The 27th Symposium on Information Theory
and Its Applications (SITA2004)
Gero, Gifu, Japan, Dec. 14-17, 2004

多機能ICカード向けPKI機能

The PKI Function for Multi-Application Smart Cards

梅澤 克之*† 洲崎 誠一† 手塚 悟† 平澤 茂一*
Katsuyuki Umezawa Seiichi Susaki Satoru Tezuka Shigeichi Hirasawa

Abstract— We propose a new PKI function for multi-application smart cards. It is possible to use this function from two or more applications in an integrated circuit card. It is also possible to use as a PKI authentication token by transfer of the command from a host side.

Keywords— Smart Card, Java Card, PKI, Certification Authiroty

1 はじめに

内部にOS (Operating System) およびCPU (Central Processing Unit) を搭載し，暗号演算機能を有するICカードが存在する。従来よりこのようなICカードを，PKI認証方式の非公開鍵およびその鍵を用いた演算を秘匿するためのPKI認証トークンとして利用してきた。また近年では，上述のような単一の機能にとどまらず複数の機能を実現できるJava Card^{TM}[1]のような，マルチアプリケーションOSを搭載したICカードが実用化され始めている。

本研究では，Java Card^{TM} OSを搭載したマルチアプリケーションカードに搭載された複数のアプリケーションが共通的に利用できるPKI機能を提案する。さらに，ホスト側からのコマンド授受により認証トークンとしても利用可能なPKI機能を提案する。

以下では，2章で従来方式，3章で提案方式について述べる。4章で評価および考察を行い，5章でまとめと今後の課題を示す。

2 従来方式

2.1 ICカードコマンド

ICカードと，PCや店舗端末等のホスト間のコマンドは，図1に示すように，ICカードがホストからのAPDU (Application Protocol Data Units) コマンドを受け取り，内部で演算を行い，レスポンスを返す，という流れで処理される。図2に，コマンドとレスポンスのデータ構造を示す。

*〒 169-8555 東京都新宿区大久保 3-4-1, 早稲田大学大学院理工学研究科, Graduate School of Science & Engineering, Waseda University, 3-4-1 Okubo Shinjuku-ku Tokyo, 169-8555 Japan.
†〒 212-8567 神奈川県川崎市幸区鹿島田 890 日立システムプラザ, (株) 日立製作所 システム開発研究所, Hitachi Ltd., Systems Development Laboratory, Hitachi System Plaza Shinkawasaki, 890 Kashimada, Saiwai-ku, Kawasaki-shi, Kanagawa, 212-8567 Japan.

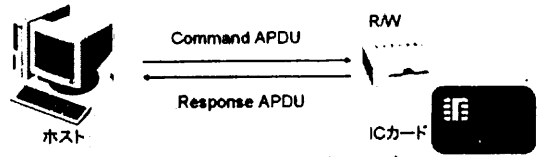

図 1: ICカード通信モデル

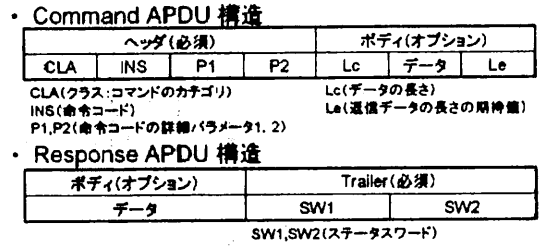

図 2: APDU コマンド

2.2 PKI認証トークンとしての利用

ICカードをPKI認証トークンとして利用するための規格はいくつか提案されている。PKCS#15[2]は，PKI関連データ（非公開鍵や公開鍵証明書等）に関するICカードの内部構造を定義している。PKCS#11[3]は，PC等のホストからICカードのPKI機能を利用するためのアプリケーションプログラムインタフェース (API) を定義している。また，WIM(Wireless Identification Module)[4]では，PKI関連データ構造と，それらにアクセスするためのAPDUコマンド等を定義している。

2.3 多機能マルチアプリケーションICカード

マルチアプリケーション対応のICカードOSとして，JavaCard^{TM} [1]やMULTOS^{TM} [2]がある。本研究では，広く仕様が公開されているJavaCard^{TM}を対象とする。図3にJavaCard^{TM}の通信モデルを示す。ホストから送られたAPDUコマンドは，JavaCard^{TM}の実行環境を経由して事前に選択されたICカード内アプリケーションプログラム (AP) に伝えられる。APはAPDUコマンドを解釈して処理を実行した後，実行結果としてのレスポンスを返す。JavaCard^{TM}には，共通鍵ベースの暗号 (DES,AES等)，公開鍵ベースの暗号 (RSA, 楕円等) 等が規定されているが，公開鍵証明書を取り扱うインタフェースは規定されていない。

[1] Java^{TM} およびJavaCard^{TM} は，米国Sun Microsystems社の登録商標です。
[2] MULTOS^{TM} は，MAOSCOの商標です。

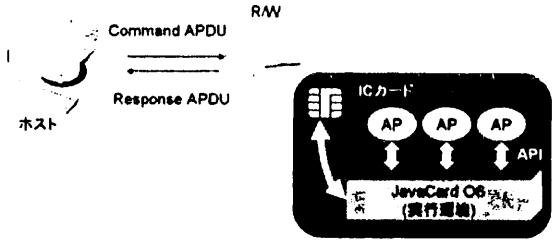

図 3: JavaCard^{TM} 通信モデル

2.4 トラストモデル

PKI 機能を実装する際の認証局 (Certification Authority:CA) の構成に基づく一般的なトラストモデルを示す。

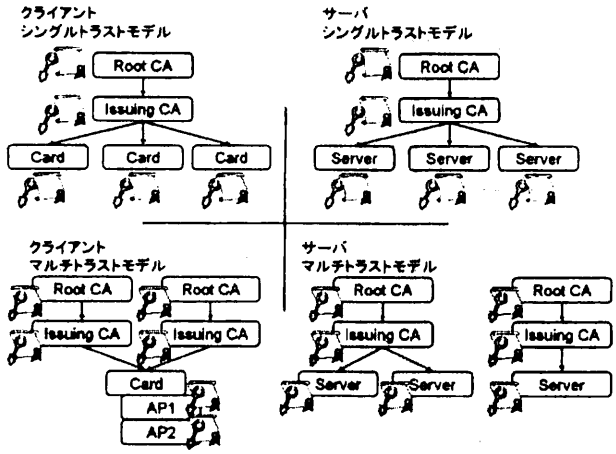

図 4: トラストモデル

図 4 に示すように，信頼できる CA 証明書としてのルート CA 証明書の数によるモデル化を行うことができる。特にマルチトラストモデルにおいては，信頼できる CA が異なるので相互認証の技術が必要とされる。[6] では，PKI の相互認証方式について以下のように分類している。

- **相互認証（Cross Certification）方式**：認証局間で相互証明書を発行する方式
- **第三者方式**：相互認証を要求する両認証局が信頼のおける第三者機関を設定し，各ドメインのエンドエンティティ間において認証を行う際，この第三者機関へ検証を依頼することにより相手エンティティの認証を確立する方式
- **独立方式**：認証者（証明書の検証者）が複数の信頼する認証ドメインの認証局証明書を保有する方式

3 提案方式

3.1 概要

以下に提案方式の考え方を示す。

(1) 従来方式で述べたように，JavaCard^{TM} には，公開鍵証明書を取り扱うインタフェースが規定されていない。この課題を解決するために端末やサーバで用いられる Java^{TM} の技術を応用し，IC カード向けの公開鍵証明書を取り扱う機能と内部 API を提案する。その際に，IC カードのリソース制限を考慮し，複数の AP から共通的に利用可能な PKI 機能を実現する。

(2) 上記機能で利用する鍵類は，複数の AP が利用するので，内部 API 経由ではなく，外部ホストから適切な管理者により APDU コマンドを用いて管理されなければならない。このような PKI 機能の鍵管理を含めたホスト側からの利用機能を提案する。この際に，コマンド変換等に伴うオーバーヘッドを減らすために上記内部 API と外部 APDU コマンドの対応付けを考慮した方式を提案する。

(3) ビジネス要件により異なるトラストモデルが要求されることが考えられるのですべてのトラストモデルに対応できる方式を提案する。

図 5 は，上記考え方に基づく提案方式の概略図である。

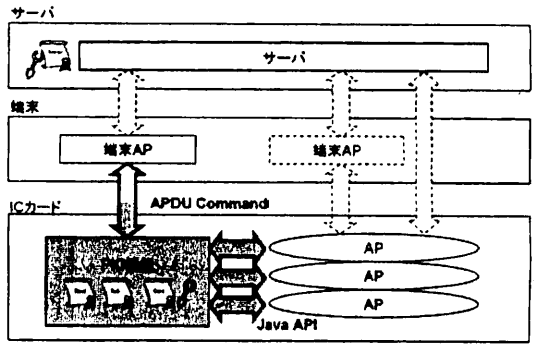

図 5: 提案方式全体概要

以降では，鍵および証明書の保管形式，実現する機能，内部 API と外部 APDU コマンドのマッピングを示す。

3.2 鍵および証明書の保管形式

図 6 に鍵および証明書を保管する鍵ストアの構成を示す。鍵ストアには複数の鍵エントリと信頼できる証明書エントリを持たせる。鍵エントリには，非公開鍵，公開鍵，公開鍵証明書連鎖，秘密鍵（共通鍵）を保持する。信頼できる証明書エントリには，公開鍵証明書連鎖を保持する。またそれぞれのエントリには別名という識別子を割り当て，各エントリへのアクセスはこの別名を用いて行うものとする。

3.3 実現する機能

前節で定義した鍵ストアを利用して多機能 IC カード向け PKI 機能を実現するには，下記の 2 つの大別された機能を実現する必要がある。

- 鍵ストアに設定された鍵を使った暗号関連機能
- 鍵ストアの管理機能

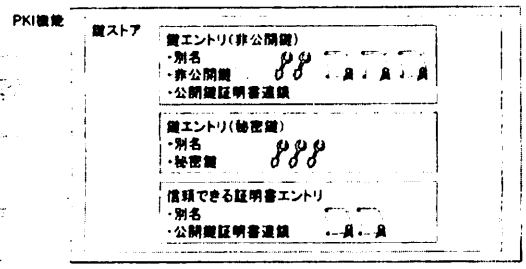

図 6: 鍵ストア

以下にそれぞれの機能の詳細を示す。

3.3.1 鍵ストアに設定された鍵を使った暗号機能

鍵ストアに設定された鍵を使った暗号関連機能を下記に示す。

- **暗号化，復号**
  - 初期化：動作モード，アルゴリズム，鍵を指定する。
  - 更新：初期化情報に基づき暗号処理を行う（中間情報は保持）。
  - 暗号化：初期化情報に基づき暗号処理を行う。
- **署名，検証**
  - 初期化：動作モード，アルゴリズム，鍵を指定する。
  - 更新：署名対象データを蓄積する。
  - 署名：署名対象データに対して署名演算を行う。
  - 検証：署名値の検証を行う。
- **乱数生成**
  - 真正乱数：真正乱数を生成する。
  - 擬似乱数：シードを与えて擬似乱数を生成する。

3.3.2 鍵ストア管理機能

鍵ストア管理用の機能を下記に示す。ここで APDU のみと記載した機能に関しては複数 AP で共有することを考慮し，内部 API では提供しない機能を表す。

- **鍵ストアの管理**
  - 選択：鍵ストアを選択する。
  - 生成：鍵ストアを生成する。(APDU のみ)
  - 削除：鍵ストアを削除する。(APDU のみ)
- **エントリの管理**
  - サイズ取得：鍵ストア内のエントリの数を取得する。
  - 列挙：鍵ストア内のエントリを列挙する。
  - 包含検査：鍵ストア内に指定した別名のエントリが存在するかチェックする。
  - 削除：鍵ストア内の指定した別名のエントリを削除する。(APDU のみ)
- **鍵エントリの管理**
  - 検査：指定した別名が鍵エントリか否かをチェックする。
  - 情報取得：指定した別名の鍵情報を取得する。
  - 設定：指定した別名で鍵を設定する。(APDU のみ)
  - 取得：指定した別名の鍵を取得する。(APDU のみ)
  - 生成：指定した別名の鍵を生成する。(APDU のみ)
- **証明書エントリの管理**
  - 検査：指定した別名が証明書エントリか否かをチェックする。
  - 取得：指定した別名の証明書を取得する。
  - 連鎖取得：指定した別名の証明書連鎖を取得する。
  - 別名取得：指定した証明書の別名を取得する。
  - 設定：指定した別名で証明書を設定する。(APDU のみ)

本節で定義した機能を IC カード内 AP から利用する際の内部 API の関数名と外部ホストから利用する際の APDU コマンド名とのマッピングを表 1 に示す[3]。

[3]個々の関数の引数や APDU コマンドのパラメータは省略している。

4 評価および考察

内部 API による IC カード内 AP からの利用例と，APDU コマンドによる外部ホストからの利用例を示す。

4.1 内部 AP からの利用

以下に IC カード内 AP から本提案の機能を利用するサンプル実装例を示す。AP は事前に設定された鍵を利用できるので，煩雑な鍵管理が必要ないことがわかる。また，鍵ストアと別名のみを指定して，署名や復号演算に必要な非公開鍵は秘匿されていることが確認できる。

```
//鍵ストアの取得
ks = KeyStore.getInstance("HITACHI");

// 署名
sig = Signature.getInstance(Signature.ALG_RSA_SHA_PKCS1);
sig.init(ks, "MyKey", Signature.MODE_SIGN );
sig.sign(in_data, in_off, in_len, sig_buf, sig_off);

// 復号
rsa = Cipher.getInstance(Cipher.ALG_RSA_NOPAD, false);
rsa.init(ks, "MyKey" Cipher.MODE_DECRYPT);
rsa.doFinal(in_data, in_off, in_len, out_buf, out_off);
```

4.2 外部ホストからの利用

以下に外部ホストから APDU コマンドにより本提案の機能を利用するサンプル実装例を示す。コマンドの種類や，アルゴリズムの指定など，上記内部 API と対応関係が取れていることが確認できる。

```
// SELECT KEYSTORE (鍵ストアの選択)
80 80 00 00 07 48 49 54 41 43 48 49
; 90 00

// SIGNATURE INIT (P1:署名(00) P2:ALG_RSA_SHA_PKCS1(42))
80 24 00 42 05 4D 79 4B 65 79
; 90 00

// SIGNATURE SIGN
80 2C 80 00 08 01 02 03 04 05 06 07 08 00
; 署名データ 90 00

// CIPHER INIT (P1:復号(01) P2:ALG_RSA_NOPAD(20)
80 22 01 20 05 4D 79 4B 65 79
; 90 00

// CIPHER DO FINAL
80 2A 80 00 08 01 02 03 04 05 06 07 08 00
; 復号データ 90 00
```

4.3 複数トラストモデルへの対応

2.4 で示した相互認証方式を IC カードで実現する場合，Cross Certification 方式では相互証明書の解析が困難であるという問題がある。また，第三者方式では常にオンラインで第三者機関に接続していなければならないという制限がある。IC カードには，複数の信頼できる CA 証明書を保有する独立方式が向いていると考えられる。ビジネス要件により要求されるトラストモデルは異なるが，本提案方式では下記に示すようにすべてのトラストモデルに対応できる。

- **シングルクライアントトラストモデル**：単一の鍵エントリに非公開鍵および公開鍵証明書連鎖を設定することで実現可能
- **シングルサーバトラストモデル**：単一の証明書エントリに信頼できる証明書連鎖を設定することで実現可能

表 1: 内部 API と APDU コマンドのマッピング

分類		機能	説明	内部 API	外部 APDU コマンド
鍵を利用した暗号処理	暗号化，復号	初期化	動作モード，アルゴリズム，鍵を指定する。	getInstance(), init()	CIPHER IINIT FOR DECRYPT
		更新	初期化情報に基づき暗号処理を行う（中間情報は保持）。	update()	CIPHER UPDATE
		暗号化	初期化情報に基づき暗号処理を行う。	doFinal()	CIPHER DO FINAL
	署名，検証	初期化	動作モード，アルゴリズム，鍵を指定する。	getInstance(), init()	SIGNATURE INIT FOR SIGN
		更新	署名対象データを蓄積する。	update()	SIGNATURE UPDATE
		署名	署名対象データに対して署名演算を行う。	sign()	SIGNATURE SIGN
		検証	署名値の検証を行う。	verify()	SIGNATURE VERIFY
	乱数	真正乱数	真正乱数を生成する。	getInstance(), generate-Data()	ASK SECURE RANDOM
		擬似乱数	シードを与えて擬似乱数を生成する。	getInstance(), setSeed(), generateData()	ASK PSEUDO RANDOM
鍵ストアの管理	鍵ストアの管理	選択	鍵ストアを選択する。	getInstance()	SELECT KEYSTORE
		生成	鍵ストアを生成する。	-	GENERATE KEYSTORE
		削除	鍵ストアを削除する。	-	DELETE KEY STORE
	エントリの管理	サイズ取得	鍵ストア内のエントリの数を取得する。	size()	GET ENTRY SIZE
		列挙	鍵ストア内のエントリを列挙する。	aliases()	GET ALIASES
		包含検査	鍵ストア内の指定した別名のエントリの存在をチェックする。	containsAlias()	CHECK ALIAS
		削除	鍵ストア内の指定した別名のエントリを削除する。	-	DELETE ENTRY
	鍵エントリの管理	検査	指定した別名が鍵エントリか否かをチェックする。	isKeyEntry()	CHECK KEY ENTRY
		設定	指定した別名で鍵を設定する。	-	SET KEY ENTRY
		取得	指定した別名の鍵を取得する。	-	GET KEY
		情報取得	指定した別名の鍵情報を取得する。	getKeyInfo()	GET KEY INFO
		生成	指定した別名の鍵を生成する。	-	GENERATE KEY
	証明書エントリの管理	検査	指定した別名が証明書エントリか否かをチェックする。	isCertificateEntry()	CHECK CERTIFICATE ENTRY
		設定	指定した別名で証明書を設定する。	-	SET CERTIFICATE ENTRY
		取得	指定した別名の証明書を取得する。	getCertificate()	GET CERTIFICATE
		連鎖取得	指定した別名の証明書連鎖を取得する。	getCertificateChain()	GET CERTIFICATE CHAIN
		別名取得	指定した証明書の別名を取得する。	getCertificateAlias()	GET CERTIFICATE ALIAS

表 2: 従来方式と提案方式の機能比較

		公開鍵証明書インタフェース		複数トラストモデル対応
		複数の IC カード AP 用	外部ホスト用	
従来方式	Java Card™	×	×	×
	WIM, PKCS	×	○	○
提案方式		○	○	○

- **マルチクライアントトラストモデル**：複数の鍵エントリに非公開鍵および公開鍵証明書連鎖を設定することで実現可能
- **マルチサーバトラストモデル**：複数の証明書エントリに信頼できる証明書連鎖を設定することで実現可能

5 まとめと今後の課題

IC カード内の複数 AP から利用することが可能であり，ホスト側からの APDU コマンドの授受によって PKI 認証トークンとして利用することも可能な多機能 IC カード向け PKI 機能を提案した。また，複数のトラストモデルに対応可能なことを示した。表 2 に従来方式と提案方式の機能比較，下記に今後の課題を示す。

- 鍵の種類や暗号アルゴリズムの詳細化
- 鍵ストア管理機能のアクセス制御の詳細化
- 性能評価

参考文献

[1] "Java CardTM 2.2.1 Application Programming Interface, Sun Microsystems, Inc.

[2] "PKCS #15 v1.1: Cryptographic Token Information Syntax Standard", RSA Security Inc., June 6, 2000

[3] "PKCS #11 v2.20: Cryptographic Token Interface Standard", RSA Security Inc., June 28, 2004

[4] "Wireless Identity Module Version 1.1 Part: Security Version 24-Oct-2002", Open Mobile Alliance, Ltd.

[5] "GlobalPlatform Card Specification Version 2.1.1", GlobalPlatform Inc., March 2003

[6] 田中 俊昭, 関野 公彦, 菊地 仁, 梅澤 克之, "モバイルコマースにおける PKI の現状と課題 -mITF モバイルコマース部会認証WGの活動状況-", 情報処理学会研究報告「コンピュータセキュリティ」, No.22-28, 2003.