TLV-Strukturen

TLV steht für Tag-Length-Value und bezeichnet eine Art, Daten variabler Länge zu organisieren. Datenstrukturen fester Länge und Organisation neigen regelmäßig dazu, früher oder später zu eng oder zu beschränkt zu werden. Eine Lösung stellen TLV-Strukturen dar. Sie werden häufig mit der *abstract syntax notation one* (ASN.1 , [ASN.1]) definiert. Die ASN.1 ist eine Sprache, mit der Datenstrukturen und ihre Kodierungen beschrieben werden können. In der [ISO/IEC8825] sind die basic encoding rules (BER) für ASN.1 zusammengefasst. Diesen Regeln folgende Datenobjekte heißen BER-TLV-kodierte Datenobjekte.

Die Idee bei der BER-TLV-Kodierung ist, dass alle Datenobjekte mit einem Kennzeichen (Tag) und einer Längenangabe (Length) neben ihrem eigentlichen Inhalt (Value) versehen werden. Die Verwendung von Datenstrukturen variabler Länge sind auf diese Weise flexibler. Bei der Auswertung der Daten vergleicht ein Programm dann zu allererst das Kennzeichen. Stimmt es mit einem bekannten überein, dann kann es den tatsächlichen Inhalt parsen. Ist es ein unbekanntes, kann es das Datenobjekt recht einfach übergehen, denn es ist ja in der Lage die Länge des Datenobjekts zu ermitteln. Folglich ist die Reihenfolge bei der Übertragung der Datenobjekte nicht mehr relevant. Es können auch beliebig Datenobjekte mit unbekannten Kennzeichen übertragen werden, die z.B. nur für eine neuere Softwareversion interpretierbar sind. Das erhöht die Kompatibilität. Man kann

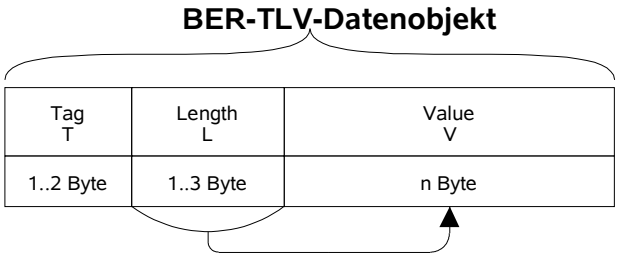

Abbildung 1: BER-TLV-Datenobjekt nach ASN.1

ein neues Datenobjekt hinzufügen ohne dabei zu einem alten Parser inkompatibel zu werden. Diese Art der Speicherung eignet sich auch vorzüglich um Speicherplatz zu sparen, was mit einer der Gründe für den Einsatz von BER-TLV-Datenobjekten auf Chipkarten ist. Wenn man optionale Datenfelder an das Ende des Inhalts stellt, kann man regelmäßig noch ein paar Bytes sparen.

Byte 1	b8	b7	b6	b5	b4	b3	b2	b1	Bedeutung
	0	0	-	-	-	-	-	-	Universal class
	0	1	-	-	-	-	-	-	Application class
	1	0	-	-	-	-	-	-	Context specific class
	1	1	-	-	-	-	-	-	Private class
	-	-	0	-	-	-	-	-	Primitive (einfaches Datenobjekt)
	-	-	1	-	-	-	-	-	Constructed (komponiertes Datenobjekt)
	-	-	-	n	n	n	n	n	Nummer des Kennzeichens (0..30)
	-	-	-	1	1	1	1	1	Ein weiteres Kennzeichenbyte folgt
Byte 2	b8			...				b1	
	0	n	n	n	n	n	n	n	Nummer des Kennzeichens (31..127)
	1	n	n	n	n	n	n	n	Noch ein Kennzeichenbyte folgt.
Byte 3	b8			...				b1	
	m	m	m	m	m	m	m	m	Ergebniskennzeichen ist dann n und m in Reihe.

Tabelle 1: Bitweise Kodierung des Kennzeichens (Tag, T) nach BER-Kodierung

Das Tag bzw. Kennzeichen besteht also je nach Bedarf aus ein bis drei Bytes. Dabei folgt die Kodierung der Bytes festen Regeln, sodass es auch einem unwissenden Parser möglich ist, sich bis

zur Längenangabe durchzuarbeiten und das Datenobjekt zu überspringen. Reichen einer Anwendung die standardmäßigen 30 Kennzeichen nicht aus, kann man den Raum der Kennzeichen recht einfach vergrößern. Man setzt die Bits b1 mit b5 auf 1 und kann dann ein weiteres Kennzeichenbyte einbauen. In dieser Konfiguration stehen einem dann bis zu 127 Kennzeichen zur Verfügung. Die Bits b1 mit b5 fallen jetzt weg, denn sie müssen als Indikator für das zweite Kennzeichenbyte auf 1 bleiben. Sollte dies abermals nicht reichen, besteht noch die Möglichkeit das höchstwertigste Bit des zweiten Kennzeichenbytes zu setzen und ein drittes Kennzeichenbyte hinzuzufügen. Da ein viertes Byte nicht vorgesehen ist, hat man hier nun acht Bits zur Verfügung plus die sieben des zweiten Bytes. Das ergibt also bei drei Kennzeichenbytes bestenfalls 2^{15} mögliche Kennzeichen für eine Anwendung.

Die Längenangabe funktioniert nach einem ähnlichen Prinzip, wie das Kennzeichen. Je nach Anforderung, also Länge des Nutzdatenbereichs kommen für die Längenangabe ein bis drei Bytes zum Einsatz. Ist die Größe der Nutzdaten bis zu 127 Bytes lang, reicht noch ein Längenbyte aus. Bei längeren Nutzdaten (bis 64kB) werden dann bis zu drei Bytes eingesetzt. Dabei dient das Bit b8 als Indikator. Ist es gesetzt und somit der Wert des ersten Bytes größer 127, dienen die restlichen Bits als Indikator für die Anzahl der nun folgenden Bytes. Jedoch sind nur maximal drei Bytes für die Längenangabe zulässig, sodass sich die Zahl der Möglichkeiten für das Byte1 als Indikator auf die Werte 81_{16} und 82_{16} reduziert. In den Längenbytes zwei und drei findet sich dann die Länge als 8- oder 16-Bit-Zahl.

Byte1	Byte2	Byte3	Bedeutung
0..127	-	-	Bis 127 ist nur ein Byte nötig.
81_{16}	128..255	-	Bei zwei Bytes setzt man das erste Byte auf 81_{16}.
82_{16}	256..65535		Ab 256 werden drei Bytes verwendet, Indikator: 82_{16}.

Tabelle 2: Die Längenangabe nach BER-Kodierung (L)

Ein oft eingesetztes Mittel ist das Verschachteln (Konstruieren) von TLV-Datenobjekten. Man erreicht durch die Verschachtelung eine Art lokaler Kennzeichensemantik. Hat man alle Kennzeichen auf der obersten Ebene, gerät man leichter in Konflikt mit anderen Kennzeichen. Baut man dagegen „lokale“ TLV-Objekte in die Nutzdaten eines TLV-Datenobjekts, kann man wieder alle Kennzeichen verwenden ohne die Gefahr einer Verwechslung mit anderen TLVs. Für die Verschachtelung gibt es keinen Standard, schließlich ist es egal, was im Nutzdatenteil steht. Es gibt nur die Vereinbarung, dass bei konstruierten TLV-Datenobjekten nach ASN.1 das Bit b6 gesetzt sein sollte.

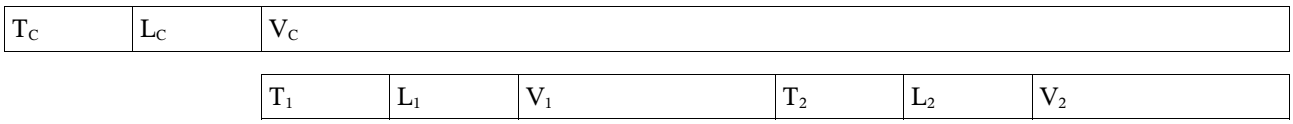

Abbildung 2: Verschachtelung von ADN.1-Datenobjekten

Die TLV-Strukturen sind ein Mittel um Informationen variabler Länge und teils unbekannten Inhalts strukturiert zu speichern und zu übertragen. Weil bei Anwendungen im Bereich der Chipkarten der Speicher immer knapp ist, wird diese Art der Darstellung fast ausschließlich verwendet. Nachteilig sind bei ASN.1-Datenobjekten allerdings der nicht geringe Rechenaufwand bei deren Verarbeitung und die Tatsache, dass bei geringen Nutzdaten das Verhältnis der Verwaltungsdaten zu den Nutzdaten unökonomisch wird (2:1). Gewichtiger Vorteil aber ist die hohe Kompatibilität, die die ASN.1-Datenobjekte mit sich bringen.

Neben der reinen Lehre mit Kennzeichen, Länge und Wert bei den TLV-Datenobjekten gibt es noch eine Reihe Variationen:

TLV	Tag, Length, Value: Das bisher beschriebene. Eigentlich ein degeneriertes TLAV, bei dem das Attribut fehlt.

TLAV	Tag, Length, Attribute, Value: Das ist das universellste Datenobjekt – der Maximalausbau. Dazu gleich mehr.
TLA	Tag, Length, Attribute: Es handelt sich hierbei um ein degeneriertes TLAV. Hier nimmt das Attribut die gesamte Länge des Datenbereichs ein.
T0	Tag, Length: Hier fehlt der Datenbereich, was dazu führt, dass die Länge immer null ist, wodurch der Name T0 zustande kommt.
LV	Length, Value: Hier fehlt das Tag. Diese Variante wird verwendet, wenn die Reihenfolge der Datenobjekte invariant ist. Die Daten werden dann sozusagen im Blindflug, nur unter Beachtung der Länge, gelesen.
VEL	Value, External, Length: Dieses müsste eigentlich lediglich V heißen, denn es ist ein TLV, dass nur noch aus dem Inhalt besteht. Es ist somit darauf angewiesen, dass seine Länge extern gespeichert wird bzw. bekannt ist.
BLOCK	Das gleiche wie VEL.

Tabelle 3: Variationen der TLV-Idee

Das TLAV ist das universellste aller Datenobjekte und eigentlich ist das TLV ein degeneriertes TLAV. Wie das TLV besteht es aus einem Kennzeichen, einer Längenangabe und dem Inhalt. Zusätzlich hat es aber noch vor dem eigentlichen Inhalt ein oder mehrere Attributbytes. Die Längenangabe bezieht sich dabei dann nicht alleine auf den Inhalt sondern auf die Summe von

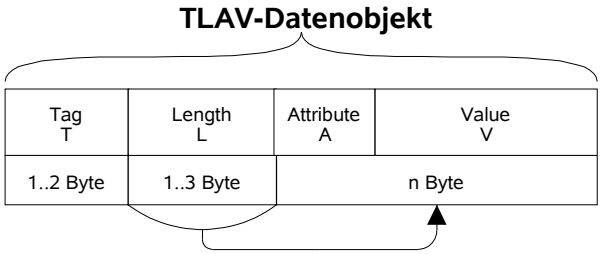

Abbildung 3: TLAV-Datenobjekt

Attribut(en) und Inhalt. Es muss schließlich weiterhin möglich sein das Datenobjekt anhand seiner Längenangabe zu übergehen. Die eigentliche Idee dahinter ist, dass man so neben den Kennzeichen noch eine weitere Gliederungsebene bekommt.

Der Indikator, ob ein Attribut vor kommt oder nicht ist das höchstwertigste Bit des Kennzeichens. Ist das Kennzeichen größer als 127, so ist mindestens das erste Byte der Nutzdaten ein Attribut. Bei dem Attribut sind allerdings nur sieben der acht Bit nutzbar. Das achte oder höchstwertigste Bit dient als Indikator, ob ein weiteres Attributbyte folgt oder nicht. Ist es gesetzt, folgt noch ein weiteres und so fort. Ist es nicht gesetzt folgt als nächstes der tatsächliche Inhalt. Es sei denn es handelt sich um ein TLA. In diesem Fall ist das Datenobjekt zu Ende bzw. die angegebene Länge erreicht.

Attributbyte(s)	b8	b7	b6	b5	b4	b3	b2	b1	Bedeutung
	1	n	n	n	n	n	n	n	Sieben Bit Daten n, ein weiteres Attributbyte folgt.
	0	n	n	n	n	n	n	n	Sieben Bit Daten n, dieses ist das letzte Attributbyte.
1. Tagbyte	b8	b7	b6	b5	b4	b3	b2	b1	
	1	n	n	n	n	n	n	n	Es handelt sich um ein TLAV- oder ein TLA-Objekt.
	0	n	n	n	n	n	n	n	Nummer des Kennzeichens (0..127).

Tabelle 4: Inhalte des Kennzeichens und des Attributs.

Literaturverzeichnis

ASN.1: Walter Gora, ASN.1, 1998
ISO/IEC8825: ISO/IEC, ASN.1 encoding rules: Specification of Packed Encoding Rules (PER), 1996