

# Semesterarbeit Kryptographie - Gruppenarbeit

## PKCS

Aregger Thomas `thomas.aregger@students.ffhs.ch`

Dünki Marc `marc.duenki@students.ffhs.ch`

Gutknecht Jürg `juerg.gutknecht@students.ffhs.ch`

Daniel David `david.daniel@students.ffhs.ch`

3. Januar 2014

## 1 ASN.1

### 1.1 Verwendete Namen und Bezeichnungen

**Object identifier** Eine Folge von Ganzzahlen welche ein spezifisches Objekt identifiziert.

## 2 PKCS 7 - Standard zur Syntax kryptographischer Nachrichten

**PKCS #7** definiert eine generelle Syntax zur Beschreibung von Inhalten welche in Verbindung mit kryptographischen Verfahren stehen können, zum Beispiel Signaturen. Die Syntax erlaubt Rekursion, so dass beispielsweise Inhalte signiert werden können, welche zuvor von einer anderen Instanz signiert wurden etc.

Folgendes sind Beispiele von Anwendungen, welche dieser Standard adressiert:

- Signieren von digitalen Nachrichten
- Digest (hash) von digitalen Nachrichten
- Authentisierung von Nachrichten (MAC)
- Verschlüsselung digitaler Inhalte

### 2.1 Aufbau

Es werden generell zwischen zwei Klassen von Inhalts-Typen unterschieden:

**base** data enthält „plain data“, also Daten, welche keine kryptographischen Erweiterungen („enhancements“) aufweisen.

**enhanced** data enthält Inhalt eines bestimmten Typs (evtl. verschlüsselt) und weitere kryptographische Erweiterungen.

Insgesamt werden vom Standard sechs Inhalts-Typen definiert, weitere könnten in der Zukunft hinzu kommen:

- data
- signed data
- enveloped data
- signed-and-enveloped data
- digested data
- encrypted data

Nur „data“ gehört zur Klasse „base“, alle weiteren Typen gehören zur Kategorie „enhanced“. Da die Typen der enhanced Klasse selbst Inhalte anderer Typen beinhalten, wird auch von dem sogenannten „inner“ und „outer“ content gesprochen. Der „inner“ content ist somit der vom „outer“ content erweiterte Inhalt.

## 2.2 Der Inhalt

Der Standard exportiert schliesslich einen Typen **ContentInfo**, welcher den entsprechenden Inhalt zu repräsentieren vermag. Eine Nachricht, ein Element resp. ein Objekt dieses Standards weist die folgende Syntax auf:

```
ContentInfo ::= SEQUENCE {  
    contentType ContentType,  
    content  
        [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL  
}
```

**contentType** benennt den Typ des Inhaltes. Es handelt sich um einen Object identifier, welcher den Inhalts-Typen gemäss obiger Liste 2.1 (data, signed data etc.) enthält. Er weist die folgende Syntax auf:

**ContentType ::= OBJECT IDENTIFIER**

**content** ist der Inhalt der Nachricht. Das Feld ist optional und falls es nicht enthalten ist, muss der gewünschte Inhalt anderweitig zur Verfügung gestellt werden (wird mittels **contentType** kommuniziert).

## 2.3 Die Inhalts-Typen

### 2.3.1 Data

Der data content type ist ein arbiträrer octet string, welcher generell keine interne Struktur aufweist (jedoch aufweisen kann) und in Form einer ASCII Zeichenketten betrachtet wird. Die genaue Interpretation des Inhaltes wird der Anwendung überlassen.

### 2.3.2 Signed-data

Der signed-data content type besteht aus dem signierten Inhalt eines beliebigen Typs sowie verschlüsselter Digests des Inhaltes, welche von einer beliebigen Anzahl Instanzen signiert wurden.

Die Syntax des signed-data content type ist folgendermassen gegeben [PKCS7, S.10]:

```
SignedData ::= SEQUENCE {  
    version Version,  
    digestAlgorithms DigestAlgorithmIdentifiers,  
    contentInfo ContentInfo,  
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates  
        OPTIONAL,  
    crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,  
    signerInfos SignerInfos  
}  
DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier  
SignerInfos ::= SET OF SignerInfo
```

Der Prozess, wie signierter Inhalt erzeugt wird, wird folgendermassen festgehalten [PKCS7, S.10]:

1. Pro jede signierende Instanz wird der Message Digest des Inhaltes gemäss dem Algorithmus der signierenden Instanz erstellt.
2. Pro signierende Instanz wird mit dessen privatem Schlüssel jeder Message Digest dieser Instanz und dessen zugehörigen Angaben verschlüsselt.
3. Pro signierende Instanz werden der verschlüsselte Message Digest und andere Instanz-spezifische Informationen in einem **SignerInfo** Objekt abgelegt. Die Zertifikate und certificate-revocation Listen werden in diesem Schritt ermittelt.
4. Sämtliche Message-Digest Algorithmen und **SignerInfo** Objekte für alle signierenden Instanzen werden mit dem Inhalt im **SignedDataValue** Objekt abgelegt.

### 2.3.3 Enveloped

Der enveloped-data content type beinhaltet verschlüsselte Daten sowie die verschlüsselten Schlüssel für eine beliebige Anzahl Empfänger, womit der Inhalt wieder entschlüsselt werden kann. Die Idee ist ein sogenanntes „Digital Envelope“, dadurch können die Vorteile des Public Key Algorithmus mit den Vorzügen der symmetrischen Verschlüsselung genutzt werden (hybrid).

Das Element enthält neben einer Version und den verschlüsselten Inhalten eine Menge von Empfänger Angaben. Darin werden der verschlüsselte Schlüssel sowie die Angaben über den zugrundeliegenden Algorithmus hinterlegt:

```
EnvelopedData ::= SEQUENCE {  
    version Version,  
    recipientInfos RecipientInfos,  
    encryptedContentInfo EncryptedContentInfo
```

```

}
RecipientInfos ::= SET OF RecipientInfo
EncryptedContentInfo ::= SEQUENCE {
    contentType ContentType,
    contentEncryptionAlgorithm
    ContentEncryptionAlgorithmIdentifier,
    encryptedContent
    [0] IMPLICIT EncryptedContent OPTIONAL
}
EncryptedContent ::= OCTET STRING

```

### 2.3.4 Signed and enveloped

Der signed and enveloped content type kombiniert quasi die signed und die enveloped Typen. Im Besonderen wird der genaue Prozess zur Erstellung des entsprechenden Inhaltes genannt [PKCS7, S.22] und der genaue Aufbau des Typs beschrieben [PKCS7, S.23].

### 2.3.5 Digested

Der digested content type beinhaltet einen Inhalt beliebigen Typs sowie einen Message Digest dazu. Dies dient grundsätzlich dazu, die Integrität des Inhaltes zu gewährleisten. Dieser Inhalt wird daher typischerweise in einen enveloped content type integriert.

### 2.3.6 Encrypted

Der encrypted content type Inhalt enthält selbst keine Empfänger oder Schlüssel. Der Typ ist eher dazu gedacht, für lokale Verschlüsselung verwendet zu werden. Der Typ beinhaltet lediglich eine Version und den verschlüsselten Inhalt wie er Bestandteil des enveloped content type ist (Syntax [2.3.3](#), `EncryptedContentInfo`).

## 3 PKCS 9 - Ausgewählte Objektklassen und Attribute

In PKCS#9 werden einige Objektklassen und Attribute behandelt, welche Bestandteile anderer PKCS Dokumente sind und von unterschiedlichen Dokumenten gleichermassen referenziert werden.

Der Standard führt die folgenden zwei neuen Objekt-Klassen und deren zugehörigen Attribute ein:

- `pkcsEntity`
- `naturalPerson`

### 3.1 `pkcsEntity`

Die `pkcsEntity` Objekt-Klasse ist dazu gedacht, Attribute von beliebigen PKCS Entitäten zu beherbergen. Sie wurde für die Verwendung von LDAP basierten Verzeichnis-Diensten entwickelt.

Die Syntax ist folgendermassen gegeben:

```
pkcsEntity OBJECT-CLASS ::= {  
    SUBCLASS OF { top }  
    KIND auxiliary  
    MAY CONTAIN { PKCSEntityAttributeSet }  
    ID pkcs-9-oc-pkcsEntity  
}
```

Die Klasse sieht eine ID sowie ein optionales Attribut vor. Die folgenden Attribute werden allesamt dafür verwendet, die zugehörigen Informationen in einem Verzeichnis-Dienst abzulegen.

**pKCS7PDU** Die in PKCS#7 definierten geschützten Daten (enveloped, signed etc.) werden mit diesem Attribut verwendet.

**userPKCS12** In PKCS#12 wird ein Format für den Austausch von Angaben über die persönliche Identität definiert. Dieses Attribut kann hierfür verwendet werden.

**pKCS15Token** In PKCS#15 wird ein Format für kryptographische Tokens definiert, welche in diesem Attribut abgelegt werden können.

**encryptedPrivateKeyInfo** PKCS#8 definiert ein Format für verschlüsselte private Schlüssel, welche in diesem Attribut enthalten sein können.

### 3.2 naturalPerson

Die Objekt-Klasse **naturalPerson** wurde wie die Klasse **pkcsEntity** für die Verwendung in Verzeichnis-Diensten erstellt. **naturalPerson** ist dafür gedacht, Attribute von natürlichen Personen (Menschen) zu beherbergen.

Die Syntax und der Aufbau ähneln stark derjenigen der **pkcsEntity**:

```
naturalPerson OBJECT-CLASS ::= {  
    SUBCLASS OF { top }  
    KIND auxiliary  
    MAY CONTAIN { NaturalPersonAttributeSet }  
    ID pkcs-9-oc-naturalPerson  
}
```

Für **naturalPerson** sind die folgenden Attribute definiert:

**emailAddress** spezifiziert eine oder mehrere E-Mail Adressen in Form einer unstrukturierten ASCII Zeichenkette. Es liegt an der Anwendung, die Adressen zu interpretieren. Speziell an diesem Element ist die **EQUALITY MATCHING RULE** als **pkcs9CaseIgnoreMatch**, welche besagt, dass falls zwei E-Mail Adressen miteinander verglichen werden, wird die Gross-Kleinschreibung ignoriert.

**unstructuredName** spezifiziert den oder die Namen eines Subjektes als unstrukturierte ASCII Zeichenkette. Ein unstrukturierter Name kann mehrere Attribut-Werte enthalten, es liegt auch hier an der Anwendung, den Namen zu interpretieren. Wie bei der E-Mail Adresse wird beim Vergleich zweier unstrukturierter Adressen die Gross-Kleinschreibung ignoriert.

**unstructuredAddress** nennt die Adresse des Subjektes. Auch hierbei handelt es sich um vollkommen unstrukturierten Inhalt. Auch hier gilt, dass beim Vergleich die Gross-Kleinschreibung ignoriert wird. Wie der Name kann auch die Adresse mehrere Attribut-Werte enthalten und es liegt an der Anwendung, diesen Inhalt zu interpretieren.

**dateOfBirth** wird in Form der **GeneralizedTime** geführt. Es wird ferner verlangt, dass dieses Attribut maximal einmal vorkommt (**SINGLE VALUE TRUE**).

**placeOfBirth** darf wie **dateOfBirth** maximal einmal vorkommen und nennt in einem unstrukturierten Text den Geburtsort des Subjektes. Der Geburtsort wird allerdings unter Berücksichtigung der Gross-Kleinschreibung behandelt.

**gender** nennt das Geschlecht mittels „F“, „f“, „M“ oder „m“. Dieses Attribut darf ebenfalls maximal einmal vorkommen.

**countryOfCitizenship** zählt alle Staatsangehörigkeiten des Subjektes auf, folglich darf das Attribut mehrmals vorkommen. Das Land wird als 2-stelliger Länder-Code gemäss ISO/IEC 3166-1 („CH“ für Schweiz, „DE“ für Deutschland etc.) hinterlegt.

**countryOfResidence** nennt alle Länder der Aufenthaltsorte des Subjektes, kann also mehrfach vorkommen. Das Land wird ebenfalls als zweistelliger ISO Code hinterlegt.

**pseudonym** spezifiziert ein Pseudonym des Subjektes. Neben einer ID enthält es das Pseudonym als Zeichenkette, welches unter Berücksichtigung der Gross-Kleinschreibung behandelt wird.

**serialNumber** Auf dieses Attribut wird nicht eingegangen, es ist definiert in ISO/IEC 9594-6.

### 3.3 Generelle Attribute

Neben der beiden neu definierten Objekt-Klassen **pkcsEntity** und **naturalPerson** wird im Besonderen auf einige spezifische Attribute eingegangen. Von diesen sollen nun einige genauer betrachtet werden:

**contentType** Das Attribut **contentType** spezifiziert den Inhalts-Typen des in PKCS#7 (oder S/MIME) signierten **ContentInfo** Objektes. In solchen Inhalten ist das Attribut **contentType** zwingend, falls authentifizierte Attribute aus PKCS#7 vorhanden sind.

**messageDigest** spezifiziert den Message Digest der Inhalte des **content** Feldes des **ContentInfo** Objektes. Der Message Digest wird anhand dem Algorithmus der signierenden Instanz berechnet. Dieses Attribut ist zwingend, falls authentifizierte Attribute aus PKCS#7 Verwendung finden.

**signingTime** benennt die Zeit, wann die Signatur erstellt wurde. Die Zeit wird gemäss ISO/IEC 9594-8 notiert, wobei Daten vor dem 1.1.1950 oder nach dem 31.12.2049 müssen in Form der **GeneralizedTime** codiert werden, alle anderen Zeiten als **UTCTime** [PKCS9, S.12].

**randomNonce** ist ein Attribut, welches es ermöglicht, gegen spezifische Attacken zu schützen. Beispielsweise kann ein Unterzeichner **signingTime** unterdrücken, um replay Attacken zu unterbinden. Das Attribut dient signierten Daten aus PKCS#7 und darf nur einmal vorkommen. Das Element **RandomNonce** ist ein Oktett-String und muss mind. 4 Bytes lang sein.

**counterSignature** erlaubt es, eine Signatur zu signieren. Das Attribut hat dieselbe Bedeutung wie **SignerInfo** [PKCS7, S.12], ausser:

- Das Feld **authenticatedAttributes** muss ein Attribut **messageDigest** aufweisen, falls es irgendwelche andere Attribute aufweist.
- Der Inhalt des Message Digest ist der Inhalt des **signatureValue** Feldes des **SignerInfo** Objektes. Das bedeutet, dass der signierende Prozess (welcher die Signatur signiert) den originalen Inhalt nicht zu kennen braucht. Zudem kann eine **counterSignature** selbst wieder eine **counterSignature** beinhalten, so lassen sich beliebig lange Ketten von **counterSignature** Objekten erstellen.

**challengePassword** spezifiziert ein Passwort, mit welchem eine Entität die Annullierung eines Zertifikates verlangen kann. Die Interpretation des Inhaltes ist wiederum der Anwendung überlassen, er wird jedoch unter Berücksichtigung von Gross-Kleinschreibung verglichen. Es wird bemerkt, dass der Inhalt als **PrintableString** encodiert werden soll, falls Internationalisierung dies nicht ermöglicht, sollte stattdessen **UTF8String** verwendet werden.

## 4 PKCS 13 - Elliptische Kurven

Die Kryptographie mit elliptischen Kurven erfährt eine zunehmende Popularität, da eine vergleichbare Sicherheit zu etablierten Public Key Verfahren mit kleineren Schlüsseln ermöglicht wird. Verbesserungen in der Implementierung wie der Erzeugung von elliptischen Kurven machen das Verfahren praxistauglicher als bei seiner Einführung in den 1980-er Jahren.

PKCS#13 ist bis heute noch kein definitiver Standard. Der Standard ist noch immer in Entwicklung. Der Standard soll die folgenden Aspekte der Kryptographie mit elliptischen Kurven abdecken [PKCS13-proj]:

- Parameter und Schlüssel Erzeugung und Validierung
- Digitale Signaturen
- Public Key Verschlüsselung
- Key Agreement (anstelle des verwundbaren anonymen Key-Exchanges wie z.Bsp. Diffie-Hellman)
- ASN.1 Syntax
- Überlegungen zur Sicherheit

Es sind bereits Standards in Arbeit, welche sich mit der Kryptographie mit elliptischen Kurven befassen:

**ANSI X9.62** Ist ein in Entwicklung befindlicher Standard für digitale Signaturen.

**ANSI X9.63** Ist ein in Entwicklung befindlicher Standard für Key Agreement.

**IEEE P1363** Soll eine generelle Referenz für Public Key Verfahren verschiedener Techniken, inkl. elliptischer Kurven werden.

PKCS#13 soll die anderen Standards vervollständigen, ein Profil der anderen Standards im PKCS Format liefern und eine Anleitung für die Integration in andere PKCS Anwendungen (wie beispielsweise PKCS#7) bieten.

## 4.1 Aktueller Umfang

Zurzeit umfasst der Standard Grundlagen der folgenden Bereiche, welche allerdings keine konkreten Implementierungen oder Standards nennt. Diese Themen können als Grundlagen der Kryptographie mit elliptischen Kurven betrachtet werden.

**Functions** Grundlegende Definition einer Funktion: „A function  $f$  from a set  $A$  to a set  $B$  assigns to each element  $a$  in  $A$  a unique element  $b$  in  $B$ .“ [PKCS13-func]

**Modular arithmetic** Grundlagen der modularen Arithmetik

**Groups** Grundlagen der diskreten Mathematik ( $\mathbb{Z}_p$  und  $\mathbb{Z}_p^*$ ) in Bezug auf Gruppen.

**Fields and rings** Ebenfalls mathematische Grundlagen.

**Vector spaces and lattices** Behandelt die Grundlagen der linearen Algebra und der Vektorräume.

**Boolean expressions** Die Betrachtung logischer Ausdrücke als Funktionen.

**Time estimations and some complexity** Betrachtungen zur Komplexität.

Generell sind keine konkreten Vorgaben zu diesem Standard vorhanden. Zurzeit existiert lediglich ein Vorschlag, welcher mögliche Key Agreement Schemes, Signature Schemes, Encryption Schemes und Point Representations nennt.

## A Bibliographie

[PKCS7] RSA Laboratories: PKCS#7 Cryptographic Message Syntax Standard; 1993

[PKCS9] RSA Laboratories: PKCS #9 v2.0: Selected Object Classes and Attribute Types; 2000



- [PKCS13] RSA Laboratories: PKCS #13: ELLIPTIC CURVE CRYPTOGRAPHY STANDARD <http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-13-elliptic-curve-cryptography-standard.htm>
- [PKCS13-proj] RSA Laboratories: PKCS#13 Project overview, <http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/project-overview.htm>, 1.1.2013
- [PKCS13-func] RSA Laboratories: A.1 Functions, <http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/functions.htm>, 1.1.2013