

CryptLib

für

Mainframe

Security Toolkit

API Handbuch

Version 2.4.2

CryptLib Programmierhandbuch

Copyright

Copyright © 1986-2008 XPS Software GmbH

Alle Rechte vorbehalten.

Warenzeichen

Java ist ein Warenzeichen von Sun Microsystems, Inc.

Windows ist ein Warenzeichen der Microsoft Corporation.

MVS, OS/390, z/OS, VSE, VSE/ESA, VM/CMS, OS/400, TSO, CICS und IMS sind Warenzeichen der IBM Corporation.

Andere in diesem Handbuch erwähnten Marken- und Produktnamen sind Warenzeichen der jeweiligen Rechtsinhaber und werden hiermit anerkannt.

Inhaltsverzeichnis

Inhaltsverzeichnis	3
Einführung	6
Installation	7
Systemvoraussetzungen	7
Ablauf der Installation	7
Installation unter MVS und OS/390	7
Verwendung der CryptLib	9
Installation unter VSE/ESA	10
Verwendung der CryptLib	10
Schlüsselgenerierung	11
Allgemein	11
Funktionen	11
GENERATE-KEY	11
GENERATE-RSA-KEY	12
Verschlüsselung	14
Allgemein	14
Funktionen	14
INIT-CTX	14
ENCRYPT	15
DECRYPT	16
GET-RESULT-LENGTH	16
RESET-CTX	17
CLEANUP-CTX	17
Digitale Signatur	22
Allgemein	22
Funktionen	22
SIGN-INIT	22
SIGN-UPDATE	23
SIGN-FINAL	23
VERIFY-INIT	23
VERIFY-UPDATE	24
VERIFY-FINAL	24
Hashfunktionen	28
Allgemein	28
Funktionen	28
DIGEST-INIT	28

DIGEST-UPDATE	29
DIGEST-FINAL.....	29
HMAC	30
X.509 Zertifikate	32
Allgemein.....	32
Funktionen	32
IMPORT-CERTIFICATE	32
GET-PUBLIC-KEY	33
GET-CRYPT-ALGO	33
GET-CRYPT-KEYLEN.....	33
GET-VERSION-INFO.....	34
GET-SERIAL-NUMBER	34
GET-ISSUER-DN.....	34
GET-SUBJECT-DN	35
GET-SIGNATURE-ALGO	35
GET-SIGNATURE	35
GET-START-DATE.....	36
GET-END-DATE.....	36
GET-ISSUER-DN-BLOB	37
GET-SUBJECT-DN-BLOB.....	37
GET-ISSUER-DN-BY-TYPE	37
GET-SUBJECT-DN-BY-TYPE.....	38
GET-FIRST-EXTENSION	39
GET-NEXT-EXTENSION.....	39
GET-EXTENSION-BY-OID	40
GET-FINGERPRINT.....	40
VERIFY-CERTIFICATE.....	40
CLEANUP-CERTIFICATE	41
S/MIME Objekte (PKCS#7)	46
Allgemein.....	46
Funktionen	46
IMPORT-PKCS7-DATA	46
IMPORT-SIGNED-DATA.....	47
IMPORT-ENVELOPED-DATA	47
IMPORT-ENCRYPTED-DATA	48
CREATE-PKCS7-DATA	48
CREATE-SIGNED-DATA	49
CREATE-ENVELOPED-DATA.....	49
CREATE-ENCRYPTED-DATA.....	50
ADD-PKCS7-DATA	51
ADD-MESSAGE-DIGEST.....	51
ADD-SIGNER.....	52
ADD-RECIPIENT.....	52

ADD-SIGNER-CERT	52
ADD-TRUSTED-SIGNER	53
FORCE-TRUSTED-SIGNER	53
GET-FIRST-SIGNER	54
GET-NEXT-SIGNER	54
GET-SIGNING-ALGO.....	55
GET-SIGNING-TIME	55
GET-NEXT-SIGNER-CERT	56
VERIFY-SIGNER	56
VERIFY-ALL-SIGNER	56
GET-FIRST-PKCS7-DATA.....	57
GET-NEXT-PKCS7-DATA	57
CREATE-OBJECT	58
CLEANUP-PKCS7	58
PKCS#12 private Key	70
Allgemein.....	70
Funktionen	70
IMPORT-PKCS12	70
GET-PRIVATE-KEY	71
GET-FIRST-CERT	71
GET-NEXT-CERT	71
CLEANUP-PKCS12	72
GZIP	74
Allgemein.....	74
Funktionen	74
GZIP	74
GUNZIP.....	74
Hilfsfunktionen	77
Allgemein.....	77
Funktionen	77
ASN2PEM	77
PEM2ASN	78
CLEANUP-PEM	78
READ-FILE	78
WRITE-FILE.....	79
CLEANUP-FILE	79
EBCDIC-TO-ASCII	79
ASCII-TO-EBCDIC	80
Nachrichten.....	81

Einführung

'Kryptographie ist die Wissenschaft, die sich mit der Absicherung von Nachrichten beschäftigt' (Zitat aus 'Angewandte Kryptographie' von Bruce Schneier, Seite 1, Addison-Wesley GmbH, 1996).

Die Entwicklung neuer Verfahren und Methoden zur Absicherung von Nachrichten wurde in der Vergangenheit vor allem durch Anforderungen im militärischen Bereich vorangetrieben. Schon die Römer setzten simple Chiffrierverfahren ein, um den Klartext von Nachrichten vor ihren Feinden zu verbergen. Die so genannte 'Caesar-Chiffrierung' ist ein Beispiel dafür.

In den letzten Jahren haben sich kryptographische Anwendungen ihren Weg in eine Vielzahl von Geschäftsbereichen gebahnt. Einer der Hauptgründe hierfür ist die Tatsache, dass die weltweite Vernetzung von Computern durch das Internet und die damit entstandenen neuen Möglichkeiten zur Kommunikation natürlich Fragen bezüglich der Sicherheit von übermittelten Informationen aufwerfen. Dabei spielen sowohl persönliche Interessen, wie sie etwa beim Online-Banking, als auch geschäftliche Interessen, wie sie etwa beim Abwickeln von Transaktionen zwischen Geschäftspartnern über das Internet auftreten, eine Rolle.

Der zunehmende Bedarf an kryptographischen Verfahren hat dazu geführt, dass die Entwicklungen auf diesem Gebiet stark vorangetrieben wurden. Die Tatsache, dass die breite Öffentlichkeit heute Zugang zu geprüften, sicheren und einfach anzuwendenden kryptographischen Verfahren hat, kann durchaus als Resultat dieser Entwicklung angesehen werden.

Private Computeranwender können Daten heute problemlos und komfortabel mit als sicher geltenden Methoden schützen. 'Sicher' bedeutet in diesem Zusammenhang, dass selbst der Einsatz von Rechenleistung, die zurzeit als unrealistisch groß angesehen werden muss, keine systematische Möglichkeit bietet, den Chiffriertext in angemessener Zeit in den Klartext zurück zu überführen. Dies kann nur gelingen, wenn eine bestimmte geheime Information bekannt ist, die als 'Schlüssel' bezeichnet wird.

Im Laufe der Zeit haben sich einige Verfahren als Standards etabliert. Dies liegt daran, dass diese Verfahren öffentlich dokumentiert und daher gut getestet und auf Sicherheit geprüft werden konnten. Dabei ist festzustellen, dass die Sicherheit dieser Verfahren allein auf der Wahl der verwendeten Schlüssel basiert. Die Kenntnis der benutzten Algorithmen schwächt die Sicherheit dieser Verfahren nicht.

Mit CryptLib bietet die XPS Software GmbH Programmierern eine Bibliothek mit standardisierten Verfahren aus der Kryptographie zur Einbindung in selbst entwickelte Applikationen an. CryptLib ist für die Betriebssysteme Win32, Linux, OS/2, OS/400, IBM iSeries, VSE/ESA, MVS/ESA, OS/390 und IBM zSeries verfügbar. Die Funktionalität erstreckt sich von Verfahren zur Hashwertbildung, symmetrischer und asymmetrischer Verschlüsselung über die Verarbeitung von X.509 Zertifikaten, die Erstellung und Prüfung digitaler Signaturen bis hin zur Unterstützung der Public Key Cryptography Standards PKCS#7 (S/MIME) und PKCS#12 (public Key).

Installation

Systemvoraussetzungen

Betriebssystemvoraussetzungen

CryptLib V1.5 läuft unter MVS/ESA ab Version 5.0, unter OS/390 ab Version 1.3, sowie unter z/OS ab Version 1.1.

Ebenso kann die VSE-Version von CryptLib ab VSE/ESA Version 2.3 eingesetzt werden.

Hardwarevoraussetzungen

CryptLib ist für die Installation auf einem IBM System/390 oder kompatibelem Prozessor ausgelegt, auf dem die unter Betriebssystemvoraussetzungen genannten benötigten Softwarekomponenten ablauffähig sind. Zum Einspielen des Installationsdatenträgers wird entweder ein Band-/Cartridgelaufwerk oder ein Client-PC mit einem CD-Rom Laufwerk und einer FTP-Anbindung an den Host benötigt.

Ablauf der Installation

§ Einspielen des Installationsdatenträgers

Installation unter MVS und OS/390

Einspielen des CryptLib-Installationsbandes

Folgende Bibliotheken sollten für die Installationsdateien vorbereitet werden:

Name	Space	Lrecl	Blksz	Recfm
XPSCRYPT.V150.LOADLIB	6144,(100,50,50)		6144	U
XPSCRYPT.V150.MACLIB	3200,(25,5,10)	80	3200	FB

Die XPSCRYPT.V150.LOADLIB enthält den ausführbaren Programmcode. In der XPSCRYPT.V150.MACLIB befinden sich die Copy-Strecken, die Beispiel-Programme, die Lizenzdatei sowie die Beispiel-Zertifikate.

Beispieljob:

```
//XPSC150 JOB , 'INSTALL CRYPTLIB' ,
//          CLASS=c,MSGCLASS=x
//TAPL     EXEC PGM=IEBCOPY
//LOADIN   DD DISP=(OLD,PASS) ,VOL=( ,RETAIN,SER=XPSC15) ,
//          LABEL=(1,SL) ,DSN=XPSC150.LOADLIB,
//          UNIT=cart
//LOADOUT  DD DISP=(NEW,CATLG) ,DSN=xpscrypt.V150.loadlib,
//          SPACE=(6144,(100,50,50) , , ,ROUND) ,DCB=SYS1.LINKLIB,
//          VOL=SER=mvs001 ,UNIT=dasd
//MACIN    DD DISP=(OLD,PASS) ,VOL=( ,RETAIN,SER=XPSC15) ,
//          LABEL=(2,SL) ,DSN=XPSC150.MACLIB,
//          UNIT=cart
//MACOUT   DD DISP=(NEW,CATLG) ,DSN=xpscrypt.V150.maclib,
//          SPACE=(3200,(25,5,10) , , ,ROUND) ,DCB=SYS1.MACLIB,
//          VOL=SER=mvs001 ,UNIT=dasd
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
COPY      INDD=LOADIN ,OUTDD=LOADOUT
COPY      INDD=MACIN ,OUTDD=MACOUT
```

Abb. 1: Installations-Job MVS

Einspielen der CryptLib-Installations-CD-Rom

Die Installationsbibliotheken auf der Installations-CD-Rom müssen mit Hilfe eines FTP-Clientprogramms zum Hostrechner übertragen werden, auf dem CryptLib installiert werden soll. Die Bibliotheken sowie ein Backup der Hilfedatei stehen im TSO-Transmit-Format auf der CD-Rom und müssen binär zum Host übertragen werden. Es ist nötig, die Dateien auf dem empfangenden Server vor der Übertragung zu allokiieren. Folgende Werte sollten dabei angegeben werden:

Name	Space	Lrecl	Blksz	Recfm
XMIT.XPSCRYPT.V150.LOADLIB	600,(100)	80	3200	FB
XMIT.XPSCRYPT.V150.MACLIB	200,(20)	80	3200	FB

Danach können folgende Dateien aus dem CD-Rom Unterverzeichnis '\MVS' vom Client zum Host gesendet werden. Dabei müssen die Dateien folgendermaßen umbenannt werden:

Clientname	Hostname
XPSC150L.BIN	XMIT.XPSCRYPT.V150.LOADLIB
XPSC150M.BIN	XMIT.XPSCRYPT.V150.MACLIB

Anschließend müssen die TSO-Transfer-Dateien durch folgende TSO-Befehle transferiert werden:

§ Für die Loadlib:

```
RECEIVE INDSN(XMIT.XPSCRYPT.V150.LOADLIB)
```

Nach Eingabe des 'RECEIVE'-Befehls erscheint folgender Prompt:

```
INMR901I Dataset XPSCRYPT.V150.LOADLIB from XPSSYST on NODENAME
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```

Hier muss der gewünschte Dateiname folgendermaßen angegeben werden:

```
DSN(xpencrypt.v150.loadlib)
```

§ Für die Maclib:

```
RECEIVE INDSN(XMIT.XPSCRYPT.V150.MACLIB)
```

Nach Eingabe des 'RECEIVE'-Befehls erscheint folgender Prompt:

```
INMR901I Dataset XPSCRYPT.V150.MACLIB from XPSSYST on NODENAME  
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```

Hier muss der gewünschte Dateiname folgendermaßen angegeben werden:

```
DSN(xpencrypt.v150.maclib)
```

Verwendung der CryptLib

Da die CryptLib ausschliesslich in 390-Assembler programmiert wurde, sind für den Einsatz keinerlei systemabhängige Voraussetzungen zu erfüllen. D. h., die CryptLib kann sowohl von Batch-Programmen als auch von CICS- oder IMS-Programmen verwendet werden. CryptLib ist von allen am Mainframe verfügbaren Programmiersprachen wie COBOL, PL/1, RPG oder Assembler aufrufbar.

Es sind lediglich folgende Job-Control Anweisungen in den Ausführungsjob hinzuzufügen:

Beispieljob:

```
//STEPLIB DD DISP=SHR,DSN=xpencrypt.v150.loadlib  
//XPDATA DD DISP=SHR,DSN=xpencrypt.v150.maclib
```

Abb. 2: Job-Control MVS

Installation unter VSE/ESA

Einspielen des CryptLib-Installationsbandes

Beispieljob:

```
// JOB XPSCRYPT      INSTALL XPS-CRYPT150
// ASSGN SYS006,tape
// EXEC LIBR
//   RESTORE SUB=XPS.CRYPTLIB:lib.sublib -
//   TAPE=SYS006 LIST=YES REPLACE=YES
/*
/ &
```

Abb. 3: Installations-Job VSE

Verwendung der CryptLib

Da die CryptLib ausschliesslich in 370-Assembler programmiert wurde, sind für den Einsatz keinerlei systemabhängige Voraussetzungen zu erfüllen. d. h., die CryptLib kann sowohl von Batch-Programmen als auch von CICS-Programmen verwendet werden. CryptLib ist von allen am Mainframe verfügbaren Programmiersprachen wie COBOL, PL/1, RPG oder Assembler aufrufbar.

Die Programmbeispiele in Assembler und COBOL sind in der Installationsbibliothek unter dem Typen „Z“ abgelegt.

Für die Ausführung ist lediglich folgende Job-Control Anweisung hinzuzufügen:

Beispieljob:

```
// LIBDEF PHASE,SEARCH=(xps.cryptlib)
```

Abb. 4: Job-Control VSE

Schlüsselgenerierung

Allgemein

Zufallszahlen spielen in der Kryptographie eine große Rolle. Die Erzeugung eines neuen Schlüssels startet sowohl bei symmetrischer als auch bei asymmetrischer Verschlüsselung mit dem Generieren einer Zufallszahl. Ist diese Zufallszahl vorhersehbar, kann mit dem entsprechenden Verfahren auch der Schlüssel berechnet werden. Von der Geheimhaltung der Schlüssel hängt das ganze Sicherheitssystem ab. Aus diesem Grund ist es möglich, durch den Parameter *seed* einen eigenen Initialisierungswert vorzugeben.

Mit der Funktion *GENERATE-RSA-KEY* kann ein RSA Public-/Private Schlüsselpaar erzeugt werden. Die Funktion *GENERATE-KEY* kann zur Erzeugung von zufälligen Schlüsseln, Initialisierungsvektoren (iv) oder sonstigen Zufallswerten benutzt werden.

Funktionen

GENERATE-KEY

Ein zufälliger Schlüssel für die symmetrische Ver- bzw. Entschlüsselung wird generiert.

Syntax	Cobol	
	<pre>MOVE GENERATE-KEY TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, KEY, KEYLEN, SEED, SEEDLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GENERATE_KEY, KEY, KEYLEN, SEED, SEEDLEN, RC), VL</pre>	
Returncode (RC)	Keiner.	
Parameter	Beschreibung	Verwendung
<i>KEY</i>	Speicheradresse des Rückgabebereiches für den zu erzeugenden symmetrischen Schlüssel.	Ausgabe
<i>KEYLEN</i>	Länge des zu erzeugenden Schlüssels.	Eingabe
<i>SEED</i>	Speicheradresse der variablen Daten, die in die Schlüsselgenerierung mit einfließen.	Eingabe
<i>SEEDLEN</i>	Länge der variablen Daten.	Eingabe

COBOL-Beispiel:

```
*-----*
*      XPS-CRYPTLIB SAMPLE GENERATE KEY      *
*-----*
ID DIVISION.
```

```

PROGRAM-ID.
  SAMPLE.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 SEED          PIC X(32)          VALUE X"0F0E0D0C0B0A09080706
- "050403020100F0E0D0C0B0A09080706050403020100" .
01 KEY          PIC X(32) .
*
77 CRYPT-FUNCTION PIC X.
77 KEYLEN        PIC 9(8)  COMP VALUE 32.
77 SEEDLEN      PIC 9(8)  COMP VALUE 32.
77 RC          PIC 9(8)  COMP VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**          PROCEDURE DIVISION          **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
* GENERATE RANDOM AES-KEY              *
*-----*
      MOVE GENERATE-KEY TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, KEY, KEYLEN, SEED, SEEDLEN, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
      STOP RUN.
ENDRUN.

```

GENERATE-RSA-KEY

Ein zufälliges RSA Public-/Private-Key Paar wird generiert.

Syntax	Cobol	
	MOVE GENERATE-RSA-KEY TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, PRIVKEY, PUBKEY, SEED, SEEDLEN, KEYLEN, RC.	
	Assembler	
	CALL XPSCRYPT, (GENERATE_RSA_KEY, PRIVKEY, PUBKEY, SEED, SEEDLEN, KEYLEN, RC), VL	
Returncode (RC)	Keiner.	
Parameter	Beschreibung	Verwendung
<i>PRIVKEY</i>	Speicheradresse des Rückgabebereiches für den zu erzeugenden privaten RSA Schlüssel.	Ausgabe
<i>PUBKEY</i>	Speicheradresse des Rückgabebereiches für den zu erzeugenden öffentlichen RSA Schlüssel.	Ausgabe
<i>SEED</i>	Speicheradresse der variablen Daten, die in die Schlüsselgenerierung mit einfließen.	Eingabe
<i>SEEDLEN</i>	Länge der variablen Daten.	Eingabe
<i>KEYLEN</i>	Länge des zu erzeugenden Schlüssels (maximal 4096).	Eingabe

COBOL-Beispiel:

```

*-----*
* XPS-CRYPTLIB SAMPLE GENERATE RSA-KEY *
*-----*
ID DIVISION.
PROGRAM-ID.
  SAMPLE.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 SEED          PIC X(32)          VALUE X"0F0E0D0C0B0A09080706

```

```
- "0504030201000F0E0D0C0B0A09080706050403020100".
*
77 CRYPT-FUNCTION PIC X.
77 KEYLEN          PIC 9(8) COMP VALUE 1024.
77 SEEDLEN        PIC 9(8) COMP VALUE 32.
77 RC              PIC 9(8) COMP VALUE ZEROES.
*
COPY XPSCLRSA.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**                PROCEDURE DIVISION                **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* GENERATE RANDOM RSA-KEY                               *
* ----- *
      MOVE GENERATE-RSA-KEY TO CRYPT-FUNCTION.
      CALL 'XPCRYPT' USING CRYPT-FUNCTION, RSA-PRIVATE-KEY,
        RSA-PUBLIC-KEY, SEED, SEEDLEN, KEYLEN, RC.
      IF RC < 0
        DISPLAY RC
        GOBACK.
      STOP RUN.
ENDRUN. }
```

Verschlüsselung

Allgemein

Man unterscheidet zwei Arten von Verschlüsselungsverfahren:

Symmetrische Verfahren

Bei symmetrischen Verfahren wird zum Verschlüsseln und zum Entschlüsseln der gleiche Schlüssel verwendet. Daraus folgt, dass Sender und Empfänger den gleichen Schlüssel verwenden müssen. Verwendet man den Operationsmodus CBC, so benötigt der Chiffre zusätzlich einen so genannten Initialisierungsvektor (iv). Der Initialisierungsvektor ist immer genau so groß wie die Blocklänge des Chiffre (DES, TripleDES, RC2, RC4, Blowfish = 8 Byte, AES = 16 Byte).

Asymmetrische Verfahren (PublicKey-Verfahren)

Sind beide Schlüssel verschieden, spricht man von einem asymmetrischen Verfahren. Ein Schlüssel wird zum Verschlüsseln, der andere zum Entschlüsseln verwendet. Beide Schlüssel werden gemeinsam bei der Schlüsselgenerierung erzeugt und es ist unmöglich, von einem der beiden Schlüssel auf den anderen zu schließen. Aus diesem Grund ist es zulässig, einen der beiden Schlüssel öffentlich bekannt zu geben (PublicKey). Mit diesem öffentlichen Schlüssel ist man dazu in der Lage, dem Besitzer des Schlüsselpaares eine verschlüsselte Nachricht zu senden. Dieser verwendet dann den geheimen Schlüssel (PrivateKey) um die Nachricht zu entschlüsseln.

CryptLib stellt eine Reihe von symmetrischen Verschlüsselungsroutinen zur Verfügung (**AES, DES, TripleDES, RC2, RC4, Blowfish**). Außerdem ist die asymmetrische Verschlüsselungsroutine **RSA** Public-/Private-Key implementiert. Die Art der gewünschten Ver-/Entschlüsselung wird beim Initialisieren der Kryptokontexts festgelegt (Funktion *INIT-CTX*). Danach ist nur noch eine der für jede Verschlüsselungsart identischen Routinen *ENCRYPT* bzw. *DECRYPT* aufzurufen, um die Ver-/Entschlüsselung durchzuführen.

Funktionen

INIT-CTX

Initialisierung des Kryptokontexts.

Syntax	Cobol
	<pre>MOVE INIT-CTX TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, KEY, RC.</pre>

	Assembler	
	CALL XPSCRIPT , (INIT_CTX , CTX , KEY , RC) , VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Ver-/Entschlüsseln benötigten Kontexts. Die Copy-Strecke dieser Struktur ist in dem Copybuch <i>XPSCCLCTX</i> bzw. <i>XPSCCLASM</i> (COBOL/Assembler) enthalten.	Eingabe
Felder	Beschreibung	
<i>CX-ALGO</i>	Algorithmus zum Chiffrieren der Daten. Folgende Algorithmen werden unterstützt: AES Advanced Encryption Standard (Rijndael) DES Data Encryption Standard mit ctx.keylength = 56 TripleDES EDE2 Data Encryption Standard mit ctx.keylength = 112 TripleDES EDE3 Data Encryption Standard mit ctx.keylength = 168 RC2 Rivest Cipher No. 2 RC4 Rivest Cipher No. 4 Blowfish Schneier RSA Public-/Private-Key Verfahren von Rivest, Shamir und Adleman	
<i>CX-MODE</i>	Verarbeitungsmodus. Unterstützte Modi bei symmetrischer Verschlüsselung (AES, DES, RC2, RC4, Blowfish): ECB Electronic-Codebook-Mode CBC Cipher-Block-Chaining Unterstützte Modi bei asymmetrischer Verschlüsselung (RSA): PUBLIC (Public-Key Ver-/Entschlüsselung) unter <i>ctx.key</i> muss die Adresse einer RSA_PUBLIC_KEY Struktur übergeben werden PRIVATE (Public-Key Ver-/Entschlüsselung) unter <i>ctx.key</i> muss die Adresse einer RSA_PRIVATE_KEY Struktur übergeben werden	
<i>CX-KLEN</i>	Länge des Schlüssels. Folgende Schlüssellängen werden unterstützt: AES 128, 192, 256 DES 56, 112, 168 RC2 40, 64, 128 RC4 40, 64, 128 Blowfish 128 RSA 512, 1024, 2048, 4096	
<i>CX-IV</i>	Speicheradresse des Initialisierungs-Vektors (iv) bei symmetrischer Verschlüsselung.	
Parameter	Beschreibung	Verwendung
<i>KEY</i>	Speicheradresse des Schlüssels.	Eingabe

ENCRYPT

Verschlüsseln von Daten. Die Art der Verschlüsselung ist abhängig vom Parameter *CX-ALGO* bei der Funktion *INIT-CTX*.

Syntax	Cobol	
	MOVE ENCRYPT TO CRYPT-FUNCTION.	
	CALL 'XPSCRIPT' USING CRYPT-FUNCTION, CTX, INPUT, INPUTLEN, OUTPUT, OUTPUTLEN, RC.	
	Assembler	

	CALL XPCRYPT, (ENCRYPT, CTX, INPUT, INPUTLEN, OUTPUT, OUTPUTLEN, RC), VL	
Returncode (RC)	Länge der verschlüsselten Daten oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Verschlüsseln benötigten Kontexts.	Eingabe
<i>INPUT</i>	Speicheradresse der zu verschlüsselnden Daten.	Eingabe
<i>INPUTLEN</i>	Länge der zu verschlüsselnden Daten.	Eingabe
<i>OUTPUT</i>	Speicheradresse des Rückgabebereiches für die verschlüsselten Daten.	Ausgabe
<i>OUTPUTLEN</i>	Länge des unter <i>OUTPUT</i> angegebenen Speicherbereiches.	Ausgabe

DECRYPT

Entschlüsseln von Daten. Die Art der Entschlüsselung ist abhängig vom Parameter *CX-ALGO* bei der Funktion *INIT-CTX*.

Syntax	Cobol	
	MOVE DECRYPT TO CRYPT-FUNCTION. CALL 'XPCRYPT' USING CRYPT-FUNCTION, CTX, INPUT, INPUTLEN, OUTPUT, OUTPUTLEN, RC.	
	Assembler	
	CALL XPCRYPT, (DECRYPT, INPUT, INPUTLEN, OUTPUT, OUTPUTLEN, RC), VL	
Returncode (RC)	Länge der entschlüsselten Daten oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Entschlüsseln benötigten Kontexts.	Eingabe
<i>INPUT</i>	Speicheradresse der zu entschlüsselnden Daten.	Eingabe
<i>INPUTLEN</i>	Länge der zu entschlüsselnden Daten.	Eingabe
<i>OUTPUT</i>	Speicheradresse des Rückgabebereiches für die entschlüsselten Daten.	Ausgabe
<i>OUTPUTLEN</i>	Länge des unter <i>OUTPUT</i> angegebenen Speicherbereiches.	Ausgabe

GET-RESULT-LENGTH

Ermitteln des Speicherbedarfs für die zu ver-/entschlüsselnden Daten.
 Hinweis: Bei Verwendung von RSA ist diese Funktion nur zur Ermittlung der Ausgabelänge für die Verschlüsselung möglich. Bei der Entschlüsselung ist die Ausgabelänge immer kleiner oder gleich der Eingabelänge!

Syntax	Cobol	
	MOVE GET-RESULT-LENGTH TO CRYPT-FUNCTION. CALL 'XPCRYPT' USING CRYPT-FUNCTION, CTX, INPUTLEN, RC.	
	Assembler	
	CALL XPCRYPT, (GET_RESULT_LENGTH, CTX, INPUTLEN, RC), VL	
Returncode (RC)	Länge der ver-/entschlüsselten Daten oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Ver-/Entschlüsseln benötigten Kontexts.	Eingabe
<i>INPUTLEN</i>	Länge der zu ver-/entschlüsselnden Daten.	Ausgabe

RESET-CTX

Rücksetzen des Initialisierungs-Vektors (iv) bei symmetrischer Ver-/Entschlüsselung.

Syntax	Cobol	
	MOVE RESET-CTX TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, RC.	
	Assembler	
	CALL XPSCRYPT, (RESET_CTX, CTX, RC), VL	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des Kontexts.	Eingabe

CLEANUP-CTX

Freigeben des bei den Ver-/Entschlüsselungsaktionen benötigten Speichers.

Syntax	Cobol	
	MOVE CLEANUP-CTX TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, RC.	
	Assembler	
	CALL XPSCRYPT, (CLEANUP_CTX, CTX, RC), VL	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des Kontexts.	Eingabe

COBOL-Beispiel (AES):

```

*-----*
*       XPS-CRYPTLIB SAMPLE PROGRAM AES-ENCRYPTION       *
*-----*
ID DIVISION.
PROGRAM-ID.
    AESTSTC.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01  BLOCK1          PIC X(32)          VALUE "XPS Software GmbH, Ha
-                               "ar/Muenchen".
01  OUT             PIC X(32)          VALUE SPACES.
01  OUT2            PIC X(32)          VALUE SPACES.
01  SEED1           PIC X(32)          VALUE X"0F0E0D0C0B0A09080706
-   "0504030201000F0E0D0C0B0A09080706050403020100".
01  SEED2           PIC X(32)          VALUE X"00010203040506070809
-   "0A0B0C0D0E0F".
01  AESKEY          PIC X(32).
01  AESIV           PIC X(16).
COPY XPSCLCTX.
*
77  CRYPT-FUNCTION  PIC X.
77  KEYLEN          PIC 9(8)          COMP VALUE 32.
77  IVLEN           PIC 9(8)          COMP VALUE 16.
77  BLOCKLEN       PIC 9(8)          COMP VALUE 32.
77  OUTLEN         PIC 9(8)          COMP VALUE 32.
77  OUTLEN2        PIC 9(8)          COMP VALUE 32.
77  SEEDLEN        PIC 9(8)          COMP VALUE 16.
77  RC             PIC 9(8)          COMP VALUE ZEROES.

```

```

*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**          PROCEDURE DIVISION          **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
* GENERATE RANDOM AES-KEY                *
*-----*
      MOVE GENERATE-KEY TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, AESKEY, KEYLEN, SEED1, SEEDLEN, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
*-----*
* GENERATE RANDOM IV (INITIALIZATION VECTOR FOR CIPHER-CBC) *
*-----*
      MOVE GENERATE-KEY TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, AESIV, IVLEN, SEED2, SEEDLEN, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
*-----*
* FILL CIPHER-CONTEXT AND INIT CONTEXT   *
*-----*
      MOVE AES TO CX-ALGO.
      MOVE CBC TO CX-MODE.
      MOVE 256 TO CX-KLEN.
      MOVE AESIV TO CX-IV.
      MOVE INIT-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, AESKEY, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
*-----*
* ENCRYPT DATA                          *
*-----*
      MOVE ENCRYPT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, BLOCK1, BLOCKLEN,
          OUT, OUTLEN, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
*-----*
* RESET CONTEXT (CHANGE FROM ENCRYPT TO DECRYPT) *
*-----*
      MOVE RESET-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
*-----*
* DECRYPT DATA                          *
*-----*
      MOVE DECRYPT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, OUT, OUTLEN,
          OUT2, OUTLEN2, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
*-----*
* CLEANUP CONTEXT                       *
*-----*
      MOVE CLEANUP-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, RC.
      STOP RUN.
ENDRUN.

```

COBOL-Beispiel (TripleDES):

```

*-----*
*          XPS-CRYPTLIB SAMPLE PROGRAM DES-ENCRYPTION          *
*-----*
ID DIVISION.
PROGRAM-ID.
    DESTSTC.
*

```

```

DATA DIVISION.
WORKING-STORAGE SECTION.
01 BLOCK1          PIC X(32)          VALUE "XPS Software GmbH, Ha
-                                     "ar/Muenchen".
01 OUT             PIC X(32)          VALUE SPACES.
01 OUT2           PIC X(32)          VALUE SPACES.
01 DESKEY         PIC X(24)          VALUE X"0F0E0D0C0B0A09080706
- "0504030201000F0E0D0C0B0A0908".
01 DESIV         PIC X(8)           VALUE X"0001020304050607".
COPY XPSCLCTX.
*
77 CRYPT-FUNCTION PIC X.
77 KEYLEN        PIC 9(8)          COMP VALUE 24.
77 IVLEN         PIC 9(8)          COMP VALUE 8.
77 BLOCKLEN     PIC 9(8)          COMP VALUE 32.
77 OUTLEN       PIC 9(8)          COMP VALUE 32.
77 OUTLEN2      PIC 9(8)          COMP VALUE 32.
77 RC           PIC 9(8)          COMP VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**                PROCEDURE DIVISION                **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* FILL CIPHER-CONTEXT AND INIT CONTEXT                *
* ----- *
      MOVE DES TO CX-ALGO.
      MOVE CBC TO CX-MODE.
      MOVE 168 TO CX-KLEN.
      MOVE DESIV TO CX-IV.
      MOVE INIT-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, DESKEY, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* ENCRYPT DATA                                        *
* ----- *
      MOVE ENCRYPT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, BLOCK1, BLOCKLEN,
          OUT, OUTLEN, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* RESET CONTEXT (CHANGE FROM ENCRYPT TO DECRYPT)      *
* ----- *
      MOVE RESET-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* DECRYPT DATA                                        *
* ----- *
      MOVE DECRYPT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, OUT, OUTLEN,
          OUT2, OUTLEN2, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* CLEANUP CONTEXT                                    *
* ----- *
      MOVE CLEANUP-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, RC.
      STOP RUN.
ENDRUN.

```

COBOL-Beispiel (RSA):

```

*-----*
* XPS-CRYPTLIB SAMPLE PROGRAM RSA-ENCRYPTION        *
*-----*
ID DIVISION.
PROGRAM-ID.
RSATSTC.
*

```

```

DATA DIVISION.
WORKING-STORAGE SECTION.
01 BLOCK1          PIC X(32)          VALUE "XPS Software GmbH, Ha
-                                     "ar/Muenchen".
01 OUT              PIC X(128)        VALUE SPACES.
01 OUT2             PIC X(128)        VALUE SPACES.
01 SEED             PIC X(32)         VALUE X"0F0E0D0C0B0A09080706
-                                     "0504030201000F0E0D0C0B0A09080706050403020100".
COPY XPSCLCTX.
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION  PIC X.
77 KEYLEN          PIC 9(8)  COMP  VALUE 512.
77 BLOCKLEN        PIC 9(8)  COMP  VALUE 32.
77 OUTLEN          PIC 9(8)  COMP  VALUE 128.
77 OUTLEN2         PIC 9(8)  COMP  VALUE 128.
77 SEEDLEN         PIC 9(8)  COMP  VALUE 32.
77 RC              PIC 9(8)  COMP  VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**                PROCEDURE DIVISION                **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* GENERATE RANDOM RSA-KEY *
* ----- *
      MOVE GENERATE-RSA-KEY TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, RSA-PRIVATE-KEY,
          RSA-PUBLIC-KEY, SEED, SEEDLEN, KEYLEN, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* FILL CIPHER-CONTEXT AND INIT CONTEXT *
* ----- *
      MOVE RSA      TO CX-ALGO.
      MOVE PUBLIC TO CX-MODE.
      MOVE KEYLEN TO CX-KLEN.
      MOVE INIT-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, RSA-PUBLIC-KEY, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* ENCRYPT DATA *
* ----- *
      MOVE ENCRYPT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, BLOCK1, BLOCKLEN,
          OUT, OUTLEN, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* ENCRYPT DATA *
* ----- *
      MOVE CLEANUP-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
      MOVE RSA      TO CX-ALGO.
      MOVE PRIVATE TO CX-MODE.
      MOVE KEYLEN TO CX-KLEN.
      MOVE INIT-CTX TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, RSA-PRIVATE-KEY, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* RESET CONTEXT (CLEANUP OLD CONTEXT AND INIT NEW CONTEXT) *
* (CHANGE FROM PUBLIC-KEY TO PRIVATE-KEY) *
* ----- *
      MOVE DECRYPT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, CIPHER-CTX, OUT, OUTLEN,
          OUT2, OUTLEN2, RC.
      IF RC < 0
          DISPLAY RC
          GOBACK.
* ----- *
* CLEANUP CONTEXT *
* ----- *
      MOVE CLEANUP-CTX TO CRYPT-FUNCTION.

```

```
CALL 'XPCRYPT'  
  USING CRYPT-FUNCTION, CIPHER-CTX, RC.  
STOP RUN.  
ENDRUN.
```

Digitale Signatur

Allgemein

Eine der wichtigsten Anwendungsmöglichkeiten asymmetrischer Verschlüsselungsverfahren ist die Implementierung digitaler Signaturen. Dabei wird aus den Daten, die elektronisch unterschrieben werden sollen, eine charakteristische Prüfsumme (Hashwert) gebildet und diese dann mit dem geheimen Schlüssel verschlüsselt. Der so entstandene Schlüsseltext kann mit dem dazugehörigen öffentlichen Schlüssel wieder entschlüsselt werden.

CryptLib unterstützt digitale Signaturen mit dem Algorithmus **RSA**. Als Hashfunktion können **MD2**, **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-386**, **SHA-512** und **RipeMD160** benutzt werden.

Die Prozedur zum Erzeugen einer digitalen Signatur umfasst folgende Schritte:

- § Initialisieren des Kontexts mit der Funktion *SIGN-INIT* zur Festlegung des Hashtyps.
- § Hinzufügen der zu signierenden Daten mit der Funktion *SIGN-UPDATE*. Diese Aktion kann beliebig oft ausgeführt werden.
- § Zum Abschluss muss die Funktion *SIGN-FINAL* aufgerufen werden, die die digitale Signatur zurückgibt.

Die Prozedur zum Verifizieren einer digitalen Signatur umfasst folgende Schritte:

- § Initialisieren des Kontexts mit der Funktion *VERIFY-INIT* zur Festlegung des Hashtyps.
- § Hinzufügen der zu prüfenden Daten mit der Funktion *VERIFY-UPDATE*. Diese Aktion kann beliebig oft ausgeführt werden.
- § Zum Abschluss muss die Funktion *VERIFY-FINAL* aufgerufen werden, die die digitale Signatur verifiziert.

Funktionen

SIGN-INIT

Initialisierung des Signaturkontexts.

Syntax	Cobol
	MOVE SIGN-INIT TO CRYPT-FUNCTION.

	CALL 'XPSCRYPT' USING CRYPT-FUNCTION, HASHALGO, CTX, RC.	
	Assembler	
	CALL XPSCRYPT, (SIGN_INIT, HASHALGO, CTX, RC), VL	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>HASHALGO</i>	Der zu verwendende Hashtyp. Unterstützt werden die Hashtypen MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 und RipeMD160.	Eingabe
<i>CTX</i>	Adresspointer zur Aufnahme der Speicheradresse des erstellten Kontexts. Dieser wird für die weitere Bearbeitung benötigt.	Ausgabe

SIGN-UPDATE

Hinzufügen der zu signierenden Daten.

Syntax	Cobol	
	MOVE SIGN-UPDATE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, DATA, DATALEN, RC.	
	Assembler	
	CALL XPSCRYPT, (SIGN_UPDATE, CTX, DATA, DATALEN, RC), VL	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Signieren benötigten Kontexts.	Eingabe
<i>DATA</i>	Speicheradresse der zu signierenden Daten.	Eingabe
<i>DATALEN</i>	Länge der zu signierenden Daten.	Eingabe

SIGN-FINAL

Erstellen der digitalen Signatur mit Hilfe des privaten Schlüssels.

Syntax	Cobol	
	MOVE SIGN-FINAL TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, SIGN, SIGNLEN, PRIVKEY, RC.	
	Assembler	
	CALL XPSCRYPT, (SIGN_FINAL, CTX, SIGN, SIGNLEN, PRIVKEY, RC), VL	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Signieren benötigten Kontexts.	Eingabe
<i>SIGN</i>	Speicheradresse des Rückgabebereichs für die erstellte Signatur.	Ausgabe
<i>SIGNLEN</i>	Speicheradresse für die Rückgabe der Länge der erstellten Signatur.	Ausgabe
<i>PRIVKEY</i>	Privater RSA Schlüssel des Unterzeichners.	Eingabe

VERIFY-INIT

Initialisierung des Verifizierungskontexts.

Syntax	Cobol	
	<pre>MOVE VERIFY-INIT TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, HASHALGO, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (VERIFY-INIT, HASHALGO, CTX, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>HASHALGO</i>	Der zu verwendende Hashtyp. Unterstützt werden die Hashtypen MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 und RipeMD160.	Eingabe
<i>CTX</i>	Adresspointer zur Aufnahme der Speicheradresse des erstellten Kontexts. Dieser wird für die weitere Bearbeitung benötigt.	Ausgabe

VERIFY-UPDATE

Hinzufügen der zu verifizierenden Daten.

Syntax	Cobol	
	<pre>MOVE VERIFY-UPDATE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, DATA, DATALEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (VERIFY_UPDATE, CTX, DATA, DATALEN, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Verifizieren benötigten Kontexts.	Eingabe
<i>DATA</i>	Speicheradresse der zu verifizierenden Daten.	Eingabe
<i>DATALEN</i>	Länge der zu verifizierenden Daten.	Eingabe

VERIFY-FINAL

Prüfen der digitalen Signatur mit Hilfe des öffentlichen Schlüssels.

Syntax	Cobol	
	<pre>MOVE VERIFY-FINAL TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, SIGN, SIGNLEN, PUBKEY, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (VERIFY_FINAL, CTX, SIGN, SIGNLEN, PUBKEY, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Verifizieren benötigten Kontexts.	Eingabe
<i>SIGN</i>	Speicheradresse der Signatur.	Eingabe
<i>SIGNLEN</i>	Länge der Signatur.	Eingabe

PUBKEY	Öffentlicher RSA Schlüssel des Unterzeichners.	Eingabe
--------	--	---------

COBOL-Beispiel:

```

*-----*
*       XPS-CRYPTLIB SAMPLE PROGRAM: DIGITALE SIGNATURE       *
*-----*
ID DIVISION.
PROGRAM-ID.
        SIGNTSTC.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)          VALUE "XPSDATA".
01 PKCS12-FILE     PIC X(8)          VALUE "XPSUSERP".
01 PWD             PIC X(8)          VALUE "xpsuser1".
01 DATA1          PIC X(32)         VALUE "XPS Software GmbH, Haar
-                                     "/Muenchen".
01 DATA2          PIC X(21)         VALUE "Muenchener Strasse 17".
01 HASHSHA1        PIC 9(8) COMP     VALUE 26.
01 HASHMD5         PIC 9(8) COMP     VALUE 5.
01 SIGNATURE       PIC X(128).
01 PKCS12-CTX      POINTER.
01 CERT-CTX        POINTER.
01 SIGN-CTX        POINTER.
01 ADDR-PKCS12     POINTER.
01 ADDR-CERT       POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION  PIC X.
77 DATALEN        PIC 9(8) COMP     VALUE ZEROES.
77 DATALEN1       PIC 9(8) COMP     VALUE 32.
77 DATALEN2       PIC 9(8) COMP     VALUE 21.
77 PWDLEN          PIC 9(8) COMP     VALUE 8.
77 RC              PIC S9(8) COMP     VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**          PROCEDURE DIVISION          **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
* READ PKCS-12 FILE "XPSUSERP" FROM MACLIB *
*-----*
        MOVE READ-FILE TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
                DDNAME, PKCS12-FILE, ADDR-PKCS12, DATALEN, RC.
        IF RC < 0
            DISPLAY "FILE 'XPSUSERP' NOT FOUND RC = " RC
            GOBACK.
*-----*
* CONVERT PASSWORD FROM EBCDIC TO ASCII (PASSWORD LOWER CASE|) *
*-----*
        MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT'
                USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
*-----*
* IMPORT PKCS-12 FILE *
*-----*
        MOVE IMPORT-PKCS12 TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS12,
                DATALEN, PWD, PWDLEN, PKCS12-CTX, RC.
        IF RC < 0
            DISPLAY "ERROR IMPORT-FILE: RC = " RC
            GOBACK.
*-----*
* GET PRIVATE-KEY FROM PKCS-12 FILE *
*-----*
        MOVE GET-PRIVATE-KEY TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
                PKCS12-CTX, RSA-PRIVATE-KEY, RC.
        IF RC < 0
            DISPLAY "ERROR GET-PRIVATE-KEY: RC = " RC
            GOBACK.
*-----*
* GET CERTIFICATE FROM PKCS-12 FILE *
*-----*
        MOVE GET-FIRST-CERT TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
                PKCS12-CTX, ADDR-CERT, RC.
        IF RC < 0
            DISPLAY "ERROR GET-FIRST-CERT: RC = " RC

```

```

      GOBACK.
* -----*
* IMPORT X.509 CERTIFICATE*
* -----*
      MOVE IMPORT-CERTIFICATE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           ADDR-CERT, CERT-CTX, RC.
      IF RC < 0
         DISPLAY "ERROR IMPORT-CERTIFICATE: RC = " RC
         GOBACK.
* -----*
* GET PUBLIC-KEY FROM CERTIFICATE*
* -----*
      MOVE RSA-PUBLIC-LEN TO DATALEN
      MOVE GET-PUBLIC-KEY TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           CERT-CTX, RSA-PUBLIC-KEY, DATALEN, RC.
      IF RC < 0
         DISPLAY "ERROR GET-PUBLIC-KEY: RC = " RC
         GOBACK.
* -----*
* CLEANUP CERTIFICATE CONTEXT*
* -----*
      MOVE CLEANUP-CERTIFICATE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           CERT-CTX, RC.
      IF RC < 0
         DISPLAY "ERROR CLEANUP-CERTIFICATE: RC = " RC
         GOBACK.
* -----*
* CLEANUP PKCS-12 CONTEXT*
* -----*
      MOVE CLEANUP-PKCS12 TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
           USING CRYPT-FUNCTION, PKCS12-CTX, RC.
* -----*
* RELEASE FILE-STORAGE "XPSTESTP"*
* -----*
      MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           ADDR-PKCS12, RC.
      IF RC < 0
         DISPLAY "ERROR RELEASE-FILE: RC = " RC
         GOBACK.
* -----*
* INITIALIZE SIGNATURE-CONTEXT*
* -----*
      MOVE SIGN-INIT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           HASHSHA1, SIGN-CTX, RC.
      IF RC < 0
         DISPLAY "ERROR SIGN-INIT: RC = " RC
         GOBACK.
* -----*
* UPDATE DATA*
* -----*
      MOVE SIGN-UPDATE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           SIGN-CTX, DATA1, DATALEN1, RC.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           SIGN-CTX, DATA2, DATALEN2, RC.
* -----*
* FINALIZE SIGNATURE*
* -----*
      MOVE SIGN-FINAL TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           SIGN-CTX, SIGNATURE, DATALEN, RSA-PRIVATE-KEY, RC.
      IF RC < 0
         DISPLAY "ERROR SIGN-FINAL: RC = " RC
         GOBACK.
* -----*
* INITIALIZE VERIFY-CONTEXT*
* -----*
      MOVE VERIFY-INIT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           HASHSHA1, SIGN-CTX, RC.
      IF RC < 0
         DISPLAY "ERROR VERIFY-INIT: RC = " RC
         GOBACK.
* -----*
* UPDATE DATA*
* -----*
      MOVE VERIFY-UPDATE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           SIGN-CTX, DATA1, DATALEN1, RC.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           SIGN-CTX, DATA2, DATALEN2, RC.
* -----*
* FINALIZE SIGNATURE*
* -----*

```

```
MOVE VERIFY-FINAL TO CRYPT-FUNCTION.  
CALL 'XPCRYPT' USING CRYPT-FUNCTION,  
SIGN-CTX, SIGNATURE, DATALEN, RSA-PUBLIC-KEY, RC.  
STOP RUN.  
ENDRUN.
```

6

Hashfunktionen

Allgemein

Hashfunktionen spielen im Zusammenhang mit Sicherheitsverfahren eine sehr wichtige Rolle. Sie werden immer dann benötigt, wenn aus Eingabedaten beliebiger Länge ein eindeutiger Hashwert erzeugt werden soll. Diese Prüfsumme wird dann zur Überprüfung der Unversehrtheit von Daten verwendet.

CryptLib unterstützt die Message Digest Funktionen **MD2**, **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-386**, **SHA-512** und **RipeMD160**. Außerdem wird **HMAC** (Keyed-Hashing for Message Authentication) unterstützt.

Die Prozedur zum Erzeugen eines Hashwerts umfasst folgende Schritte:

- § Initialisieren des Kontexts mit der Funktion *DIGEST-INIT* zur Festlegung des Hashtyps.
- § Hinzufügen der zu hashenden Daten mit der Funktion *DIGEST-UPDATE*. Diese Aktion kann beliebig oft ausgeführt werden.
- § Zum Abschluss muss die Funktion *DIGEST-FINAL* aufgerufen werden, die den Hashwert zurückgibt.

Funktionen

DIGEST-INIT

Initialisierung des Signaturkontexts.

Syntax	Cobol	
	<pre>MOVE DIGEST-INIT TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, HASHALGO, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (DIGEST_INIT, HASHALGO, CTX, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>HASHALGO</i>	Der zu verwendende Hashtyp. Unterstützt werden die Hashtypen MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 und RipeMD160.	Eingabe
<i>CTX</i>	Speicheradresse des Rückgabebereichs für den zum Hashen benötigten Kontext.	Ausgabe

DIGEST-UPDATE

Hinzufügen der zu hashenden Daten.

Syntax	Cobol	
	<pre>MOVE DIGEST-UPDATE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, DATA, DATALEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (DIGEST_UPDATE, CTX, DATA, DATALEN, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Hashen benötigten Kontexts.	Eingabe
<i>DATA</i>	Speicheradresse der zu hashenden Daten.	Eingabe
<i>DATALEN</i>	Länge der zu hashenden Daten.	Eingabe

DIGEST-FINAL

Erstellen des Hashwertes.

Syntax	Cobol	
	<pre>MOVE DIGEST-FINAL TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, HASHDATA, HASHLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (DIGEST_FINAL, CTX, HASHDATA, HASHLEN, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des zum Hashen benötigten Kontexts.	Eingabe
<i>HASHDATA</i>	Speicheradresse des Rückgabebereiches für den zu erstellenden Hashwert.	Ausgabe
<i>HASHLEN</i>	Speicheradresse für die Rückgabe der Länge des erstellten Hashwerts.	Ausgabe

Cobol-Beispiel:

```
*-----*
*       XPS-CRYPTLIB SAMPLE PROGRAM: HASH-ROUTINES       *
*-----*
ID DIVISION.
PROGRAM-ID.
      HASHTSTC.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01  DATA1          PIC X(32)          VALUE "XPS Software GmbH, Haar
-                               "/Muenchen".
01  DATA2          PIC X(21)          VALUE "Muenchener Strasse 17".
01  TESTKEY         PIC X(7)           VALUE "testkey".
01  HASHSHA1        PIC 9(8) COMP VALUE 26.
01  HASHMD5         PIC 9(8) COMP VALUE 5.
01  DIGEST-MD5      PIC X(16).
01  DIGEST-SHA1     PIC X(20).
01  DIGEST-HMAC     PIC X(20).
01  HASHCTX        POINTER.
```

```

*
77 CRYPT-FUNCTION PIC X.
77 DATALEN1 PIC 9(8) COMP VALUE 32.
77 DATALEN2 PIC 9(8) COMP VALUE 21.
77 KEYLEN PIC 9(8) COMP VALUE 7.
77 HASHMD5-LEN PIC 9(8) COMP VALUE 16.
77 HASHSHA1-LEN PIC 9(8) COMP VALUE 20.
77 RC PIC S9(8) COMP VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
** PROCEDURE DIVISION **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* INITIALIZE HASH-CONTEXT MD5 *
* ----- *
MOVE DIGEST-INIT TO CRYPT-FUNCTION.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
HASHMD5, HASHCTX, RC.
IF RC < 0
DISPLAY "ERROR DIGEST-INIT: RC = " RC
GOBACK.
* ----- *
* UPDATE DATA *
* ----- *
MOVE DIGEST-UPDATE TO CRYPT-FUNCTION.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
HASHCTX, DATA1, DATALEN1, RC.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
HASHCTX, DATA2, DATALEN2, RC.
* ----- *
* FINALIZE HASH *
* ----- *
MOVE DIGEST-FINAL TO CRYPT-FUNCTION.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
HASHCTX, DIGEST-MD5, HASHMD5-LEN, RC.
IF RC < 0
DISPLAY "ERROR DIGEST-FINAL: RC = " RC
GOBACK.
* ----- *
* INITIALIZE HASH-CONTEXT SHA-1 *
* ----- *
MOVE DIGEST-INIT TO CRYPT-FUNCTION.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
HASHSHA1, HASHCTX, RC.
IF RC < 0
DISPLAY "ERROR DIGEST-INIT: RC = " RC
GOBACK.
* ----- *
* UPDATE DATA *
* ----- *
MOVE DIGEST-UPDATE TO CRYPT-FUNCTION.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
HASHCTX, DATA1, DATALEN1, RC.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
HASHCTX, DATA2, DATALEN2, RC.
* ----- *
* FINALIZE HASH *
* ----- *
MOVE DIGEST-FINAL TO CRYPT-FUNCTION.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
HASHCTX, DIGEST-SHA1, HASHSHA1-LEN, RC.
IF RC < 0
DISPLAY "ERROR DIGEST-FINAL: RC = " RC
GOBACK.
* ----- *
* HMAC *
* ----- *
MOVE HMAC TO CRYPT-FUNCTION.
CALL 'XPSCRYPT' USING CRYPT-FUNCTION, DATA1, DATALEN1,
TESTKEY, KEYLEN, DIGEST-HMAC, HASHSHA1, RC.
IF RC < 0
DISPLAY "ERROR HMAC: RC = " RC
GOBACK.
STOP RUN.
ENDRUN.

```

HMAC

Erstellen eines MAC. Ein Message-Authentication-Code oder MAC ist eine schlüsselabhängige Einweg-Hashfunktion. Daher kann der MAC nur mit Hilfe eines Schlüssels erzeugt oder verifiziert werden. Dadurch wird verhindert, dass ein Unbefugter eine durch einen MAC geschützte Nachricht verändert und

den Hashwert neu berechnet. Ein MAC dient somit zur Sicherstellung von Datenintegrität ohne die Daten verschlüsseln zu müssen.

Syntax	Cobol	
	<pre>MOVE HMAC TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, DATA, DATALEN, PWD, PWDLEN, HMAC, HASHALGO, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (HMAC, DATA, DATALEN, PWD, PWDLEN, HMAC, HASHALGO, RC), VL</pre>	
Returncode (RC)	Länge des HMAC oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>DATA</i>	Speicheradresse der Eingabedaten.	Eingabe
<i>DATALEN</i>	Länge der Eingabedaten.	Eingabe
<i>KEY</i>	Speicheradresse des Schlüssels.	Eingabe
<i>KEYLEN</i>	Länge des Schlüssels.	Eingabe
<i>HMAC</i>	Speicheradresse des Rückgabebereiches für den erstellten Hashwert.	Ausgabe
<i>HASHALGO</i>	Der zu verwendende Hashtyp. Unterstützt werden die Hashtypen MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 und RipeMD160.	Eingabe

X.509 Zertifikate

Allgemein

Zertifikate gibt es seit der Erfindung von Public-Key-Algorithmen. Sie werden auch als elektronische Ausweise bezeichnet. Ein Zertifikat ist nichts anderes als ein signierter Datensatz. Beim Ausstellen eines Zertifikats sind zwei Parteien beteiligt: ein Zertifikatsaussteller und ein Zertifikatsantragsteller. Beide müssen ein asymmetrisches Schlüsselpaar besitzen. Will der Antragsteller zertifiziert werden, so übergibt er seinen öffentlichen Schlüssel dem Aussteller. Dieser bildet einen Datensatz, der sich aus dem Namen des Ausstellers, dem Namen des Antragstellers und dessen öffentlichen Schlüssel zusammensetzt. Dieser Datensatz wird anschließend mit dem privaten Schlüssel des Ausstellers signiert. Zusammen mit der Signatur bildet er das Zertifikat. Der Aussteller bescheinigt somit, dass der Zertifikatsinhaber und dessen öffentlicher Schlüssel zusammengehören. Mit dem öffentlichen Schlüssel des Zertifizierers kann der elektronische Ausweis jederzeit auf seine Unverfälschtheit überprüft werden.

CryptLib bietet Funktionen an, mit denen man die Unverfälschtheit eines Zertifikats überprüfen, sowie sämtliche Daten eines Zertifikats auslesen kann.

Funktionen

IMPORT-CERTIFICATE

Einlesen eines Zertifikatobjekts sowie Prüfung auf formale Richtigkeit.

Syntax	Cobol	
	<pre>MOVE IMPORT-CERTIFICATE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CERT, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (IMPORT_CERTIFICATE, CERT, CTX, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CERT</i>	Speicheradresse eines X.509 Zertifikats. Sowohl binäre als auch Base64-verschlüsselte Formate werden unterstützt.	Eingabe
<i>CTX</i>	Adresspointer zur Aufnahme der Speicheradresse des erstellten Zertifikatskontexts. Dieser wird für die weitere Bearbeitung des Zertifikats benötigt.	Ausgabe

GET-PUBLIC-KEY

Extrahieren des öffentlichen Schlüssels aus dem Zertifikat.

Syntax	Cobol	
	<pre>MOVE GET-PUBLIC-KEY TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, PUBKEY, PUBLLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_PUBLIC_KEY, CTX, PUBKEY, PUBLLEN, RC), VL</pre>	
Returncode (RC)	Länge des öffentlichen Schlüssels oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>PUBKEY</i>	Speicheradresse des Rückgabebereiches für den öffentlichen Schlüssel.	Ausgabe
<i>PUBLLEN</i>	Größe des verfügbaren Speicherplatzes für den öffentlichen Schlüssel.	Eingabe

GET-CRYPT-ALGO

Extrahieren des Verschlüsselungs-Algorithmus des öffentlichen Schlüssels.

Syntax	Cobol	
	<pre>MOVE GET-CRYPT-ALGO TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_CRYPT_ALGO, CTX, OUT, OUTLEN, RC), VL</pre>	
Returncode (RC)	Länge der Bezeichnung des Verschlüsselungs-Algorithmus oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für die Bezeichnung des Verschlüsselungs-Algorithmus.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-CRYPT-KEYLEN

Extrahieren der Schlüssellänge des öffentlichen Schlüssels.

Syntax	Cobol	
	<pre>MOVE GET-CRYPT-KEYLEN TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_CRYPT_KEYLEN, CTX, RC), VL</pre>	
Returncode (RC)	Länge des öffentlichen Schlüssels oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung

<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
------------	--	---------

GET-VERSION-INFO

Extrahieren der Versionsnummer des Zertifikats.

Syntax	Cobol	
	<pre>MOVE GET-VERSION-INFO TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_VERSION_INFO,OUT,OUTLEN,RC),VL</pre>	
Returncode (RC)	Länge der Versionsnummer oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für die extrahierte Versionsnummer.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-SERIAL-NUMBER

Extrahieren der Seriennummer des Zertifikats.

Syntax	Cobol	
	<pre>MOVE GET-SERIAL-NUMBER TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_SERIAL_NUMBER,CTX,OUT,OUTLEN,RC),VL</pre>	
Returncode (RC)	Länge der Seriennummer oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für die extrahierte Seriennummer.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-ISSUER-DN

Extrahieren der Zertifikatsausstellerdaten.

Syntax	Cobol	
	<pre>MOVE GET-ISSUER-DN TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_ISSUER_DN,CTX,OUT,OUTLEN,RC),VL</pre>	

Returncode (RC)	Länge der Zertifikatsausstellerdaten oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für die extrahierten Ausstellerdaten. Die einzelnen Elemente (CN, O, OU usw.) sind durch Kommata getrennt.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-SUBJECT-DN

Extrahieren der Zertifikatsinhaberdaten.

Syntax	Cobol	
	<pre>MOVE GET-SUBJECT-DN TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_SUBJECT_DN, CTX, OUT, OUTLEN, RC), VL</pre>	
Returncode (RC)	Länge der Zertifikatsinhaberdaten oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für die extrahierten Inhaberdaten. Die einzelnen Elemente (CN, O, OU usw.) sind durch Kommata getrennt.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-SIGNATURE-ALGO

Extrahieren des vom Zertifikatsaussteller zur Signatur verwendeten Algorithmus.

Syntax	Cobol	
	<pre>MOVE GET-SIGNATURE-ALGO TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_SIGNATURE_ALGO, CTX, OUT, OUTLEN, RC), VL</pre>	
Returncode (RC)	Länge der Bezeichnung des Verschlüsselungs-Algorithmus oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für die Bezeichnung des Verschlüsselungs-Algorithmus.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-SIGNATURE

Extrahieren der Zertifikatssignatur.

Syntax	Cobol	
	MOVE GET-SIGNATURE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.	
	Assembler	
	CALL XPSCRYPT, (GET_SIGNATURE,CTX,OUT,OUTLEN,RC),VL	
Returncode (RC)	Länge der Signatur oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des Zertifikatskontexts.	Eingabe
OUT	Speicheradresse des Rückgabebereiches für die extrahierte Signatur.	Ausgabe
OUTLEN	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-START-DATE

Extrahieren des Gültigkeitsbeginns des Zertifikats.

Syntax	Cobol	
	MOVE GET-START-DATE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.	
	Assembler	
	CALL XPSCRYPT, (GET_START_DATE,CTX,OUT,OUTLEN,RC),VL	
Returncode (RC)	Länge des Datums oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des Zertifikatskontexts.	Eingabe
OUT	Speicheradresse des Rückgabebereiches für den Gültigkeitsbeginn. Das Datum wird in der Form DD.MM.YYYY HH:MM:SS zurückgegeben.	Ausgabe
OUTLEN	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-END-DATE

Extrahieren des Gültigkeitsendes des Zertifikats.

Syntax	Cobol	
	MOVE GET-END-DATE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, RC.	
	Assembler	
	CALL XPSCRYPT, (GET_END_DATE,CTX,OUT,OUTLEN,RC),VL	
Returncode (RC)	Länge des Datums oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des Zertifikatskontexts.	Eingabe
OUT	Speicheradresse des Rückgabebereiches für das Gültigkeitsende. Das Datum wird in der Form DD.MM.YYYY HH:MM:SS zurückgegeben.	Ausgabe

<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe
---------------	--	---------

GET-ISSUER-DN-BLOB

Extrahieren der Zertifikatsausstellerdaten in binärer Form, einschließlich der ASN.1 Steuerzeichen. Dieser Blob kann dazu verwendet werden, den weiteren Zertifizierungspfad zu verfolgen.

Syntax	Cobol	
	<pre>MOVE GET-ISSUER-DN-BLOB TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_ISSUER_DN_BLOB, CTX, OUT, RC), VL</pre>	
Returncode (RC)	Länge des Zertifikatsaussteller-Blobs oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Adresspointer zur Aufnahme der Speicheradresse des Rückgabebereiches des extrahierten Zertifikatsaussteller-Blobs.	Ausgabe

GET-SUBJECT-DN-BLOB

Extrahieren der Zertifikatsinhaberdaten in binärer Form, einschließlich der ASN.1 Steuerzeichen. Dieser Blob kann dazu verwendet werden, den weiteren Zertifizierungspfad zu verfolgen.

Syntax	Cobol	
	<pre>MOVE GET-SUBJECT-DN-BLOB TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, KEY, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_SUBJECT_DN_BLOB, OUT, RC), VL</pre>	
Returncode (RC)	Länge des Zertifikatsinhaber-Blobs oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches des extrahierten Zertifikatsinhaber-Blobs.	Ausgabe

GET-ISSUER-DN-BY-TYPE

Extrahieren des unter *TYPE* spezifizierten Zertifikatsausstellerelements.

Syntax	Cobol	
	<pre>MOVE GET-ISSUER-DN-BY-TYPE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, TYPE, OUT, OUTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_ISSUER_DN_BY_TYPE, CTX, TYPE, OUT, OUTLEN, RC), VL</pre>	

Returncode (RC)	Länge der Zertifikatsausstellerdaten oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>TYPE</i>	DN-Typ des Ausstellers. Folgende Typen sind möglich: DN_C Country DN_SP State/Province DN_L Locality DN_O OrganizationName DN_OU OrganizationUnit DN_CN CommonName DN_EMAIL E-Mail DN_STREET Street DN_PHONE Phone DN_POSTAL PostalCode DN_TITLE Title	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für die extrahierten Ausstellerdaten.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-SUBJECT-DN-BY-TYPE

Extrahieren des unter *TYPE* spezifizierten Zertifikatsinhaberelements.

Syntax	Cobol	
	<pre>MOVE GET-SUBJECT-DN-BY-TYPE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, TYPE, OUT, OUTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_SUBJECT_DN_BY_TYPE, CTX, TYPE, OUT, OUTLEN, RC), VL</pre>	
Returncode (RC)	Länge der Zertifikatsinhaberdaten oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>TYPE</i>	DN-Typ des Inhabers. Folgende Typen sind möglich: DN_C Country DN_SP State/Province DN_L Locality DN_O OrganizationName DN_OU OrganizationUnit DN_CN CommonName DN_EMAIL E-Mail DN_STREET Street DN_PHONE Phone DN_POSTAL PostalCode DN_TITLE Title	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für die extrahierten Inhaberdaten.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe

GET-FIRST-EXTENSION

Die Standardfelder der X.509-Zertifikate reichen für viele Anwendungen nicht aus. Aus diesem Grund wurde die Syntax der Version 3 mit einer Extension-Komponente erweitert. Mit dieser ist es möglich, beliebige Daten in einem Zertifikat anzugeben.

Syntax	Cobol	
	<pre>MOVE GET-FIRST-EXTENSION TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, EXTENSION, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT,(GET_FIRST_EXTENSION,CTX,EXTENSION,RC),VL</pre>	
Returncode (RC)	Länge des Extension-Bereichs (0 falls nicht vorhanden) oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>EXTENSION</i>	Adresspointer auf die nachfolgend beschriebene Struktur. Die Copy-Strecke dieser Struktur ist in dem Copybuch <i>XPSCLEXT</i> bzw. <i>XPSCCLASM</i> (COBOL/Assembler) enthalten.	Eingabe
Felder	Beschreibung	
<i>C-OIDBIN</i>	Binärer Wert des OIDs (Object Identifier).	
<i>C-OIDCHR</i>	Character Wert des OIDs (Object Identifier).	
<i>C-ISCRIT</i>	Diese Komponente markiert eine Erweiterung als kritisch oder unkritisch. Kritische Erweiterungen müssen immer beachtet werden. Stößt ein Programm auf eine solche und kennt sie nicht, darf das Zertifikat nicht verwendet werden. Nichtkritische Erweiterungen sind unproblematisch. Sie können gegebenenfalls ignoriert werden.	
<i>C-FTYPE</i>	Der Feldtyp dieser Extension. Folgende Feldtypen sind möglich: BER_BOOLEAN BER_INTEGER BER_BITSTRING BER_OCTETSTRING BER_OBJECT_IDENTIFIER BER_OBJECT_DESCRIPTOR BER_EXTERNAL BER_REAL BER_ENUMERATED BER_EMBEDDED_PDV BER_STRING_UTF8 BER_RELATIVE_OID BER_STRING_NUMERIC BER_STRING_PRINTABLE BER_STRING_T61 BER_STRING_GRAPHIC BER_STRING_ISO646 BER_STRING_VIDEOTEXT BER_STRING_GENERAL BER_STRING_UNIVERSAL BER_CHAR_STRING BER_STRING_BMP	
<i>C-VALUE</i>	Integer Wert bei den Typen BER_BOOLEAN, BER_INTEGER, BER_ENUMERATED.	
<i>C-DATA</i>	Speicheradresse des binären Datenbereichs der Extension.	
<i>C-DLEN</i>	Länge des binären Datenbereichs der Extension.	

GET-NEXT-EXTENSION

Extrahieren der nächsten Zertifikats-Extension.

Syntax	Cobol	
	<pre>MOVE GET-NEXT-EXTENSION TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, EXTENSION, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT,(GET_NEXT_EXTENSION,CTX,EXTENSION,RC),VL</pre>	

Returncode (RC)	Länge des Extension-Bereichs (0 falls nicht vorhanden) oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>EXTENSION</i>	Speicheradresse des Rückgabebereiches für die extrahierte Extension. Die Struktur des Rückgabebereiches ist unter 'GET-FIRST-EXTENSION' beschrieben.	Ausgabe

GET-EXTENSION-BY-OID

Extrahieren der Zertifikats-Extension mit diesem Object-Identifizier (OID).

Syntax	Cobol	
	<pre>MOVE GET-EXTENSION-BY-OID TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OID, EXTENSION, RC.</pre>	
	Assembler	
	CALL XPSCRYPT, (GET_EXTENSION_BY_OID, CTX, OID, EXTENSION, RC), VL	
Returncode (RC)	Länge des Extension-Bereichs (0 falls nicht vorhanden) oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OID</i>	Speicheradresse des binären Object-Identifizier, der extrahiert werden soll.	Eingabe
<i>EXTENSION</i>	Speicheradresse des Rückgabebereiches für die extrahierte Extension. Die Struktur des Rückgabebereiches ist unter 'GET-FIRST-EXTENSION' beschrieben.	Ausgabe

GET-FINGERPRINT

Erzeugen eines Fingerprints (Hashwert) des Zertifikats. Der Fingerprint dient zur visuellen Überprüfung eines Zertifikats.

Syntax	Cobol	
	<pre>MOVE GET-FINGERPRINT TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OUT, OUTLEN, HASHALGO, RC.</pre>	
	Assembler	
	CALL XPSCRYPT, (GET_FINGERPRINT, CTX, OUT, OUTLEN, HASHALGO, RC), VL	
Returncode (RC)	Länge des Fingerprints oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Zertifikatskontexts.	Eingabe
<i>OUT</i>	Speicheradresse des Rückgabebereiches für den erzeugten Fingerprint.	Ausgabe
<i>OUTLEN</i>	Größe des verfügbaren Speicherplatzes.	Eingabe
<i>HASHALGO</i>	Der zu verwendende Hashtyp. Unterstützt werden die Hashtypen MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 und RipeMD160.	Eingabe

VERIFY-CERTIFICATE

Überprüfung der Gültigkeit eines Zertifikats.

Syntax	Cobol	
	<pre>MOVE VERIFY-CERTIFICATE TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION, CTX, PUBKEY, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRIPT, (VERIFY_CERTIFICATE, CTX, PUBKEY, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des Zertifikatskontexts.	Eingabe
PUBKEY	Öffentlicher Schlüssel des Zertifikatsausstellers.	Eingabe

CLEANUP-CERTIFICATE

Freigeben des von den Zertifikatsroutinen benötigten Speichers.

Syntax	Cobol	
	<pre>MOVE CLEANUP-CERTIFICATE TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRIPT, (CLEANUP_CERTIFICATE, CTX, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des Zertifikatskontexts.	Eingabe

COBOL-Beispiel:

```
*-----*
*   XPS-CRYPTLIB SAMPLE PROGRAM: GET X.509 CERTIFICATE   *
*-----*
ID DIVISION.
PROGRAM-ID.
    CERTTSTC.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)           VALUE "XPSDATA".
01 CERTUSER        PIC X(8)           VALUE "XPSUSERC".
01 CERITRST        PIC X(8)           VALUE "XPSTESTC".
01 OUT              PIC X(1024)        VALUE LOW-VALUES.
01 HEADER          PIC X(20)          VALUE SPACES.
01 HEADER1         PIC X(20)          VALUE "VERSION:".
01 HEADER2         PIC X(20)          VALUE "SERIAL-NUMBER:".
01 HEADER3         PIC X(20)          VALUE "ISSUER-DN:".
01 HEADER4         PIC X(20)          VALUE "START-DATE:".
01 HEADER5         PIC X(20)          VALUE "END-DATE:".
01 HEADER6         PIC X(20)          VALUE "CRYPT-ALGO:".
01 HEADER7         PIC X(20)          VALUE "CRYPT-KEYLEN:".
01 HEADER8         PIC X(20)          VALUE "PUBLIC-KEY:".
01 HEADER9         PIC X(20)          VALUE "SIGNATURE-ALGO:".
01 HEADER10        PIC X(20)          VALUE "SIGNATURE:".
01 HEADER11        PIC X(20)          VALUE "EXTENSION-OID:".
01 HEADER12        PIC X(20)          VALUE "EXTENSION-VALUE:".
01 HEADER13        PIC X(20)          VALUE "FINGERPRINT-SHA1:".
01 HEADER14        PIC X(20)          VALUE "FINGERPRINT-MD5:".
01 OID             PIC X(32)          VALUE X"06096086480186F8420103".
01 HASHSHA         PIC 9(8)           COMP VALUE 26.
01 HASHMD5         PIC 9(8)           COMP VALUE 5.
01 CERT-CTX        POINTER.
01 ADDR-CERT       POINTER.
```

```

01 DNTAB.
05 DNNAME.
    10 PIC X(8)      VALUE "CN:".
    10 PIC X(8)      VALUE "O:".
    10 PIC X(8)      VALUE "OU:".
    10 PIC X(8)      VALUE "L:".
    10 PIC X(8)      VALUE "SP:".
    10 PIC X(8)      VALUE "STREET:".
    10 PIC X(8)      VALUE "POSTAL:".
    10 PIC X(8)      VALUE "C:".
    10 PIC X(8)      VALUE "EMAIL:".
    10 PIC X(8)      VALUE "PHONE:".
    10 PIC X(8)      VALUE "TITEL:".
05 DNID.
    10 PIC 9(8) COMP VALUE 6.
    10 PIC 9(8) COMP VALUE 4.
    10 PIC 9(8) COMP VALUE 5.
    10 PIC 9(8) COMP VALUE 3.
    10 PIC 9(8) COMP VALUE 2.
    10 PIC 9(8) COMP VALUE 8.
    10 PIC 9(8) COMP VALUE 10.
    10 PIC 9(8) COMP VALUE 1.
    10 PIC 9(8) COMP VALUE 7.
    10 PIC 9(8) COMP VALUE 9.
    10 PIC 9(8) COMP VALUE 11.
01 REDEFINES DNTAB.
05 DN-NAME PIC X(8)      OCCURS 11.
05 DN-ID   PIC 9(8) COMP OCCURS 11.
*
COPY XPSCLRSA.
COPY XPSCLEXT.
*
77 CRYPT-FUNCTION PIC X.
77 DATALEN      PIC 9(8)  COMP VALUE ZEROES.
77 DUMPLEN       PIC 9(8)  COMP VALUE ZEROES.
77 OUTLEN        PIC 9(8)  COMP VALUE 1024.
77 RC            PIC S9(8)  COMP VALUE ZEROES.
77 IX            PIC S9(2)  VALUE ZERO.
*
LINKAGE SECTION.
COPY XPSCLCOB.
01 EXTDATA      PIC X.
*****
**              PROCEDURE DIVISION              **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* READ X.509 FILE "XPSTESTC" FROM MACLIB (TRUSTED-SIGNER CERT) *
* ----- *
    MOVE READ-FILE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        DDNAME, CERTTRST, ADDR-CERT, DATALEN, RC.
    IF RC < 0
        DISPLAY "FILE 'XPSTESTC' NOT FOUND: RC = " RC
        GOBACK.
* ----- *
* IMPORT X.509 CERTIFICATE (TRUSTED-SIGNER CERT) *
* ----- *
    MOVE IMPORT-CERTIFICATE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        ADDR-CERT, CERT-CTX, RC.
    IF RC < 0
        DISPLAY "ERROR IMPORT-CERTIFICATE: RC = " RC
        GOBACK.
* ----- *
* GET PUBLIC-KEY FROM CERTIFICATE (TRUSTED-SIGNER CERT) *
* ----- *
    MOVE RSA-PUBLIC-LEN TO DATALEN
    MOVE GET-PUBLIC-KEY TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, RSA-PUBLIC-KEY, DATALEN, RC.
    IF RC < 0
        DISPLAY "ERROR GET-PUBLIC-KEY: RC = " RC
        GOBACK.
* ----- *
* RELEASE CERTIFICATE CONTEXT (TRUSTED-SIGNER CERT) *
* ----- *
    MOVE CLEANUP-CERTIFICATE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, RC.
    IF RC < 0
        DISPLAY "ERROR CLEANUP-CERTIFICATE: RC = " RC
        GOBACK.
* ----- *
* RELEASE FILE-STORAGE "XPSTESTC" *
* ----- *
    MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        ADDR-CERT, RC.

```

```

IF RC < 0
  DISPLAY "ERROR RELEASE-FILE: RC = " RC
  GOBACK.
* ----- *
* READ X.509 FILE "XPSUSERC" FROM MACLIB (USER-CERTIFICATE) *
* ----- *
  MOVE READ-FILE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION, DDNAME,
    CERTUSER, ADDR-CERT, DATALEN, RC.
  IF RC < 0
    DISPLAY "FILE 'XPSUSERC' NOT FOUND: RC = " RC
    GOBACK.
* ----- *
* IMPORT X.509 CERTIFICATE (USER-CERTIFICATE) *
* ----- *
  MOVE IMPORT-CERTIFICATE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    ADDR-CERT, CERT-CTX, RC.
  IF RC < 0
    DISPLAY "ERROR IMPORT-CERTIFICATE: RC = " RC
    GOBACK.
* ----- *
* VERIFY USER-CERTIFICATE WITH TRUSTED-SIGNER PUBLIC-KEY *
* ----- *
  MOVE VERIFY-CERTIFICATE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    CERT-CTX, RSA-PUBLIC-KEY, RC.
  IF RC < 0
    DISPLAY "ERROR VERIFY: " RC
    GOBACK.
* ----- *
* THE CERTIFICATE IS OKAY, SO WE CAN GET DATA FROM THE CERT *
* ----- *
  MOVE GET-VERSION-INFO TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    CERT-CTX, OUT, OUTLEN, RC.
  IF RC < 0
    DISPLAY "ERROR GET-VERSION-INFO: RC = " RC
    GOBACK.
  MOVE RC TO DUMPLEN.
  MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    HEADER1, OUT, DUMPLEN, RC.
*
  MOVE GET-SERIAL-NUMBER TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    CERT-CTX, OUT, OUTLEN, RC.
  IF RC < 0
    DISPLAY "ERROR GET-SERIAL-NUMBER: RC = " RC
    GOBACK.
  MOVE RC TO DUMPLEN.
  MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    HEADER2, OUT, DUMPLEN, RC.
*
  MOVE GET-ISSUER-DN TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    CERT-CTX, OUT, OUTLEN, RC.
  IF RC < 0
    DISPLAY "ERROR GET-ISSUER-DN: RC = " RC
    GOBACK.
  MOVE RC TO DUMPLEN.
  MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    HEADER3, OUT, DUMPLEN, RC.
*
  MOVE GET-START-DATE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    CERT-CTX, OUT, OUTLEN, RC.
  IF RC < 0
    DISPLAY "ERROR GET-START-DATE: RC = " RC
    GOBACK.
  MOVE RC TO DUMPLEN.
  MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    HEADER4, OUT, DUMPLEN, RC.
*
  MOVE GET-END-DATE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    CERT-CTX, OUT, OUTLEN, RC.
  IF RC < 0
    DISPLAY "ERROR GET-END-DATE: RC = " RC
    GOBACK.
  MOVE RC TO DUMPLEN.
  MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    HEADER5, OUT, DUMPLEN, RC.
*
PERFORM VARYING IX
  FROM 1 BY 1

```

```

        UNTIL IX > 11
        MOVE GET-SUBJECT-DN-BY-TYPE TO CRYPT-FUNCTION
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            CERT-CTX, OUT, OUTLEN, DN-ID (IX), RC
        IF RC > 0
            MOVE DN-NAME (IX) TO HEADER
            MOVE RC TO DUMPLEN
            MOVE DUMP-STORAGE TO CRYPT-FUNCTION
            CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
                HEADER, OUT, DUMPLEN, RC
        END-IF
    END-PERFORM.
*
    MOVE GET-CRYPT-ALGO TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, OUT, OUTLEN, RC.
    IF RC < 0
        DISPLAY "ERROR GET-CRYPT-ALGO: RC = " RC
        GOBACK.
    MOVE RC TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HEADER6, OUT, DUMPLEN, RC.
*
    MOVE GET-CRYPT-KEYLEN TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, RC.
    IF RC < 0
        DISPLAY "ERROR GET-CRYPT-KEYLEN: RC = " RC
        GOBACK.
    MOVE 4 TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HEADER7, RC, DUMPLEN, RC.
*
    MOVE GET-PUBLIC-KEY TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, OUT, OUTLEN, RC.
    IF RC < 0
        DISPLAY "ERROR GET-PUBLIC-KEY: RC = " RC
        GOBACK.
    MOVE RC TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HEADER8, OUT, DUMPLEN, RC.
*
    MOVE GET-SIGNATURE-ALGO TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, OUT, OUTLEN, RC.
    IF RC < 0
        DISPLAY "ERROR GET-SIGNATURE-ALGO: RC = " RC
        GOBACK.
    MOVE RC TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HEADER9, OUT, DUMPLEN, RC.
*
    MOVE GET-SIGNATURE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, OUT, OUTLEN, RC.
    IF RC < 0
        DISPLAY "ERROR GET-SIGNATURE: RC = " RC
        GOBACK.
    MOVE RC TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HEADER10, OUT, DUMPLEN, RC.
*
    MOVE GET-FIRST-EXTENSION TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, CERT-EXT, RC.
    PERFORM UNTIL RC <= ZEROES
        PERFORM GET-EXTENSION
    END-PERFORM.
*
    MOVE GET-EXTENSION-BY-OID TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, CERT-EXT, OID, RC.
    IF RC > ZEROES
        PERFORM GET-EXTENSION
    END-IF.
*
    MOVE GET-FINGERPRINT TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, OUT, OUTLEN, HASHSHA, RC.
    IF RC < 0
        DISPLAY "ERROR GET-FINGERPRINT: RC = " RC
        GOBACK.
    MOVE RC TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.

```

```
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,  
HEADER13, OUT, DUMPLEN, RC.  
*  
MOVE GET-FINGERPRINT TO CRYPT-FUNCTION.  
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,  
CERT-CTX, OUT, OUTLEN, HASHMD5, RC.  
IF RC < 0  
  DISPLAY "ERROR GET-FINGERPRINT: RC = " RC  
  GOBACK.  
MOVE RC TO DUMPLEN.  
MOVE DUMP-STORAGE TO CRYPT-FUNCTION.  
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,  
HEADER14, OUT, DUMPLEN, RC.  
STOP RUN.  
*  
GET-EXTENSION SECTION.  
MOVE 32 TO DUMPLEN.  
MOVE DUMP-STORAGE TO CRYPT-FUNCTION.  
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,  
HEADER11, C-OIDCHR, DUMPLEN, RC.  
IF C-DLEN = ZEROES  
  MOVE 4 TO DUMPLEN  
  MOVE DUMP-STORAGE TO CRYPT-FUNCTION  
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,  
  HEADER12, C-VALUE, DUMPLEN, RC  
ELSE  
  MOVE C-DLEN TO DUMPLEN  
  SET ADDRESS OF EXTDATA TO C-DATA  
  MOVE DUMP-STORAGE TO CRYPT-FUNCTION  
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,  
  HEADER12, EXTDATA, DUMPLEN, RC  
END-IF.  
MOVE GET-NEXT-EXTENSION TO CRYPT-FUNCTION.  
CALL 'XPSCRYPT' USING CRYPT-FUNCTION,  
CERT-CTX, CERT-EXT, RC.  
GET-EXTENSION-END.  
EXIT.  
ENDRUN.
```

S/MIME Objekte (PKCS#7)

Allgemein

PKCS#7 wird als *Cryptographic Message Syntax Standard* bezeichnet und beschreibt eine Syntax, nach der Daten durch kryptographische Maßnahmen wie digitale Signaturen oder Verschlüsselung geschützt werden können. CryptLib unterstützt folgende Inhaltstypen (Content Types):

Data	wird zur Modellierung von Daten verwendet und bietet keinerlei kryptographische Funktionalität.
Signed-data	beschreibt ein Format um die Integrität von Daten und die Sender-Authentizität durch Verwendung von digitalen Signaturen und Zertifikaten zu gewährleisten.
Enveloped-data	wird zur empängerspezifischen Verschlüsselung von Daten verwendet, um zu verhindern, dass ein Unbefugter den Inhalt lesen kann. (Vertraulichkeit).
Encrypted-data	wird zur Datenverschlüsselung verwendet.

Funktionen

IMPORT-PKCS7-DATA

Einlesen eines PKCS#7-Data Objekts sowie Prüfung auf formale Richtigkeit. Der Inhaltstyp *Data* beschreibt eine beliebige Folge von Datenbytes. Die Daten können mit den Funktionen *GET-FIRST-DATA* sowie *GET-NEXT-DATA* extrahiert werden.

Syntax	Cobol	
	<pre>MOVE IMPORT-PKCS7-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, APKCS7, PKCS7LEN, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (IMPORT_PKCS7_DATA, APKCS7, PKCS7LEN, CTX, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
APKCS7	Speicheradresse eines PKCS#7-Data Objekts. Unterstützt werden sowohl binäre, Base64-verschlüsselte als auch S/MIME Formate.	Eingabe
PKCS7LEN	Länge des PKCS#7-Data Objekts.	Eingabe
CTX	Speicheradresse des importierten PKCS#7-Kontexts. Dieser wird für die weitere Bearbeitung des Objekts benötigt.	Ausgabe

IMPORT-SIGNED-DATA

Einlesen eines PKCS#7 Signed-data Objekts sowie Prüfung auf formale Richtigkeit. Der Inhaltstyp *Signed-data* definiert eine Syntax zur Berechnung und Übertragung von digitalen Signaturen. Die Anzahl der Parteien, die eine Nachricht signieren, ist nicht eingeschränkt. Das heißt, es ist erlaubt, dass dieselben Daten von mehr als nur einer Partei unterzeichnet werden.

Die Daten können mit den Funktionen *GET-FIRST-DATA* sowie *GET-NEXT-DATA* extrahiert werden. Mit den Funktionen *GET-FIRST-SIGNER* sowie *GET-NEXT-SIGNER* können die einzelnen Unterzeichner ausgelesen werden. Die Funktionen *VERIFY-SIGNER* sowie *VERIFY-ALL-SIGNER* können zur Überprüfung der Integrität der Daten verwendet werden. Mit der Funktion *ADD-SIGNER-CERT* können Zertifikatsaussteller hinzugefügt werden.

Syntax	Cobol	
	<pre>MOVE IMPORT-SIGNED-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, APKCS7, PKCS7LEN, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (IMPORT_SIGNED_DATA,APKCS7,PKCS7LEN,CTX,RC),VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
APKCS7	Speicheradresse eines PKCS#7-Data Objekts. Unterstützt werden sowohl binäre, Base64-verschlüsselte als auch S/MIME Formate.	Eingabe
PKCS7LEN	Länge des PKCS#7-Data Objekts.	Eingabe
CTX	Speicheradresse des importierten PKCS#7-Kontexts. Dieser wird für die weitere Bearbeitung des Objekts benötigt.	Ausgabe

IMPORT-ENVELOPED-DATA

Einlesen eines PKCS#7 Enveloped-data Objekts sowie Prüfung auf formale Richtigkeit. Der Inhaltstyp *Enveloped-data* definiert eine Syntax, nach der die Nachricht 'empfänger-spezifisch' verschlüsselt wird, also Informationen über den beabsichtigten Empfänger miteinbezieht. Dazu wird eine Technik verwendet, die man als 'Digital Enveloping' bezeichnet. In Äquivalenz zum *Signed-data*-Typ, bei dem eine Nachricht von mehreren Unterzeichnern signiert werden kann, erlaubt es der *Enveloped-data*-Typ, mehrere Empfänger einzubeziehen. Informationen über einen Empfänger werden im Hilfstyp 'RecipientInfo' zusammengefasst.

Die Daten können mit den Funktionen *GET-FIRST-DATA* sowie *GET-NEXT-DATA* extrahiert werden.

Syntax	Cobol	
	<pre>MOVE IMPORT-ENVELOPED-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, APKCS7, PKCS7LEN, APKCS12, PKCS12LEN, PWD, PWDLEN, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (IMPORT_ENVELOPED_DATA,APKCS7,PKCS7LEN,APKCS12, PKCS12LEN,PWD,PWDLEN,CTX,RC),VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
APKCS7	Speicheradresse eines PKCS#7-Data Objekts. Unterstützt werden sowohl binäre, Base64-verschlüsselte als auch S/MIME Formate.	Eingabe
PKCS7LEN	Länge des PKCS#7-Data Objekts.	Eingabe
APKCS12	Speicheradresse eines PKCS#12-Objekts. PKCS#12-Objekte stellen eine Syntax für den Austausch von Schlüsseln und Zertifikaten zur	Eingabe

	Verfügung. Ein PKCS#12-Objekt enthält Schlüsseltaschen (key-bags) und Zertifikatstaschen (certificate-bags). Mit Hilfe der Zertifikatstaschen kann ermittelt werden, ob im PKCS#7-Objekt eine RecipientInfo für diesen Benutzer enthalten ist. Falls eine RecipientInfo enthalten ist, kann dann aus der Schlüsseltasche der private Schlüssel des Anwenders extrahiert werden. Mit dem privaten Schlüssel wird der <i>Content-Encryption</i> -Schlüssel dekodiert. Mit diesem wiederum können dann die Daten entschlüsselt werden.	
<i>PKCS12LEN</i>	Länge des PKCS#12-Objekts.	Eingabe
<i>PWD</i>	PKCS#12-Objekte sind mit einem Passwort gesichert. Hier ist die Speicheradresse des Passwortes zu übergeben.	Eingabe
<i>PWDLEN</i>	Länge des Passwortes.	Eingabe
<i>CTX</i>	Speicheradresse des importierten PKCS#7-Kontexts. Dieser wird für die weitere Bearbeitung des Objekts benötigt.	Ausgabe

IMPORT-ENCRYPTED-DATA

Einlesen eines PKCS#7 Encrypted-data Objekts sowie Prüfung auf formale Richtigkeit. Der Inhaltstyp *Encrypted-data* beschreibt eine Syntax zur Datenverschlüsselung. Im Unterschied zum Typ *Enveloped-data* wird hier davon ausgegangen, dass der Empfänger bereits im Besitz des *Content-Encryption*-Schlüssels ist und dieser daher nicht eigens angegeben werden muss. Dieser Typ eignet sich vor allem für Applikationen, in denen Daten verschlüsselt auf ein Speichermedium geschrieben werden. Ein prominentes Anwendungsbeispiel ist der *Personal-Information-Exchange-Syntax*-Standard PKCS#12. Die Daten können mit den Funktionen *GET-FIRST-DATA* sowie *GET-NEXT-DATA* extrahiert werden.

Syntax	Cobol	
	<pre>MOVE IMPORT-ENCRYPTED-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, APKCS7, PKCS7LEN, PWD, PWDLEN, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (IMPORT_ENCRYPTED_DATA, APKCS7, PKCS7LEN, PWD, PWDLEN, CTX, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>APKCS7</i>	Speicheradresse eines PKCS#7-Data Objekts. Unterstützt werden sowohl binäre, Base64-verschlüsselte als auch S/MIME Formate.	Eingabe
<i>PKCS7LEN</i>	Länge des PKCS#7 Encrypted-data Objekts.	Eingabe
<i>PWD</i>	Speicheradresse des Passwortes, mit dem der <i>Content-Encryption</i> -Schlüssel verschlüsselt wurde.	Eingabe
<i>PWDLEN</i>	Länge des Passwortes.	Eingabe
<i>CTX</i>	Speicheradresse des importierten PKCS#7-Kontexts. Dieser wird für die weitere Bearbeitung des Objekts benötigt.	Ausgabe

CREATE-PKCS7-DATA

Erstellen eines PKCS#7-Data-Objektes. Ein PKCS#7-Data-Objekt wird zur Modellierung von Daten verwendet und bietet keinerlei kryptographische Funktionalität. Daten werden mit der Funktion *ADD-PKCS7-DATA* hinzugefügt.

Syntax	Cobol
---------------	--------------

	MOVE CREATE-PKCS7-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, CTX, RC.	
	Assembler	
	CALL XPSCRYPT, (CREATE_CTX_DATA, OPTION, CTX, RC), VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>OPTION</i>	HEADER_INCLUDED Bei Angabe dieser Option wird der Contentype hinzugefügt.	Eingabe
<i>CTX</i>	Speicheradresse des erstellten PKCS#7-Kontexts.	Ausgabe

CREATE-SIGNED-DATA

Erstellen eines PKCS#7 Signed-data-Objektes. Der Inhaltstyp *Signed-data* definiert eine Syntax zur Berechnung und Übertragung von digitalen Signaturen. Die Anzahl der Parteien, die eine Nachricht signieren, ist unbeschränkt. Das heißt, es ist erlaubt, dass dieselben Daten von mehr als nur einer Partei unterzeichnet werden. Unterzeichner werden mit der Funktion ADD-SIGNER, Daten mit der Funktion ADD-PKCS-DATA hinzugefügt. Mit der Funktion ADD-SIGNER-CERT können zusätzliche Zertifikate hinzugefügt werden.

Syntax	Cobol	
	MOVE CREATE-SIGNED-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, CTX, RC.	
	Assembler	
	CALL XPSCRYPT, (CREATE_CTX_DATA, OPTION, CTX, RC), VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>OPTION</i>	HEADER_INCLUDED Bei Angabe dieser Option wird der Contentype hinzugefügt. DATA_IMPLICIT Bei Angabe dieser Option werden die Daten in das Signed-data Objekt miteinbezogen. Das bedeutet, dass dann das content-Feld für die Aufnahme des Nachrichteninhalts vorhanden ist. Wenn diese Option nicht gesetzt wird, fehlt das content-Feld und die Daten müssen auf andere Art und Weise übertragen werden. CERT_IMPLICIT Bei Angabe dieser Option werden sämtliche in der PKCS#12-Datei enthaltenen Zertifikate des Unterzeichners in das Signed-data-Objekt mit aufgenommen.	Eingabe
<i>CTX</i>	Speicheradresse des erstellten PKCS#7-Kontexts.	Ausgabe

CREATE-ENVELOPED-DATA

Erstellen eines PKCS#7 Enveloped-data-Objektes. Der Inhaltstyp *Enveloped-data* definiert eine Syntax, nach der die Nachricht 'empfänger-spezifisch' verschlüsselt wird, also Informationen über den beabsichtigten Empfänger miteinbezieht. Dazu wird eine Technik verwendet, die man 'Digital Enveloping' bezeichnet. In Äquivalenz zum *Signed-data*-Typ, bei dem eine Nachricht von mehreren Unterzeichnern signiert werden kann, erlaubt es der *Enveloped-data*-Typ, mehrere Empfänger einzubeziehen.

Empfänger werden mit der Funktion *ADD-RECIPIENT*, Daten mit der Funktion *ADD-PKCS7-DATA* hinzugefügt.

Syntax	Cobol	
	<pre>MOVE CREATE-ENVELOPED-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, CTX, ENCRALGO, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (CREATE_ENVELOPED_DATA,OPTION,CTX,ENCRALGO,RC),VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>OPTION</i>	HEADER_INCLUDED Bei Angabe dieser Option wird der ContentType hinzugefügt.	Eingabe
<i>CTX</i>	Speicheradresse des erstellten PKCS#7-Kontexts.	Ausgabe
<i>ENCRALGO</i>	Algorithmus mit dem die Daten verschlüsselt werden sollen. Folgende Algorithmen werden unterstützt: DESEDE3CBC Triple DES 168Bit DESCBC DES 56Bit RC2CBC_128 RC2 128Bit RC2CBC_64 RC2 64Bit RC2CBC_40 RC2 40Bit RC4_128 RC4 128Bit RC4_64 RC4 64Bit RC4_40 RC4 40Bit	Eingabe

CREATE-ENCRYPTED-DATA

Erstellen eines PKCS#7 Encrypted-data-Objektes. Der Inhaltstyp *Encrypted-data* beschreibt eine Syntax zur Datenverschlüsselung. Im Unterschied zum Typ *Enveloped-data* wird hier davon ausgegangen, dass der Empfänger bereits im Besitz des *Content-Encryption*-Schlüssels ist und dieser daher nicht eigens angegeben werden muss. Dieser Typ eignet sich vor allem für Applikationen, bei denen Daten verschlüsselt auf ein Speichermedium geschrieben werden. Die Daten werden mit der Funktion *ADD-PKCS7-DATA* hinzugefügt.

Syntax	Cobol	
	<pre>MOVE CREATE-ENCRYPTED-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, CTX, PBEALGO, PWD, PWDLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (CREATE_ENCRYPTED_DATA,OPTION,CTX,PBEALGO,PWD, PWDLEN,RC),VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>OPTION</i>	HEADER_INCLUDED Bei Angabe dieser Option wird der ContentType hinzugefügt.	Eingabe
<i>CTX</i>	Speicheradresse des erstellten PKCS#7-Kontexts.	Ausgabe
<i>PBEALGO</i>	Algorithmus mit dem die Daten verschlüsselt werden sollen. Folgende Algorithmen werden unterstützt: PBE3DES_3Key Triple DES 168Bit PBE3DES_2Key Triple DES 112Bit PBERC2_128 RC2 128Bit	Eingabe

	PBERC2_40 RC2 40Bit PBERC4_128 RC4 128Bit PBERC4_40 RC4 40Bit	
<i>PWD</i>	Speicheradresse des Passwortes, das zur Schlüsselgenerierung benötigt wird.	Eingabe
<i>PWDLEN</i>	Länge des Passwortes.	Eingabe

ADD-PKCS7-DATA

Import Funktionen:

Bei *Signed-data* Objekts können die Daten IMPLICIT oder EXPLICIT verarbeitet werden. Wird der Modus IMPLICIT festgelegt, werden die Daten in das *Signed-data*-Objekt miteinbezogen. Genauer ausgedrückt bedeutet dies, dass dann das Feld *content* für die Aufnahme des Nachrichteninhalts vorhanden ist. Wird jedoch der Modus EXPLICIT gewählt, fehlt das *content*-Feld und die Daten müssen auf andere Art und Weise übertragen werden. Diese Funktion bietet die Möglichkeit, Daten an das *Signed-data*-Objekt zu übergeben.

Create Funktionen:

Hiermit werden bei den Funktion *CREATE-PKCS7-DATA*, *CREATE-SIGNED-DATA*, *CREATE-ENVELOPED-DATA* sowie *CREATE-ENCRYPTED-DATA* Daten zum PKCS#7-Objekt hinzugefügt.

Syntax	Cobol	
	<pre>MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, DATA, DATALEN, RC.</pre>	
Syntax	Assembler	
	<pre>CALL XPSCRYPT, (ADD_PKCS7_DATA, CTX, DATA, DATALEN, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>DATA</i>	Speicheradresse der zu übergebenden Daten.	Eingabe
<i>DATALEN</i>	Länge der Daten.	Eingabe

ADD-MESSAGE-DIGEST

Bei *Signed-data* Objekts die EXPLICIT verarbeitet werden, kann mit dieser Funktion der Message Digest (Hashwert über die Daten) hinzugefügt werden. Beim Hinzufügen des extern errechneten Message Digest darf die Funktion *ADD-PKCS7-DATA* nicht ausgeführt werden.

Syntax	Cobol	
	<pre>MOVE ADD-MESSAGE-DIGEST TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, DATA, DATALEN, RC.</pre>	
Syntax	Assembler	
	<pre>CALL XPSCRYPT, (ADD_MESSAGE_DIGEST, CTX, DATA, DATALEN, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>DATA</i>	Speicheradresse des Message Digest.	Eingabe

<i>DATALEN</i>	Länge des Message Digest.	Eingabe
----------------	---------------------------	---------

ADD-SIGNER

Ein *Signed-data* Objekt muss von einem oder mehreren Unterzeichnern signiert werden. Mit dieser Funktion wird aus der *PKCS#12*-Datei, die den geheimen Schlüssel sowie das X.509 Zertifikat des Unterzeichners enthält, eine *SignerInfo* Struktur erstellt, mit der das Objekt signiert werden kann.

Syntax	Cobol	
	<pre>MOVE ADD-SIGNER TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, APKCS12, PKCS12LEN, PWD, PWDLEN, RC.</pre>	
	Assembler	
	CALL XPSCRYPT, (ADD_SIGNER, CTX, APKCS12, PKCS12LEN, PWD, PWDLEN, RC), VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>APKCS12</i>	Speicheradresse des PKCS#12-Objekts des Unterzeichners.	Eingabe
<i>PKCS12LEN</i>	Länge des PKCS#12-Objekts.	Eingabe
<i>PWD</i>	Speicheradresse des Passwortes, mit dem das PKCS#12-Objekt verschlüsselt wurde.	Eingabe
<i>PWDLEN</i>	Länge des Passwortes.	Eingabe

ADD-RECIPIENT

Beim Erstellen eines *EnvelopedData* Objekts müssen Informationen über den beabsichtigten Empfänger mit einbezogen werden. Das heißt, die Nachricht wird 'empfänger-spezifisch' verschlüsselt. Mit dieser Funktion wird das X.509 Zertifikat eines Empfängers übergeben. In dem Zertifikat ist der öffentliche Schlüssel enthalten, der zur Verschlüsselung des symmetrischen *Content-Encryption*-Schlüssels verwendet wird.

Syntax	Cobol	
	<pre>MOVE ADD-RECIPIENT TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, CERT, CERTLEN, RC.</pre>	
	Assembler	
	CALL XPSCRYPT, (ADD_RECIPIENT, CTX, CERT, CERTLEN, RC), VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>CERT</i>	Speicheradresse des X.509 Zertifikates des Empfängers.	Eingabe
<i>CERTLEN</i>	Länge des Zertifikates.	Eingabe

ADD-SIGNER-CERT

Import Funktionen:

Bei Verifizierung von *Signed-data* Objekts sollte die Zertifikathierarchie der Unterzeichner überprüft

werden. Falls die Aussteller-Zertifikate im PKCS#7-Objekt nicht enthalten sind, bietet diese Funktion die Möglichkeit, die Zertifikate an das *Signed-data*-Objekt zu übergeben.

Create Funktionen:

Bei der Funktion CREATE-SIGNED-DATA werden alle Zertifikate der PKCS#12-Datei der Unterzeichner hinzugefügt. Falls weitere Zertifikate zu dem Objekt hinzugefügt werden sollen, kann dies mit dieser Funktion geschehen.

Syntax	Cobol	
	<pre>MOVE ADD-SIGNER-CERT TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION, CTX, CERT, CERTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRIPT, (ADD_SIGNER_CERT, CTX, CERT, CERTLEN, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des PKCS#7-Kontexts.	Eingabe
CERT	Speicheradresse des X.509 Zertifikates des Unterzeichners.	Eingabe
CERTLEN	Länge des Zertifikates.	Eingabe

ADD-TRUSTED-SIGNER

Bei der Verifizierung von *Signed-data* Objekten kann das Zertifikat des Unterzeichners auf einen vertrauenswürdigen Aussteller (Trusted Signer) überprüft werden. Mit dieser Funktion können vertrauenswürdige Unterzeichner geladen werden.

Syntax	Cobol	
	<pre>MOVE ADD-TRUSTED-SIGNER TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION, CTX, CERT, CERTLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRIPT, (ADD_TRUSTED_SIGNER, CTX, CERT, CERTLEN, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des PKCS#7-Kontexts.	Eingabe
CERT	Speicheradresse des X.509 Zertifikates des vertrauenswürdigen Ausstellers.	Eingabe
CERTLEN	Länge des Zertifikates.	Eingabe

FORCE-TRUSTED-SIGNER

Mit dieser Funktion können vor der Verifizierung von *Signed-data* Objekten Zertifikate mit der gleichen Identität (Issuer- und Subject-BLOB) ausgetauscht werden, die sich bereits im PKCS#7-Object befinden.

Syntax	Cobol	
	<pre>MOVE FORCE-TRUSTED-SIGNER TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION, CTX, CERT, CERTLEN, RC.</pre>	
	Assembler	

	CALL XPSCRYPT, (FORCE_TRUSTED_SIGNER,CTX,CERT,CERTLEN,RC),VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>CERT</i>	Speicheradresse des X.509 Zertifikates des vertrauenswürdigen Ausstellers.	Eingabe
<i>CERTLEN</i>	Länge des Zertifikates.	Eingabe

GET-FIRST-SIGNER

Bei *Signed-data* Objekten können die Informationen über die Unterzeichner (*SignerInfos*) des Objekts extrahiert werden. Mit dieser Funktion wird die erste SignerInfo-Struktur ausgelesen.

Syntax	Cobol	
	<pre>MOVE GET-FIRST-SIGNER TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, SIGNER, RC.</pre>	
	Assembler	
	CALL XPSCRYPT, (GET_FIRST_SIGNER,CTX,SIGNER,RC),VL	
Returncode (RC)	0 falls Unterzeichner vorhanden, sonst kleiner 0.	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>SIGNER</i>	Speicheradresse der ausgelesenen SIGNERINFO. Die Struktur enthält Informationen über Zertifikat, Serial-Nr., Zertifikatsaussteller, Attribute, Message Digest, Digest Algorithm sowie die Signatur. Die Copy-Strecke dieser Struktur ist in dem Copybuch <i>XPSCPCOB</i> bzw. <i>XPSCPCASM</i> unter dem Namen SIGNINFO enthalten.	Ausgabe

GET-NEXT-SIGNER

Bei *Signed-data* Objekten können die Informationen über die Unterzeichner (*SignerInfos*) des Objekts extrahiert werden. Mit dieser Funktion kann jeweils die nächste SignerInfo-Struktur ausgelesen werden.

Syntax	Cobol	
	<pre>MOVE GET-NEXT-SIGNER TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, SIGNER, RC.</pre>	
	Assembler	
	CALL XPSCRYPT, (GET_NEXT_SIGNER,CTX,SIGNER,RC),VL	
Returncode (RC)	0 falls Unterzeichner vorhanden, sonst kleiner 0.	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>SIGNER</i>	Speicheradresse der ausgelesenen SIGNERINFO. Die Struktur enthält Informationen über Zertifikat, Serial-Nr., Zertifikatsaussteller, Attribute, Message Digest, Digest Algorithm sowie die Signatur. Die Copy-Strecke dieser Struktur ist in dem Copybuch <i>XPSCPCOB</i> bzw. <i>XPSCPCASM</i> unter dem Namen SIGNINFO enthalten.	Ausgabe

GET-SIGNING-ALGO

Bei *Signed-data* Objekten können die Art der Verschlüsselung und der Hash-Algorithmus des Dokuments extrahiert werden. Diese Funktion benötigt als Eingabe eine SignerInfo-Struktur, die bei Ausführung der Funktionen *GetFirstSigner* und *GetNextSigner* erzeugt wird.

Syntax	Cobol	
	<pre>MOVE GET-SIGNING-ALGO TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, SIGNER, SIGALGO, ALGOLEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT,(GET_SIGNING_ALGO,CTX,SIGNER,SIGALGO,ALGOLEN,RC),VL</pre>	
Returncode (RC)	Stringlänge der <i>SIGALGO</i> .	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>SIGNER</i>	Adresspointer der Speicheradresse der zuvor ausgelesenen <i>SIGNERINFO</i> .	Eingabe
<i>SIGALGO</i>	Speicheradresse an der die gewünschte Information gespeichert werden soll. Die ermittelte Information wird als Zeichenkette zurückgegeben, bei der der Verschlüsselungsalgorithmus und die Hashmethode durch einen Schrägstrich getrennt sind. Beispiel: "rsaEncryption/sha-1".	Ein-/Ausgabe
<i>ALGOLEN</i>	Länge des von der <i>AlgoInfo</i> belegten Speicherbereichs.	Eingabe

GET-SIGNING-TIME

Bei *Signed-data* Objekten kann der Zeitpunkt der Signatur des Dokuments extrahiert werden. Diese Funktion benötigt als Eingabe eine SignerInfo-Struktur, die bei Ausführung der Funktionen *GetFirstSigner* und *GetNextSigner* erzeugt wird.

Syntax	Cobol	
	<pre>MOVE GET-SIGNING-TIME TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, SIGNER, SIGTIME, TIMELEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT,(GET_SIGNING_TIME,CTX,SIGNER,SIGTIME,TIMELEN,RC),VL</pre>	
Returncode (RC)	Stringlänge von <i>SIGTIME</i> .	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>SIGNER</i>	Adresspointer der Speicheradresse der zuvor ausgelesenen <i>SIGNERINFO</i> .	Eingabe
<i>SIGTIME</i>	Speicheradresse an der die gewünschte Information gespeichert werden soll. Der Zeitpunkt der Erstellung der Signatur wird im folgenden Format zurückgegeben: "YYMMDDHHMMZ". Die einzelnen Positionen haben dabei die folgenden Bedeutungen: YY letzte 2 Stellen des Jahres	Ein-/Ausgabe

	MM Monat (01 bis 12) DD Tag des Monats (01 bis 31) HH Stunde (00 bis 23) MM Minuten (00 bis 59) Z Das Zeichen "Z" bezeichnet Greenwich Mean Time (GMT).	
<i>TIMELEN</i>	Länge des von der <i>SigTime</i> belegten Speicherbereichs.	Eingabe

GET-NEXT-SIGNER-CERT

Mit dieser Funktion wird das nächste Ausstellerzertifikat des aktuellen Unterzeichners ausgelesen.

Syntax	Cobol	
	<pre>MOVE GET-NEXT-SIGNER-CERT TO CRYPT-FUNCTION. CALL 'XPCRYPT' USING CRYPT-FUNCTION, CTX, CERT, RC.</pre>	
	Assembler	
	<pre>CALL XPCRYPT, (GET_NEXT_SIGNER_CERT, CTX, CERT, RC), VL</pre>	
Returncode (RC)	0 falls kein Zertifikat mehr vorhanden, sonst Länge des Zertifikats.	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>CERT</i>	Speicheradresse der ausgelesenen Unterzeichner Zertifikats.	Ausgabe

VERIFY-SIGNER

Das *Signed-data* Objekt wird auf Gültigkeit überprüft. Bei dieser Funktion wird jedoch nicht die Gesamtheit überprüft, sondern nur die Gültigkeit dieses bestimmten Unterzeichners. Die *SIGNERINFO*-Struktur muss vorher mit der Funktion *GET-FIRST-SIGNER/GET-NEXT-SIGNER* ermittelt worden sein.

Syntax	Cobol	
	<pre>MOVE VERIFY-SIGNER TO CRYPT-FUNCTION. CALL 'XPCRYPT' USING CRYPT-FUNCTION, CTX, SIGNER, CHECKCERT, RC.</pre>	
	Assembler	
	<pre>CALL XPCRYPT, (VERIFY_SIGNER, CTX, SIGNER, CHECKCERT, RC), VL</pre>	
Returncode (RC)	0 falls Unterzeichner verifiziert, sonst Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>SIGNER</i>	Speicheradresse der zu verifizierenden <i>SIGNERINFO</i> Struktur.	Eingabe
<i>CHECKCERT</i>	Bei Angabe von 0 werden der Zertifikatspfad des Unterzeichners sowie vertrauenswürdige Aussteller nicht überprüft. Bei Angabe von 1 wird der Zertifikatspfad bis zum Wurzelzertifikat überprüft. Ausserdem muss der Zertifikatsaussteller mit der Funktion <i>ADD-TRUSTED-SIGNER</i> vorher geladen worden sein.	Eingabe

VERIFY-ALL-SIGNER

Alle Unterzeichner des *Signed-data* Objekts werden auf Gültigkeit überprüft.

Syntax	Cobol	
	<pre>MOVE VERIFY-ALL-SIGNER TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, CHECKCERT, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (VERIFY_ALL_SIGNER, CTX, CHECKCERT, RC), VL</pre>	
Returncode (RC)	0 falls alle Unterzeichner verifiziert, sonst Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>CHECKCERT</i>	Bei Angabe von 0 werden der Zertifikatspfad des Unterzeichners sowie vertrauenswürdige Aussteller nicht überprüft. Bei Angabe von 1 wird der Zertifikatspfad bis zum Wurzelzertifikat überprüft. Ausserdem muss der Zertifikatsaussteller mit der Funktion <i>ADD-TRUSTED-SIGNER</i> vorher geladen worden sein.	Eingabe

GET-FIRST-PKCS7-DATA

Die Daten des PKCS#7-Objekts werden ausgelesen. Diese Funktion ist bei allen unterstützten PKCS#7-Typen vorhanden.

Syntax	Cobol	
	<pre>MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, DATA, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_FIRST_PKS7_DATA, CTX, DATA, RC), VL</pre>	
Returncode (RC)	Länge der Daten, bei 0 keine Daten vorhanden, sonst Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>DATA</i>	Speicheradresse der ausgelesenen Daten.	Ausgabe

GET-NEXT-PKCS7-DATA

Die nächsten Daten des PKCS#7-Objekts werden ausgelesen. Diese Funktion ist bei allen unterstützten PKCS#7-Typen vorhanden.

Syntax	Cobol	
	<pre>MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, DATA, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_NEXT_PKS7_DATA, CTX, DATA, RC), VL</pre>	
Returncode (RC)	Länge der Daten, bei 0 keine Daten vorhanden, sonst Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#7-Kontexts.	Eingabe
<i>DATA</i>	Speicheradresse der ausgelesenen Daten.	Ausgabe

CREATE-OBJECT

Mit dieser Funktion wird das Erstellen von PKCS#7-Objekten abgeschlossen.

Syntax	Cobol	
	MOVE CREATE-OBJECT TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, OBJECT, RC.	
	Assembler	
	CALL XPSCRYPT, (CREATE_OBJECT , CTX , OBJECT , RC) , VL	
Returncode (RC)	Länge des PKCS#7-Datenobjekts oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des PKCS#7-Kontexts.	Eingabe
OBJECT	Speicheradresse des erstellten PKCS#7-Objekts.	Ausgabe

CLEANUP-PKCS7

Freigeben des von den PKCS#7-Routinen benötigten Speichers.

Syntax	Cobol	
	MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, RC.	
	Assembler	
	CALL XPSCRYPT, (CLEANUP_PKCS7 , CTX , RC) , VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des PKCS#7-Kontexts.	Eingabe

COBOL-Beispiel (Create PKCS7Data):

```

*-----*
*      CREATE PKCS-7 DATA OBJECT      *
*-----*
ID DIVISION.
PROGRAM-ID.
    PK7WR1C.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)          VALUE "XPSDATA".
01 PKCS7-FILE      PIC X(8)          VALUE "PK7DA".
01 DATA1          PIC X(32)         VALUE "XPS Software GmbH, Haar
-                                     "/Muenchen".
01 DATA2          PIC X(21)         VALUE "Muenchener Strasse 17".
01 PKCS7-CTX       POINTER.
01 ADDR-PKCS7-OBJ POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION  PIC X.
77 OPTION          PIC X.
77 OBJECT-LENGTH  PIC 9(8) COMP VALUE ZEROES.
77 DATALEN1      PIC 9(8) COMP VALUE 32.
77 DATALEN2      PIC 9(8) COMP VALUE 21.
77 RC             PIC 9(8) COMP VALUE ZEROES.
77 RCC            PIC 9(8) COMP VALUE ZEROES.
*
LINKAGE SECTION.
*
COPY XPSCLCOB.
    
```

```

*****
**          PROCEDURE DIVISION          **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
* CREATE PKCS-7 DATA-OBJECT           *
*-----*
      MOVE HEADER-INCLUDED TO OPTION.
      MOVE CREATE-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, PKCS7-CTX, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR CREATE-PKCS7-DATA: RC = " RCC
          GOBACK.
*-----*
* ADD DATA TO PKCS7 DATA-OBJECT      *
*-----*
      MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, DATA1, DATALEN1, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
          GOBACK.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, DATA2, DATALEN2, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
          GOBACK.
*-----*
* CREATE PKCS-7 DATA-OBJECT           *
*-----*
      MOVE CREATE-OBJECT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, ADDR-PKCS7-OBJ RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR CREATE-OBJECT: RC = " RCC
          GOBACK.
      MOVE RC TO OBJECT-LENGTH.
*-----*
* WRITE PKCS-7 DATA-OBJECT TO MACLIB  *
*-----*
      MOVE WRITE-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          DDNAME, PKCS7-FILE, ADDR-PKCS7-OBJ, OBJECT-LENGTH, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR PUT-FILE: RC = " RCC
          GOBACK.
*-----*
* CLEANUP PKCS-7 CONTEXT               *
*-----*
      MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, PKCS7-CTX, RC.
      STOP RUN.
ENDRUN.

```

COBOL-Beispiel (Read PKCS7Data):

```

*-----*
*          TEST CHECK PKCS-7 DATA OBJECT          *
*-----*
ID DIVISION.
PROGRAM-ID.
    PK7RD1C.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)          VALUE "XPSDATA".
01 PKCS7-FILE      PIC X(8)          VALUE "PK7DA".
01 HEADER1         PIC X(20)         VALUE "DATA:".
01 PKCS7-CTX       POINTER.
01 ADDR-PKCS7      POINTER.
01 ADDR-DATA       POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION  PIC X.
77 PKCS7LEN        PIC 9(8)          COMP VALUE ZEROES.
77 DUMPLEN         PIC 9(8)          COMP VALUE ZEROES.
77 RC              PIC 9(8)          COMP VALUE ZEROES.
77 RCC             PIC 9(8)          VALUE ZEROES.

```

```

*
LINKAGE SECTION.
*
COPY XPSCLCOB.
01 PKCS7-DATA      PIC X(1).
*****
**                PROCEDURE DIVISION                **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
* READ PKCS-7 DATA OBJECT "PK7DA" FROM MACLIB      *
*-----*
      MOVE READ-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            DDNAME, PKCS7-FILE, ADDR-PKCS7, PKCS7LEN, RC.
      IF RC < 0
        MOVE RC TO RCC
        DISPLAY "FILE 'PK7DA' NOT FOUND RC = " RCC
        GOBACK.
*-----*
* IMPORT PKCS-7 DATA OBJECT                          *
*-----*
      MOVE IMPORT-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7, PKCS7LEN,
            PKCS7-CTX, RC.
      IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR IMPORT-PKCS-DATA: RC = " RCC
        GOBACK.
*-----*
* GET ALL DATA                                       *
*-----*
      MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            PKCS7-CTX, ADDR-DATA, RC.
      PERFORM UNTIL RC <= ZEROS
        PERFORM GET-DATA
      END-PERFORM.
*-----*
* CLEANUP PKCS-7 CONTEXT                             *
*-----*
      MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, PKCS7-CTX, RC.
*-----*
* RELEASE FILE-STORAGE "PK7DA"                       *
*-----*
      MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            ADDR-PKCS7, RC.
      IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR RELEASE-FILE: RC = " RCC
        GOBACK.
      STOP RUN.
*
GET-DATA SECTION.
      SET ADDRESS OF PKCS7-DATA TO ADDR-DATA.
      MOVE RC TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            HEADER1, PKCS7-DATA, DUMPLEN, RC.
      MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            PKCS7-CTX, ADDR-DATA, RC.
GET-DATA-END.
EXIT.
ENDRUN.

```

COBOL-Beispiel (Create SignedData):

```

*-----*
* CREATE PKCS-7 SIGNED DATA OBJECT                  *
*-----*
ID DIVISION.
PROGRAM-ID.
      PK7WR2C.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)          VALUE "XPSDATA".
01 PKCS7-FILE      PIC X(8)          VALUE "PK7SD".
01 PKCS12-FILE     PIC X(8)          VALUE "XPSUSERP".
01 PWD             PIC X(8)          VALUE "xpsuser1".
01 DATA1          PIC X(32)         VALUE "XPS Software GmbH, Haar

```

```

-
01 DATA2          PIC X(21)          VALUE "/Muenchen".
01 DATA3          PIC X(10)         VALUE "Muenchener Strasse 17".
01 ADDR-PKCS12     POINTER.
01 PKCS7-CTX       POINTER.
01 PEM-CTX         POINTER.
01 ADDR-PKCS7-OBJ  POINTER.
01 ADDR-PEM-OBJ    POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION  PIC X.
77 OPTION          PIC X.
77 PKCS12-LENGTH   PIC 9(8)          COMP VALUE ZEROES.
77 PWDLEN          PIC 9(8)          COMP VALUE 8.
77 OBJECT-LENGTH   PIC 9(8)          COMP VALUE ZEROES.
77 PEM-LENGTH      PIC 9(8)          COMP VALUE ZEROES.
77 DATALEN1       PIC 9(8)          COMP VALUE 32.
77 DATALEN2       PIC 9(8)          COMP VALUE 21.
77 DATALEN3       PIC 9(8)          COMP VALUE 10.
77 RC              PIC 9(8)          COMP VALUE ZEROES.
77 RCC             PIC 9(8)          VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**                PROCEDURE DIVISION                **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* READ PKCS-12 FILE "XPSUSERP" FROM MACLIB          *
* ----- *
      MOVE READ-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            DDNAME, PKCS12-FILE, ADDR-PKCS12, PKCS12-LENGTH, RC.
      IF RC < 0
        MOVE RC TO RCC
        DISPLAY "FILE 'XPSUSERP' NOT FOUND RC = " RCC
        GOBACK.
* ----- *
* CONVERT PASSWORD/DATA FROM EBCDIC TO ASCII        *
* ----- *
      MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
      CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, DATA1, DATALEN1, RC.
      CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, DATA2, DATALEN2, RC.
      CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, DATA3, DATALEN3, RC.
* ----- *
* CREATE PKCS-7 SIGNED-DATA-OBJECT                  *
* ----- *
      MOVE DATA-CERT-HEADER TO OPTION.
      MOVE CREATE-SIGNED-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, PKCS7-CTX, RC.
      IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR CREATE-SIGNED-DATA: RC = " RCC
        GOBACK.
* ----- *
* ADD DATA TO PKCS7 SIGNED-DATA-OBJECT              *
* ----- *
      MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            PKCS7-CTX, DATA1, DATALEN1, RC.
      IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
        GOBACK.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            PKCS7-CTX, DATA2, DATALEN2, RC.
      IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
        GOBACK.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            PKCS7-CTX, DATA3, DATALEN3, RC.
      IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
        GOBACK.
* ----- *
* ADD SIGNER TO PKCS-7 SIGNED-DATA-OBJECT           *
* ----- *
      MOVE ADD-SIGNER TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, PKCS7-CTX,
            ADDR-PKCS12, PKCS12-LENGTH, PWD, PWDLEN, RC.

```

```

IF RC < 0
  MOVE RC TO RCC
  DISPLAY "ERROR ADD-SIGNER: RC = " RCC
  GOBACK.
* -----*
* CREATE PKCS-7 SIGNED-DATA-OBJECT *
* -----*
  MOVE CREATE-OBJECT TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    PKCS7-CTX, ADDR-PKCS7-OBJ RC.
  IF RC < 0
    MOVE RC TO RCC
    DISPLAY "ERROR CREATE-OBJECT: RC = " RCC
    GOBACK.
  MOVE RC TO OBJECT-LENGTH.
* -----*
* CONVERT ASN1-FORMAT TO PEM-FORMAT *
* -----*
  MOVE ASN-2-PEM TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7-OBJ,
    OBJECT-LENGTH, PKCS7-FILE, ADDR-PEM-OBJ, PEM-CTX, RC.
  IF RC < 0
    MOVE RC TO RCC
    DISPLAY "ERROR ASN-2-PEM: RC = " RCC
    GOBACK.
  MOVE RC TO PEM-LENGTH.
* -----*
* WRITE PKCS-7 SIGNED-DATA-OBJECT TO MACLIB *
* -----*
  MOVE WRITE-FILE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    DDNAME, PKCS7-FILE, ADDR-PEM-OBJ, PEM-LENGTH, RC.
  IF RC < 0
    MOVE RC TO RCC
    DISPLAY "ERROR PUT-FILE: RC = " RCC
    GOBACK.
* -----*
* CLEANUP PKCS-7 CONTEXT *
* -----*
  MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT'
    USING CRYPT-FUNCTION, PKCS7-CTX, RC.
  MOVE CLEANUP-PEM TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT'
    USING CRYPT-FUNCTION, PEM-CTX, RC.
* -----*
* RELEASE FILE-STORAGES *
* -----*
  MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    ADDR-PKCS12, RC.
  STOP RUN.
ENDRUN.

```

COBOL-Beispiel (Read SignedData):

```

* -----*
* TEST CHECK PKCS-7 SIGNED-DATA OBJECT *
* -----*
ID DIVISION.
PROGRAM-ID.
  PK7RD2C.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)          VALUE "XPSDATA".
01 PKCS7-FILE      PIC X(8)          VALUE "PK7SD".
01 CERT-FILE       PIC X(8)          VALUE "XPSTESTC".
01 HEADER1         PIC X(20)         VALUE "SIGNER-CERTIFICATE:".
01 HEADER2         PIC X(20)         VALUE "DATA:".
01 PKCS7-CTX       POINTER.
01 CERT-CTX        POINTER.
01 ADDR-PKCS7      POINTER.
01 ADDR-CERT       POINTER.
01 ADDR-DATA       POINTER.
01 ADDR-SIGNER     POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION  PIC X.
77 DUMPLEN        PIC 9(8) COMP VALUE ZEROES.
77 PKCS7LEN       PIC 9(8) COMP VALUE ZEROES.
77 CERTLEN        PIC 9(8) COMP VALUE ZEROES.
77 BOOLEAN-TRUE   PIC 9(8) COMP VALUE 1.
77 RC             PIC 9(8) COMP VALUE ZEROES.
77 RCC           PIC 9(8) COMP VALUE ZEROES.

```

```

*
LINKAGE SECTION.
COPY XPSCLCOB.
01 PKCS7-DATA      PIC X(1).
01 SIGNER-CERT    PIC X(1).
*****
**                PROCEDURE DIVISION                **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* READ PKCS-7 SIGNED DATA OBJECT "PK7SD" FROM MACLIB *
* ----- *
      MOVE READ-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           DDNAME, PKCS7-FILE, ADDR-PKCS7, PKCS7LEN, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "FILE 'PK7SD' NOT FOUND RC = " RCC
          GOBACK.
* ----- *
* READ TRUSTED-SIGNER CERTIFICATE "XPSTESTC" *
* ----- *
      MOVE READ-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           DDNAME, CERT-FILE, ADDR-CERT, CERTLEN, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "FILE 'PK7SD' NOT FOUND RC = " RCC
          GOBACK.
* ----- *
* IMPORT SIGNED DATA OBJECT *
* ----- *
      MOVE IMPORT-SIGNED-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7, PKCS7LEN,
           PKCS7-CTX, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR IMPORT-SIGNED-DATA: RC = " RCC
          GOBACK.
* ----- *
* ADD TRUSTED SIGNER CERTIFICATE *
* ----- *
      MOVE ADD-TRUSTED-SIGNER TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           PKCS7-CTX, ADDR-CERT, CERTLEN, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR ADD-TRUSTED-SIGNER: RC = " RCC
          GOBACK.
* ----- *
* GET ALL SIGNERS *
* ----- *
      MOVE GET-FIRST-SIGNER TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           PKCS7-CTX, ADDR-SIGNER, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR GET-FIRST-SIGNER: RC = " RCC
          GOBACK.
      PERFORM UNTIL RC < ZEROES
          PERFORM GET-SIGNERS
      END-PERFORM.
* ----- *
* VERIFY SIGNER *
* ----- *
      MOVE VERIFY-SIGNER TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           PKCS7-CTX, ADDR-SIGNER, BOOLEAN-TRUE, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR VERIFY-SIGNER: RC = " RCC
          GOBACK.
* ----- *
* GET ALL DATA *
* ----- *
      MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           PKCS7-CTX, ADDR-DATA, RC.
      PERFORM UNTIL RC <= ZEROES
          PERFORM GET-DATA
      END-PERFORM.
* ----- *
* CLEANUP PKCS-7 CONTEXT *
* ----- *
      MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
           USING CRYPT-FUNCTION, PKCS7-CTX, RC.
* ----- *
* RELEASE FILE-STORAGES *
* ----- *

```

```

* ----- *
      MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          ADDR-PKCS7, RC.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          ADDR-CERT, RC.
      STOP RUN.
* ----- *
* PERFORM ROUTINES *
* ----- *
GET-SIGNERS SECTION.
      SET ADDRESS OF SIGNINFO TO ADDR-SIGNER.
      SET ADDRESS OF SIGNER-CERT TO S-CERT.
      MOVE S-LCERT TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER1, SIGNER-CERT, DUMPLEN, RC.
      MOVE GET-NEXT-SIGNER TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, ADDR-SIGNER, RC.
GET-SIGNERS-END.
      EXIT.
*
GET-DATA SECTION.
      SET ADDRESS OF PKCS7-DATA TO ADDR-DATA.
      MOVE RC TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER2, PKCS7-DATA, DUMPLEN, RC.
      MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, ADDR-DATA, RC.
GET-DATA-END.
      EXIT.
ENDRUN.

```

COBOL-Beispiel (Create EnvelopedData):

```

* ----- *
*       CREATE PKCS-7 ENVELOPED DATA OBJECT *
* ----- *
ID DIVISION.
PROGRAM-ID.
    PK7WR3C.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)          VALUE "XPSDATA" .
01 PKCS7-FILE     PIC X(8)          VALUE "PK7EV" .
01 CERT-FILE      PIC X(8)          VALUE "XPSUSERC" .
01 DATA1         PIC X(32)         VALUE "XPS Software GmbH, Haar
-                                     "/Muenchen" .
01 ADDR-CERT      POINTER.
01 PKCS7-CTX     POINTER.
01 PEM-CTX       POINTER.
01 ADDR-PKCS7-OBJ POINTER.
01 ADDR-PEM-OBJ  POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION PIC X.
77 OPTION         PIC X.
77 CRYPT-TYPE     PIC X.
77 CERT-LENGTH   PIC 9(8)          COMP VALUE ZEROES.
77 OBJECT-LENGTH PIC 9(8)          COMP VALUE ZEROES.
77 PEM-LENGTH     PIC 9(8)          COMP VALUE ZEROES.
77 DATALEN1     PIC 9(8)          COMP VALUE 32.
77 RC             PIC 9(8)          COMP VALUE ZEROES.
77 RCC           PIC 9(8)          VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**              PROCEDURE DIVISION              **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* READ RECIPIENT CERTIFICATE-FILE 'XPSUSERC' FROM MACLIB *
* ----- *
      MOVE READ-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          DDNAME, CERT-FILE, ADDR-CERT, CERT-LENGTH, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "FILE 'XPSUSERC' NOT FOUND RC = " RCC

```

```

      GOBACK.
* -----*
* CONVERT DATA FROM EBCDIC TO ASCII *
* -----*
      MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, DATA1, DATALEN1, RC.
* -----*
* CREATE PKCS-7 ENVELOPED-DATA-OBJECT *
* -----*
      MOVE HEADER-INCLUDED TO OPTION.
      MOVE DESEDE3CBC      TO CRYPT-TYPE.
      MOVE CREATE-ENVELOPED-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, PKCS7-CTX,
          CRYPT-TYPE, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR CREATE-ENVELOPED-DATA: RC = " RCC
          GOBACK.
* -----*
* ADD DATA TO PKCS7 ENVELOPED-DATA-OBJECT *
* -----*
      MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, DATA1, DATALEN1, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
          GOBACK.
* -----*
* ADD RECIPIENT TO PKCS-7 ENVELOPED-DATA-OBJECT *
* -----*
      MOVE ADD-RECIPIENT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, PKCS7-CTX,
          ADDR-CERT, CERT-LENGTH, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR ADD-RECIPIENT: RC = " RCC
          GOBACK.
* -----*
* CREATE PKCS-7 ENVELOPED-DATA-OBJECT *
* -----*
      MOVE CREATE-OBJECT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, ADDR-PKCS7-OBJ RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR CREATE-OBJECT: RC = " RCC
          GOBACK.
      MOVE RC TO OBJECT-LENGTH.
* -----*
* CONVERT ASN1-FORMAT TO PEM-FORMAT *
* -----*
      MOVE ASN-2-PEM TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7-OBJ,
          OBJECT-LENGTH, PKCS7-FILE, ADDR-PEM-OBJ, PEM-CTX, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR ASN-2-PEM: RC = " RCC
          GOBACK.
      MOVE RC TO PEM-LENGTH.
* -----*
* WRITE PKCS-7 ENVELOPED-DATA-OBJECT TO MACLIB *
* -----*
      MOVE WRITE-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          DDNAME, PKCS7-FILE, ADDR-PEM-OBJ, PEM-LENGTH, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR PUT-FILE: RC = " RCC
          GOBACK.
* -----*
* CLEANUP PKCS-7 CONTEXT *
* -----*
      MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, PKCS7-CTX, RC.
      MOVE CLEANUP-PEM TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, PEM-CTX, RC.
* -----*
* RELEASE FILE-STORAGES *
* -----*
      MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          ADDR-CERT, RC.
      STOP RUN.
ENDRUN.

```

COBOL-Beispiel (Read EnvelopedData):

```

-----*
*   TEST CHECK PKCS-7 ENVELOPED-DATA OBJECT   *
-----*
ID DIVISION.
PROGRAM-ID.
  PK7RD3C.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)          VALUE "XPSDATA" .
01 PKCS7-FILE      PIC X(8)          VALUE "PK7EV" .
01 PKCS12-FILE     PIC X(8)          VALUE "XPSUSERP" .
01 PWD             PIC X(8)          VALUE "xpsuser1" .
01 HEADER1        PIC X(20)         VALUE "DATA:".
01 PKCS7-CTX      POINTER.
01 ADDR-PKCS7     POINTER.
01 ADDR-PKCS12    POINTER.
01 ADDR-DATA      POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION PIC X.
77 DUMPLEN        PIC 9(8)          COMP VALUE ZEROES.
77 PKCS7LEN       PIC 9(8)          COMP VALUE ZEROES.
77 PKCS12LEN      PIC 9(8)          COMP VALUE ZEROES.
77 PWDLEN         PIC 9(8)          COMP VALUE 8.
77 RC             PIC 9(8)          COMP VALUE ZEROES.
77 RCC           PIC 9(8)          VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
01 PKCS7-DATA     PIC X(1).
*****
**      PROCEDURE DIVISION      **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
* CONVERT PASSWORD FROM EBCDIC TO ASCII (PASSWORD LOWER CASE|) *
*-----*
  MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT'
    USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
*-----*
* READ PKCS-7 ENVELOPED DATA OBJECT "PK7EV" FROM MACLIB *
*-----*
  MOVE READ-FILE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    DDNAME, PKCS7-FILE, ADDR-PKCS7, PKCS7LEN, RC.
  IF RC < 0
    MOVE RC TO RCC
    DISPLAY "FILE 'PK7EV' NOT FOUND RC = " RCC
    GOBACK.
*-----*
* READ RECIPIENT PKCS12-FILE "XPSUSERP" FROM MACLIB *
*-----*
  MOVE READ-FILE TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    DDNAME, PKCS12-FILE, ADDR-PKCS12, PKCS12LEN, RC.
  IF RC < 0
    MOVE RC TO RCC
    DISPLAY "FILE 'XPSUSERP' NOT FOUND RC = " RCC
    GOBACK.
*-----*
* IMPORT ENVELOPED DATA OBJECT *
*-----*
  MOVE IMPORT-ENVELOPED-DATA TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7, PKCS7LEN,
    ADDR-PKCS12, PKCS12LEN, PWD, PWDLEN, PKCS7-CTX, RC.
  IF RC < 0
    MOVE RC TO RCC
    DISPLAY "ERROR IMPORT-ENVELOPED-DATA: RC = " RCC
    GOBACK.
*-----*
* GET ALL DATA *
*-----*
  MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
    PKCS7-CTX, ADDR-DATA, RC.
  PERFORM UNTIL RC <= ZEROES
    PERFORM GET-DATA
  END-PERFORM.
*-----*
* CLEANUP PKCS-7 CONTEXT *
*-----*
  MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
  CALL 'XPSCRYPT'

```

```

        USING CRYPT-FUNCTION, PKCS7-CTX, RC.
* -----*
* RELEASE FILE-STORAGES *
* -----*
        MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            ADDR-PKCS7, RC.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            ADDR-PKCS12, RC.
        STOP RUN.
*
* -----*
* PERFORM ROUTINES *
* -----*
GET-DATA SECTION.
        SET ADDRESS OF PKCS7-DATA TO ADDR-DATA.
        MOVE RC TO DUMPLEN.
        MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            HEADER1, PKCS7-DATA, DUMPLEN, RC.
        MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            PKCS7-CTX, ADDR-DATA, RC.
GET-DATA-END.
        EXIT.
ENDRUN.

```

COBOL-Beispiel (Create EncryptedData):

```

*-----*
* CREATE PKCS-7 ENCRYPTED DATA OBJECT *
*-----*
ID DIVISION.
PROGRAM-ID.
    PK7WR4C.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DNAME          PIC X(8)          VALUE "XPSDATA".
01 PKCS7-FILE     PIC X(8)          VALUE "PK7EC".
01 DATA1         PIC X(32)         VALUE "XPS Software GmbH, Haar
-                                     "/Muenchen".
01 PWD            PIC X(12)         VALUE "testpassword".
01 PKCS7-CTX     POINTER.
01 PEM-CTX       POINTER.
01 ADDR-PKCS7-OBJ POINTER.
01 ADDR-PEM-OBJ  POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION PIC X.
77 OPTION         PIC X.
77 CRYPT-TYPE     PIC X.
77 OBJECT-LENGTH PIC 9(8) COMP VALUE ZEROES.
77 PEM-LENGTH     PIC 9(8) COMP VALUE ZEROES.
77 DATALEN1     PIC 9(8) COMP VALUE 32.
77 PWDLEN        PIC 9(8) COMP VALUE 12.
77 RC            PIC 9(8) COMP VALUE ZEROES.
77 RCC           PIC 9(8) COMP VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
*****
**          PROCEDURE DIVISION          **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
* CONVERT PWD/DATA FROM EBCDIC TO ASCII *
*-----*
        MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
        CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, DATA1, DATALEN1, RC.
        CALL 'XPSCRYPT'
*-----*
* CREATE PKCS-7 ENCRYPTED-DATA-OBJECT *
*-----*
        MOVE HEADER-INCLUDED TO OPTION.
        MOVE PBE3DES-3KEY TO CRYPT-TYPE.
        MOVE CREATE-ENCRYPTED-DATA TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, PKCS7-CTX,
            CRYPT-TYPE, PWD, PWDLEN, RC.
        IF RC < 0
            MOVE RC TO RCC

```

```

        DISPLAY "ERROR CREATE-ENCRYPTED-DATA: RC = " RCC
        GOBACK.
* -----*
* ADD DATA TO PKCS7 ENCRYPTED-DATA-OBJECT *
* -----*
        MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            PKCS7-CTX, DATA1, DATALEN1, RC.
        IF RC < 0
            MOVE RC TO RCC
            DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
            GOBACK.
* -----*
* CREATE PKCS-7 ENCRYPTED-DATA-OBJECT *
* -----*
        MOVE CREATE-OBJECT TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            PKCS7-CTX, ADDR-PKCS7-OBJ RC.
        IF RC < 0
            MOVE RC TO RCC
            DISPLAY "ERROR CREATE-OBJECT: RC = " RCC
            GOBACK.
        MOVE RC TO OBJECT-LENGTH.
* -----*
* CONVERT ASN1-FORMAT TO PEM-FORMAT *
* -----*
        MOVE ASN-2-PEM TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7-OBJ,
            OBJECT-LENGTH, PKCS7-FILE, ADDR-PEM-OBJ, PEM-CTX, RC.
        IF RC < 0
            MOVE RC TO RCC
            DISPLAY "ERROR ASN-2-PEM: RC = " RCC
            GOBACK.
        MOVE RC TO PEM-LENGTH.
* -----*
* WRITE PKCS-7 ENCRYPTED-DATA-OBJECT TO MACLIB *
* -----*
        MOVE WRITE-FILE TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
            DDNAME, PKCS7-FILE, ADDR-PEM-OBJ, PEM-LENGTH, RC.
        IF RC < 0
            MOVE RC TO RCC
            DISPLAY "ERROR PUT-FILE: RC = " RCC
            GOBACK.
* -----*
* CLEANUP PKCS-7 CONTEXT *
* -----*
        MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, PKCS7-CTX, RC.
        MOVE CLEANUP-PEM TO CRYPT-FUNCTION.
        CALL 'XPSCRYPT'
            USING CRYPT-FUNCTION, PEM-CTX, RC.
        STOP RUN.
ENDRUN.

```

COBOL-Beispiel (Read EncryptedData):

```

* -----*
* TEST CHECK PKCS-7 ENCRYPTED-DATA OBJECT *
* -----*
ID DIVISION.
PROGRAM-ID.
    PK7RD4C.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DDNAME          PIC X(8)          VALUE "XPSDATA".
01 PKCS7-FILE      PIC X(8)          VALUE "PK7EC".
01 PWD             PIC X(12)         VALUE "testpassword".
01 HEADER1        PIC X(20)         VALUE "DATA:".
01 PKCS7-CTX      POINTER.
01 ADDR-PKCS7     POINTER.
01 ADDR-DATA      POINTER.
*
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION  PIC X.
77 DUMPLEN        PIC 9(8)          COMP VALUE ZEROES.
77 PKCS7LEN       PIC 9(8)          COMP VALUE ZEROES.
77 PWDLEN         PIC 9(8)          COMP VALUE 12.
77 RC             PIC 9(8)          COMP VALUE ZEROES.
77 RCC           PIC 9(8)          COMP VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.

```

```

01 PKCS7-DATA      PIC X(1).
*****
**                PROCEDURE DIVISION                **
*****
PROCEDURE DIVISION.
MAIN SECTION.
* ----- *
* CONVERT PASSWORD FROM EBCDIC TO ASCII (PASSWORD LOWER CASE|) *
* ----- *
      MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
* ----- *
* READ PKCS-7 ENCRYPTED DATA OBJECT "PK7EC" FROM MACLIB          *
* ----- *
      MOVE READ-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          DDNAME, PKCS7-FILE, ADDR-PKCS7, PKCS7LEN, RC.
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "FILE 'PK7EC' NOT FOUND RC = " RCC
          GOBACK.
* ----- *
* IMPORT ENCRYPTED DATA OBJECT                                  *
* ----- *
      MOVE IMPORT-ENCRYPTED-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7, PKCS7LEN,
          PWD, PWDLEN, PKCS7-CTX, RC.
*
      IF RC < 0
          MOVE RC TO RCC
          DISPLAY "ERROR IMPORT-ENCRYPTED-DATA: RC = " RCC
          GOBACK.
* ----- *
* GET ALL DATA                                                *
* ----- *
      MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, ADDR-DATA, RC.
      PERFORM UNTIL RC <= ZEROES
          PERFORM GET-DATA
      END-PERFORM.
* ----- *
* CLEANUP PKCS-7 CONTEXT                                       *
* ----- *
      MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, PKCS7-CTX, RC.
* ----- *
* RELEASE FILE-STORAGES                                        *
* ----- *
      MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          ADDR-PKCS7, RC.
      STOP RUN.
* ----- *
* PERFORM ROUTINES                                             *
* ----- *
GET-DATA SECTION.
      SET ADDRESS OF PKCS7-DATA TO ADDR-DATA.
      MOVE RC TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER1, PKCS7-DATA, DUMPLEN, RC.
      MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS7-CTX, ADDR-DATA, RC.
GET-DATA-END.
      EXIT.
ENDRUN.

```

PKCS#12 private Key

Allgemein

PKCS#12-Objekte (*Personal-Information-Exchange-Syntax-Standard*) stellen eine Syntax für den Austausch von Schlüsseln und Zertifikaten zur Verfügung. Ein PKCS#12-Objekt enthält Schlüsseltaschen (key-bags) und Zertifikatstaschen (certificate-bags). PKCS#12 ist der Standard zum sicheren Speichern von Privaten Schlüsseln und Zertifikaten. Es wird vor allem von Internet Browsern wie Netscape (Exportformat .p12) und Microsoft Internet Explorer (Exportformat .pfx) verwendet.

Funktionen

IMPORT-PKCS12

Einlesen eines PKCS#12-Objekts sowie Prüfung auf formale Richtigkeit.

Syntax	Cobol	
	<pre>MOVE IMPORT-PKCS12 TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, APKCS12, PKCS12LEN, PWD, PWDLEN, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (IMPORT_PKCS12,APKCS12,PKCS12LEN,PWD,PWDLEN, CTX,RC),VL</pre>	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
APKCS12	Speicheradresse des PKCS#12-Objekts.	Eingabe
PKCS12LEN	Länge des PKCS#12-Objekts.	Eingabe
PWD	Speicheradresse des Passwortes, mit dem das PKCS#12-Objekts verschlüsselt wurde.	Eingabe
PWDLEN	Länge des Passwortes.	Eingabe
CTX	Speicheradresse des eingelesenen PKCS#12-Objekts. Dieses wird für die weitere Bearbeitung des Objekts benötigt.	Ausgabe

GET-PRIVATE-KEY

Extrahieren des geheimen Schlüssels aus dem PKCS#12-Objekt.

Syntax	Cobol	
	<pre>MOVE GET-PRIVATE-KEY TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, PRIVKEY, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_PRIVATE_KEY, CTX, PRIVKEY, RC), VL</pre>	
Returncode (RC)	Länge der privateKey Struktur oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#12-Kontexts.	Eingabe
<i>PRIVKEY</i>	Speicheradresse, an der der extrahierte geheime Schlüssel gespeichert werden soll. Die Definition der Struktur befindet sich im Copybuch <i>XPSCLRSA</i> bzw. <i>XPSCCLASM</i> (COBOL/Assembler).	Ausgabe

GET-FIRST-CERT

Extrahieren des Benutzerzertifikates aus dem PKCS#12-Objekt.

Syntax	Cobol	
	<pre>MOVE GET-FIRST-CERT TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, CERT, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_FIRST_CERT, CTX, CERT, RC), VL</pre>	
Returncode (RC)	Länge des X.509 Zertifikates oder 0, falls kein Zertifikat vorhanden.	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#12-Kontexts.	Eingabe
<i>CERT</i>	Speicheradresse des extrahierten X.509 Zertifikats.	Ausgabe

GET-NEXT-CERT

Extrahieren des nächsten Zertifikates aus dem PKCS#12-Objekt. Ein PKCS#12-Objekt kann neben dem Benutzerzertifikat sämtliche Ausstellerzertifikate bis hin zum Wurzelzertifikat enthalten.

Syntax	Cobol	
	<pre>MOVE GET-NEXT-CERT TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, CERT, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (GET_NEXT_CERT, CTX, CERT, RC), VL</pre>	
Returncode (RC)	Länge des X.509 Zertifikates oder 0, falls kein Zertifikat vorhanden.	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des PKCS#12-Kontexts.	Eingabe

CERT	Speicheradresse des extrahierten X.509 Zertifikats.	Ausgabe
------	---	---------

CLEANUP-PKCS12

Freigeben des bei den PKCS#12-Routinen benötigten Speichers.

Syntax	Cobol	
	MOVE CLEANUP-PKCS12 TO CRYPT-FUNCTION. CALL 'XPCRYPT' USING CRYPT-FUNCTION, CTX, RC.	
	Assembler	
	CALL XPCRYPT, (CLEANUP_PKCS12,CTX,RC),VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
CTX	Speicheradresse des PKCS#12-Kontexts.	Eingabe

COBOL-Beispiel:

```

*-----*
*   XPS-CRYPTLIB SAMPLE PROGRAM: GET PRIVATE-KEY   *
*-----*
ID DIVISION.
PROGRAM-ID.
    PK12TSTC.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 PASSWRD          PIC X(32)          VALUE "xpsuser1".
01 PKCS12-FILE      PIC X(8)          VALUE "XPSUSERP".
01 DDNAME           PIC X(8)          VALUE "XPSDATA".
01 PKCS12-CTX       POINTER.
01 ADDR-PKCS12-FILE POINTER.
01 ADDR-CERTIFICATE POINTER.
COPY XPSCLRSA.
*
77 CRYPT-FUNCTION   PIC X.
77 DATALEN        PIC 9(8)  COMP VALUE ZEROES.
77 PWDLEN          PIC 9(8)  COMP VALUE 8.
77 RC              PIC 9(8)  COMP VALUE ZEROES.
*
LINKAGE SECTION.
*
COPY XPSCLCOB.
*****
**          PROCEDURE DIVISION          **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
* READ PKCS-12 FILE "XPSUSERP" FROM MACLIB   *
*-----*
    MOVE READ-FILE TO CRYPT-FUNCTION.
    CALL 'XPCRYPT' USING CRYPT-FUNCTION,
        DDNAME, PKCS12-FILE, ADDR-PKCS12-FILE, DATALEN, RC.
    IF RC < 0
        DISPLAY "FILE 'XPSUSERP' NOT FOUND RC = " RC
        GOBACK.
*-----*
* CONVERT PASSWORD FROM EBCDIC TO ASCII (PASSWORD LOWER CASE|) *
*-----*
    MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
    CALL 'XPCRYPT'
        USING CRYPT-FUNCTION, PASSWRD, PWDLEN, RC.
*-----*
* IMPORT PKCS-12 FILE                               *
*-----*
    MOVE IMPORT-PKCS12 TO CRYPT-FUNCTION.
    CALL 'XPCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS12-FILE,
        DATALEN, PASSWRD, PWDLEN, PKCS12-CTX, RC.
    IF RC < 0
        DISPLAY "ERROR IMPORT-FILE: RC = " RC
        GOBACK.

```

```
* ----- *
* GET PRIVATE-KEY FROM PKCS-12 FILE *
* ----- *
      MOVE GET-PRIVATE-KEY TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS12-CTX, RSA-PRIVATE-KEY, RC.
      IF RC < 0
          DISPLAY "ERROR GET-PRIVATE-KEY: RC = " RC
          GOBACK.
* ----- *
* GET ALL CERTIFICATES FROM PKCS-12 FILE *
* ----- *
      MOVE GET-FIRST-CERT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS12-CTX, ADDR-CERTIFICATE, RC.
      PERFORM UNTIL RC <= ZEROES
          PERFORM GET-CERTS
      END-PERFORM.
* ----- *
* CLEANUP PKCS-12 CONTEXT *
* ----- *
      MOVE CLEANUP-PKCS12 TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT'
          USING CRYPT-FUNCTION, PKCS12-CTX, RC.
      STOP RUN.
*
GET-CERTS SECTION.
      MOVE GET-NEXT-CERT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          PKCS12-CTX, ADDR-CERTIFICATE, RC.
GET-CERTS-END.
      EXIT.
ENDRUN.
```

10

GZIP

Allgemein

CryptLib bietet die Möglichkeit mit der Funktion *GZIP* Daten zu komprimieren sowie mit der Funktion *GUNZIP* die Daten wieder zu entpacken.

Funktionen

GZIP

Daten werden im gzip-Format gepackt.

Syntax	Cobol	
	<pre>MOVE GZIP TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, INPUT, INPUTLEN, AOUTPUT, OUTLEN, FILENAME, RC.</pre>	
	Assembler	
	CALL XPSCRYPT, (GZIP, INPUT, INPUTLEN, AOUTPUT, OUTLEN, FILENAME, RC), VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>INPUT</i>	Speicheradresse der zu komprimierenden Daten.	Eingabe
<i>INPUTLEN</i>	Länge der zu komprimierenden Daten.	Eingabe
<i>AOUTPUT</i>	Speicheradresse der komprimierten Daten.	Ausgabe
<i>OUTLEN</i>	Speicheradresse eines Feldes, in dem die Länge der komprimierten Daten zurückgegeben wird.	Ausgabe
<i>FILENAME</i>	Dateiname für den ZIP-Header. Der Name muss mit BLANK x'00' terminiert sein.	Eingabe

GUNZIP

Mit GZIP gepackten Daten werden entpackt.

Syntax	Cobol	
	<pre>MOVE GUNZIP TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, INPUT, INPUTLEN, AOUTPUT, OUTLEN, AFILENAME, RC.</pre>	
	Assembler	

	CALL XPSCRYPT, (GUNZIP, INPUT, INPUTLEN, AOUTPUT, OUTLEN, AFILENAME, RC), VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	Beschreibung	Verwendung
<i>INPUT</i>	Speicheradresse der komprimierten Daten.	Eingabe
<i>INPUTLEN</i>	Länge der komprimierten Daten.	Eingabe
<i>AOUTPUT</i>	Speicheradresse der entkomprimierten Daten.	Ausgabe
<i>OUTLEN</i>	Speicheradresse eines Feldes, in dem die Länge der entkomprimierten Daten zurückgegeben wird.	Ausgabe
<i>FILENAME</i>	Speicheradresse, an die der Dateiname aus dem ZIP-Header kopiert wird. Falls kein Dateiname enthalten ist, wird dieser Pointer auf NULL gesetzt. Ansonsten ist der Dateiname mit x'00' terminiert.	Ausgabe

COBOL-Beispiel:

```

*-----*
*       XPS-CRYPTLIB SAMPLE PROGRAM ZIP       *
*-----*
ID DIVISION.
PROGRAM-ID.
    AESTSTC.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01  XPS                PIC X(160)           VALUE "XPS Software GmbH
-                                     "Muenchener Str. 17
-                                     "85540 Haar/Muenchen
-                                     "Tel. 0049-89-456989-0
-                                     "
-                                     "Internet: www.xps.biz
-                                     "
01  FILENAME          PIC X(16)           VALUE "XPSHOME.TXT".
01  HEADER1           PIC X(20)           VALUE "ZIP-DATA:".
01  HEADER2           PIC X(20)           VALUE SPACES.
01  ADDR-ZIPDATA      POINTER.
01  ADDR-UNZIPDATA    POINTER.
01  ADDR-FILENAME     POINTER             VALUE NULL.
*
COPY XPSCLCTX.
*
77  CRYPT-FUNCTION    PIC X.
77  ZIPLLEN           PIC 9(8)           COMP VALUE ZEROES.
77  UNZIPLLEN        PIC 9(8)           COMP VALUE ZEROES.
77  XPSLEN            PIC 9(8)           COMP VALUE 160.
77  RC                PIC 9(8)           COMP VALUE ZEROES.
77  RCC               PIC 9(8)           VALUE ZEROES.
77  NULL-PARM        PIC 9(8)           COMP VALUE ZEROES.
*
LINKAGE SECTION.
COPY XPSCLCOB.
01  OUT-DATA          PIC X(1).
01  OUT-FILENAME     PIC X(12).
*****
**                   PROCEDURE DIVISION                   **
*****
PROCEDURE DIVISION.
MAIN SECTION.
*-----*
*  ZIP XPS                                                  *
*-----*
    MOVE GZIP TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        XPS, XPSLEN, ADDR-ZIPDATA, ZIPLLEN, FILENAME, RC.
    IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR GZIP: RC = " RCC
        GOBACK.
    SET ADDRESS OF OUT-DATA TO ADDR-ZIPDATA.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT'
        USING CRYPT-FUNCTION, HEADER1, OUT-DATA, ZIPLLEN, RC.
*-----*
*  UNZIP XPS                                               *
*-----*
    MOVE GUNZIP TO CRYPT-FUNCTION.

```

```

CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OUT-DATA,
    ZIPLN, ADDR-UNZIPDATA, UNZIPLN, ADDR-FILENAME, RC.
IF RC < 0
    MOVE RC TO RCC
    DISPLAY "ERROR GUNZIP: RC = " RCC
    GOBACK.
SET ADDRESS OF OUT-DATA TO ADDR-UNZIPDATA.
IF ADDR-FILENAME NOT = NULL
    SET ADDRESS OF OUT-FILENAME TO ADDR-FILENAME
    MOVE OUT-FILENAME TO HEADER2
END-IF.
MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
CALL 'XPSCRYPT'
    USING CRYPT-FUNCTION, HEADER2, OUT-DATA, UNZIPLN, RC.
* ----- *
* CLEANUP CONTEXT *
* ----- *
MOVE CLEANUP-GZIP TO CRYPT-FUNCTION.
CALL 'XPSCRYPT'
    USING CRYPT-FUNCTION, ADDR-ZIPDATA, RC.
CALL 'XPSCRYPT'
    USING CRYPT-FUNCTION, ADDR-UNZIPDATA, RC.
STOP RUN.
ENDRUN.

```

Hilfsfunktionen

Allgemein

CryptLib enthält einige Hilfsfunktionen, die den Entwickler bei der Programmierung kryptographischer Anwendungen unterstützen. Hierzu zählen Funktionen zur Konvertierung der Darstellung von ASN.1-Objekten von binär nach US-ASCII und umgekehrt, Funktionen zum Lesen und Schreiben von Dateien sowie Funktionen zum Übersetzen von Daten von EBCDIC nach ASCII und umgekehrt.

Funktionen

ASN2PEM

BER/DER-kodierte ASN.1-Objekte liegen in binärer Form vor. Es gibt jedoch Übertragungsprotokolle, die nicht in der Lage sind, binäre Daten transparent wiederzugeben. Um BER/DER-kodierte Daten für derartige Applikationen 'lesbar' zu machen, müssen sie von der binären Form in die US-ASCII-Repräsentation übersetzt werden. Dies geschieht mit der, im RFC 1521 definierten, *Base64*-Methode. Mit dieser Funktion kann ein binäres Objekt in ein Base64-Objekt umgewandelt werden.

Syntax	Cobol	
	<pre>MOVE ASN2PEM TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, INPUT, INPUTLEN, SMIME, AOUTPUT, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (ASN2PEM, INPUT, INPUTLEN, SMIME, AOUTPUT, CTX, RC), VL</pre>	
Returncode (RC)	Länge des Base64-Objekts oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>INPUT</i>	Speicheradresse eines binären ASN.1-Objekts.	Eingabe
<i>INPUTLEN</i>	Länge des ASN.1-Objekts.	Eingabe
<i>SMIME</i>	Falls ein S/MIME-Name übergeben wird, wird vor dem PEM-Objekt folgender S/MIME-Header eingefügt: <pre>Content-Disposition: attachment; filename="smime.p7m" Content-Type: application/x-pkcs7-mime; name="smime.p7m" Content-Transfer-Encoding: base64</pre>	Eingabe
<i>AOUTPUT</i>	Speicheradresse des erstellten Base64-Objekts.	Ausgabe
<i>CTX</i>	Speicheradresse eines erstellten Kontexts. Dieser wird zum Freigeben des verwendeten Speichers benötigt.	Ausgabe

PEM2ASN

Mit dieser Funktion kann ein Base64-Objekt in ein binäres Objekt zurückgewandelt werden.

Syntax	Cobol	
	<pre>MOVE PEM2ASN TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, INPUT, INPUTLEN, AOUTPUT, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (PEM2ASN, INPUT, INPUTLEN, AOUTPUT, CTX, RC), VL</pre>	
Returncode (RC)	Länge des binären Objects oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>INPUT</i>	Speicheradresse eines Base64-Objekts.	Eingabe
<i>INPUTLEN</i>	Länge des Base64-Objekts.	Eingabe
<i>AOUTPUT</i>	Speicheradresse des erstellten binären Objekts.	Ausgabe
<i>CTX</i>	Speicheradresse eines erstellten Kontexts. Dieser wird zum Freigeben des verwendeten Speichers benötigt.	Ausgabe

CLEANUP-PEM

Freigeben des von den Base64-Routinen benötigten Speichers.

Syntax	Cobol	
	<pre>MOVE CLEANUP-PEM TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (CLEANUP_PEM, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>CTX</i>	Speicheradresse des Kontexts.	Eingabe

READ-FILE

Mit dieser Funktion kann eine Datei aus der mit DDNAME bezeichneten MACLIB (MVS) bzw. Library-Sublib (VSE) eingelesen werden. Das DD-Statement (MVS) bzw. LIBDEF SEARCH (VSE) muss im Job-Control angegeben werden.

Syntax	Cobol	
	<pre>MOVE READ-FILE TO CRYPT-FUNCTION. CALL 'XPSCRYPT' USING CRYPT-FUNCTION, DDNAME, FILENAME, AFILE, FILELEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRYPT, (READ_FILE, DDNAME, FILENAME, AFILE, FILELEN, RC), VL</pre>	
Returncode (RC)	Länge der Datei oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>DDNAME</i>	Name des Job-Control DD-Statements (MVS) bzw. Library-Sublib-Name	Eingabe

	(VSE).	
<i>FILENAME</i>	Name der einzulesenden Datei.	Eingabe
<i>AFILE</i>	Speicheradresse der eingelesenen Datei.	Ausgabe
<i>FILELEN</i>	Speicheradresse eines Feldes, in dem die Länge der eingelesenen Datei zurückgegeben wird.	Ausgabe

WRITE-FILE

Mit dieser Funktion kann eine Datei auf die mit DDNAME bezeichnete MACLIB (MVS) bzw. Library-Sublib (VSE) geschrieben werden. Das DD-Statement (MVS) bzw. LIBDEF SEARCH (VSE) muss im Job-Control angegeben werden.

Syntax	Cobol	
	<pre>MOVE WRITE-FILE TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION, DDNAME, FILENAME, AFILE, FILELEN, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRIPT, (WRITE_FILE, DDNAME, FILENAME, AFILE, FILELEN, RC), VL</pre>	
Returncode (RC)	0 oder Fehlercode (<0).	
Parameter	Beschreibung	Verwendung
<i>DDNAME</i>	Name des Job-Control DD-Statements (MVS) bzw. Library-Sublib-Name (VSE).	Eingabe
<i>FILENAME</i>	Name der zu schreibenden Datei.	Eingabe
<i>AFILE</i>	Speicheradresse der zu schreibenden Daten.	Eingabe
<i>FILELEN</i>	Länge der zu schreibenden Daten.	Eingabe

CLEANUP-FILE

Freigeben des mit READ-FILE eingelesenen Datenspeichers.

Syntax	Cobol	
	<pre>MOVE CLEANUP-FILE TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION, AFILE, RC.</pre>	
	Assembler	
	<pre>CALL XPSCRIPT, (CLEANUP_FILE, AFILE, RC), VL</pre>	
Returncode (RC)	Keiner.	
Parameter	Beschreibung	Verwendung
<i>AFILE</i>	Adresse des freizugebenden Speichers.	Eingabe

EBCDIC-TO-ASCII

Mit dieser Funktion können Daten vom EBCDIC-Format in das ASCII-Format konvertiert werden.

Syntax	Cobol	
	<pre>MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION,</pre>	

	DATA, DATALEN, RC.	
	Assembler	
	CALL XPSCRIPT, (EBCDIC_TO_ASCII, DATA, DATALEN, RC), VL	
Returncode (RC)	Keiner.	
Parameter	Beschreibung	Verwendung
<i>DATA</i>	Speicheradresse der zu konvertierenden EBCDIC-Daten.	Ein-/Ausgabe
<i>DATALEN</i>	Länge der EBCDIC-Daten.	Eingabe

ASCII-TO-EBCDIC

Mit dieser Funktion können Daten vom ASCII-Format in das EBCDIC-Format konvertiert werden.

Syntax	Cobol	
	MOVE ASCII-TO-EBCDIC TO CRYPT-FUNCTION. CALL 'XPSCRIPT' USING CRYPT-FUNCTION, DATA, DATALEN, RC.	
	Assembler	
	CALL XPSCRIPT, (ASCII_TO_EBCDIC, DATA, DATALEN, RC), VL	
Returncode (RC)	Keiner.	
Parameter	Beschreibung	Verwendung
<i>DATA</i>	Speicheradresse der zu konvertierenden ASCII-Daten.	Ein-/Ausgabe
<i>DATALEN</i>	Länge der ASCII-Daten.	Eingabe

ERR_TOOMANYPARMS **-001**

Beschreibung: Der aufgerufenen Funktion wurden zu viele Parameter übergeben.

ERR_UNSUPPFUNC **-002**

Beschreibung: Der im Feld Crypt-Function übergebene Wert ist nicht gültig.

ERR_NOSTORAGE **-003**

Beschreibung: Es ist kein virtueller Speicher mehr vorhanden.

ERR_INVPARMS **-004**

Beschreibung: Der aufgerufenen Funktion wurden zu wenig Parameter übergeben.

ERR_ALGORITHM **-100**

Beschreibung: Der bei der Funktion *INIT-CTX* angegebene Algorithmus wird nicht unterstützt. Unterstützte Algorithmen sind **AES, DES, RC2, RC4, Blowfish** sowie **RSA**.

ERR_KEYLENGTH **-101**

Beschreibung: Die bei der Funktion *INIT-CTX* angegebene Schlüssellänge wird nicht unterstützt.

ERR_MODE **-102**

Beschreibung: Der bei der Funktion *INIT-CTX* angegebene Modus wird nicht unterstützt. Unterstützte Modi sind bei symmetrischer Verschlüsselung **ECB** und **CBC**, bei asymmetrischer Verschlüsselung **PUBLIC** und **PRIVATE**.

ERR_BUILDKEY -103

Beschreibung: Bei der Funktion *INIT-CTX* ist ein Fehler bei der Algorithmus-Initialisierung aufgetreten.

ERR_BUILDIV -104

Beschreibung: Bei der Funktion *INIT-CTX* ist ein Fehler bei der Bildung des Initialisierungsvektors aufgetreten.

ERR_CTX -105

Beschreibung: Der bei der Funktion angegebene Kontext ist ungültig oder nicht initialisiert.

ERR_OUTLEN -106

Beschreibung: Der bei der Funktion angegebene Speicherbereich ist zu klein.

ERR_LICENSE -150

Beschreibung: Es ist keine gültige Lizenzdatei vorhanden. Bitte wenden Sie sich an den XPS Vertriebspartner.

ERR_CONTENTENC -200

Beschreibung: Bei der Base64 zu PEM Umsetzung trat ein Fehler auf.

ERR_DATA -201

Beschreibung: Die Daten, die der RSA-Funktion übergebenen wurden, sind nicht korrekt.

ERR_DIGALGO -202

Beschreibung: Der Hashtyp, der bei der RSA-Funktion angegebene wurde, wird nicht unterstützt.

ERR_ENCODING -203

Beschreibung: Bei der PEM zu Base64 Umsetzung trat ein Fehler auf.

ERR_RSAKEY -204

Beschreibung: Der an die Funktion übergebene RSA-Schlüssel ist nicht korrekt.

ERR_RSALENGTH -205

Beschreibung: Die der RSA-Funktion übergebene Datenlänge ist nicht korrekt.

ERR_MODULUS -206**Beschreibung:** Der Modulus des übergebenen RSA-Schlüssels ist nicht korrekt.

ERR_RANDOM -207**Beschreibung:** Die Random-Struktur wurde nicht initialisiert.

ERR_PRIVKEY -208**Beschreibung:** Der der Funktion übergebene private RSA-Schlüssel (`privateKey`) ist nicht korrekt.

ERR_PUBKEY -209**Beschreibung:** Der der Funktion übergebene öffentliche RSA-Schlüssel (`publicKey`) ist nicht korrekt.

ERR_SIGNATURE -210**Beschreibung:** Die zur Verifizierung übergebene Signatur stimmt mit dem Original nicht überein.

ERR_ENCRALGO -211**Beschreibung:** Der in der RSA-Funktion verwendete Verschlüsselungs-Algorithmus ist nicht bekannt.

ERR_CERTPARAM -300**Beschreibung:** Bei der Funktion *IMPORT-CERTIFICATE* wurde ein ungültiger Parameter angegeben.

ERR_CERTIMPORT -301**Beschreibung:** Bei der Funktion *IMPORT-CERTIFICATE* wurde ein nicht unterstütztes X.509 Zertifikat übergeben.

ERR_CERTLENGTH -302**Beschreibung:** Der der Zertifikats-Extrahier-Funktion übergebene Speicherbereich ist zu klein.

ERR_CERTALGO -303**Beschreibung:** Bei X.509 Zertifikaten wird nur der Verschlüsselungs-Algorithmus RSA unterstützt.

ERR_CERTHASH -304**Beschreibung:** Bei X.509 Zertifikaten werden nur die Hashtypen MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 und RipeMD160 unterstützt.

ERR_CERTSTART -305

Beschreibung: Beim Verifizieren des X.509 Zertifikats wurde festgestellt, dass das aktuelle Datum kleiner als das Zertifikats-Beginndatum ist. Die Signatur des Zertifikats ist korrekt.

ERR_CERTEND -306

Beschreibung: Beim Verifizieren des X.509 Zertifikats wurde festgestellt, dass das Ablaufdatum des Zertifikats überschritten ist. Die Signatur des Zertifikats ist korrekt.

ERR_CERTOID -307

Beschreibung: Die, mit der Funktion *GET-EXTENSION-BY-OID* gesuchte, Erweiterung ist nicht vorhanden.

ERR_ASN1 -400

Beschreibung: Das zu importierende Objekt hat eine fehlerhafte oder nicht unterstützte ASN.1 Struktur.

ERR_ASN1TABLE -401

Beschreibung: Das importierte ASN.1-Objekt ist nicht mit der aufgerufenen Funktion kompatibel.

ERR_HASHOID -402

Beschreibung: Das importierte PKCS-Objekt enthält einen nicht unterstützten Hashtype.

ERR_CONTENTINF -403

Beschreibung: Das importierte PKCS-Objekt enthält eine nicht unterstützte Content-Information.

ERR_HMAC -404

Beschreibung: Der HMAC des importierten PKCS#12-Objekts ist nicht gültig. Eventuell wurde ein falsches Passwort übergeben.

ERR_AUTHSAFE -405

Beschreibung: Das importierte PKCS#12-Objekt enthält einen nicht unterstützten Authenticated Safe.

ERR_PBETYPE -406

Beschreibung: Das importierte PKCS#12-Objekt enthält einen nicht unterstützten Pbe-Typ (PBE=PasswordBasedEncryption). Unterstützt werden folgende Algorithmen: *pbeWithSHAAnd128BitRC4*, *pbeWithSHAAnd40BitRc4*, *pbeWithSHAAnd3KeyTripleDES-CBC*, *pbeWithSHAAnd2KeyTripleDES-CBC*, *pbeWithSHAAnd128BitRC2-CBC* und *pbeWithSHAAnd40BitRC2-CBC*.

ERR_CERTBAG -407**Beschreibung:** Das importierte PKCS#12-Objekt enthält einen nicht unterstützten Certification Bag.

ERR_CERTBAGTYPE -408**Beschreibung:** Der im importierten PKCS#12-Objekt enthaltene Certification Bag enthält ein nicht unterstütztes Zertifikatsformat. Die Formate x509Certificate und sdsiCertificate werden unterstützt.

ERR_NOAUTHSAFE -409**Beschreibung:** Das importierte PKCS#12-Objekt enthält keinen Authenticated Safe.

ERR_SAFE BAG -410**Beschreibung:** Das importierte PKCS#12-Objekt enthält keinen Safe Bag Type 'pkcs8-shroudedKeybag'.

ERR_PRIVKEY -411**Beschreibung:** Das importierte PKCS#12-Objekt enthält keinen geheimen Schlüssel (PrivateKey).

ERR_RSAENC -412**Beschreibung:** Bei PKCS#12-Objekten wird nur der Verschlüsselungs-Algorithmus RSA unterstützt.

ERR_NOKEYBAG -413**Beschreibung:** Das importierte PKCS#12-Objekt enthält keinen Key Bag.

ERR_HMACALGO -414**Beschreibung:** Das importierte PKCS#12-Objekt enthält einen nicht unterstützten HMAC Algorithmus. Unterstützt werden MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 und RipeMD160.

ERR_ALGO -415**Beschreibung:** Das importierte PKCS#7-Objekt enthält einen nicht unterstützten Hash Type. Unterstützt werden MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 und RipeMD160.

ERR_EALGO -416**Beschreibung:** Das importierte PKCS#7-Objekt enthält einen nicht unterstützten Encryption Algorithm. Unterstützt wird nur RSA.

ERR_NOSIGNER -417

Beschreibung: Für den im importierten PKCS#7-Signed-data Objekt enthalten Unterzeichner (Signer) wurde kein Aussteller-Zertifikat (Trusted Signer) gefunden.

ERR_NOSIGNERCERT -418

Beschreibung: Für den im importierten PKCS#7-Signed-data Objekt enthalten Unterzeichner (Signer) wurde kein Zertifikat gefunden.

ERR_MESSAGEDIG -419

Beschreibung: Der Prüfwert (Message Digest) des im importierten PKCS#7-Signed-data Objekt enthaltenen Unterzeichners (Signer) ist nicht gültig.

ERR_VERIFY -420

Beschreibung: Die Verifizierung der Signatur des im importierten PKCS#7-Signed-data Objekt enthalten Unterzeichners (Signer) ist nicht gültig.

ERR_UNKNOWNSIGNER -421

Beschreibung: Der bei der Funktion *VERIFY-SIGNER* übergebene Unterzeichner ist im importierten PKCS#7-Signed-data Objekt nicht enthalten.

ERR_NODATA -422

Beschreibung: Das importierte PKCS#7 Objekt enthält keine Daten.

ERR_NOCERT -423

Beschreibung: Das importierte PKCS#7 Objekt enthält kein Zertifikat.

ERR_NORECIPIENT -424

Beschreibung: Das importierte PKCS#7 Enveloped-data Objekt enthält keinen Empfänger (Recipient).

ERR_P12NOTVALID -425

Beschreibung: Das der Funktion *IMPORT-ENVELOPED-DATA* übergebene PKCS#12 Objekt ist nicht gültig.

ERR_INVOPT -426

Beschreibung: Die der Funktion *CREATE...DATA* übergebene Option ist nicht gültig.

ERR_NOTRUSTEDSIGN -427

Beschreibung: Für den der Funktion *VERIFY-SIGNER* übergebenen Unterzeichner ist kein vertrauenswürdiger Aussteller (Trusted Signer) vorhanden.

ERR_MAXDATA -428

Beschreibung: Die Funktion *ADD-PKCS7-DATA* hat die maximal zu verarbeitende Datenmenge überschritten.