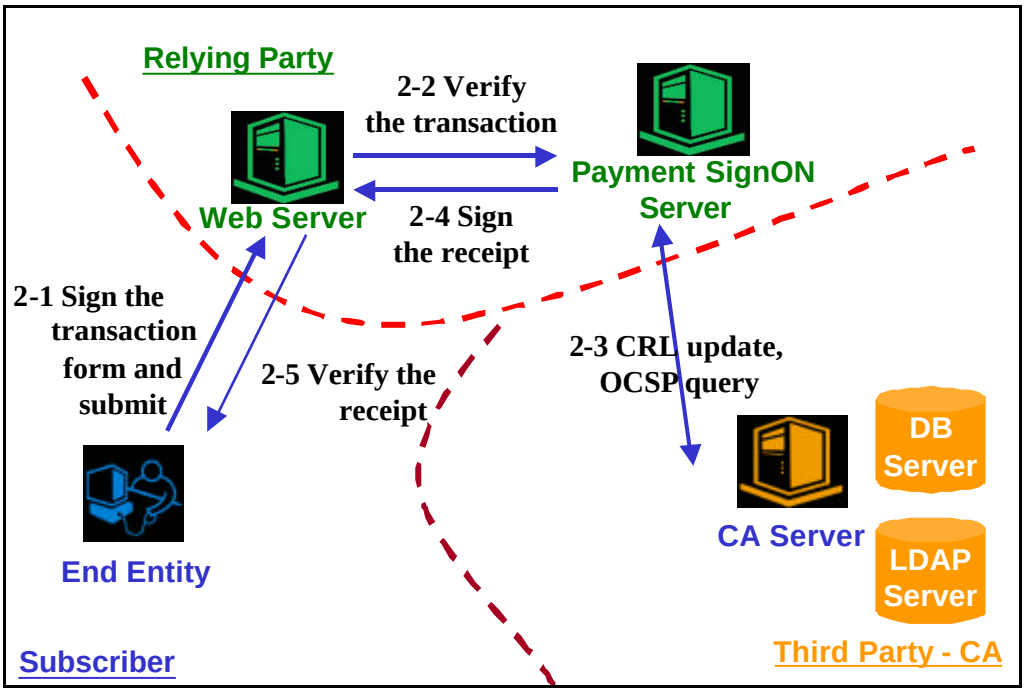
Component	Applications Utilized
CA Server	NEC PKI Server/Carassuit E-Government edition
LDAP Server	NEC EnterpriseDirectoryServer
VA Server	Hitachi Certificate Validation Server
e-ProcurementServer	Fujitsu ProcureMART
Client Application	Fujitsu e-ProcurementClient

4.1.2.3 Chinese Taipei Test Environment

4.1.2.3.1 PaymentSignOn Test Environment

The test environment of PaymentSignOn is illustrated as below: 4 servers with different functionalities were set up including web server, LDAP repository server, CA server and Database Server.





For the subscriber side, the end entity (EE) has to install PKI modules released by the CA. For the Relying Party and CA sides, the main components of 4 servers are briefed as follow:

(1) Web Server: web server is located in the DMZ area serving as a transaction receiver. The web server is responsible for receiving the signature data from the testing users and handling the signature data validation tasks. The validation task is done based on the "Path Validation Model" defined in the JT CA-CA Experiment.
(2) Payment-SignON Server: Including RA server, Secure server and Log server, response for certificate related function, sign/verify function and log functions. The validation task is done based on the "Path Validation Model" defined in the JT CA-CA Experiment.
(3) LDAP Repository Server: LDAP server is located in the DMZ area and it contains CA certificates, CRL, ARL and the end user certificate bulletin; testing users can access to LDAP server to retrieve the necessary information.
(4) CA Server: CA server is located in the Intranet area and it issues a variety of certificates (CA, CC, EE, CRL and ARL) based on the JT Experiment Certificate and CRL Profile. The CA server stores certificates into the database server and publish valid certificates in the LDAP server.
(5) DB Server: DB server is located in the Intranet area and it stores log files as well as certificates issued by the CA.

The table below summarizes the specifications of the Payment SignOn experiment environment:

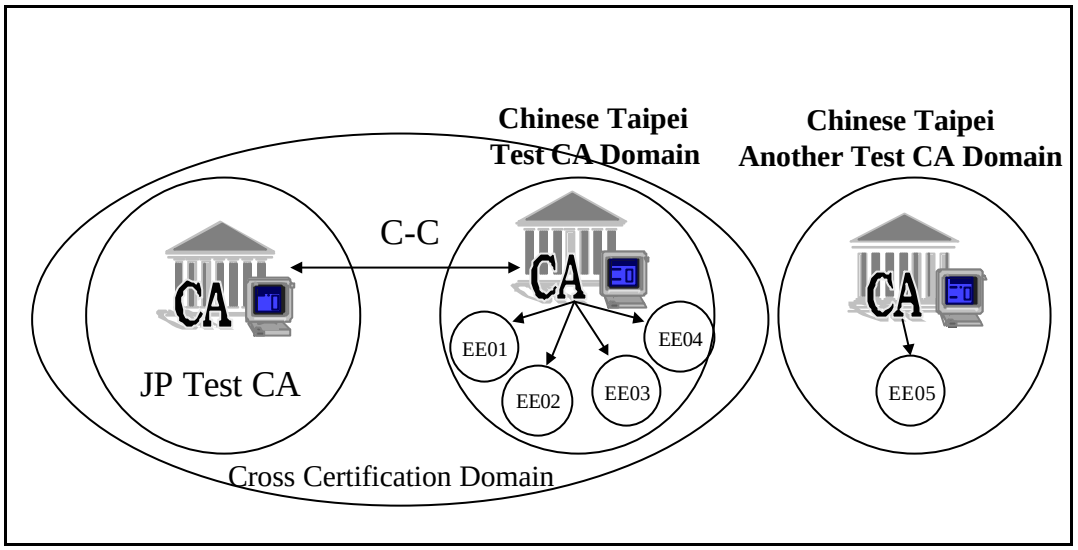
Item	Context
Base Specification	TWCA
Certificate profile	X.509 (97) v3, RFC3280
Certificate encoding format	DER
CRL profile	X.509 (97) v2, RFC3280
CRL encoding format	DER
Cross-Cert request format	PKCS#10
Cross-Cert response format	X/509/DER
The method of sending fingerprint	E-mail
POP (proof of possession)	Verification of digital signature on certificate request format
Storage device	End Entity : CAPI compatible Token or IC Card. Relying Party : HSM
EE Certificate response format	PKCS#12 (Private-key included)
Repository access protocol(e.g., LDAPv2, LDAPv3, DAP)	LDAPv3
Certificate path validation method	RFC2459 (son of)
Certificate validation entity	EE

The experiment applications utilized in each component are listed in the following table:

Component	Applications Utilized
CA Server	Baltimore UniCERT CA
LDAP Server	Netscape iPlanet Directory Server 4
Web Server	JRun Web Server
Client Application	MS-CAPI ActiveX(genkey pair and make signature)

4.1.2.3.2 Secure Email Test Environment

In the experiment, ChungHwa Telecom (CHT) Lab simulated two CAs, which are basically similar to Chinese Taipei GPKI specifications. This guarantees the cross-certification between CHT PKI and Chinese Taipei GPKI. These CAs are cross-certified with the Japan CA in the experiment.



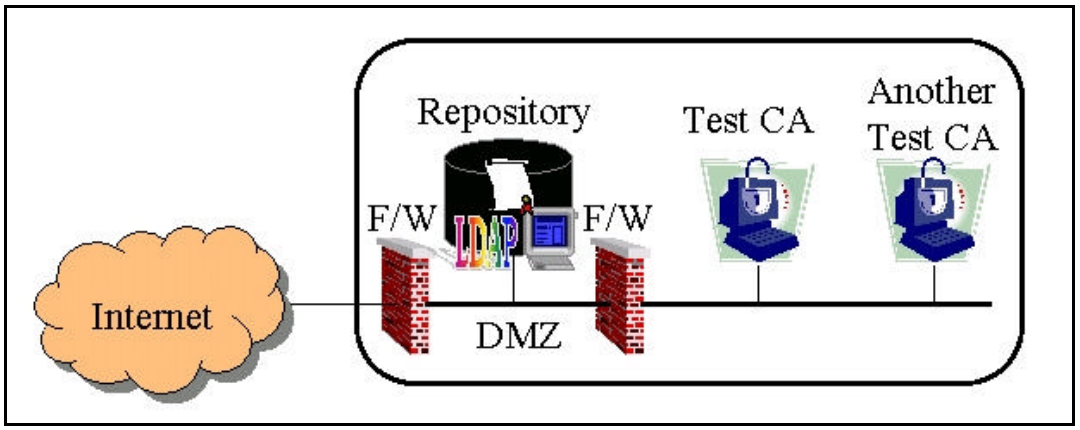
CA server is located in the CHT Lab Intranet area; it issues a variety of certificates (CC, EE, CRL and ARL) based on the JT Experiment Certificate and CRL Profile. The CA server stores certificates into the database server and publish valid certificates in the LDAP server.

Chinese Taipei Client and Server

(1) Secure E-mail clients with Microsoft Outlook Express 6. End Entities are located inside the DMZ of the CHT Lab simulation center.

(2) Outlook Express is more than an application; it also plays the role of Certificate Validation , which validate the certificate path on behalf of the End Entity.

(3) LDAP server is located in the DMZ area and it contains the Cross certificate pair (CCP), CRL, ARL (for JP end entity and CA, respectively); due to the characteristic of the secure E-mail, testing users from Chinese Taipei need to access to JP LDAP server to retrieve the necessary certificate information.



The table below summarizes the specifications of the Secure E-mail experiment environment:

Item	Context
Base Specification	Chinese Taipei GPKI specifications
Certificate profile	X.509 (97) v3, RFC3280
Certificate encoding format	DER
CRL profile	X.509 (97) v2, RFC3280
CRL encoding format	DER
Cross-Cert request format	PKCS#10
Cross-Cert response format	X/509/DER
The method of sending fingerprint	E-mail
POP (proof of possession)	Verification of digital signature on certificate request format
Storage device	FDD
EE Certificate response format	PKCS#12 (Private-key included)
Repository access protocol(e.g., LDAPv2, LDAPv3, DAP)	LDAPv3
Certificate path validation method	RFC2459 (son of) and RFC 3280
Certificate validation entity	EE

The experiment applications utilized in each component are listed in the following table:

Component	Applications Utilized
CA Server	CHT CA Server (Developed by CHT)
LDAP Server	Netscape Directory Server 4
Client Application	MS-Outlook Express 6
Client Operating Systems	MS Windows 98, Windows 2000, Windows XP

4.1.3 Test Plans and Results

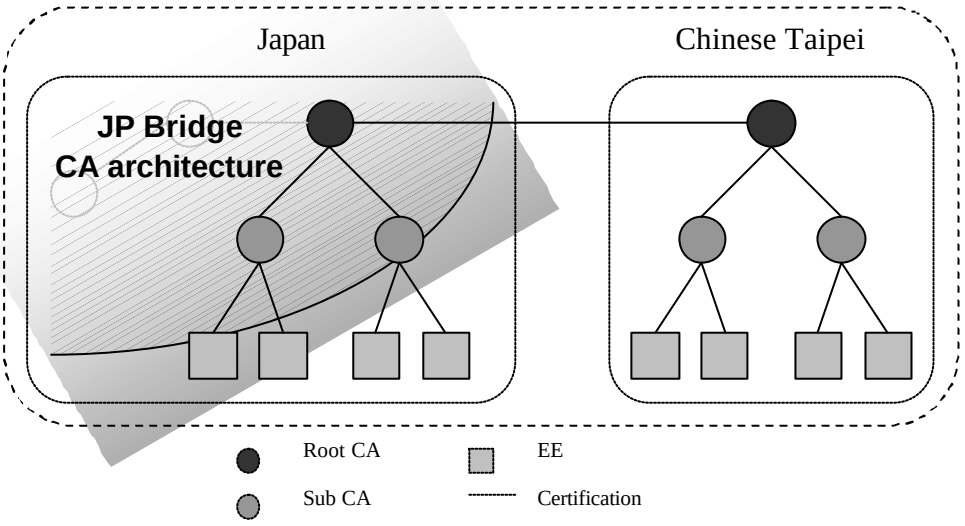
4.1.3.1 Overall Test Plan

4.1.3.1.1 Test scenario

The test plan was established based on the aforementioned CA-CA models and test scenario. There are two CA operators from Chinese Taipei participating in this experiment; one represents Taiwan Government PKI (Chunghwa Telecom) and the other stands for a local commercial CA (TWCA). Each CA was defined differently for the experiment phases.

Scenario 1. Chunghwa Telecom CA to Japan CA

This scenario included two simulated Cross Certification environments in Japan GPKI and Chinese Taipei GPKI, respectively. In order to carry out the test, the test plan was divided into two phases. The first phase featured the test cases in which a Chinese Taipei CA cross-certified with the Japan GPKI CA. The second phase featured the test cases in which a Japanese CA cross-certified with the Chinese Taipei CA. The following figure shows the test scenario of the first phase.



Scenario 2. TWCA CA to Japan CA

This scenario included two CA-CA models. In order to carry out the test, the test plan in the scenario 2 was divided into the two phases. The first phase featured the test cases in which a TWCA and the Japan CA reached interoperability via cross recognition model while the second phase featured TWCA and the Japan CA reached interoperability via cross certification model.

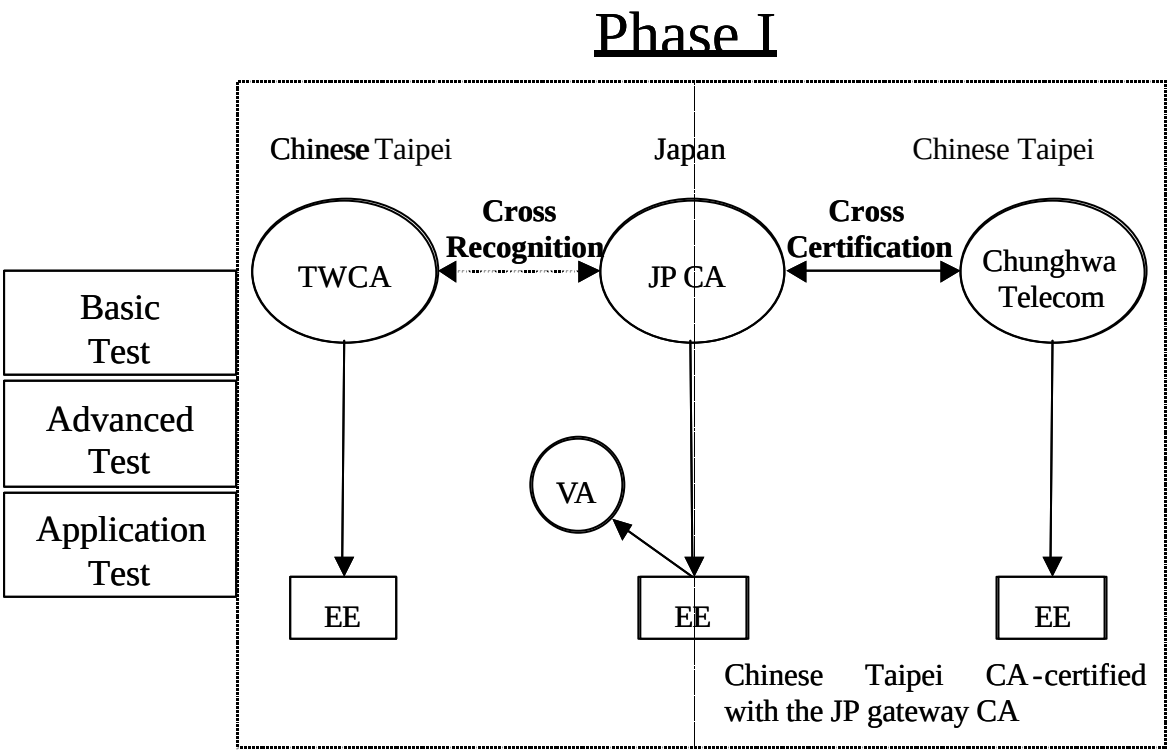
4.1.3.1.2 Test cases

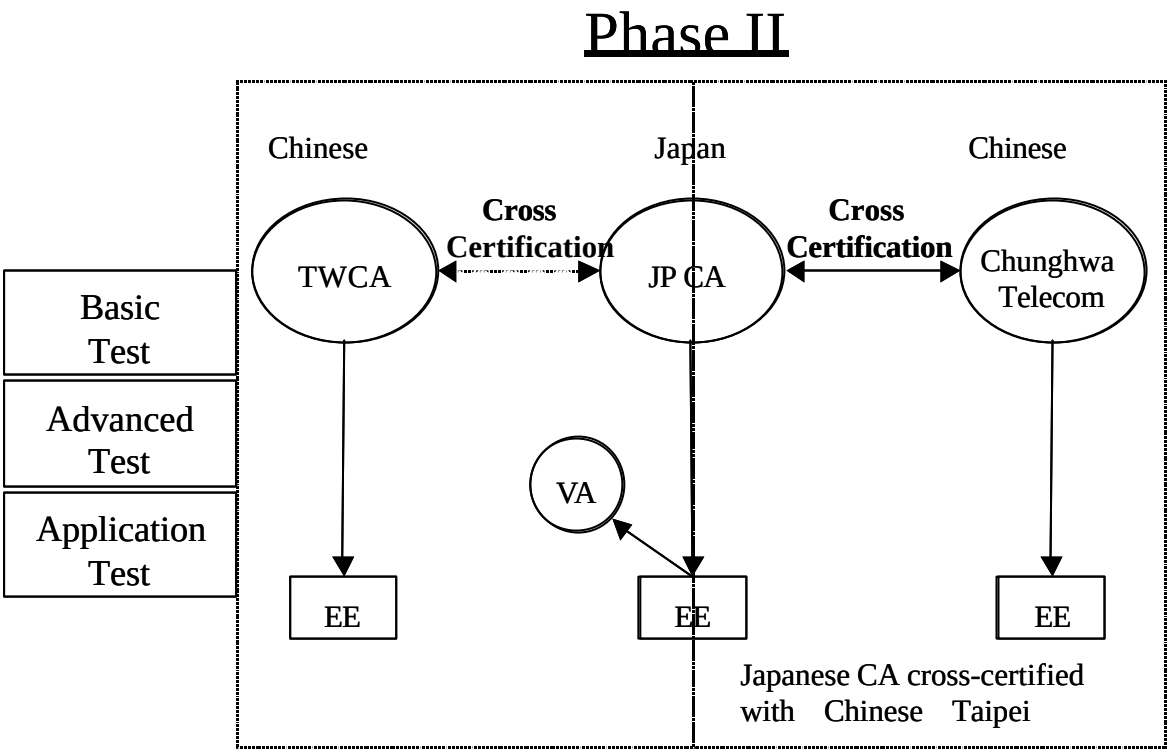
The test cases were developed based on the "2001 JKS Interoperability Project" including three segments as of "basic test", "advanced test" and "application test". Additionally, the test included 2 phases in each test segment. The definitions of each test segment are described

below: Basic Test includes the test cases for the cross certification process; Advanced Test includes basic Path Construction and Path Validation test cases; Application Test includes the transactions using different certificates with the business scenarios established.

Test Phase	Contents
Basic Test	Issue certificates
	Revoke certificates and issue CRL/ARL
	Update certificates
	Renew certificates
Advanced Test	Path Construction and Validation
	Key Changeover (Pending)
Application Test	Japan application business scenario TWCA application business scenario ChungHwa Telecom business scenario

The two figures below show the summarized test environments in the test plan.





4.1.3.2 Overall Test Results

Japan and Chinese Taipei members conducted the experiment and obtained the experiment All the test items were completed successfully and some issues have been identified as well.

		Chinese Taipei			
		TWCA		ChungHwa Telecom	
JAPAN		Phase1: CR	Phase2: CC	Phase1: CC	Phase2: CC
	BasicTest	All scenarios completed	All scenarios completed	All scenarios completed	All scenarios completed
	Advanced Test	All scenarios completed	All scenarios completed	All scenarios completed	All scenarios completed
	Application Tests	All scenarios completed	All scenarios completed	All scenarios completed	All scenarios completed

In the case of Chunghwa Telecom CA to interoperate with Japan CA using secure email as the application, it was found that, when the certificates were issued with the "CertificatePolicy" extension set as "Critical", the mail application Outlook Express (on the OS of Windows 98/2000) was not able to choose such certificate for implementing either digital signature or encryption function. Meanwhile, when using Outlook Express in Windows XP platform, certificates with "CertificatePolicy" extension as of "Critical" or "Non-Critical" were acceptable. However, in the Critical "CertificatePolicy" extension case, Outlook Express would simply accept the certificate even though it did not really understand the meaning of the critical

Certificate Policy OID.

Issues related to CRLDP were also identified: from the analysis of network sniffer tool when performing S/MIME test, it was realized that Windows Operation System would try to retrieve the CRL specified by the CRLDP to confirm the status of the received EE certificates. It was also found that, windows OS supports both LDAP URI and HTTP URI, which is with higher priority, in the CRLDP. In the LDAP URI case, its attribute part can be with or without binary option. However, if the attribute part was omitted from the LDAP URI, Outlook Express would not be able to successfully download the updated CRL from the LDAP server. The risk of accepting revoked EE certificates would therefore rise.

To resolve the concerns found in the experiment, the output of "IWG Certificate and CRL Profile" was changed in order to simulate the test scenarios in more realistic ways. The table below shows the amendments of the profile.

Cross certification (CC) certificates, intermediate CA certificates and EE certificates			
Domain name	JPGPKI	CTGPKI	NOTE
issuingDistributionPoint	mandatory if CRLDP is set	optional	In real cases now, CTGPKI is using "HTTP URI"
certificatePolicies	critical	either critical or non-critical	To make usable in existing applications

4.1.4 Recommendations

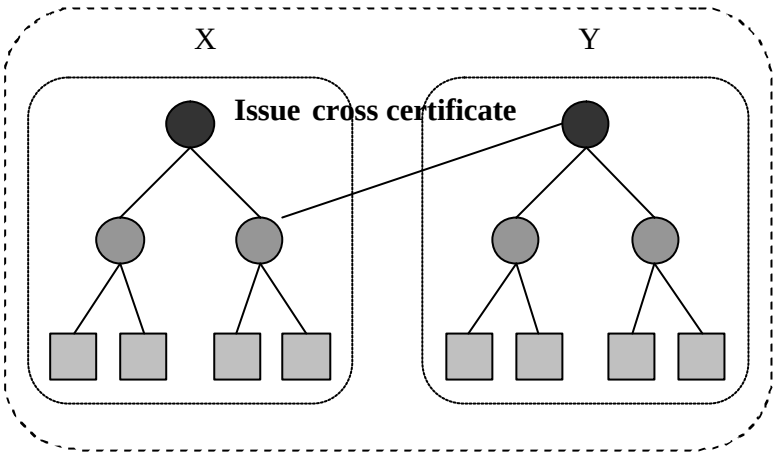
4.1.4.1 CA-CA model refined

The experiment explored the CA-CA interoperability by adopting one of the CA-CA models; CC model, in which a party in one domain issues a certificate to connect to the infrastructure in other domains. In the CC model, usually the root CAs, which often called principal CA, issue cross certificates each other and establish the CC relationship. However, there are some CC cases that do not fall into. When a root CA in one domain issues a cross certificate to a subordinate CA in other domains, there are some consideration that should be taken.

(1) The trust relationship disassociation and its level ambiguity
In the PKI system, Root CA is typically the source of trust and distributing the public key of the trust anchor. Subordinate CAs normally plays the role of issuing the certificates to EE and other subordinate CA, specifying the purpose of the public key usage and other information in certificate that are issuing.
Technically, it is feasible to achieve the CC relationship between Root CA and Subordinate CAs. However, it is very ambiguous and even causing policy mapping confusion. Clear objective and policy mapping control is highly required.

(2) Requirement for Path Construction

When a Root CA in one domain and a subordinate CA in other domain cross-certify each other, there are some requirements to achieve correct path construction. For example, when the root CA in domain Y cross certifies with one of the subordinate CAs in X domain shown as the chart below, and EE in domain X attempts to validate a certificate of EE in domain Y,



the following certificate and CRL information should be stored in repository.

	Root CA as Trust Anchor	Subordinate CA as Trust Anchor
Forwared Path Construction	Subordinate CA certificate stored in crossCertificatePair.issuedByThisCA of Root CA-X entry	No requirement
Reverse Path Construction	Subordinate CA certificate stored in crossCertificatePair.issuedToThisCA of Subordinate CA -X entry	No requirement
Hybrid Path Construction[6]	No requirement	No requirement

When client application adopts the forward path construction, the subordinate CA certificate must be stored in crossCertificatePair.issuedByThisCA of Root CA-X entry. When it adopts the reverse path construction, the subordinate CA certificate must be stored in the crossCertificatePair.issuedToThisCA of Subordinate CA -X entry. When it employs the subordinate CA as trust anchor, no such requirement should be mandatory.

Without having the requirements, the software client in domain X must set the subordinate CA to its trust anchor to start the certificate path processing. This maybe complicate the fact that the software client should be clever enough to switch the trust anchor when a Relying Party has a certificate to be validated in the particular CC relationship.

[6] This combined the path construction between the forward path construction and reverse path construction.

In sum, this CC model needs to understand the trust relationship and its level of equality and path construction requirement in repository or/and client application.

4.1.4.2 CRL distribution security considerations

There are some security considerations on CRL distribution and profile requirement when a CA-CA model is established between different PKI domains. There are at least 4 types of the CRL distribution (excluding the delta CRL and indirect CRL).

(1) CA publishes one full CRL
(2) CA publishes partitioned CRLs only
(3) CA publishes one complete CRL and one complete ARL
(4) CA publishes partitioned CRLs, and one complete ARL or partitioned ARLs

Before going to the discussion, the following terms are defined tentatively.

1. full CRL is a CRL that lists all revoked certificates including the all EE and CA certificates.
2. complete CRL(ARL) is a CRL that lists all revoked certificates within two given scopes. One is the set of the certificates covered by the CRL that contains all the EE certificates only. The other is the set of the certificates covered by the CRL that contains all the CA certificates only.
3. partitioned CRL is a partition of a full CRL or complete CRL(ARL), partitioned with some kinds of the criteria such as the range of the certificate serial numb er or some other ad hoc range. The criteria depend on the CA policy. The CA makes sure that the union of the full set of the partitioned CRL should be equivalent to a full CRL. This profile assumes that the partitioned CRL must be published at the locations of the cRLDistributionPoint.DistriubtionPoint.fullName and issuingDistributionPoint.distributionPoint.fullName fields.

It is very important to coordinate CRL distribution policy with the profile requirement in order to expect all PKI software to validate the CRL with commonly agreed range of Revocation List information that CRL covers. The CRL publication policy and the covered range of Revocation List information are expected to comply with the following assumptions:

(1) A CRL without the issuingDistributionPoint(iDP) extension is expected to cover all the revocation information of all unexpired certificates, including both CA and EE certificates, all issued by the CRL issuer (assuming the certificate-issuing CA). The CRL must be a full CRL.
(2) A CRL with the iDP extension with the onlyContainsUserCerts field is expected to cover all the revocation information of all unexpired EE certificates, all issued by the CRL issuer (assuming the certificate-issuing CA). A CRL with the iDP extension with the onlyContainsCACerts field is expected to cover all the revocation information of all unexpired CA certificates, all issued by the CRL issuer (assuming the certificate-issuing CA). The CRLs must be a complete CRL and complete ARL.
(3) A CRL with the iDP extension with distributionPoint.fullName field is expected to be a partitioned CRL, (or a full CRL and complete CRL/ARL published at the location of the

distributionPoint.fullName). It is important that the RP must make sure that one of the names in the distribution point fields in the CRL distribution point extension (cRLDP) must match one of the names in the distribution point field in iDP to prevent the CRL substitution attack. The CA should make one of the names in the distribution point field of the certificate's cRLDP EXACTLY the same as one of the names in the distribution point fields in the CRL's iDP. It is also important that this is the CA's responsibility to issue valid partitioned CRLs with a given scope.

The values of issuingDistributionPoints are specified based on the CRL publication policies above.

(1) CA publishes only one full CRL (no ARL)
iDP -- Optional (critical/non-critical)
distributionPoint -- Optional
fullName – **Optional** (EXACTLY the same value as one of the names in the cRLDP in the certificate)
nameRelativeToCRLIssuer -- not defined
onlyContainsUserCerts -- forbidden to use
onlyContainsCACerts -- forbidden to use

(2) CA publishes partitioned CRLs (no ARL)
iDP -- Mandatory (critical)
fullName –**Mandatory** (EXACTLY the same value as one of the names in the cRLDP in the certificate)
nameRelativeToCRLIssuer -- not defined
onlyContainsUserCerts -- forbidden to use
onlyContainsCACerts -- forbidden to use

(3) CA publishes one complete CRL and one complete ARL
iDP -- Mandatory (critical)
distributionPoint -- Optional
fullName –Optional (EXACTLY the same value as one of the names in the cRLDP in the certificate)
nameRelativeToCRLIssuer -- not defined
onlyContainsUserCerts -- **Mandatory in CRL**
onlyContainsCACerts -- **Mandatory in ARL**

(4) CA publishes partitioned CRLs and ARL(s)
iDP -- Mandatory (critical)
distributionPoint -- **Mandatory**
fullName –Mandatory (EXACTLY the same value as one of the names in the cRLDP in the certificate)
nameRelativeToCRLIssuer -- not defined
onlyContainsUserCerts -- **Mandatory in CRL**
onlyContainsCACerts -- **Mandatory in ARL**

4.1.4.3 CRLDP: URI format and binary option

The experiment showed that when adopting LDAP URI in CRLDP, the following format should create the least problems:

ldap://*hostname*[:*portnumber*]/*dn*?*attribute*[;binary]

The portnumber part is optional; the attribute part should be mandatory and the binary option is optional.

Please note that IETF LDAPbis WG had already decided to removed all mention of transfer encodings and the binary attribute option from the LDAPv3 "core" specification and leaved the IETF PKIX WG to decide whether to retain the binary option in the PKIX-related LDAP schema RFC. Until the moment of finalizing this report, it seems that the two mentioned working groups have not yet reached consensus regarding the binary option issue.

In the experiment, some members encountered the issue on the encoding of "DistinguishedName(DN)" in LDAP URI form. When the comma character is used in the DN in LDAP URI, the characters should be encoded using the RFC 2255. However there should be a caution that the encoding must go through the following two steps:

1) DN in certificate to DN of LDAP string representation (RFC 2253)
2) DN of LDAP string representation to DN of LDAP URI (RFC 2255)

For example, when the following DN is written in a certificate

Country = JP
Organization = ABC Co., Ltd.

The software first should escape the comma character to distinguish the delimiter function by doing:

1. o=ABC Co.\2c Ltd., c=JP
2. o=ABC Co.\, Ltd., c=JP
3. o="ABC Co., Ltd.", c=JP (LDAP v2)

Then the software should escape the special characters by doing:

1'. ldap://example.url/o=ABC%20Co.%5c2c%20Ltd., c=JP
2'. ldap://example.url/o=ABC%20Co.%5c,%20Ltd., c=JP
3'. ldap://example.url/o=%22ABC%20Co.,%20Ltd.%22, c=JP

The caution should be made when the above 2' and 3' LDAP URIs are processed. There are comma characters (not a function of delimiter, but a string representation) that appears in the string RDN sequence. For the interoperability purpose, the comma character is recommended to be escaped when used as string representation. However the un-escaped comma character can appear in a string RDN sequence above and it is not illegal.

4.1.4.4 ASN1 INTEGER TYPE and serial number

The implementation issues on the serial number have been raised in the experiment. The serial number is defined in ASN.1 INTEGER TYPE and expected to have longer than the 4 bytes in general. However some application cannot process the longer serial number since the application expect the serial number to be within the 4 bytes long.

This issue simply reveals the fact that the ASN.1 INTEGER TYPE and Data Type 'int' in program language are not associated with each other.

4.1.4.5 DN matching rule refinement

For some PKI implementations, such as Taiwan Government PKI, the use of UTF-8 is mandatory. The IWG profile strongly recommends using the UTF8STRING as default encoding. However the PrintableString is acceptable and still valid to maintain the backward compatibility with legacy and web browser systems. What has been learned in the experiment is that, for DN matching rule, it is highly recommended to follow the rules stated in the PKIX standard:

Conforming implementations are REQUIRED to implement the following name comparison rules:

(a) attribute values encoded in different types (e.g., PrintableString and BMPString) MAY be assumed to represent different strings;

(b) attribute values in types other than PrintableString are case sensitive (this permits matching of attribute values as binary objects);

(c) attribute values in PrintableString are not case sensitive (e.g., "Marianne Swanson" is the same as "MARIANNE SWANSON"); and

(d) attribute values in PrintableString are compared after removing leading and trailing white space and converting internal substrings of one or more consecutive white space characters to a single space.

Please refer to the section 4.2.6.3.

4.2 Path Processing Experiment

4.2.1 Experiment Overview & Scope

As the IWG Certificate and CRL profiles have been created, we hope that the certificates issued according to the IWG profiles will be widely used in the multi PKI domains. But we confronted the realities that PKI S/Ws in respective countries might deal with those compliant certificates in different ways. If any, it shall damage the PKI Interoperability we have strived for.

We adopted the IWG Certificate and CRL Profile and RFC3280 Path Processing Algorithm for this experiment. In the multi PKI domains interoperability (especially different vendors in different countries involved), when no levels of conformance are guaranteed in terms of path validation processing, it would be difficult to ensure a Relying Party application in one country will validate the certificate and its path in the same way that the other does in other countries, and to achieve the reliable infrastructure where secure business transactions are conducted.

In the experiment, we've created the certificate path processing implementation guideline and common agreeable test suites and the guideline on which PKI developers can implement the complicated RFC3280 Path processing algorithms well and test if the implementations are compliant in the multiple CA topology and trust models.

4.2.2 Path Processing Guidelines

4.2.2.1 Goals and Concepts

The objective of the Certificate Path Processing Implementation Guideline(for short PPIG) is to help the PKI developers to implement the complicated Certificate Path Processing algorithms, since they are essential to make the multi PKI domains interoperable with each other.

Also the objective of the Certificate Path Processing Testing Guideline(for short PPTG) is to help the PKI developers and evaluators to narrow down the test requirements from the complicated algorithm.

The certificate path validation algorithms in RFC 3280 are so complex and difficult that it is possible for PKI developers to make mistakes in implementing its algorithms.

Since IWG members hope that all PKI S/Ws output one same result on the certain certificate paths in the interoperable PKI domains, PPIG is developed to minimize any possible mistakes and errors by the PKI developers at implementation levels, and PPTG is developed to select minimal requirements to test by PKI developers and evaluators at evaluation levels.

Path Processing Experiment establishes a test environment by participating parties. The potential developers and evaluators can actually test them by themselves, using the environment.

4.2.2.2 Characteristics of Guidelines

4.2.2.2.1 PPIG

The procedures of validating certificate paths, so called “Certificate Path Processing”, consists of two main parts. One is the certificate path validation and the other is certificate path construction. In this guideline, those two main parts of certificate path processing and some additional considerations are described .

In the first part, the certificate path validation algorithm, which is based on the algorithm described in RFC 3280 will be covered. According to the algorithms, the conformant implementation must output the same results for the same inputs. To help such a conformant implementation, this guideline will give detailed explanations about the algorithm. And this guideline will derive some interoperability requirements between different PKI domains and will describe some considerations to support the IWG recommended profiles.

For the second part, this guideline will include the restricted certificate path construction algorithm with some environmental assumptions. The certificate path construction procedure depends on many environmental aspects, such as interoperability model, repository and CRL distribution model. For the reasons listed above, it is very difficult to derive a general algorithm for certificate path construction. Hence, Some assumptions have been made on which the restricted certificate path construction algorithm have been developed.

At the end of this guideline, Some considerations for the certificate validation software will be covered.

4.2.2.2.2 PPTG

What RFC3280 described about certificate path processing is solely an algorithm example. It does not describe the test requirement and the test method. And, the test requirement is also complicated because the certificate path processing algorithm is complicated.

Therefore, the testing guideline defines the test requirement and the test method for satisfying the path validation requirement in RFC3280. This helps to extract a minimal test requirement for trust model.

This guideline consists of the testing models and its testing requirements, testing assumptions, and testing items for each model.

4.2.3 Path Processing Test Participants

The following companies from IWG member countries Japan, Korea, and Chinese Taipei joined the Path Processing Experiment.

Japan PKI Forum
Japan

Hitachi, Ltd.

Japan

Korea Information Security Agency
Korea

National Information Infrastructure Enterprise Promotion Association
Chinese Taipei

Chunghwa Telecom Co., Ltd.
Chinese Taipei

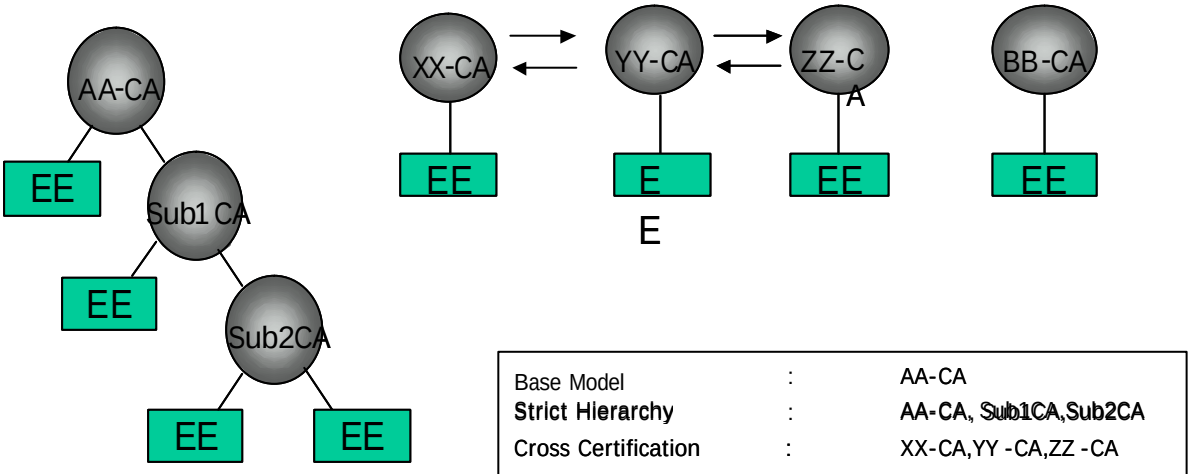
4.2.4 Path Processing Test Environments

This section describes the test environments used by the IWG members to perform the Path Processing tests. IWG members generated the certificates and CRLs for the selected test cases and stored them in repository. Each IWG member developed or/and used the client applications or Certificate Validation Servers with built-in Path Processing capability.

All the certificates and CRLs (including invalid data) are generated based on the Path Processing Testing Guidelines and tested by each IWG members to see if each test result will be matched with the expected result corresponding to the particular test case in the guideline.

4.2.4.1 CA Hierarchical structure

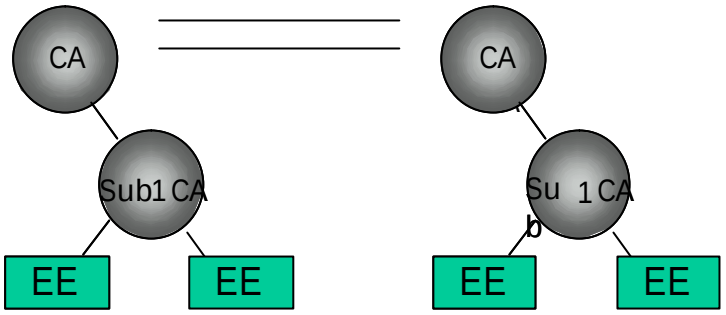
This section describes the CA structure for the test cases in the guideline. The following picture shows the overall structure constructing CA relationship and test cases.



Base Model	:	AA-CA
Strict Hierarchy	:	AA-CA, Sub1CA,Sub2CA
Cross Certification	:	XX-CA,YY -CA,ZZ -CA
Cross Recognition	:	AA-CA, YY -CA
Servic	:	AA-CA
Revocation	:	BB-CA

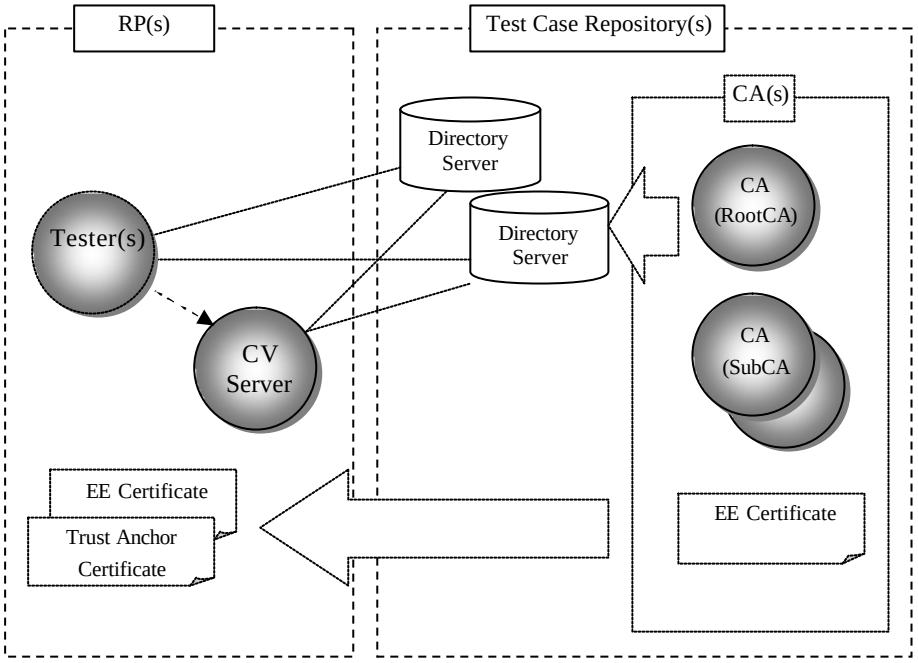
The Base Model and Strict Hierarchy Model employ the AA and its sub CA hierarchy. The Cross Certification and Cross Recognition Model use the XX, YY, and ZZ CA structure. The Service Model and Revocation Model use the AA CA and BB CA respectively.

To construct more realistic Cross Certification Model test environments among the parties in the experiment, the following CA hierarchical structures are adopted. Each CA has its subordinate CA that issues certificates to end entities. Each trust anchor CA (root CA) cross certifies each other.



4.2.4.2 Relying Party Test Environments

This section describes the RP test environment in each party participated in the experiment.



Each party prepares the Relying Party (RP) environment and Test Case Repository. The test CA and its subordinate CA issue the test certificates and send the EE certificates to the RPs. In the test case repository, the CA stores all the information necessary for the RP to validate certificates and its path. In the RP environment, a tester initiate each test case using the trust anchor information and EE certificate to be validated. In some environment, the RP delegates the path processing function to a remote Certificate Validation (CV) server. The application or Certificate Validation (CV) server access to the repository and collect all the information and then validate the certificate and its path.

4.2.4.2.1 Japan Relying Party and Test Case Repository Environment

Test Environment	Details
Path Processing Module/Application	Hitachi Certificate Validation Server
Repository	NEC EnterpriseDirectoryServer
	Netscape Directory Server 4.12
	iPlanet Directory Server 5.1

4.2.4.2.2 Korea Relying Party and Test Case Repository Environment

Test Environment	Details
Path Processing Module/Application	PKI Client that has the functions of Path Construction and Validation based on the CRL model. Platform : Windows2000 Professional

Repository	Two Netscape Directory Servers version 4.1
	One for RootCA Directory and Other for SubCA Directory Referral to Japanese Directory

4.2.4.2.3 Chinese Taipei Relying Party and Test Case Repository Environment

Test Environment	Details
Path Processing Module/Application	Chunghwa Telecom HiSecure SDK Enterprise version 5.0 This SDK contains path validation related functions designed to be conformable to RFC 3280. (A small program was written to drive the path validation functions of the SDK to go through all test cases for CC model.)
Repository	Netscape Directory Server 4.12

4.2.5 Path Processing Test Experiment Results

4.2.5.1 Test Models and Test Results by Japan

This section describes the test models and test results performed by Japan. Japan selected all Test models and CC model with Korea and Chinese Taipei members.

4.2.5.1.1 Test Models

The table below describes the test models selected and initial information setting in Japan.

Test Model	Entity	Profile	Parameters	
Base	1) AA Self-signed CA 2) End Entity	IWG Profile	user-initial-policy-set	any-policy
			trustAnchorInfo	RP's AA Self-signed CA
			initial-explicit-policy	False
Strict Hierarchy	1) AA Self-signed CA 2) AA Sub1 CA 3) AA Sub2 CA 4) End Entity	IWG Profile	user-initial-policy-set	policy-AA
			trustAnchorInfo	RP's AA Self-signed CA
			initial-explicit-policy	True
Cross Certification	1) JP Self-signed CA 2) Sub1 CA 3) End Entity	IWG Profile	user-initial-policy-set	policy-JP, policy-KR or policy-CT
			trustAnchorInfo	JP Self-signed CA

	4) KR Self-signed CA 5) KR Sub1 CA 6) KR End Entity 4) CT Self-signedCA 5) CT Sub 1 CA 6) CT End Entity		initial-explicit-policy	True
Cross Recognition	1) AA Self-signed CA 2) AA Sub1 CA 3) AA Sub2 CA 4) AA End Entity 5) YY Self-signed CA 6) YY Sub1 CA 7) YY End Entity	IWG Profile	user-initial-policy-set	policy-AA, policy-YY
			trustAnchorInfo	RP's AA Self-signed CA, YY Self-signed CA
			initial-explicit-policy	True
Service	1) AA Self-signed CA 2) End Entity	IWG Profile	user-initial-policy-set	policy-AA
			trustAnchorInfo	RP's AA Self-signed CA
			initial-explicit-policy	True
Revocation	1) BB Self-signed CA 2) End Entity	IWG Profile	user-initial-policy-set	Unspecified
			trustAnchorInfo	RP's BB Self-signed CA
			initial-explicit-policy	Unspecified

4.2.5.1.2 Test Results

The tables below describes the test results obtained by Japan

Base Model

Test Case	Expected Result	Test Result
Base.RP.07.01	Success	Success
Base.RP.08.01	Failure	Failure
Base.RP.09.01	**Failure**	**Success**
Base.RP.10.01	Failure	Failure
Base.RP.11.01	Failure	Failure
Base.RP.12.01	Failure	Failure
Base.RP.13.01	Failure	Failure

Base.RP.14.01	Failure	Failure
Base.RP.15.01	Failure	Failure
Base.RP.16.01	Failure	Failure
Base.RP.17.01	Failure	Failure
Base.RP.18.01	Failure	Failure
Base.RP.19.01	Failure	Failure
Base.RP.20.01	Failure	Failure

Base RP.09.01 is a test case where the RP checks if the white space can be significant or not in DN. The test case assumed the strict UTF8 String matching rule described in the RFC 3280. However, the path processing module is expected to implement X.501 name comparison algorithm.

Strict Hierarchy Model

Test Case	Expected Result	Test Result
Int.SH.RP.08.01	Success	Success
Int.SH.RP.09.01	Failure	Failure
Int.SH.RP.10.01	Failure	Failure
Int.SH.RP.11.01	Failure	Failure
Int.SH.RP.12.01	Failure	Failure
Int.SH.RP.13.01	**Failure**	**Success**
Int.SH.RP.14.01	Success	Failure
Int.SH.RP.14.02	Failure	Failure
Int.SH.RP.15.01	**Failure**	**Success**
Int.SH.RP.16.01	Failure	Failure
Int.SH.RP.17.01	**Failure**	**Success**
Int.SH.RP.18.01	Failure	Failure
Int.SH.RP.19.01	Success	Success
Int.SH.RP.19.02	Failure	Failure
Int.SH.RP.20.01	Failure	Failure
Int.SH.RP.21.01	Success	Success
Int.SH.RP.21.01	Failure	Failure
Int.SH.RP.22.01	Failure	Failure
Int.SH.RP.23.01	Failure	Failure

Int.SH.RP.13.01 is a test case where the RP checks if the critical flag appears in a particular extension field in the certificate. The test case assumed that a non-critical extension in the basicConstraints should be failed. However, the criticality is not an exact value to be validated in the profile.

Int.SH.RP.15.01 is a test case where the RP checks if the intermediate CA certificate has the KeyUsage and critical extension. The test case assumed that this extension should appear in the extension. However, the RFC 3280 path processing algorithm allows the certificate to be validated reagardless of the KeyUsage Extension.

Int.SH.RP.17.01 is the same test case as in the Int.SH.13.01

Cross Certification Model

Test Case	Expected Result	Test Result
Int.CC.RP.19.01	Success	Success
Int.CC.RP.20.01	Failure	Failure
Int.CC.RP.21.01	Failure	OUT OF SCOPE
Int.CC.RP.22.01	Failure	Failure
Int.CC.RP.23.01	Success	Success
Int.CC.RP.23.02	Failure	Failure
Int.CC.RP.24.01	Success	Success
Int.CC.RP.24.02	Failure	Failure
Int.CC.RP.25.01	Success	Success
Int.CC.RP.25.02	Failure	Failure
Int.CC.RP.26.01	Failure	Failure
Int.CC.RP.27.01	Failure	Failure
Int.CC.RP.28.01	**Failure**	**Success**
Int.CC.RP.29.01	Success	Success
Int.CC.RP.29.02	Failure	Failure
Int.CC.RP.30.01	**Failure**	**Success**
Int.CC.RP.31.01	Failure	Failure
Int.CC.RP.32.01	**Failure**	**Success**
Int.CC.RP.33.01	Success	Success
Int.CC.RP.33.02	Failure	Failure
Int.CC.RP.34.01	Success	Success
Int.CC.RP.34.02	Failure	Failure
Int.CC.RP.35.01	Success	Success
Int.CC.RP.35.02	Failure	Failure
Int.CC.RP.36.01	Success	Success
Int.CC.RP.36.02	Failure	Failure
Int.CC.RP.37.01	Failure	Failure
Int.CC.RP.38.01	Failure	Failure
Int.CC.RP.39.01	Failure	Failure

Int.CC.RP.28.01 is designed for RP to check if the critical flag appears in a particular extension field in the certificate. The test case assumed that a non-critical extension should be failed. However, the criticality is not an exact value to be validated in the profile. In Similar case, Int.CC.RP.32.01 is a test case where the RP checks if the intermediate CA certificate has the Key Usage and critical extension. However the criticality itself is not a value to be validated.

Int.CC.RP.30.01 is designed for RP to reject the path processing if the an intermedia CA certificate that does not include the keyUsage. The test case assumes that this extension should appear in the extension. However, the RFC 3280 path processing algorithm allows the certificate to be validated reagardless of the KeyUsage Extension.

Cross Recognition Model

Test Case	Expected Result	Test Result
Int.CR.RP.05.01	Success	Success
Int.CR.RP.06.01	Failure	OUT OF SCOPE
Int.CR.RP.07.01	Failure	Failure
Int.CR.RP.08.01	Failure	Failure
Int.CR.RP.09.01	Failure	Failure
Int.CR.RP.10.01	Failure	Failure
Int.CR.RP.11.01	Success	Success
Int.CR.RP.11.02	Failure	Failure
Int.CR.RP.12.01	Success	Success
Int.CR.RP.12.02	Failure	Failure
Int.CR.RP.13.01	Success	Success
Int.CR.RP.13.02	Failure	Failure

In the Cross Recognition Model, all the test cases are verified and succeeded. No particular test cases are found.

Service Model

Test Case	Expected Result	Test Result
Svc.DS.RP.06.01	Success	Success
Svc.DS.RP.07.01	**Failure**	**Success**
Svc.DS.RP.07.02	**Failure**	**Success**
Svc.DS.RP.07.03	**Failure**	**Success**
Svc.DS.RP.07.04	Success	Success
Svc.DS.RP.08.01	Failure	Failure
Svc.DS.RP.09.01	Success	Success
Svc.DS.RP.09.02	Failure	Failure
Svc.DS.RP.10.01	Success	Success
Svc.DS.RP.10.02	Failure	Failure
Svc.DS.RP.11.01	Success	Success
Svc.DS.RP.11.02	Failure	Failure

Svc.DS.RP.07.01 is a test case where RP checks if the user certificate has the KeyUsage and critical extension. The test case assumed that this extension should appear in the extension. However, the RFC 3280 path processing algorithm allows the certificate to be validated reagardless of the KeyUsage Extension.

Svc.DS.RP.08.01 is a test case where RP checks if the user certificate has inconsistent Key usage extension (Key Encipherment instead of Digital Signature). This test case is only run when the configuration accepts the all the Key Usage bits.

Svc.DS.RP.09.01 is a test case where RP checks if the critical flag appears in a particular extension field in the certificate. The test case assumed that a non-critical extension should be

failed. However, the criticality is not an exact value to be validated in the profile.

Revocation Model

Test Case	Expected Result	Test Result
Rvk.CRL.RP.08.01	Success	Success
Rvk.CRL.RP.09.01	Failure	Failure
Rvk.CRL.RP.10.01	Failure	Failure
Rvk.CRL.RP.11.01	Failure	Failure
Rvk.CRL.RP.11.02	Failure	Failure
Rvk.CRL.RP.12.01	Success	Success
Rvk.CRL.RP.12.02	Failure	Failure
Rvk.CRL.RP.13.01	Success	Success
Rvk.CRL.RP.13.02	**Failure**	**Success**
Rvk.CRL.RP.13.03	Failure	Failure
Rvk.CRL.RP.13.04	Failure	Failure
Rvk.CRL.RP.13.05	**Failure**	**Success**
Rvk.CRL.RP.14.01	Failure	Failure
Rvk.CRL.RP.15.01	Failure	OUT OF SCOPE
Rvk.CRL.RP.15.02	Failure	OUT OF SCOPE
Rvk.CRL.RP.16.01	Failure	OUT OF SCOPE
Rvk.CRL.RP.17.01	Failure	Failure
Rvk.CRL.RP.17.02	Success	Success
Rvk.CRL.RP.18.01	Success	Success
Rvk.CRL.RP.19.01	Failure	Failure
Rvk.CRL.RP.20.01	Failure	Failure
Rvk.CRL.RP.21.01	Failure	Failure
Rvk.CRL.RP.22.01	Success	Success
Rvk.CRL.RP.23.01	Failure	Failure
Rvk.CRL.RP.24.01	Success	Success
Rvk.CRL.RP.25.01	Failure	Failure
Rvk.CRL.RP.26.01	Failure	Failure
Rvk.CRL.RP.27.01	Success	Success
Rvk.CRL.RP.28.01	Failure	Failure
Rvk.CRL.RP.29.01	**Failure**	**Success**
Rvk.CRL.RP.30.01	Success	Success
Rvk.CRL.RP.31.01	Failure	Failure

Rvk.CRL.RP.13.02 is a test case where RP checks if the critical flag appears in a particular extension field in the certificate. The test case assumed that a non-critical extension should be failed. However, the criticality is not an exact value to be validated in the profile.

Rvk.CRL.RP.13.05 is the test case where RP checks if the user certificate has the KeyUsage and critical extension. The test case assumed that this extension should appear in the extension.

However, the RFC 3280 path processing algorithm allows the certificate to be validated reagardless of the KeyUsage Extension. Apparently, this test case is extremely redundant. This test case should be deleted.

Rvk.CRL.RP.29.01 is a test case where the value of the full name in IssuingDistributionPoint extension should matched to the one in the CRLDistributionPoints extension. This is not the error case. This is only required in Japan GPKI architecture. So this test case is not needed to be changed at all.

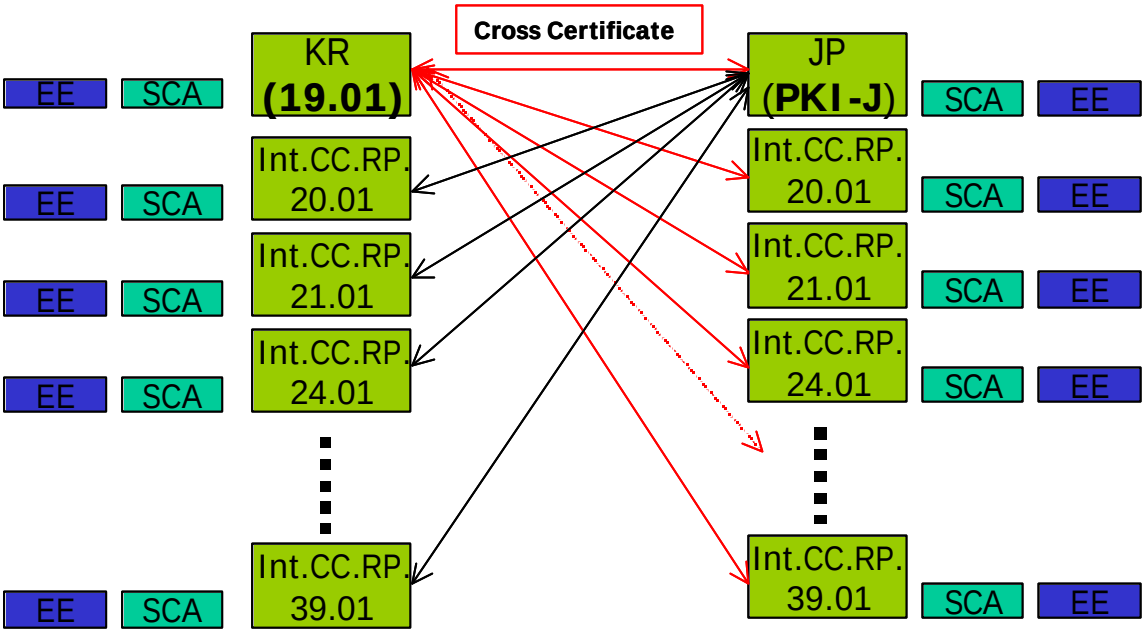
4.2.5.2 Test Models and Test Results by Korea

Test Models

Test Model	Entity	Profile	Parameters	
Cross Certification	JP Self-signed CA JP Sub CA JP End Entity	IWG Profile	user-initial-policy-set	policy-KR / Any policy
			trustAnchorInfo	KR Self-signed CA
			initial-explicit-policy	True /False
			initial-any-policy-inhibit	True/False
			initial -policy-mapping- inhibit	True/False

Korea chose the Cross Certification as the test model for this year experiment, however, has the plans to cover other test models in the future.

To make the experiments more practical, Korea has created 3-Level PKI (RootCA – SubCA – End Entity) structure and asked Japan to setup the equivalent PKI Structure and prepared two repositories which support the LDAPv3; one for RootCA and the other for Subordinate CAs. Following figure shows how the KR RootCA has been cross certified with Japan RootCAs for the tests.



Test Results

Test Case	Expected Result	Test Result
Int.CC.RP.19.01	Success	Success
Int.CC.RP.20.01	Failure	Not Tested
Int.CC.RP.21.01	Failure	Not Tested
Int.CC.RP.22.01	Failure	Failure or Success
Int.CC.RP.23.01	Success	Success
Int.CC.RP.23.02	Success	Not Tested
Int.CC.RP.24.01	Success	Success
Int.CC.RP.24.02	Failure	Failure or Success
Int.CC.RP.25.01	Success	Success
Int.CC.RP.25.02	Failure	Failure or Success
Int.CC.RP.26.01	Failure	Failure
Int.CC.RP.27.01	Failure	Failure
Int.CC.RP.28.01	Failure	Success
Int.CC.RP.29.01	Success	Failure
Int.CC.RP.29.02	Failure	Not Tested
Int.CC.RP.30.01	Failure	Failure
Int.CC.RP.31.01	Failure	Failure
Int.CC.RP.32.01	Failure	Failure
Int.CC.RP.33.01	Success	Failure
Int.CC.RP.33.02	Failure	Failure
Int.CC.RP.34.01	Success	Failure or Success
Int.CC.RP.34.02	Failure	Success
Int.CC.RP.35.01	Success	Failure
Int.CC.RP.35.02	Failure	Failure
Int.CC.RP.36.01	Success	Success

Int.CC.RP.36.02	Failure	Not Tested
Int.CC.RP.37.01	Failure	Failure
Int.CC.RP.38.01	Failure	Failure
Int.CC.RP.39.01	Failure	Failure

Here is the summary of test results by Korea.

Classification	Number of Test Items	Remarks
Identical to the expected Results	19 / 29	The results of test items are Int.CC.RP.22.01, 24.02, 34.01 and 34.02
NOT Identical to Expected Results	5 / 29	3-Level PKI Structure NOT considered (2) (29.01, 33.01) Korea's DN Encoding Problem (1) Non-Critical Extension Processings (1) Test condition is not suitable (1)
Test NOT Performed	5 / 29	Could NOT Generated Test Data (4) NOT registered in repository (1)

Korea found the test items whose expected results can be changed depending on such initial setting as initial-explicit-policy. Korea proposes setting the additional requirements to test items of PPTG as the following table indicates.

Test Items	Exp. Results	Additional Requirements to set in Test Guideline
Int.CC.RP.22.01	FAILURE	Initial-explicit-policy setting(T)
Int.CC.RP.24.02	FAILURE	Initial-explicit-policy setting(T)
Int.CC.RP.25.02	FAILURE	Initial-explicit-policy setting(T)
Int.CC.RP.34.01	SUCCESS	Initial-explicit-policy setting(F)

And Korea has output 5 different test results from the expected test results of PPTG.
Korea proposes setting new requirements to the test items at issues as the following table indicates.

Test Items	Exp. Results	Test Results	New Requirements to set
Int.CC.RP.28.01	FAILURE	SUCCESS	No need to test on RP or Need to change the expected results to SUCCESS
Int.CC.RP.29.01	SUCCESS	FAILURE	Let's change the test requirements 'PathLengthConstraint = 1', considering 3-Level PKI Structure.
Int.CC.RP.33.01	SUCCESS	FAILURE	Let's change the test requirements 'requireExplicitPolicy = 3', considering 3-Level PKI Structure.
Int.CC.RP.34.02	FAILURE	SUCCESS	Test conditions are not suitable, Always success
Int.CC.RP.35.01	SUCCESS	FAILURE	Failure from the fact that the DN of

			Japan is UTF8 Encoded but the DN of Korea is PrintableString encoded.

4.2.5.3 Test Models and Test Results by Chinese Taipei

Test Models

Test Model	Entity	Profile	Parameters	
Cross Certification	1) CT Root CA 2) JP Root CA 3) JP Sub CA 4) JP End Entity	IWG Profile	user-initial-policy-set	policy-CT
			trustAnchorInfo	CT Root CA
			initial-explicit-policy	False (for Int.CC.RP.33.01 and Int.CC.RP.33.02 test cases) True (for all other test cases)
			initial-any-policy-inhibit	True
			initial -policy-mapping- inhibit	False

In the PPTG experiment of this year, Chinese Taipei (CT) participated in the experiment of the CC model. In the experiment, CT created a 3-level PKI hierarchy to help creating cross-domain certification paths as designed by the test case designer. Also, CT tested CT's path validation module (the RP) with those cross-domain certification paths from CT's Root CA to JP's End Entity.

The test case designer of PPTG originally assumed that the local test environment for the CC model will be a 2-level PKI (RootCA – End Entity). However, later the participant countries (Japan, Korea and Chinese Taipei) all agreed to extended the local test environment to be a 3-level PKI (RootCA – SubCA – End Entity) to make the experiment more close to the real case. Therefore, the experiment involved validating the certification path from CT's Root CA to JP's End Entity for all the test cases of the CC model. Every certification path from CT's Root CA to JP's End Entity contains four certificates:

1) the self-signed certificate of CT's Root CA
2) the CA certificate issued from CT's Root CA to JP's Root CA
3) the CA certificate issued from JP's Root CA to JP's Sub CA
4) the EE certificate issued from JP's Sub CA to JP's End Entity

In general, CT use the following initial inputs to the path validation module (the RP):

1) The user-initial-policy-set contains only one CP OID. Our RP is supposed to accept CT's Domain CP OID as the only acceptable CP OID.
2) The trust anchor information comes from the self-signed certificate of CT's Root CA. Our RP is suppose to trust CT's Root CA as the only one trust anchor.
3) The initial-explicit-policy is set to 'True'. Our RP want to make sure all the certificates in the path after the trust anchor contains at least one valid CP OID.

4) The initial-any-policy-inhibit is set to 'True'. Our RP does not consider that the special anyPolicy OID as a valid CP OID.
5) The initial-policy-mapping-inhibit is set to 'False'. To accept cross certificates that bridge different PKI domain, our RP has to initially accept policy mappings.

However, there are two exceptions that the values of initial-explicit-policy are set to 'False' for test case Int.CC.RP.33.01 and Int.CC.RP.33.02. The reason is that during the path validation process, the state variable explicit_policy has the following characteristics:

1) It may be decreased, but may not be increase; and
2) If the initial-explicit-policy is set to 'True', then the initial value of explicit_policy is 0.

That means the initial-explicit-policy setting has the higher priority than the existence of the requireExplicitPolicy field in the Policy Constraints extension. Therefore, unless their initial-explicit-policy are set to 'False', the requireExplicitPolicy constraint in the cross certificate will never become effective and thus make test case Int.CC.RP.33.01 and Int.CC.RP.33.02 meaningless.

Test Results

Test Case	Expected Result	Test Result
Int.CC.RP.19.01	Success	Success
Int.CC.RP.20.01	Failure	Failure
Int.CC.RP.21.01	Failure	OUT OF SCOPE
Int.CC.RP.22.01	Failure	Failure
Int.CC.RP.23.01	Success	Success
Int.CC.RP.23.02	Failure	Failure
Int.CC.RP.24.01	Success	Success
Int.CC.RP.24.02	Failure	Failure
Int.CC.RP.25.01	Success	Success
Int.CC.RP.25.02	Failure	Failure
Int.CC.RP.26.01	Failure	Failure
Int.CC.RP.27.01	Failure	Failure
Int.CC.RP.28.01	**Failure**	**Success**
Int.CC.RP.29.01	**Success**	**Failure (with pathLengthConstraint = 0)** **Success (with pathLengthConstraint = 1)**
Int.CC.RP.29.02	Failure	Failure
Int.CC.RP.30.01	**Failure**	**Success**
Int.CC.RP.31.01	Failure	Failure
Int.CC.RP.32.01	**Failure**	**Success**
Int.CC.RP.33.01	**Success**	**Failure (with requireExplicitPolicy = 1)** **Success (with**

		requireExplicitPolicy = 3)
Int.CC.RP.33.02	Failure	Failure
Int.CC.RP.34.01	Success	Success
Int.CC.RP.34.02	**Failure**	**Success**
Int.CC.RP.35.01	Success	Success
Int.CC.RP.35.02	Failure	Failure
Int.CC.RP.36.01	Success	Success
Int.CC.RP.36.02	Failure	Failure
Int.CC.RP.37.01	Failure	Failure
Int.CC.RP.38.01	Failure	Failure
Int.CC.RP.39.01	Failure	Failure

Test case Int.CC.RP.28.01 is designed to check how the RP behaves if the Basic Constraints extension is marked as critical in the cross certificate. Test case Int.CC.RP.32.01 is designed to how the RP behaves if the Key Usage extension is marked critical in the cross certificate. It seems that the test case designer assumed that the RP should reject the cross certificate if these extensions appear in the cross certificate but are marked non-critical. It is true that the X.509 standard**,** PKIX profile (RFC 3280) or IWG profile all stipulate that these extensions SHOULD be marked critical. However, the test results revealed that the Basic Path Validation algorithm presented in section 6.1 of RFC 3280 does not make any check on the criticality of any extension. That is why the test results of those two cases are 'Success'.

Test case Int.CC.RP.30.01 is designed to check how the RP behaves if the cross certificate has the Key Usage extension. It seems that the test case designer assumed that this extension must appear in the cross certificate, or it should be rejected by the RP. It is true that the X.509 standard, PKIX profile and IWG profile all stipulate that the Key Usage extension MUST appear in a cross certificate. However, the test results revealed that the preparation steps of the Basic Path Validation algorithm specified in section 6.1.4 of RFC 3280 will accept a certificate as a CA certificate even if the Key Usage is absent in that certificate as long as there is a Basic Constraints extension (or some out-of-band mechanism) specifying that it is a CA certificate. This is what step (k) and step (n) of section 6.1.4 of RFC 3280 say about verifying if the certificate is a CA certificate. Note that the step (n) of section 6.1.4 of RFC 3280 says "If a key usage extension is present, verify that the keyCertSign bit is set." That implies that if no key usage extension is present, no verification on any key usage bit is needed.

Test cases Int.CC.RP.29.01 and Int.CC.RP.29.02 are designed to check if the RP be able to correctly handling the Path Length Constraint specified in the Basic Constraints extension of the cross certificate. The original test case specified that the value of the pathLengthConstraint field for test case Int.CC.RP.29.01 to be 0 because the test case designer assumed that the local test environment will be a 2-level PKI (RootCA – End Entity). However, since later the participant countries (Japan, Korea and Chinese Taipei) all agreed to extended the local test environment to be a 3-level PKI (RootCA – SubCA – End Entity) to make the experiment more close to the real case, the value of the pathLengthConstraint field for test case Int.CC.RP.29.01 should be changed to 1, or the test result will be 'Failure'.

Test cases Int.CC.RP.33.01 and Int.CC.RP.33.02 are designed to check how the RP behaves if the requireExplicitPolicy field is present with different values in the Policy Constraints extension of the cross certificate. The original test case specified that the value of the requireExplicitPolicy field for test case Int.CC.RP.33.01 to be 1 because the test case designer assumed that the local test environment will be a 2-level PKI (RootCA – End Entity). (Note that even with the original assumption of a 2-level PKI as local test environment, the value of the requireExplicitPolicy field for test case Int.CC.RP.33.01 should be 2 rather than 1 if the test case designer want to let the expected result of that test case be 'Success'.) However, since later the participant countries (Japan, Korea and Chinese Taipei) all agreed to extended the local test environment to be a 3-level PKI (RootCA – SubCA – End Entity) to make the experiment more close to the real case, the value of the requireExplicitPolicy field for test case Int.CC.RP.33.01 should be changed to 3, or the test result will be 'Failure'. For test case Int.CC.RP.33.02, any value less than 3 (ie., 0, 1, or 2) will make the test result be the intended 'Failure'.

Test cases Int.CC.RP.34.01 and Int.CC.RP.34.02 are designed to check how the RP behaves if the inhibitPolicyMapping field is present with different values in the Policy Constraints extension of the cross certificate. The original test cases design specified that the certification path contains four certificates:

1) the self-signed certificate of the RP's trust anchor (CA X)
2) the cross certificate issued from the CA-X to CA Y (with the inhibitPolicyMapping constraint field)
3) the cross certificate issued from CA Y to CA Z (with Policy Mappings)
4) the EE certificate issued by CA Z

The test case designer specified that for test case Int.CC.RP.34.01 let the value of the inhibitPolicyMapping field be 1 will make the expected test result be 'Success' and for test case Int.CC.RP.34.02 let the value of the inhibitPolicyMapping field be 1 will make the expected test result be 'Failure'. These designs seem to be fine. However, please note that these designs only work if the cross certificate issued from CA Y to CA Z contains a Policy Mappings extension since the inhibitPolicyMapping constraint only become effect after the certificate it appears. In our experiment of these two test cases, the certification path from our trust anchor to JP's EE contains four certificates:

1) the self-signed certificate of CT's Root CA (the trust anchor)
2) the CA certificate issued from CT's Root CA to JP's Root CA (with the inhibitPolicyMapping constraint field)
3) the CA certificate issued from JP's Root CA to JP's Sub CA (no policy mappings)
4) the EE certificate issued from JP's Sub CA to JP's End Entity

Since JP's Root CA and JP's Sub CA belong to the same PKI domain, there is no Policy Mappings extension exists in the CA certificate issued from JP's Root CA to JP's Sub CA. Therefore, the results of the path validation for these two test cases are both always 'Success'. We conjecture that if we further extended the input of these two test cases to be the certification path reaching KR's EE as follows:

1) the self-signed certificate of CT's Root CA (the trust anchor)
2) the CA certificate issued from CT's Root CA to JP's Root CA (with the inhibitPolicyMapping constraint field)

3) the CA certificate issued from JP's Root CA to KR's Root CA (with a Policy Mappings extension)
4) the CA certificate issued from KR's Root CA to KR's Sub CA
5) the EE certificate issued from KR's Sub CA to KR's End Entity

Since there are further policy mappings appear in the certification path, the experiments on these two test cases will generate the expected results. However, until the moment this report is written, we have not test these two test cases with the extended certification path yet. Thus that is just our conjecture.

4.2.6 Recommendations

4.2.6.1 Criticality

There are some test cases that check the value of criticality flag in the extensions in certificate. However, the criticality is used to indicate a flag to process or ignore a particular extension in application. The certificate path processing module is generally expected to validate the critical extensions, and non-critical extensions if recognized by the module.

As stated in section 6.2 of RFC 3280, the path validation algorithm presented in section 6.1 of RFC 3280 only defines the minimum conditions for a path to be considered valid. The local policy of the application (the RP) MAY want to impose some application-specific conditions for determining whether a certificate is acceptable. The extra criticality checking on some certificate extensions might be the one that the application wants to impose because the test results revealed that the RFC 3280 path validation algorithm does not make any check on the criticality of any extension. For example, the local policy might regard the certificate with a critical Subject Key Identifier extension as invalid because the X.509 standard stipulate that the Subject Key Identifier extension MUST be always non-critical. To provide the ability to allow the application to impose some application-specific conditions, the implementer of the RFC 3280 compliant path validation module might need to additionally provide a "plug-in" mechanism (for example, hook functions or call-back functions) in their implementation to let the application specify some extra certificate checking.

4.2.6.2 Key Usage

The combination of Key Usage is very confusing and sometimes can be contradictory. Also there is still unclear about the meaning when no Key Usage extension is specified in the certificate, or when non-critical Key Usage is specified.

The recommendation suggests that the certificate path processing module is expected to validate the Key Usage extension in either critical or non-critical. In addition, when the Key Usage extension is not specified, this is treated as all bits set or simply ignored in the certificate path processing module. Please note the test results revealed that, with its minimum conditions, the RFC 3280 path validation algorithm will accept a certificate as a CA certificate even if the Key Usage is absent in that certificate as long as there is a Basic Constraints extension (or some out-of-band mechanism) specifying that it is a CA certificate. Thus if the application wants to enforce that every CA certificate must contain a Key Usage extension with the keyCertSign bit being set, extra application-specific certificate checking is needed.

4.2.6.3 DN Matching Rule

The DN Matching Rule in the PPTG assumed two requirements at the beginning, derived from the section 4.1.2.4 in RFC 3280. One is that the RP should check that the names are different when they differ by white space in values other than country name. The next one is that the RP should check that the names are different when they differ by capitalization in values other than country name.

Since the Path Processing Experiment profile employs the UTF8String for the directory name in Subject and Issuer Fields (with only alphanumerical characters within the range of PrintableString defined in ISO/IEC 8824-1:1998), the white space and capitalization are significant in the DN Matching Rule.

However, the section 4.1.2.4 of RFC 3280 also allows application developers to implement the comparison rules defined in the X.500 series of specifications. The characters themselves are compared without regard to encoding, using Case Ignore Match defined in the section 6.1.1 and String matching rules in the section 6.1 of ITU-T X.520 (1993 E).

The implementations are different in interpreting the section 4.1.2.4.

Considering the character range of PrintableString being used in the certificate and potential impacts of the strict String Matching Rule in UTF8String, the discussion suggests that:

1) X.500 series comparison rule is beneficial when the certificates with the directory name of PrintableString have been already deployed in real business and need to update them with the UTF8String with significant costs.
2) We may encounter situations where UTF8String and PrintableString are messed up in the certificate path, especially when the cross certificate relationship is established. We will need a solution at such a transition period.
3) There are still unclear whether there is any business case that uses the characters other than the range of PrintableString in international context (such as mixed CJK characters).
4) There are no complaints found on X.500 series comparison rule only if a certificate contains only the characters within the range of PrintableString.

Therefore, the current conclusion suggests within the character range of the PrintableSting, the certificate validation module should recognize that leading, tailing spaces, and capitalization are insignificant in the current situation where PrintableString and UTF8String may be mixed up.

4.2.6.4 Self-Signed Certificate Validation

The validation mechanism for the self-signed Certificate is not described in the RFC 3280, since the self-signed certificate is an input to the validation of the certificate path and some out-of-band mechanism is assumed. Note that the self-signed certificate is the certificate that serves as a trust anchor.

Even though there appears the fact that RFC 3280 treats this validation mechanism is out of band, there are still issues to be considered.

1) Is the self-signed certificate checked in each time the certificate path validation is processed?
2) When is the self-signed certificate checked?
3) What kinds of checks are necessary at least?

For the issue 1), this depends on local policy. Since the RP has already accepted the self-signed certificate as trust anchor, some policy ignores the check each time. The other policy doesn' t.

Regardless of local policy configuration, at least there are some checks that should be performed when the RP accepts the certificate as trust anchor. The checks would include the issuer DN and subject DN matching, validity date checking, signature verification, and the application prompts users to accept the result.

4.2.6.5 Matching Rule of full names between CRLDP and IDP extensions

Please refer to the section 4.1.4.2.

4.3 PKCS#11 Application Interoperability Experiment

4.3.1 Experiment Overview & Scope

The PKCS#11 Application Interoperability Experiment explores issues when implementing downloadable web-based applications which interface with cryptographic functions of different vendors in different environments, making PKI applications to be portable and interoperable beyond the traditional national boundaries.

PKCS#11 is selected because of its mature-ness and popularity.

The experiment follows the following steps:

1)Creating the IWG Signature & Verification Profile and IWG Experiment Profile
2)Developing the web-based applications including the wrapper functions that integrate the application and PKCS#11 library
3)Establishing test bed environments and conducting the experiment
4)Evaluating the test results and providing the recommendations

In this experiment, each member downloads the web-applications and tests the other applications with its own PKCS#11 library for function calls related to signature generation and verification.

4.3.2 PKCS#11 IWG Conformance Profile

4.3.2.1 Goals and Concepts

IWG adopted the PKCS#11 to achieve PKI Application Interoperability in ASIA. PKCS#11 IWG Conformance Profile was developed to ensure the conformance of all PKCS#11 Libraries to be developed individually. This profile specifies the application interfaces that support sign and verify functions.

IWG assumed that the certificates and private keys in a secure token should be under the control of each nation and the code libraries as well, when interfacing with other PKI domains. Considering these conditions, IWG adopted the PKCS#11 Token interfaces that have the global recognition and can serve as a secure storage.

It became necessary to develop the PKCS#11 Conformance Profile with which the individual developers would comply at the development stages, to be interoperable with each other, regardless of whether the tokens are for S/W or H/W.

As the first step to PKI Application Interoperability, IWG limited the APIs to the functions only for sign and verify based on PKCS#11.

In order to achieve the real meaning of the PKI Application Interoperability through PKCS#11, IWG still need to consider the functions of certificate path validations, which are not covered in PKCS #11, and such security issues as the integrity of downloaded or pre-installed code libraries as well. IWG will manage them in the next experiments.

4.3.2.2 Characteristics of Profile

Referring to the RSA PKCS#11 Conformance profile, this profile has two sub-profiles; one is named the 'IWG Signing & Verification Profile' which is just-fit-profile for signing and verification functions and the other is named the 'IWG Experiment Profile' which covers the minimum requirement, only for the 2002 IWG experiment.

This profile is based on PKCS#11 v2.11, but none of advanced functionalities of this version are required compulsorily. So, all versions above 2.01 are compatible for this profile. IWG Signing & Verification Profile covers 16 RSA Base APIs and 8 additional APIs for Singing and Verification. But IWG Experiment Profile covers 18 APIs and no Base APIs. This profile does not specify the statement regarding templates. This profile supports only mechanism CKM_RSA_PKCS on which every documents to be signed must be hashed outside the token. After applications hashes the documents, then token should sign.

The following 18 PKCS#11 functions and their return codes were decided upon in this phase of PKCS#11 interoperability testing.

Function Name	Return
C_GetFunctionList	CKR_OK
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
C_Initialize	CKR_OK
	CKR_ARGUMENTS_BAD
	CKR_CANT_LOCK
	CKR_CRYPTOKI_ALREADY_INITIALIZED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_NEED_TO_CREATE_THREADS
C_Finalize	CKR_OK
	CKR_ARGUMENTS_BAD
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY

Function Name	Return
C_GetInfo	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
C_GetSlotList	CKR_OK
	CKR_BUFFER_TOO_SMALL
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR CKR_HOST_MEMORY
C_OpenSession	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_SESSION_COUNT
	CKR_SESSION_PARALLEL_NOT_SUPPORTED
	CKR_SESSION_READ_WRITE_SO_EXISTS
	CKR_SLOT_ID_INVALID
	CKR_TOKEN_NOT_PRESENT
	CKR_TOKEN_NOT_RECOGNIZED CKR_TOKEN_WRITE_PROTECTED
C_CloseSession	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
C_Login	CKR_OK CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR

Function Name	Return
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_PIN_INCORRECT
	CKR_SESSION_CLOSED CKR_SESSION_HANDLE_INVALID
	CKR_SESSION_READ_ONLY_EXISTS
	CKR_USER_ALREADY_LOGGED_IN
	CKR_USER_ANOTHER_ALREADY_LOGGED_IN
	CKR_USER_PIN_NOT_INITIALIZED
	CKR_USER_TYPE_INVALID
C_Logout	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
	CKR_USER_NOT_LOGGED_IN
C_FindObjectsInit	CKR_OK
	CKR_ATTRIBUTE_TYPE_INVALID
	CKR_ATTRIBUTE_VALUE_INVALID
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_OPERATION_ACTIVE
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
	CKR_TEMPLATE_INCONSISTENT
C_FindObjects	CKR_OK

Function Name	Return
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_OPERATION_NOT_INITIALIZED
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
C_FindObjectsFinal	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_OPERATION_NOT_INITIALIZED
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
C_CreateObject	CKR_OK
	CKR_ATTRIBUTE_READ_ONLY
	CKR_ATTRIBUTE_TYPE_INVALID
	CKR_ATTRIBUTE_VALUE_INVALID
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
	CKR_SESSION_READ_ONLY
	CKR_TEMPLATE_INCOMPLETE
	CKR_TEMPLATE_INCONSISTENT
	CKR_TOKEN_WRITE_PROTECTED
	CKR_USER_NOT_LOGGED_IN

Function Name	Return
C_DestroyObject	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_OBJECT_HANDLE_INVALID
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
	CKR_SESSION_READ_ONLY
	CKR_TOKEN_WRITE_PROTECTED
C_GetAttributeValue	CKR_OK
	CKR_ATTRIBUTE_SENSITIVE
	CKR_ATTRIBUTE_TYPE_INVALID
	CKR_BUFFER_TOO_SMALL
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_OBJECT_HANDLE_INVALID
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
	CKR_TEMPLATE_INCONSISTENT
C_SignInit	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_CANCELED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_KEY_FUNCTION_NOT_PERMITTED
	CKR_KEY_HANDLE_INVALID
	CKR_KEY_SIZE_RANGE

Function Name	Return
	CKR_KEY_TYPE_INCONSISTENT
	CKR_MECHANISM_INVALID
	CKR_MECHANISM_PARAM_INVALID
	CKR_OPERATION_ACTIVE
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
	CKR_USER_NOT_LOGGED_IN
C_Sign	CKR_OK
	CKR_BUFFER_TOO_SMALL
	CKR_CRYPTOKI_NOT_INITIALIZED CKR_DATA_INVALID
	CKR_DATA_LEN_RANGE
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_CANCELED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_OPERATION_NOT_INITIALIZED
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
C_VerifyInit	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED
	CKR_FUNCTION_CANCELED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_KEY_FUNCTION_NOT_PERMITTED
	CKR_KEY_HANDLE_INVALID
	CKR_KEY_SIZE_RANGE
	CKR_KEY_TYPE_INCONSISTENT
	CKR_MECHANISM_INVALID
	CKR_MECHANISM_PARAM_INVALID
	CKR_OPERATION_ACTIVE

Function Name	Return
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
	CKR_USER_NOT_LOGGED_IN
C_Verify	CKR_OK
	CKR_CRYPTOKI_NOT_INITIALIZED
	CKR_DATA_INVALID
	CKR_DATA_LEN_RANGE
	CKR_DEVICE_ERROR
	CKR_DEVICE_MEMORY
	CKR_DEVICE_REMOVED CKR_FUNCTION_CANCELED
	CKR_FUNCTION_FAILED
	CKR_GENERAL_ERROR
	CKR_HOST_MEMORY
	CKR_OPERATION_NOT_INITIALIZED
	CKR_SESSION_CLOSED
	CKR_SESSION_HANDLE_INVALID
	CKR_SIGNATURE_INVALID
	CKR_SIGNATURE_LEN_RANGE

IWG also confirmed the sequence of PKCS#11 function calls to be made by the PKCS#11 test application while performing different token functions. Listed below is the sequence of the function calls to perform various token functions in the PKCS#11 test application.

Sequence #1 (login)
C_GetFunctionList, C_Initialize, C_GetSlotList, C_OpenSession, C_Login

Sequence #2 (logout)
C_Logout, C_CloseSession, C_Finalize

Sequence #3 (find_object)
C_FindObjectsInit, C_FindObjects, C_FindObjectsFinal

Sequence #4 (sign)
C_SignInit, C_Sign

Sequence #5 (verify)
C_VerifyInit, C_Verify

Sequence #6 (get_attr_value)
C_GetAttributeValue

Sequence #7 (create_object)
C_CreateObject

Sequence #8 (destroy object)
C_DestroyObject

Since the PKCS#11 test application is independent of the actual PKCS#11 token library, it was decided that a common name resolution INI file would be used to store the name of the actual PKCS#11 token library. The name of the INI file decided upon is pkcs11.ini.

The format of setting in the pkcs11.ini file is as follows

[PKCS11.Driver.Name] - the section name
Dll Name - the token driver name

4.3.3 PKCS#11 Test Participants

The following companies from the four IWG member countries Japan, Korea, Chinese Taipei and Singapore participated in the PKCS#11 Application Interoperability test.

Japan PKI Forum
Japan

Hitachi, Ltd
Japan

Fujitsu
Japan

Korea Information Security Agency
Seoul
Korea

Korea Information Certificate Authority
Seoul,
Korea

Taiwan-CA.COM., INC
Taipei,
Chinese Taipei

National Information Infrastructure Enterprise Promotion Association
Taipei,
Chinese Taipei

Infocomm Development Authority of Singapore
Singapore

CrimsonLogic Pte. Ltd.
Singapore

4.3.4 PKCS#11 Test Environments

This section describes the Test Environments used by the IWG members to perform the PKCS#11 Interoperability Tests. Each IWG member country developed a server side PKCS#11 test application and the PKCS#11 Wrapper and the client side PKCS#11 Library**.** The client side PKCS#11 library provided an interface to the actual PKCS#11 token driver.

In the experiment, each participating party prepared its own PKCS#11 library and web-based application. After the full set-up of the environment, each participating party verified the application interface with each other using its respective applications and PKCS#11 library.

The PKCS#11 Wrapper was tested locally using the test application by the each IWG member country. The PKCS#11 Library of each country was tested locally. Each IWG member country setup a web server environment and deployed its PKCS#11 test application in the server. This test application and the PKCS#11 library was then used by other IWG member countries to test for PKCS#11 Interoperability with the actual PKCS#11 token driver.

The server side PKCS#11 test application was downloaded to the client while performing the PKCS#11 Interoperability test.

PKCS#11 Test Application Interface model.

PKCS#11 Interface model describes the interface between the PKCS#11 test application, PKCS#11 wrapper and the PKCS#11 library. Figure 1 shows the two interface models used by the IWG members for the PKCS#11 Interoperability test.

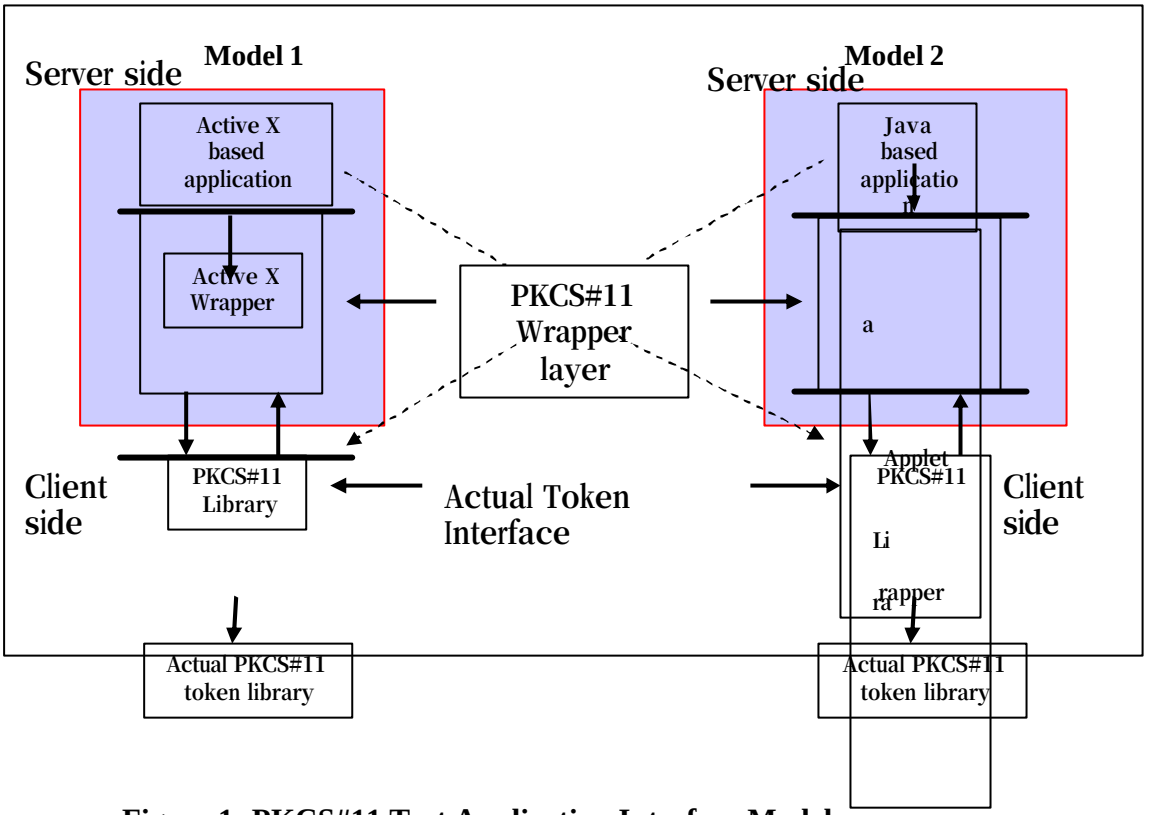


Figure 1: PKCS#11 Test Application Interface Model

In Figure 1 above, Model 1 approach was to develop an PKCS#11 test application, which interacts with a PKCS#11 Wrapper developed as an Active X control. The PKCS#11 wrapper in turn interacted with the PKCS#11 library. The Model 2 approach was to build a Java based application, which interacted with the PKCS#11 wrapper. The PKCS#11 wrapper included a Java API, which calls native functions. The native functions were implemented as a Windows JNI dynamic link library written in C. The PKCS#11 wrapper interacted with the PKCS#11 library.

IWG members Korea and Chinese Taipei followed the Model 1 approach whereas IWG members Japan and Singapore followed the Model 2 approach.

As an example, Figure 2 below describes the interface layer model of the PKCS#11 test application developed by Singapore. The topmost layer is the application layer. The next layer in the Java based PKCS#11 Wrapper which interacts with the PKCS#11 library. The lowest layer is the actual PKCS#11 token library. The arrows indicate the dependencies between the different modules.

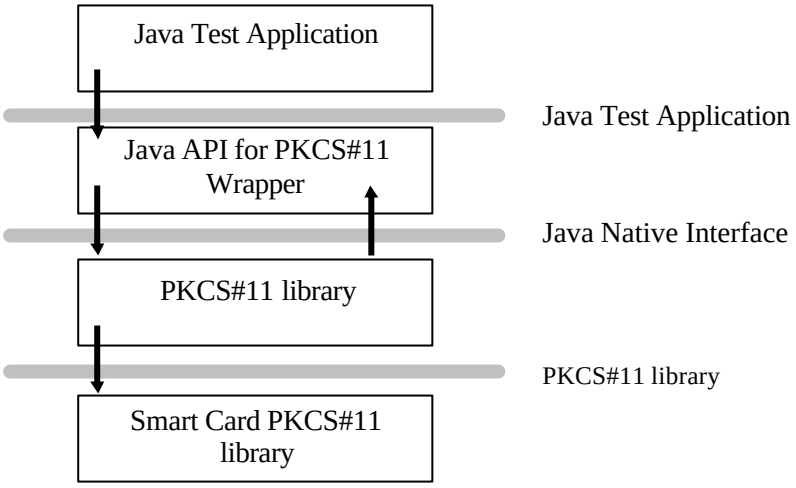


Figure 2: Layer Model of the Singapore PKCS#11 Library

PKCS#11 Interoperability Test Environments for the PKCS#11 Wrapper for a particular IWG member describes the server side test environment for that IWG member and the client side test environment used by every other IWG member to perform the interoperability test.

4.3.4.1 Japan PKCS#11 Test Environments

This section describes the Local test environments and the Application Interoperability test Environment used to test the PKCS#11 Wrapper developed by IWG member in Japan.

4.3.4.1.1 Local Test Environment

The table below lists the local test environment details for the test conducted in Japan

Local System Configuration	Details
OS	Microsoft Windows 2000 Professional
Token Used	Smart Card
JRE	Java2 Runtime Environment 1.4.1 (02) is installed.
PKCS#11 library	F3EZscl2.dll The driver for the smart card is pre-installed in the system folder of Windows and tested locally.
Configuration file	pkcs11.ini [PKCS11.Driver.Name] - the section name F3EZscl2.dll - the token driver This file is pre-installed in the Windows system folder to perform the PKCS#11 testing.
PKCS#11 Wrapper	PKCS11WrapperJNI.dl. This file is copied to the Windows system folder to perform the PKCS#11 testing. PKCS11Wrapper.jar. This file is downloaded.
Browser Supported	Internet Explorer 6.0

4.3.4.1.2 Application Interoperability Test Environment

Server Side Environment in Japan for Interoperability test

The table below lists the server side environment details in Japan for the PKCS#11 Application Interoperability test.

System Configuration	Details

OS	Microsoft Windows NT 4.0 Server
Web Server	Microsoft Internet Information Server 4.0
Website	http://ap.pki-j-sim.jp/pkcs11/test2.html Now, this site cannot be accessed.
Browser Supported	Internet Explorer 6.0

Client side Test Environment in Korea for testing Japan's PKCS#11 Wrapper
The table below lists the client side environment details in Korea to test Japan's PKCS#11 Application

Local System Configuration	**Details**
OS	Windows 2000 Professional
Token Used	S/W Token
JRE	Java2 Runtime Environment 1.4.1 (02) is installed
PKCS#11 library	The driver for the smart card is pre-installed in the system folder of Windows and tested locally.
Configuration file	pkcs11.ini [PKCS11.Driver.Name] - the section name sgpkcs.dll - the token driver This file is pre-installed in the Windows system folder to perform the PKCS#11 testing.
PKCS#11 Wrapper	PKCS11Wrapper.dll. This file is copied to the Windows system folder to perform the PKCS#11 testing.
Browser Supported	Internet Explorer 5.5 and above.

Client side Test Environment in Chinese Taipei for testing Japan's PKCS#11 Wrapper
The table below lists the client side environment details in Chinese Taipei to test Japan's PKCS#11 Application

Local System Configuration	**Details**
OS	Windows 2000 Server
Token Used	GemPKCS
JRE	Java2 Runtime Environment 1.4.1 (02) is installed
PKCS#11 library	gclib.dll
Configuration file	pkcs11.ini AsiaPKIEnv.dat
PKCS#11 Wrapper	PKCS11WrapperJNI.dll
Browser Supported	IE 5.5

Client side Environment in Singapore for testing Japan's PKCS#11 Wrapper

The table below lists the client side environment details in Singapore to test Japan's PKCS#11 Application

Local System Configuration	Details
OS	Window 2000 Professional
Hardware Token Used	Datakey Model 330 with DKR 730 Reader
JRE	JRE 1.3.1
PKCS#11 library	dkck201.dll (Provided with datakey card)
Configuration file	pkcs11.ini
PKCS#11 Wrapper	PKCS11WrapperJNI.dll
Browser Supported	IE 5.5 above

4.3.4.2 Korea PKCS#11 Test Environments

This section describes the Local test environments and the Application Interoperability test Environment used to test the PKCS#11 Wrapper developed by IWG member in Korea.

4.3.4.2.1 Local Test Environment

The table below lists the local test environment details for the test conducted in Korea.

Local System Configuration	Details
OS	Windows 2000 Professional
Token Used	S/W Token
JRE	N/A
PKCS#11 library	Sgpkcs.dll The driver for the smart card is pre-installed in the system folder of Windows and tested locally.
Configuration file	Pkcs11.ini, sgpkcs.ini
PKCS#11 Wrapper	AxSgpkcs.dll
Browser Supported	IE5.5 and above

4.3.4.2.2 Application Interoperability Test Environment

Server Side Environment in Korea for Interoperability test

The table below lists the server side environment details in Korea for the PKCS#11 Application Interoperability test.

System Configuration	Details
OS	WinNT Server
Web Server	IIS
Website	http://apptest.rootca.or.kr/
Browser Supported	IE 5.5 and above

Client side Test Environment in Japan for testing Korea's PKCS#11 Wrapper

The table below lists the client side environment details in Japan to test Korea's PKCS#11 Application

Local System Configuration	Details
OS	Microsoft Windows 2000 Professional
Token Used	Smart Card

JRE	Java2 Runtime Environment 1.4.1 (02) is installed.
PKCS#11 library	F3EZscl2.dll The driver for the smart card is pre-installed in the system folder of Windows and tested locally.
Configuration file	pkcs11.ini [PKCS11.Driver.Name] - the section name F3EZscl2.dll - the token driver This file is pre-installed in the Windows system folder to perform the PKCS#11 testing.
PKCS#11 Wrapper	PKCS11WrapperJNI.dl. This file is copied to the Windows system folder to perform the PKCS#11 testing. PKCS11Wrapper.jar. This file is downloaded.
Browser Supported	Internet Explorer 6.0

Client side Test Environment in Chinese Taipei for testing Korea's PKCS#11 Wrapper

The table below lists the client side environment details in Chinese Taipei to test Korea's PKCS#11 Application.

Local System Configuration	**Details**
OS	Windows 2000 Server
Token Used	GemPKCS
JRE	N/A
PKCS#11 library	gclib.dll
Configuration file	pkcs11.ini
PKCS#11 Wrapper	AxSgpkcs.dll
Browser Supported	IE 5.5

Client side Environment in Singapore for testing Korea PKCS#11 Wrapper

The table below lists the client side environment details in Singapore to test Korea's PKCS#11 Application.

Local System Configuration	**Details**

OS	Windows 2000 Professional
Hardware Token Used	Datakey Model 330 with DKR 730 Reader
JRE	N/A
PKCS#11 library	dkck201.dll
Configuration file	pkcs11.ini
PKCS#11 Wrapper	AxSgpkcs.dll
Browser Supported	IE 6.0

4.3.4.3 Chinese Taipei PKCS#11 Test Environments

This section describes the Local test environments and the Application Interoperability test Environment used to test the PKCS#11 Wrapper developed by IWG member in Chinese Taipei.

4.3.4.3.1 Local Test Environment

The table below lists the local test environment details for the test conducted in Chinese Taipei.

Local System Configuration	Details
OS	Windows 2000 Server
Token Used	GemPKCS
JRE	N/A
PKCS#11 library	gclib.dll
Configuration file	pkcs11.ini
PKCS#11 Wrapper	JKST0305.cab#Version=1,0,0,4
Browser Supported	IE 5.5

4.3.4.3.2 Application Interoperability Test Environment

Server Side Environment in Chinese Taipei for Interoperability test

The table below lists the server side environment details in Chinese Taipei for the PKCS#11 Application Interoperability test.

System Configuration	Details
OS	Windows 2000 Server
Web Server	IIS 5.0
Website	http://210.66.126.50/JKST/wrapper.html
Browser Supported	IE 5.5

Client side Test Environment in Japan for testing Chinese Taipei' s PKCS#11 Wrapper

The table below lists the client side environment details in Japan to test Chinese Taipei' s PKCS#11 Application.

Local System Configuration	Details
OS	Microsoft Windows 2000 Professional
Token Used	Smart Card
JRE	Java2 Runtime Environment 1.4.1 (02) is installed.

PKCS#11 library	F3EZscl2.dll The driver for the smart card is pre-installed in the system folder of Windows and tested locally.
Configuration file	pkcs11.ini [PKCS11.Driver.Name] - the section name F3EZscl2.dll - the token driver This file is pre-installed in the Windows system folder to perform the PKCS#11 testing.
PKCS#11 Wrapper	JKST0305.cab#Version=1,0,0,4
Browser Supported	Internet Explorer 6.0

Client side Test Environment in Korea for testing Chinese Taipei's PKCS#11 Wrapper

The table below lists the client side environment details in Korea to test Chinese Taipei's PKCS#11 Application.

Local System Configuration	**Details**
OS	Windows 2000 Professional
Token Used	S/W Token
JRE	N/A
PKCS#11 library	The driver for the smart card is pre-installed in the system folder of Windows and tested locally.
Configuration file	pkcs11.ini [PKCS11.Driver.Name] - the section name sgpkcs.dll - the token driver This file is pre-installed in the Windows system folder to perform the PKCS#11 testing.
PKCS#11 Wrapper	JKST0305.cab#Version=1,0,0,4
Browser Supported	Internet Explorer 5.5 and above.

Client side Environment in Singapore for testing Chinese Taipei PKCS#11 Wrapper

The table below lists the client side environment details in Singapore to test Chinese Taipei's PKCS#11 Application.

Local System Configuration	**Details**
OS	Windows 2000 Professional

Hardware Token Used	Datakey Model 330 with DKR 730 Reader
JRE	N/A
PKCS#11 library Configuration file	dkck201.dll pkcs11.ini
PKCS#11 Wrapper	JKST0305.cab#Version=1,0,0,4
Browser Supported	IE 6.0

4.3.4.4 Singapore PKCS#11 Test Environments

This section describes the Local test environments and the Application Interoperability test Environment used to test the PKCS#11 Wrapper developed by IWG member in Singapore.

4.3.4.4.1 Local Test Environment

The table below lists the local test environment details for the test conducted in Singapore.

Local System Configuration	**Details**	
OS	*Windows System*	*System Folder*
	Windows 98 SE	C:\windows\system
	Windows 2000	C:\winnt\system32
	Windows XP	C:\windows\system32
Token Used	1. Datakey Model 330 with Datakey DKR 730 card reader; 2. Rainbow ikey 2000	
JRE	Java2 Runtime Environment 1.4.1 (02) is installed.	
PKCS#11 library	The driver for the smart card is installed and the DLL file of the main body of the PKCS#11 library is stored in the system folder of Windows.	
Configuration file	pkcs11.ini The following settings are stored in the configuration file pkcs11.ini [PKCS11.Driver.Name] - the section name dkck201.dll - the token driver This file is copied to the Windows system folder to perform the PKCS#11 testing.	
PKCS#11 Wrapper	PKCS11WrapperJNI.dll. This file is copied to the Windows system folder to perform the PKCS#11 testing.	
Browser Supported	Internet Explorer 5.5 and above. Local testing was done using Internet Explorer 5.5	

4.3.4.4.2 Application Interoperability Test Environment

Server Side Environment in Singapore for Interoperability test

The table below lists the server side environment details in Singapore for the PKCS#11 Application Interoperability test.

System Configuration	Details
OS	Microsoft 2000 Server
Web Server	IIS
Website	http://203.126.248.102/iwg/pkcs11test_2.html
Browser Supported	Internet Explorer 5.5 and above. Local testing was done using Internet Explorer 5.5

Client side Test Environment in Japan for testing Singapore PKCS#11 Wrapper

The table below lists the client side environment details in Japan to test Singapore's PKCS#11 Application.

Local System Configuration	**Details**
OS	Microsoft Windows 2000 Professional
Token Used	Smart Card
JRE	Java2 Runtime Environment 1.4.1 (02) is installed.
PKCS#11 library	F3EZscl2.dll The driver for the smart card is pre-installed in the system folder of Windows and tested locally.
Configuration file	pkcs11.ini [PKCS11.Driver.Name] - the section name F3EZscl2.dll - the token driver This file is pre-installed in the Windows system folder to perform the PKCS#11 testing.
PKCS#11 Wrapper	PKCS11WrapperJNI.dl. This file is copied to the Windows system folder to perform the PKCS#11 testing. PKCS11Wrapper.jar. This file is downloaded.
Browser Supported	Internet Explorer 6.0

Client side Test Environment in Korea for testing Singapore PKCS#11 Wrapper

The table below lists the client side environment details in Korea to test Singapore's PKCS#11 Application.

Local System Configuration	Details
OS	Windows 2000 Professional
Token Used	S/W Token
JRE	Java2 Runtime Environment 1.4.1 (02) is installed
PKCS#11 library	The driver for the smart card is pre-installed in the system folder of Windows and tested locally.
Configuration file	pkcs11.ini [PKCS11.Driver.Name] - the section name sgpkcs.dll - the token driver This file is pre-installed in the Windows system folder to perform the PKCS#11 testing.
PKCS#11 Wrapper	PKCS11WrapperJNI.dll. This file is copied to the Windows system folder to perform the PKCS#11 testing.
Browser Supported	Internet Explorer 5.5 and above.

Client side Environment for Chinese Taipei for testing Singapore PKCS#11 Wrapper

The table below lists the client side environment details in Chinese Taipei to test Singapore's PKCS#11 Application.

Local System Configuration	Details
OS	Windows 2000 Server
Hardware Token Used	GemPKCS
JRE	Java2 Runtime Environment 1.4.1 (02) is installed
PKCS#11 library	gclib.dll
Configuration file	pkcs11.ini
PKCS#11 Wrapper	PKCS11WrapperJNI.dll
Browser Supported	IE 5.5

4.3.5 Test Plan/Scenarios

A minimal set of PKCS#11 function list was proposed in order to conduct the PKCS#11 Application Interoperability test. The following functions were included in the set to be tested.

PKCS#11 Function list for PKCS#11 Application Interoperability test

C_GetFunctionList
C_Initialize
C_Finalize
C_GetSlotList
C_OpenSession
C_CloseSession
C_FindObjectsInit
C_FindObjects
C_FindObjectsFinal
C_CreateObject
C_DestroyObject
C_Login
C_Logout
C_GetAttributeValue
C_SignInit
C_Sign
C_VerifyInit
C_Verify

The test scenarios used for PKCS#11 Interoperability test covered all the functions listed above.

The test scenarios make use of the PKCS#11 functions listed above. The following tables lists down the functions used by each test scenario.

Scenario	Sequence	Function
iwg_login	Login	C_GetFunctionList
		C_Initialize
		C_GetSlotList
		C_OpenSession
		C_Login
	Logout	C_Logout
		C_CloseSession
		C_Finalize

Scenario	Sequence	Function
iwg_sign	Login	C_GetFunctionList
		C_Initialize
		C_GetSlotList
		C_OpenSession
		C_Login

Scenario	Sequence	Function
	Find_ object	C_FindObjectInit
		C_FindObjects
		C_FindObjectFinal
	Sign	C_SignInit
		C_Sign
	Logout	C_Logout
		C_CloseSession
		C_Finalize

Scenario	Sequence	Function
iwg_getcert	Login	C_GetFunctionList
		C_Initialize
		C_GetSlotList
		C_OpenSession
		C_Login
	Find_ object	C_FindObjectInit
		C_FindObjects
		C_FindObjectFinal
	Get_attr_value	C_GetAttributeValue
	Logout	C_Logout
		C_CloseSession
		C_Finalize

Scenario	Sequence	Function
iwg_putcert	Login	C_GetFunctionList
		C_Initialize
		C_GetSlotList
		C_OpenSession
		C_Login
	Create_ object	C_CreateObject
	Verify	C_VerifyInit
		C_Verify
	Destroy_object	C_DestroyObject
	Logout	C_Logout
		C_CloseSession
		C_Finalize

Scenario	Sequence	Function
iwg_verify	Login	C_GetFunctionList
		C_Initialize
		C_GetSlotList
		C_OpenSession
		C_Login
	Create_ object	C_CreateObject
	Verify	C_VerifyInit

Scenario	Sequence	Function
		C_Verify
	Logout	C_Logout
		C_CloseSession
		C_Finalize

Abnormal Case

Scenario	Sequence	Function
iwg_login	Login	C_GetFunctionList
		C_Initialize
		C_GetSlotList
		C_OpenSession
		C_Login
	Logout	C_Logout
		C_CloseSession
		C_Finalize

Scenario	Sequence	Function
iwg_verify	Login	C_GetFunctionList
		C_Initialize
		C_GetSlotList
		C_OpenSession
		C_Login
	Create_ object	C_CreateObject
	Verify	C_VerifyInit
		C_Verify
	Logout	C_Logout
		C_CloseSession
		C_Finalize

Each IWG member prepared their own test scenarios for testing the PKCS#11 library with the test applications developed by all the IWG members.

The test scenarios for IWG members from Japan, Korea, Chinese Taipei and Singapore are listed below.

- **Test case scenario for Japan**

Normal Case:

No	Scenarios	Details	Expected Return
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_sign	Generate signature	CKR_OK

3	iwg_getcert	Get client' s certificate	CKR_OK
4	iwg_verify	Verify signature	CKR_OK

Abnormal Case:

No	Scenarios	Details	Expected Return
5	iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
6	iwg_verify	Verify signature with wrong data	CKR_SIGNATURE_INVALID

- **Test case scenarios for Korea**

Normal Case:

No	Scenarios	Details	Expected Return
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_getcert	Get client' s certificate	CKR_OK
3	iwg_putpubkey	Put server' s public key object	CKR_OK
4	iwg_sign	Generate signature	CKR_OK
5	iwg_verify	Verify signature	CKR_OK

Abnormal Case:

No	Scenarios	Details	Expected Return
6	iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
7	iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
8	iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

- **Test case scenarios for Chinese Taipei**

Normal Case:

No	Scenarios	Details	Expected Return
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_getcert	Get client' s certificate	CKR_OK
3	iwg_putpubkey	Put server' s public key object	CKR_OK
4	iwg_sign	Generate signature	CKR_OK
5	iwg_verify	Verify signature	CKR_OK

Abnormal Case:

No	Scenarios	Details	Expected Return
6	iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
7	iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
8	iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

- **Test case scenarios for Singapore**

Normal Case:

No	Scenarios	Details	Expected Return
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_sign	Generate signature	CKR_OK
3	iwg_getcert	Get client' s certificate	CKR_OK
4	iwg_verify	Verify signature	CKR_OK

Abnormal Case:

No	Scenarios	Details	Expected Return
5	iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
6	iwg_verify	Verify signature with wrong data	CKR_SIGNATURE_INVALID

4.3.6 PKCS#11 Test Experiment Results

Each IWG member country conducted the PKCS#11 interoperability test to test their PKCS#11 token library using PKCS#11 application and PKCS11 Wrapper developed by other IWG member countries.

Listed below are the interoperability test experiment results for each IWG member country using applications in the other IWG member country server.

4.3.6.1 Consolidated Interoperability Test Results for Japan PKCS#11 library

Using Korea Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_sign	Generate signature	CKR_OK
3	iwg_verify	Verify signature	CKR_OK
4	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
5	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
6	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

Using Chinese Taipei Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_sign	Generate signature	CKR_OK
3	iwg_verify	Verify signature	CKR_OK
4	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
5	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
6	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

Using Singapore Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_sign	Generate signature	CKR_OK
3	iwg_verify	Verify signature	CKR_OK
4	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
5	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
6	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

4.3.6.2 Consolidated Interoperability Test Results for Korea PKCS#11 library

Using Japan Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_getcert	Get client' s certificate	CKR_OK
3	iwg_sign	Generate signature	CKR_OK
4	iwg_verify	Verify signature	CKR_OK
5	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
6	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

Using Chinese Taipei Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_getcert	Get client' s certificate	CKR_OK
3	iwg_putcert	Put server' s public key object	CKR_OK
4	iwg_sign	Generate signature	CKR_OK
5	iwg_verify	Verify signature	CKR_OK
6	ab_iwg_login	Login with	CKR_PIN_INCORRECT

		incorrect PIN	
7	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
8	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

Using Singapore Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_getcert	Get client's certificate	CKR_OK
3	iwg_sign	Generate signature	CKR_OK
4	iwg_verify	Verify signature	CKR_OK
5	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
6	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

4.3.6.3 Consolidated Interoperability Test Results for Chinese Taipei PKCS#11 library

Using Japan Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_getcert	Get client's certificate	CKR_OK
3	iwg_putcert	Put server's public key object	CKR_OK
4	iwg_sign	Generate signature	CKR_OK
5	iwg_verify	Verify signature	CKR_OK
6	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
7	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
8	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

Using Korea Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_getcert	Get client's certificate	CKR_OK
3	iwg_putcert	Put server's public key object	CKR_OK
4	iwg_sign	Generate signature	CKR_OK
5	iwg_verify	Verify signature	CKR_OK
6	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
7	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
8	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

Using Singapore Server

Scenario	Functions	Details	Expected Return Code
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_getcert	Get client's certificate	CKR_OK
3	iwg_putcert	Put server's public key object	CKR_OK
4	iwg_sign	Generate signature	CKR_OK
5	iwg_verify	Verify signature	CKR_OK
6	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
7	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
8	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

4.3.6.4 Consolidated Interoperability Test Results for Singapore PKCS#11 library

Using Japan Server

Scenario	Functions	Details	Expected Return Code

1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_sign	Generate signature	CKR_OK
3	iwg_verify	Verify signature	CKR_OK
4	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
5	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
6	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

Using Korea Server

Scenario	**Functions**	**Details**	**Expected Return Code**
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_sign	Generate signature	CKR_OK
3	iwg_verify	Verify signature	CKR_OK
4	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
5	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
6	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

Using Chinese Taipei Server

Scenario	**Functions**	**Details**	**Expected Return Code**
1	iwg_login	Login with correct PIN	CKR_OK
2	iwg_sign	Generate signature	CKR_OK
3	iwg_verify	Verify signature	CKR_OK
4	ab_iwg_login	Login with incorrect PIN	CKR_PIN_INCORRECT
5	ab_iwg_sign	Sign with over 128bytes long data	CKR_DATA_LEN_RANGE
6	ab_iwg_verify	Verify signature with wrong certificate	CKR_SIGNATURE_INVALID

4.3.7 Technical Issues

The PKCS#11 library supports limited functions necessary to perform digital signing and verification. It does not support functions of the Certificate Path Validation. In addition, functions for Encrypt /Decrypt and Wrap/Unwrap are not supported.
The integrity of the PKCS#11 library and the information file "pkcs11.ini" is not checked, as they are vulnerable to any changes done by malicious users.
The PKCS#11 library does not support Dual keys in the token and multiple token slots in the PC.

4.3.8 Recommendations

4.3.8.1 Mechanism 'CKM_RSA_PKCS'

IWG recommend that only the mechanism CKM_RSA_PKCS be supported in this experiment. It is critical when we need application interoperability between server and client. The length of the signed document is not predictable. And memory is not quite big on tokens. Furthermore, the big document does not need to be copied into the token due to which the signing and verification time can be reduced.

4.3.8.2 Integrity Issues (PKCS11.INI & PKCS11 Library)

The PKCS#11 library and the information file name the 'pkcs11.ini' are pre-installed, when interfacing with any application servers of other nations. The information file 'pkcs11.ini' was used to store the token driver library. In this experiment, the integrity of the above files is not checked, even though they are vulnerable to any changes by malicious users. Security measures are needed to guarantee their integrity, which would make large money transactions secure and reliable.

One of the suggestion included adding additional information in the PKCS11.ini file like the vendor information, security related information and any other relevant information.

Another issue raised was about code signing for the PKCS#11 downloadable Applets and ActiveX controls to make them trustworthy.

4.3.8.3 Dual key and Multiple slots support.

The PKCS#11 library needs to be updated to support Dual keys in the token and multiple token slots in the PC. Chinese Taipei indicated whether we should test for multiple slots for token and Dual keys in the token during this round of Interoperability testing. But its recommendation was deferred as a future plan for the PKCS#11 interoperability test.

4.3.8.4 Advanced Functions

In the real application, the functions of Certificate Path Validation are necessary to be coupled with verification of digital signature. However, the PKCS#11 Interfaces does not cover those functions. In order to make this profile practical and well organized, it's necessary to develop the Certificate Path Validation, Encrypt/Decrypt and Wrap/Unwrap as well.