跨瀏覽器元件說明
版本1.3.4.5

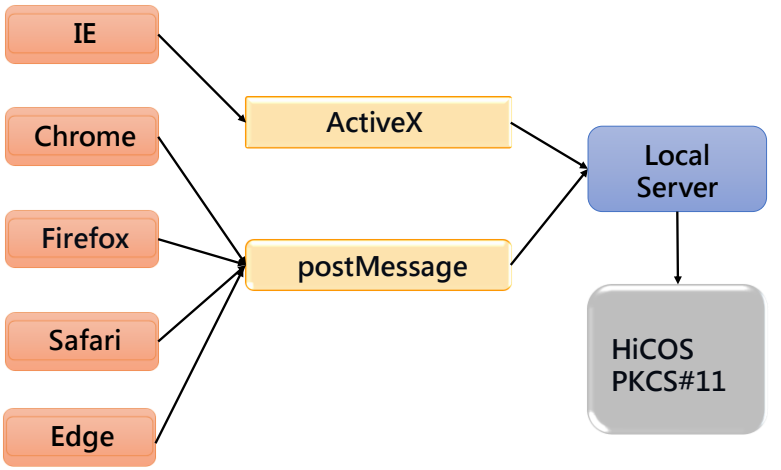
108年10月30日

大綱

- 應用系統跨瀏覽器解決方案
- 應用系統導入說明
- 範例網站介紹
- 開發注意事項

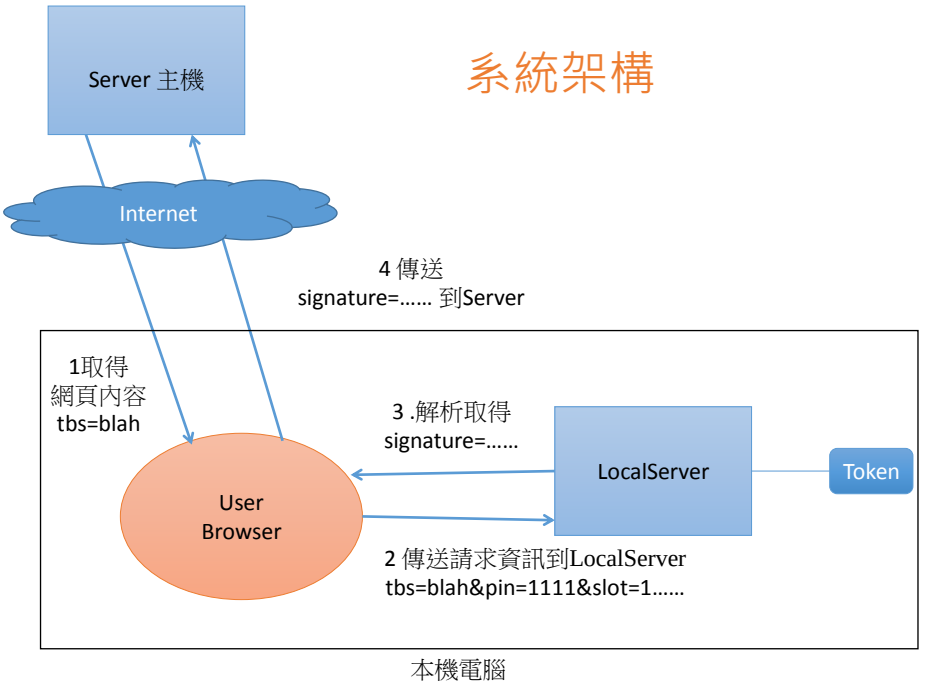
跨瀏覽器元件解決方案

應用系統實作各瀏覽器提供之標準解決方案



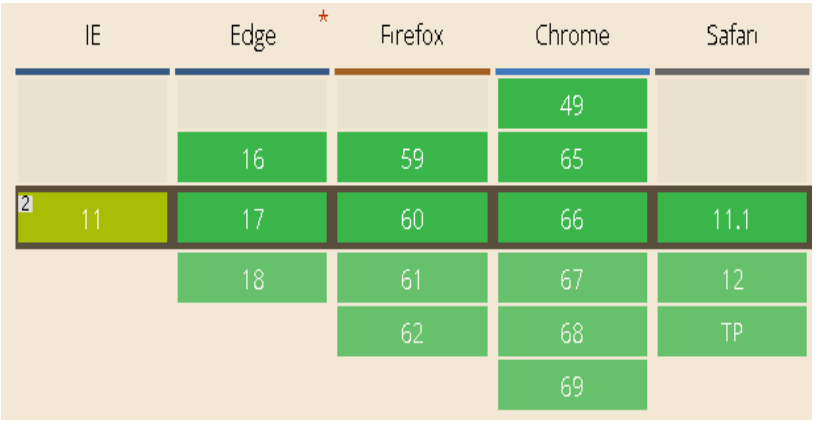
IE不支援跨window及domain的postMessage

系統架構



本機電腦

瀏覽器支援postMessage現況



資料來源：
http://caniuse.com/#search=postMessage

應用系統導入說明(1/2)

- 本範例提供數位簽章/驗證功能、加密/解密功能、卡片資訊讀取功能。
- 提供用戶端安裝包 下載點，有功能有提供檢測用戶端IC卡片檢測功能，可以支援業系統有
  - Windows版
  - Linux版
  - Mac版
- 提供範例程式語言
  - Java版本
  - .NET版本
- 範例網站的驗證功能僅包含簽章驗證，不包含憑證驗證與憑證延伸欄位解析，開發應用系統仍應遵照GPKI公鑰憑證處理安全事項檢查表規定進行檢核確定。

應用系統導入說明(2/2)

- 要驗用戶憑證是否為合法證憑、證狀查詢、憑證有效性檢查、憑證延伸欄位內容解析(例如：統一編號、身分證後四碼等相關欄位)，請搭配Hisecure Api[C++語言/Java語言]
- 如需要驗證用戶身分完整性可以搭配身分確認服務系統(詳細內容請參考:http://moica.nat.gov.tw/check_apply.html)簡稱Ics Api， 適用單位：符合內政部自然人憑證用戶身分確認服務系統申請要點第一點和第二點之規定，該應用系統之使用對象為一般民眾，並以自然人憑證作為身分驗 證方式。

Java版範例網頁介紹(1/2)

網頁	說明	備註
Index.htm	範例首頁，有元件下載功能	
checkVersion.htm	檢視元件狀態 可顯示目前的驅動程式資訊、插入卡片資訊等	
signPKCS7.htm signPKCS1.htm signPKCS7withSelectCard.htm	數簽章(PKCS#7/PKCS#1)範例網頁 withSelectCard的是可以選擇要使用哪一張卡簽章	tbs:待簽本文 pin:IC卡密碼 hashAlgorithm: SHA1/SHA256/SHA384/SHA512 nonce: 隨機亂數 checkValidity:是否檢查憑證時間 withCardSN:簽章內含卡號(true/false) tbsEncoding:待簽文編碼格式(NONE/base64)
signPKCS1withSelectcardAndSCAP	數簽章(PKCS#1)範例網頁並取的附卡授權資訊	tbs:待簽本文 pin:IC卡密碼 hashAlgorithm: SHA1/SHA256/SHA384/SHA512 nonce: 隨機亂數 tbsEncoding:待簽文編碼格式(NONE/base64) withSCAP:取得工商憑證附卡授權證

Java版範例網頁介紹(2/2)

網頁	說明	備註
signRAW.htm	RAW數簽章PKCS#1範例(先針對明文先Hash動作再進行簽章)	tbs: Base64 做RAW資料 pin:IC卡密碼 tbsEncoding:待簽文編碼格式(base64)
VerifyPKCS7.jsp VerifyPKCS1.jsp	驗證PKCS#7/PKCS#1數位簽章範例	不包含憑證驗證(憑證有效性、憑證狀態、憑證發行單位驗證等)
readCertificate.htm	讀取用戶憑證，並加密資料	cipher type: 加密格式(PKCS1/PKCS7) plainEncoding:明文編碼格式(NONE/base64)
DecryptData.jsp	使用IC卡解密資料	pin:IC卡密碼

.NET版範例網頁介紹(1/2)

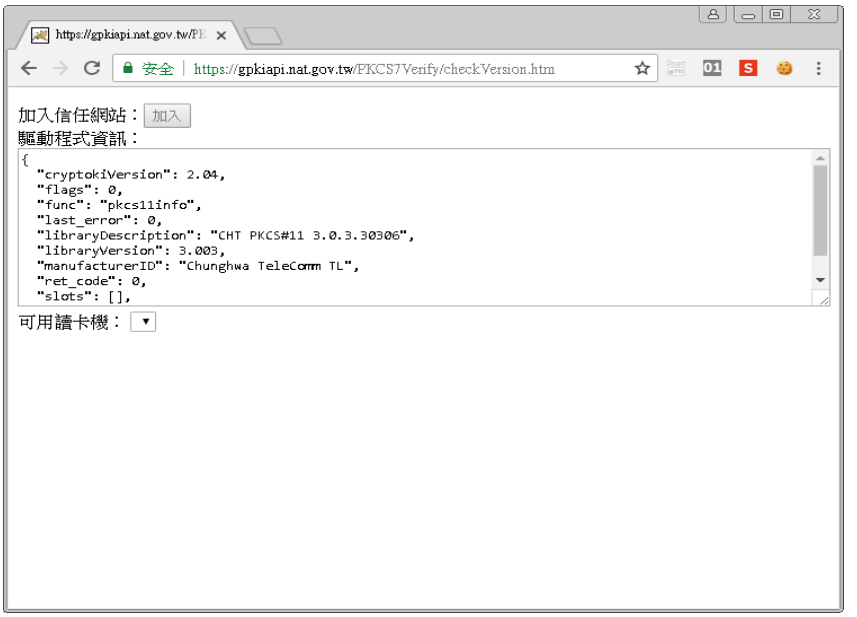
網頁b	說明	備註
Index.htm	範例首頁，有元件下載功能	
checkVersion.htm	檢視元件狀態 可顯示目前的驅動程式資訊、插入卡片資訊等	
signPKCS7.htm signPKCS1.htm signPKCS7withSelectCard.htm	數簽章(PKCS#7/PKCS#1)範例網頁 withSelectCard的是可以選擇要使用哪一張卡簽章	tbs:待簽本文 pin:IC卡密碼 hashAlgorithm: SHA1/SHA256/SHA384/SHA512 nonce: 隨機亂數 withCardSN:簽章內含卡號(true/false) tbsEncoding:待簽文編碼格式(NONE/base64)
signPKCS1withSelectcardAndSCAP	數簽章(PKCS#1)範例網頁並取的附卡授權資訊	tbs:待簽本文 pin:IC卡密碼 hashAlgorithm: SHA1/SHA256/SHA384/SHA512 nonce: 隨機亂數 tbsEncoding:待簽文編碼格式(NONE/base64) withSCAP:取得工商憑證附卡授權證

.NET版範例網頁介紹(2/2)

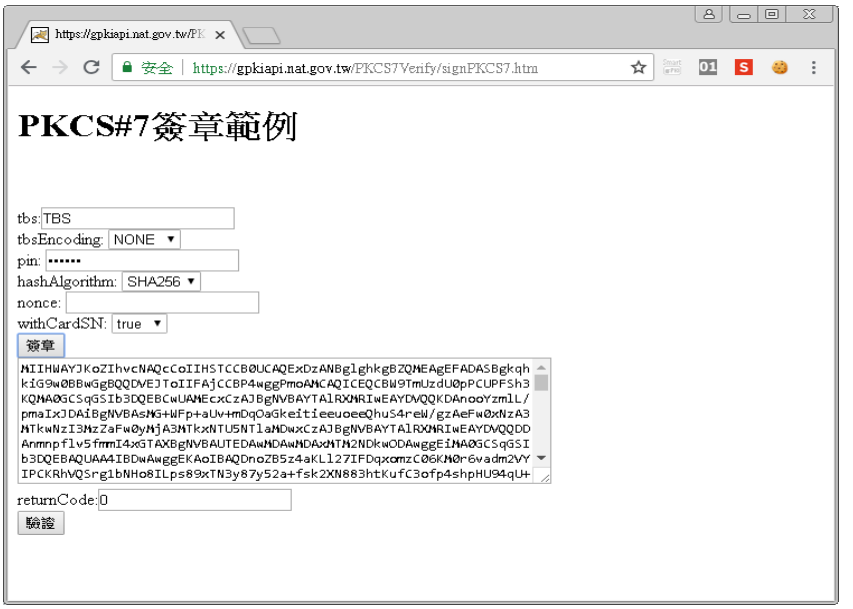
網頁b	說明	備註
signRAW.htm	RAW數簽章PKCS#1範例(先針對明文先Hash動作再進行簽章)	tbs: Base64 做RAW資料 pin:IC卡密碼 tbsEncoding:待簽文編碼格式(base64)
VerifyP7.aspx VerifyP1.aspx	驗證PKCS#7/PKCS#1數位簽章範例	不包含憑證驗證(憑證有效性、憑證狀態、憑證發行單位驗證等)
readCertificate.htm	讀取用戶憑證，並加密資料	cipher type: 加密格式(PKCS1/PKCS7) plainEncoding:明文編碼格式(NONE/base64)
DecryptData.aspx	使用IC卡解密資料	pin:IC卡密碼

index.htm

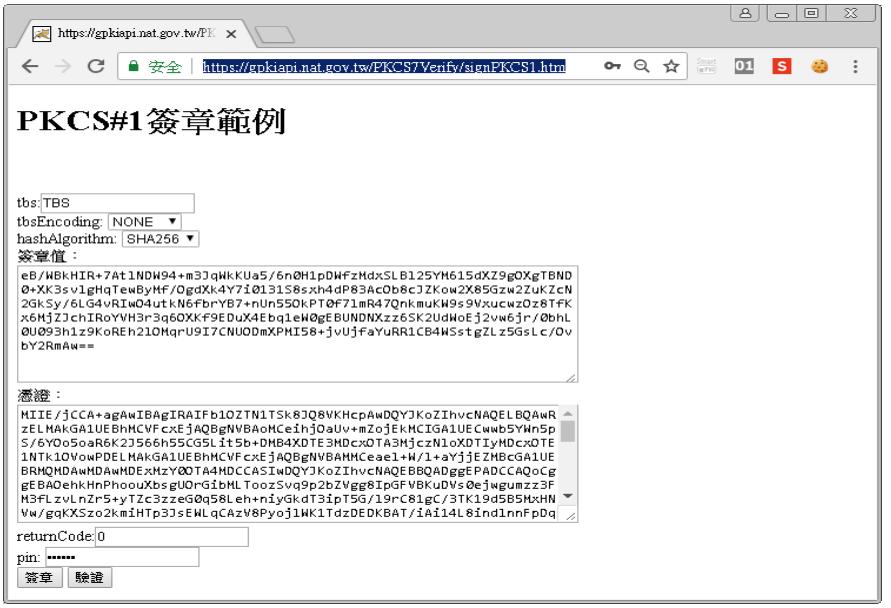
checkVersion.htm



signPKCS7.htm



signPKCS1.htm

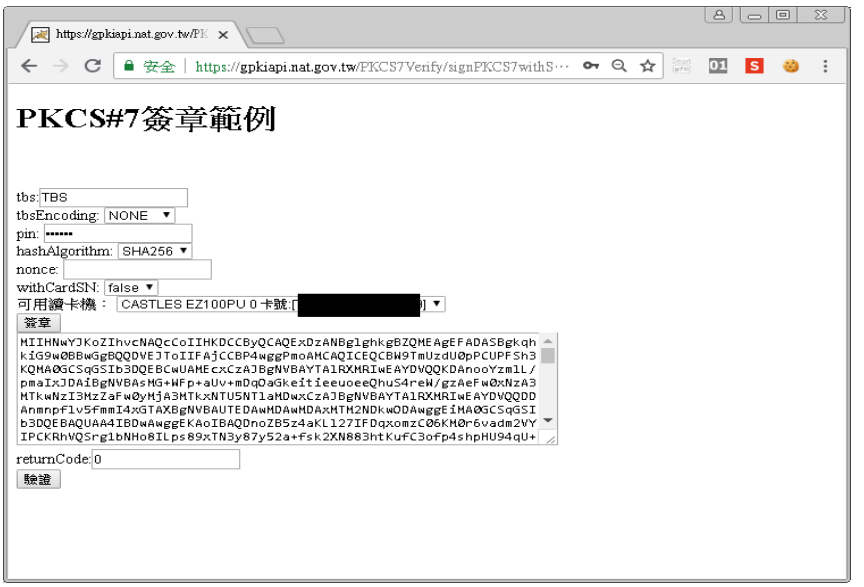


signRAW.html

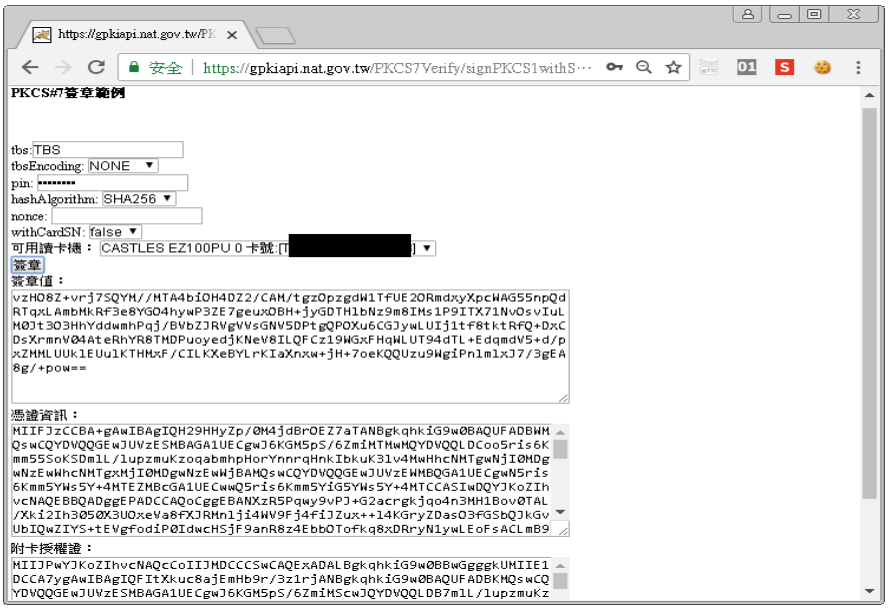
signRAW.html 補充說明

- 補充說明hashPrefixes 的值
- SHA1 prefix 值
- 0x30, 0x21, 0x30, 0x09, 0x06, 0x05, 0x2b, 0x0e, 0x03, 0x02, 0x1a, 0x05, 0x00, 0x04, 0x14
- SHA256 prefix 值
- 0x30, 0x31, 0x30, 0x0d, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x01, 0x05, 0x00, 0x04, 0x20
- SHA384 prefix 值
- 0x30, 0x41, 0x30, 0x0d, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x02, 0x05, 0x00, 0x04, 0x30
- SHA512 prefix 值
- 0x30, 0x51, 0x30, 0x0d, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x03, 0x05, 0x00, 0x04, 0x40
- 當『TBS』進行Hash(SHA1\ SHA256 \ SHA384\ SHA512)後，將**其Uint8 Byte Array prefix 值放置到已被**Hash過後的字串前面在進行Base64編碼動作。
- SHA1 Hash後的Base64 :
  MCEwCQYFKw4DAhoFAAQU52AqDFox+uJOPeG8gd86df9UY3g=
- SHA256 Hash後的Base64 :
  MDEwDQYJYIZIAWUDBAIBBQAEIMnXsP3Gf/4Y4lexWlIlnQ8CvCWzhP2Tra9hBVOo1IL8
- SHA384 Hash後的Base64 :
  MEEwDQYJYIZIAWUDBAICBQAEMMpgDnLuF3Y09QuAuPx5ydKEqMo95YsRQwkKiK3KOD4bj
  S9lfyrygM11LhwaglzmiQ==
- SHA512 Hash後的Base64 :
- MFEwDQYJYIZIAWUDBAIDBQAEQJw4NM7VUaC5/UY4MMvRRr60/3RB5tavbpF7JCSNkyy7u
  mrHXVoWd2N0JI5/md5G5VmyHh8HgFbFngcUmvZV4iA=

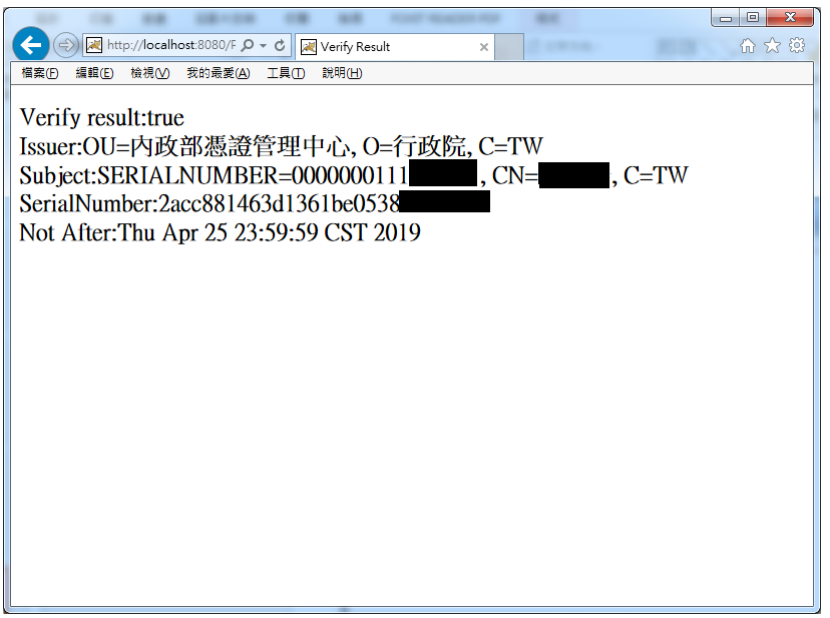
signPKCS7withSelectcard.html



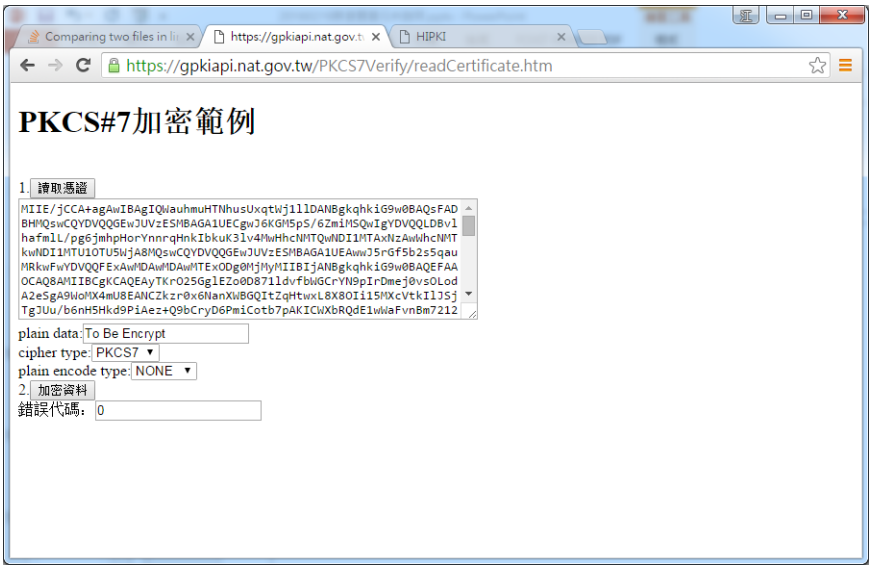
signPKCS1withSelectcardAndSCAP.html



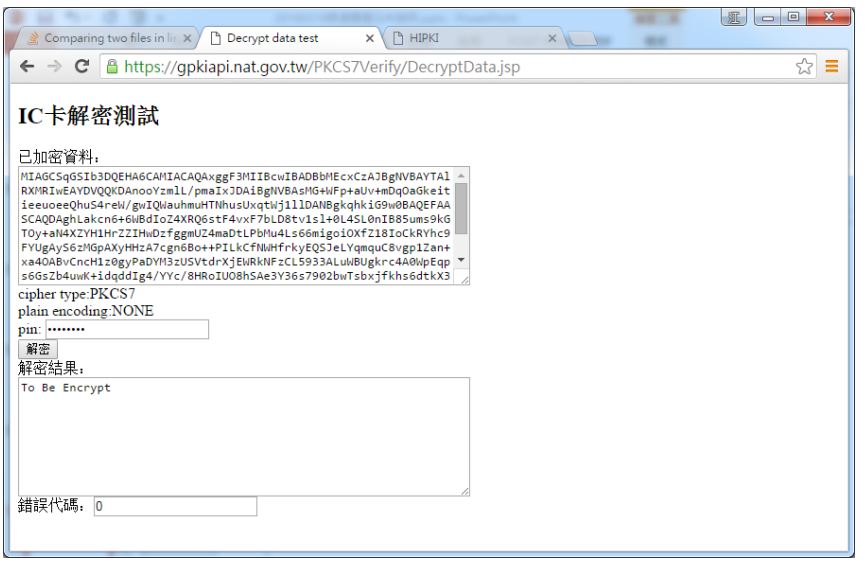
VerifySignature.jsp



讀取憑證


Comparing two files in li
https://gpkiapi.nat.gov.t
HIPKI
https://gpkiapi.nat.gov.tw/PKCS7Verify/readCertificate.htm
PKCS#7加密範例
1. 讀取憑證
MIIE/jCCA+agAwIBAgIQWauhmuHTNhusUxqtWj1llDANBgkqhkiG9w0BAQsFAD
BHMQswCQYDVQQGEwJUVzESMBAGA1UECgwJ6KGM5pS/6ZmiMSQwIgYDVQQLDBvl
hafmll/pg6jmhpHorYnnrqHnkIbkuK3lv4MwHhcNMTQwNDI1MTAxNzAwWhcNMT
kwNDI1MTU1OTU5WjA8MQswCQYDVQQGEwJUVzESMBAGA1UEAwwJ5rGf5b2s5qau
MRkwFwYDVQQFExAwMDAwMDAwMTExODg0MjMyMIIBIjANBgkqhkiG9w0BAQEFAA
OCAQ8AMIIBCgKCAQEAyTKrO25GglEZo0D871ldvfbWGCrYN9pIrDmej0vsOLod
A2eSgA9WoMX4mU8EANCZkzr0x6NanXWBGQItZqHtwxL8X8OIi15MXcVtklJSj
TgJUu/b6nH5Hkd9PiAez+Q9bCryD6PmiCotb7pAKICWXbRQdE1wWaFvnBm7212
plain data: To Be Encrypt
cipher type: PKCS7
plain encode type: NONE
2. 加密資料
錯誤代碼： 0

解密資料


Comparing two files in li
Decrypt data test
HIPKI
https://gpkiapi.nat.gov.tw/PKCS7Verify/DecryptData.jsp
IC卡解密測試
已加密資料：
MIAGCSqGSIb3DQEHA6CAMIACAQAxggF3MIIBcwIBADBbMEcxCzAJBgNVBAYTAl
RXMRIwEAYDVQQKDAnooYzmlL/pmaIxJDAiBgNVBAsMG+WFp+aUv+mDqOaGkeit
ieeuoeeQhuS4reW/gwIQWauhmuHTNhusUxqtWj1llDANBgkqhkiG9w0BAQEFAA
SCAQDAghLakcn6+6WBdIoZ4XRQ6stF4vxF7bLD8tv1sl+0L4SL0nIB85ums9kG
TOy+aN4XZYH1HrZZIHwDzfggmUZ4maDtLPbMu4Ls66migoiOXfZ18IoCkRYhc9
FYUgAyS6zMGpAXyHHzA7cgn6Bo++PILkCfNWHfrkyEQSJeLYqmquC8vgp1Zan+
xa4OABvCncH1z0gyPaDYM3zUSVtdrXjEWRkNFzCL5933ALuWBUgkrc4A0WpEqp
s6GsZb4uwK+idqddIg4/YYc/8HRoIUO8hSAe3Y36s7902bwTsbxjfkhs6dtkX3
cipher type:PKCS7
plain encoding:NONE
pin: ••••••••
解密
解密結果：
To Be Encrypt
錯誤代碼： 0

開發注意事項

- 網頁版面呈現相容性問題(IE瀏覽器)
- IE瀏覽器運行跨平台重點說明
- 建立傳送JSON參數內容說明
- 解析回應JSON參數內容說明
- 跨平台請求功能呼叫使用
- 需要轉換字串為UTF-8 編碼注意

網頁版面呈現相容性設定(IE瀏覽器)

- 在開發網頁請記住要在ＨＴＭＬ網頁語法加入這行

  <meta http-equiv="X-UA-Compatible" content="IE=EDGE" charset="utf-8"/>

- 放置位置在HTML的 <head> 裡面。

```
1 <!DOCTYPE HTML>
2 <HTML><HEAD>
3 <meta http-equiv="X-UA-Compatible" content="IE=EDGE" charset="utf-8"/>
4 <script type="text/javascript" src="errorcode.js"></script>
```

- 主要目的是讓HTML頁面在最新的IE瀏覽器正常顯示，不會因為使用者開啟相容性檢視導致HTML頁面顯示舊版IE運行。
- 可以確保用戶使用HTML 在IE 呈現頁面正常狀態 。

PS: 這個設定只針對是for Internet explorer 瀏覽器。

IE瀏覽器運行跨平台重點說明 (1/3)

- 使用誇平台網頁範例程式進行改寫時候需要注意

1. HTML語法 要注意是否這行
   加入在<BODY>的下一行

```
<BODY>
<span id="httpObject" ></span>
<H1>PKCS#7簽章範例</H1><br/>
<BR>tbs:<INPUT name="tbs" id="tbs" value="TBS"/><BR>
tbsEncoding: <SELECT name="tbsEncoding" id="tbsEncoding">
<OPTION value="NONE" selected>NONE</OPTION><OPTION value="base64" >base64</OPTION>
</SELECT><BR>
```

IE瀏覽器運行跨平台重點說明 (2/3)

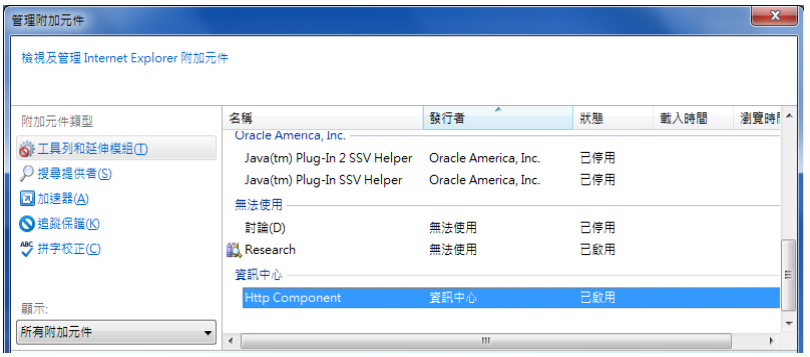
2. JavaScript語法要注意是否這行 document.getElementById（“httpObject”).innerHTML= ‘<OBJECT id=“http” width=1 height=1 style=“LEFT: 1px; TOP: 1px” type=“application/x-httpcomponent” VIEWASTEXT></OBJECT>’ ;

呼叫Activex元件傳送JSON內容請求資訊給本機端(Local Server)進行簽章或解密等相關作業。

```
var ua = window.navigator.userAgent;
if(ua.indexOf("MSIE")!=-1 || ua.indexOf("Trident")!=-1) //is IE, use ActiveX
{
    postTarget=window.open("waiting.htm?簽章中", "Signing","height=200, width=200, left=100, top=20");
    var tbsPackage=getTbsPackage();
    document.getElementById("httpObject").innerHTML='<OBJECT id="http" width=1 height=1 style="LEFT: 1px; TOP: 1px" type
    var data=postData("http://localhost:61161/sign","tbsPackage="+tbsPackage);
    postTarget.close();
    postTarget=null;
    if(!data) alert("尚未安裝元件");
    else setSignature(data);
}
else{
```

IE瀏覽器運行跨平台重點說明 (3/3)

3. 檢查IE 管理附加元件是否有出現 HttpComponent 元件名稱發行單位 資訊中心。沒有顯示元件出來建議做法就是重新安裝跨平台網頁元件需要已管理者帳戶安裝。



傳送JSON參數說明(1/6)

- 傳送PKCS7簽章 JSON定義 (可以參考網頁範例有signPKCS7.htm、signPKCS7withSelectCard.htm)

固定項目
tbsData[“tbs”]= 指定字串明文或一串Base64 字串
tbsData[“tbsEncoding”]= 是否編譯明文(NONE/base64)
tbsData[“hashAlgorithm”]=指定演算法 SHA1、SHA256、SHA386、SHA512等
tbsData[“withCardSN”]= 是否顯示使用者卡號(true->顯示卡號/ false->不顯示卡號)
tbsData[“pin”]=使用者卡片密碼
tbsData[“nonce”]= 指定已一個亂數或session 代表
tbsData[“func”]=“MakeSignature”; 產生簽章用途
tbsData[“signatureType”]= PKCS7
補充項目
tbsData[“checkValidity”]=是否驗證卡片效期
(true ->驗效期/ false ->不會驗效期)
tbsData[“slotDescription”]=指定讀卡機

傳送JSON參數說明(2/6)

- **建立傳送JSON內容方式**

```
function getTbsPackage(){
    var tbsData = {};
    tbsData["tbs"]=encodeURIComponent(document.getElementById("tbs").value);
    tbsData["tbsEncoding"]=document.getElementById("tbsEncoding").value;
    tbsData["hashAlgorithm"]=document.getElementById("hashAlgorithm").value;
    tbsData["withCardSN"]=document.getElementById("withCardSN").value;
    tbsData["pin"]=encodeURIComponent(document.getElementById("pin").value);
    tbsData["nonce"]=document.getElementById("nonce").value;
    tbsData["func"]="MakeSignature";
    tbsData["signatureType"]="PKCS7";
    tbsData["checkValidity"]=true;
    var json = JSON.stringify(tbsData);
    return json;
}
```

傳送JSON參數說明(3/6)

- 傳送PKCS1簽章 JSON定義 (可以參考網頁範例有signPKCS1.htm、signPKCS1withSelectcardAndSCAP)

tbsData[“tbs”]= 指定字串明文或一串Base64 字串
tbsData[“tbsEncoding”]= 是否編譯明文(NONE/base64)
tbsData[“hashAlgorithm”]=指定演算法 SHA1、SHA256、SHA386 、SHA512等
tbsData[“pin”]=使用者卡片密碼
tbsData[“func”]=“MakeSignature”; 產生簽章用途
tbsData[“signatureType”]= PKCS1
補充項目
tbsData[“checkValidity”]=是否驗證卡片效期
(true ->不驗效期、 false ->會驗效期)
tbsData[“slotDescription”]=指定讀卡機
tbsData[“withSCAP”]=是否要取出工商憑證附卡授權資訊
(true ->取得/false->不取得)只能讀取工商憑證卡片並做過附卡授權的卡片才能取出相關資料出來

傳送JSON參數說明(4/6)

- **建立傳送JSON內容方式**

```
function getTbsPackage(){
    var tbsData = {};
    tbsData["tbs"]=encodeURIComponent(document.getElementById("tbs").value);
    tbsData["tbsEncoding"]=document.getElementById("tbsEncoding").value;
    tbsData["hashAlgorithm"]=document.getElementById("hashAlgorithm").value;
    tbsData["pin"]=encodeURIComponent(document.getElementById("pin").value);
    tbsData["checkValidity"]=true;
    tbsData["func"]="MakeSignature";
    tbsData["signatureType"]="PKCS1";
    var json = JSON.stringify(tbsData);
    return json;
}
```

傳送JSON參數說明(5/6)

- 傳送RAW簽章 JSON定義 (可以參考網頁範例有signRAW .htm)

  tbsData["tbs"]=
  tbsData["tbsEncoding"]=
  tbsData["pin"]=使用者卡片 密碼
  tbsData["func"]="MakeSignature";
  tbsData["signatureType"]= 指定RAW ;
  補充項目
  tbsData["slotDescription"]=指定讀卡機

- **建立傳送JSON內容方式**

```
function getEncryptedPackage(){
    var tbsData = {};
    tbsData["cipher"]=encodeURIComponent(document.getElementById("cipher").value.replace(/[\r\n]/g, ''));
    tbsData["cipherType"]=document.getElementById("cipherType").innerHTML;
    tbsData["plainEncoding"]=document.getElementById("plainEncoding").innerHTML;
    tbsData["pin"]=encodeURIComponent(document.getElementById("pin").value);
    tbsData["func"]="DecryptData";
    var json = JSON.stringify(tbsData);
    return json;
}
```

傳送JSON參數說明(6/6)

- 傳送解密JSON定義 (可以參考網頁範例有DecryptData.jsp或.aspx)
  tbsData["cipher"]=加密後的資訊
  tbsData["cipherType"]= 加密方式PKCS7或PKCS1
  tbsData["plainEncoding"]=是否編譯文字 (NONE/base64)
  tbsData["pin"]=使用者卡片密碼
  tbsData["func"]=“DecryptData”；
  補充項目
  tbsData["slotDescription"]=指定讀卡機

- **建立傳送JSON內容方式**

```
function getTbsPackage(){
    var tbsData = {};
    tbsData["tbs"]=document.getElementById("tbs").value;//Base64 做RAW資料
    tbsData["tbsEncoding"]=document.getElementById("tbsEncoding").value;
    tbsData["pin"]=encodeURIComponent(document.getElementById("pin").value);
    tbsData["func"]="MakeSignature";
    tbsData["signatureType"]="RAW"; //RAW
    var json = JSON.stringify(tbsData);
    return json;
}
```

解析回應JSON參數內容說明(1/2)

- 回應格式 JSON表示，相關格式內容請參考此文件
- 解析PKCS7簽章JSON格式寫法

```
function setSignature(signature){
    var ret=JSON.parse(signature);
    document.getElementById("ResultSignedData").value=ret.signature; //取得簽章結果
    document.getElementById("returnCode").value=ret.ret_code; // 回傳代碼
    if(ret.ret_code!=0){//ret_code如果非0錯誤訊息表示
    alert(MajorErrorReason(ret.ret_code)); // 第一層跨停元件顯示錯誤代碼
        if(ret.last_error)
            alert(MinorErrorReason(ret.last_error)); // 第二層cryptographic顯示錯誤代碼
    }
}
```

- 解析PKCS1、 RWA簽章JSON格式寫法

```
function setSignature(signature){
    var ret=JSON.parse(signature);
    document.getElementById("b64Signature").value=ret.signature;//取得簽章結果
    document.getElementById("b64Cert").value=ret.certb64;//取得用戶數位憑證
    document.getElementById("returnCode").value=ret.ret_code;// 回傳代碼
    if(ret.ret_code!=0){//ret_code如果非0錯誤訊息表示
    alert(MajorErrorReason(ret.ret_code)); // 第一層跨停元件顯示錯誤代碼
        if(ret.last_error)
            alert(MinorErrorReason(ret.last_error)); // 第二層cryptographic顯示錯誤代碼
    }
}
```

解析回應JSON內容說明(2/2)

- 解析解密JSON格式寫法

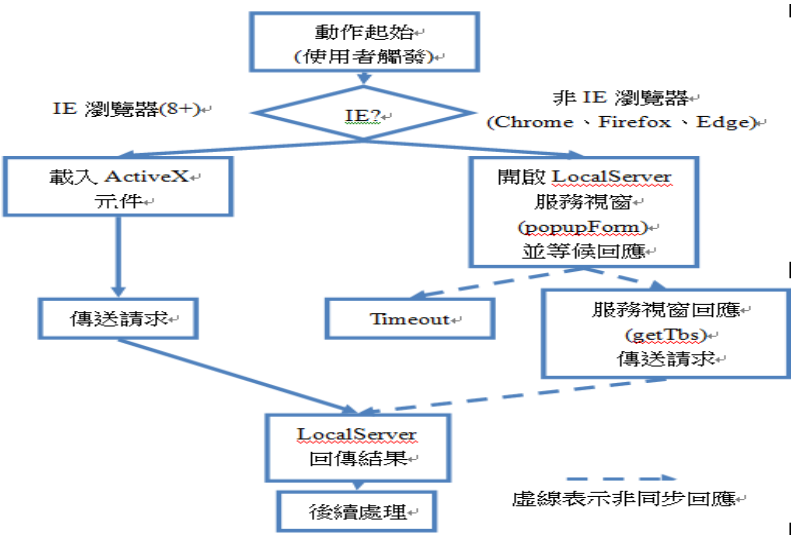
```
function setDecryptResult(result){
    var ret=JSON.parse(result);
    document.getElementById("plainData").value=ret.plain; //解密後的內容
    document.getElementById("returnCode").value=ret.ret_code;// 回傳代碼
    if(ret.ret_code!=0){//ret_code如果非0錯誤訊息表示
    alert(MajorErrorReason(ret.ret_code)); // 第一層跨停元件顯示錯誤代碼
        if(ret.last_error)
            alert(MinorErrorReason(ret.last_error)); // 第二層cryptographic顯示錯誤代碼
    }
}
```

- 解析PKCS1簽章含工商憑證附卡授權資料JSON格式寫法

```
function setSignature(signature){
    var ret=JSON.parse(signature);
    document.getElementById("b64Signature").value=ret.signature;//取得簽章結果
    document.getElementById("b64Cert").value=ret.certb64;//取得用戶數位憑證
    if(ret.SCAP!=null) //判斷是否有附卡授權資料
        document.getElementById("b64SCAP").value=ret.SCAP;//取得附卡授權資料
    document.getElementById("returnCode").value=ret.ret_code;// 回傳代碼
    if(ret.ret_code!=0){//ret_code如果非0錯誤訊息表示
        alert(MajorErrorReason(ret.ret_code)); // 第一層跨停元件顯示錯誤代碼
        if(ret.last_error)
            alert(MinorErrorReason(ret.last_error)); // 第二層cryptographic顯示錯誤代碼
    }
}
```

跨平台請求功能呼叫使用(1/5)

- **流程圖**



跨平台請求功能呼叫使用(1/4)

• 使用Activex 元件呼叫跨平台傳送JSON參數的方式

呼叫函式是 postData(“指定URL” , “tbsPackage= 指定JSON資料”)

• 指定URL 列表如下

1. http://localhost:61161/pkcs11info //取得讀卡機與卡片資訊
2. http://localhost:61161/pkcs11info?withcert=true //包含憑證以及金鑰資訊
3. http://localhost:61161/sign //進行簽章作業
4. http://localhost:61161/decrypt //進行解密作業

• 指定JSON資料傳送

只有進行簽章和解密作業才需要傳送JSON資料，其他可以給”” ，後面補充說說明。

範例說明

// 取得讀卡機與卡片資訊(參數二地方就不需要給參數)

```
var  output=postData(“http://localhost:61161/pkcs11info”,””);
```

// 解密作業

```
var data=postData("http://localhost:61161/decrypt","tbsPackage="+tbsPackage);
```

跨平台請求功能呼叫使用(2/4)

```
function postData(target,data){//呼叫Activex元件傳送資料
    if(!http.sendRequest){
        return null;
    }
    http.url=target; //指定URL
    http.actionMethod="POST"; //傳送方式
    var code=http.sendRequest(data);//傳送資料
    if(code!=0) return null;
    return http.responseText; //回傳結果
}
```

```
function makeSignature(){
    var ua = window.navigator.userAgent;//取得瀏覽器用戶代理資訊
    if(ua.indexOf("MSIE")!=-1 || ua.indexOf("Trident")!=-1){//is IE, use ActiveX
    postTarget=window.open("waiting.htm?簽章中",
    "Signing","height=200, width=200, left=100, top=20"); //開啟小視窗圖示
    var tbsPackage=getTbsPackage(); //取得JSON參數內容
    var ac = '<OBJECT id="http" width=1 height=1 style="LEFT: 1px; TOP: 1px"
    + 'type="application/x-httpcomponent" VIEWASTEXT></OBJECT>';
    document.getElementById("httpObject").innerHTML=ac;//啟動Activex元件
    var data=postData("http://localhost:61161/sign","tbsPackage="+tbsPackage);//啟動Activex函式
    postTarget.close(); //關閉小視窗
    postTarget=null;
    if(!data) alert("尚未安裝元件");
    else setSignature(data);//解析回傳的JSON資料內容
   }else{
        /*      非IE使用方式        */
    }
}
```

跨平台請求功能呼叫使用(3/4)

- 使用非IE瀏覽器方式透過 postMessage 方式傳送資料

1. 先透過 window.open(http://localhost:61161/popupForm …)

```
function checkFinish(){
    if(postTarget){
        postTarget.close();
        alert("尚未安裝元件");
    }
}

function makeSignature(){
    var ua = window.navigator.userAgent;//取得瀏覽器代理資訊
    if(ua.indexOf("MSIE")!=-1 || ua.indexOf("Trident")!=-1){
            /*IE 使用*/
    }else{
        postTarget=window.open("http://localhost:61161/popupForm", "簽章中","height=200, width=200, left=100, top=20");//啟動彈跳視窗
        timeoutId=setTimeout(checkFinish,3500);//等待沒有回應強制關閉彈跳視窗
    }
}
```

2. 透過window.addEventListener 進行監聽訊息接收，監聽Message事件，觸發呼叫執行函式receiveMessage接收到是件訊息

```
function receiveMessage(event){
    if(console) console.debug(event);

    if(event.origin!="http://localhost:61161") //安全起見，這邊應填入網站位址檢查
        return;
    try{
        var ret = JSON.parse(event.data);//JSON to String
        if(ret.func){//取得請求方式
            if(ret.func=="getTbs"){
                clearTimeout(timeoutId);
                var json=getTbsPackage()
                postTarget.postMessage(json,"*");
            }else if(ret.func=="sign"){

                setSignature(event.data);
            }
        }else{
        if(console) console.error("no func");
        }
    }catch(e){//errorhandle
        if(console) console.error(e);
    }
}
if (window.addEventListener) {
    window.addEventListener("message", receiveMessage, false);
    }else{ //for IE8 使用
        window.attachEvent("onmessage", receiveMessage);
    }
var console=console||{"log":function(){}, "debug":function(){}, "error":function(){}};//for IE8
```

3. 當是件觸發receiveMessage函式，需要檢查http://localhost:61161 網域是否正確

```
function receiveMessage(event){
    if(console) console.debug(event);

    if(event.origin!="http://localhost:61161") //安全起見，這邊應填入網站位址檢查
        return;
    try{
        var ret = JSON.parse(event.data);//JSON to String
        if(ret.func){//取得請求方式
            /*解析請求方式 */
        }else{
        if(console) console.error("no func");
        }
    }catch(e){//errorhandle
        if(console) console.error(e);
    }
}
```

4. 判斷回應功能有getTbs、 sign、 decrypt、 pkcs11info

```
var ret = JSON.parse(event.data);//JSON to String
if(ret.func){//取得回應功能
    if(ret.func=="getTbs"){//傳送JSON內容給跨平台伺服器
        clearTimeout(timeoutId);
        var json=getTbsPackage(); //建立傳送JSON內容
        postTarget.postMessage(json,"*"); //傳送JSON內容到 http://localhost:61161/popupForm
    }else if(ret.func=="sign"){ //簽章作業
        setSignature(event.data); //解析簽章作業JSON訊息格式
    }else if(ret.func=="decrypt"){//解密作業
        setDecryptResult(event.data);//解析解密作業JSON訊息格式
    }else if(ret.func=="pkcs11info"){//讀卡機資訊或卡片憑證資訊作業
        setSlotandUserCert(event.data);//解析讀卡機資訊或卡片憑證資訊作業JSON訊息格式
    }
```

跨平台請求功能呼叫使用(4/4)

- 解析回應JSON內容

```
function setSignature(signature){
    var ret=JSON.parse(signature);
    document.getElementById("ResultSignedData").value=ret.signature; //取得簽章結果
    document.getElementById("returnCode").value=ret.ret_code; // 回傳代碼
    if(ret.ret_code!=0){//ret_code如果非0錯誤訊息表示
    alert(MajorErrorReason(ret.ret_code)); // 第一層跨停元件顯示錯誤代碼
        if(ret.last_error)
            alert(MinorErrorReason(ret.last_error)); // 第二層cryptographic顯示錯誤代碼
    }
}
```

如何實現連續簽章的範例

- 請參考此檔案連結

需要轉換字串為UTF-8 編碼注意

- 需要透過 encodeURIComponent 將特別字串符號轉換成 UTF-8 編碼進行顯示。
- 會需要使用此語法只有使用者輸入卡片 PIN碼 和 明文文字(TBS) 和 加密後的資訊(Cipher)。
- 建議寫法

```
tbsData["tbs"]=encodeURIComponent(document.getElementById("tbs").value);
tbsData["cipher"]=encodeURIComponent(document.getElementById("cipher").value.replace(/[\r\n]/g, ''));
tbsData["pin"]=encodeURIComponent(document.getElementById("pin").value);
```

簡報完畢
敬請指教

45