User guide

by Nathanaël COTTIN

Version 1.0.3

Table of content

Introduction

asnViewer displays user-friendly tree views of ASN.1-encoded files. asnViewer performs a powerful interpretation and matches encoded regions with their corresponding tree view entries.

ASN.1 encodings can be found in many systems, environments and protocols which need data exchange. For example, ASN.1 is part of SNMP and most of security tools dealing with cryptography and digital certificates.

Screen captures are taken from asnViewer version 1.3.1.

Features

asnViewer decodes BER- and DER-encoded ASN.1 files and displays both a tree view and an hexadecimal view.

Requirements

asnViewer requires a 1.5 (or newer) JRE. This environment can be freely downloaded at http://java.sun.com. The Windows executable version automatically displays the appropriate link to sun website.

Overview

asnViewer graphical interface is composed of three parts (fig. 1):

1. A floating menu
2. A tree view and hexadecimal view of the opened file
3. An interpretation / raw view section. The corresponding tabs refer to primitive types only. Constructed types values are not displayed in this section.

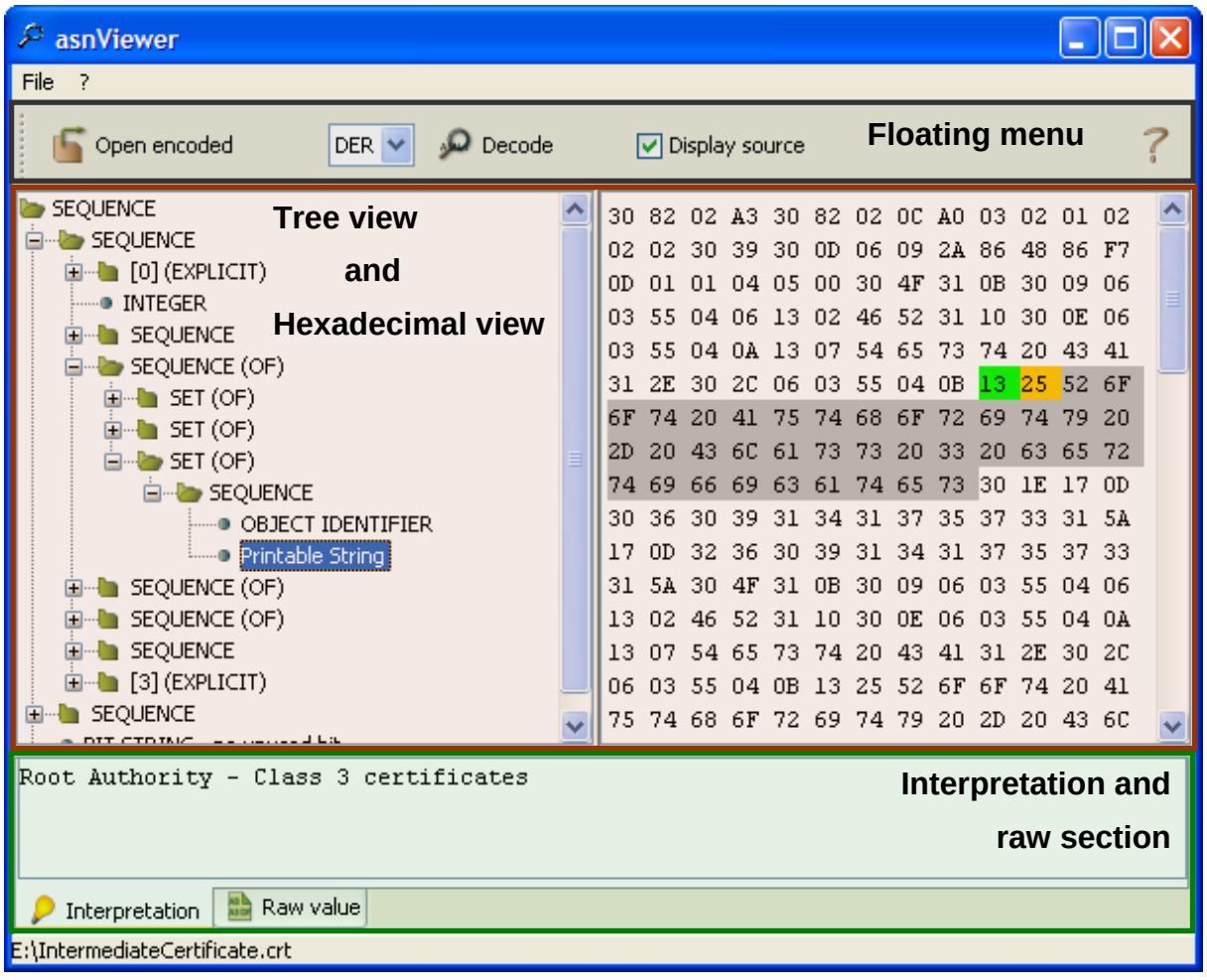

Figure 1

Usage

Executing asnViewer displays the GUI shown hereafter (fig. 2). First action consists in loading an ASN.1-encoded file by clicking on the “Open encoded” button on the top left hand corner.

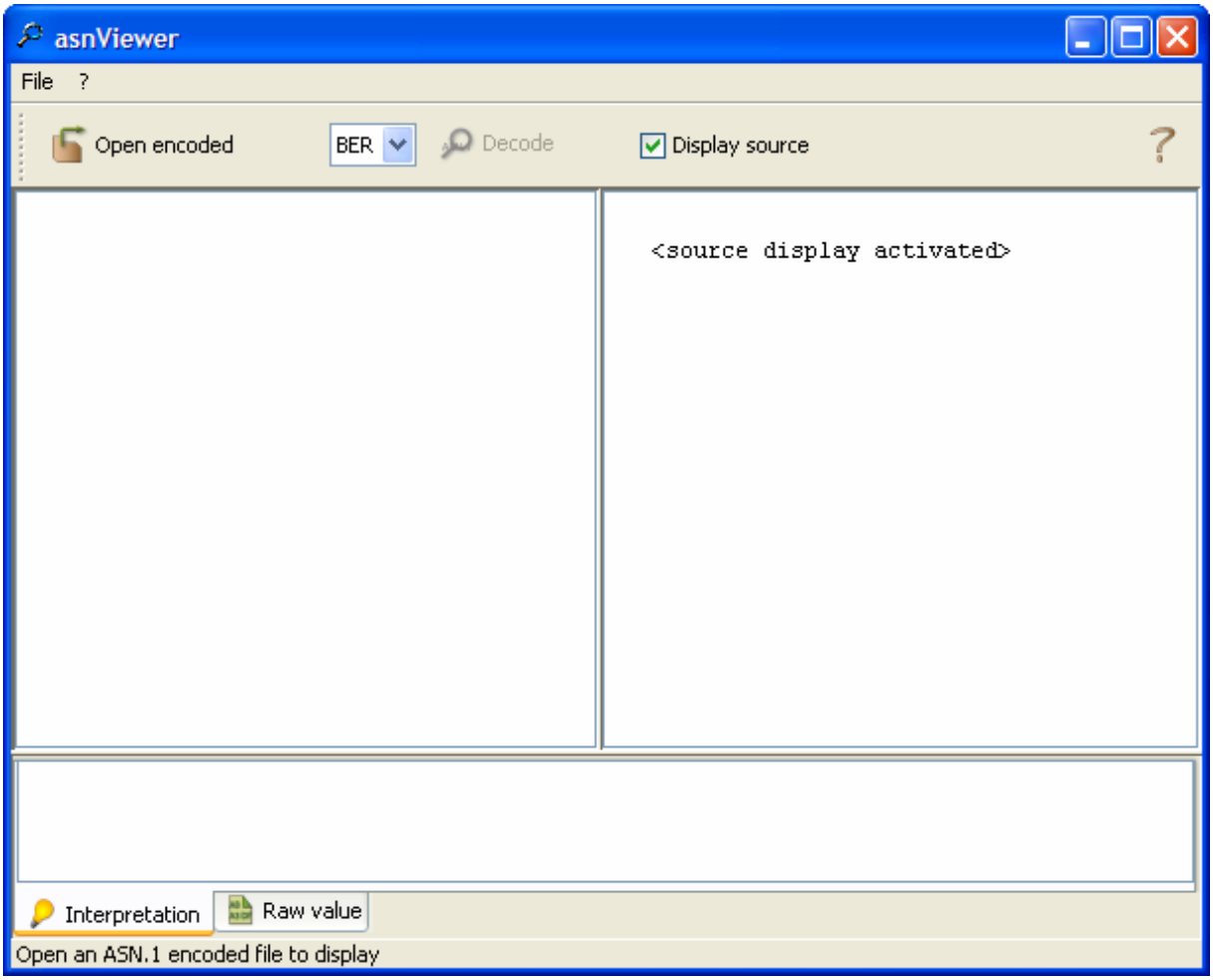

Figure 2

The selected file is then automatically displayed in hexadecimal format and its decoding tree view is created.

The selected file is not loaded in the hexadecimal view when the “display source” box is unchecked.

Selecting appropriate decoding rules

Basic Encoding Rules (BER) are selected by default. These rules allow to decode BER, Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

When opening a BER-encoded file using BER specific rules (indefinite length encoding, Boolean values, etc.), the tree view is created (fig. 3).

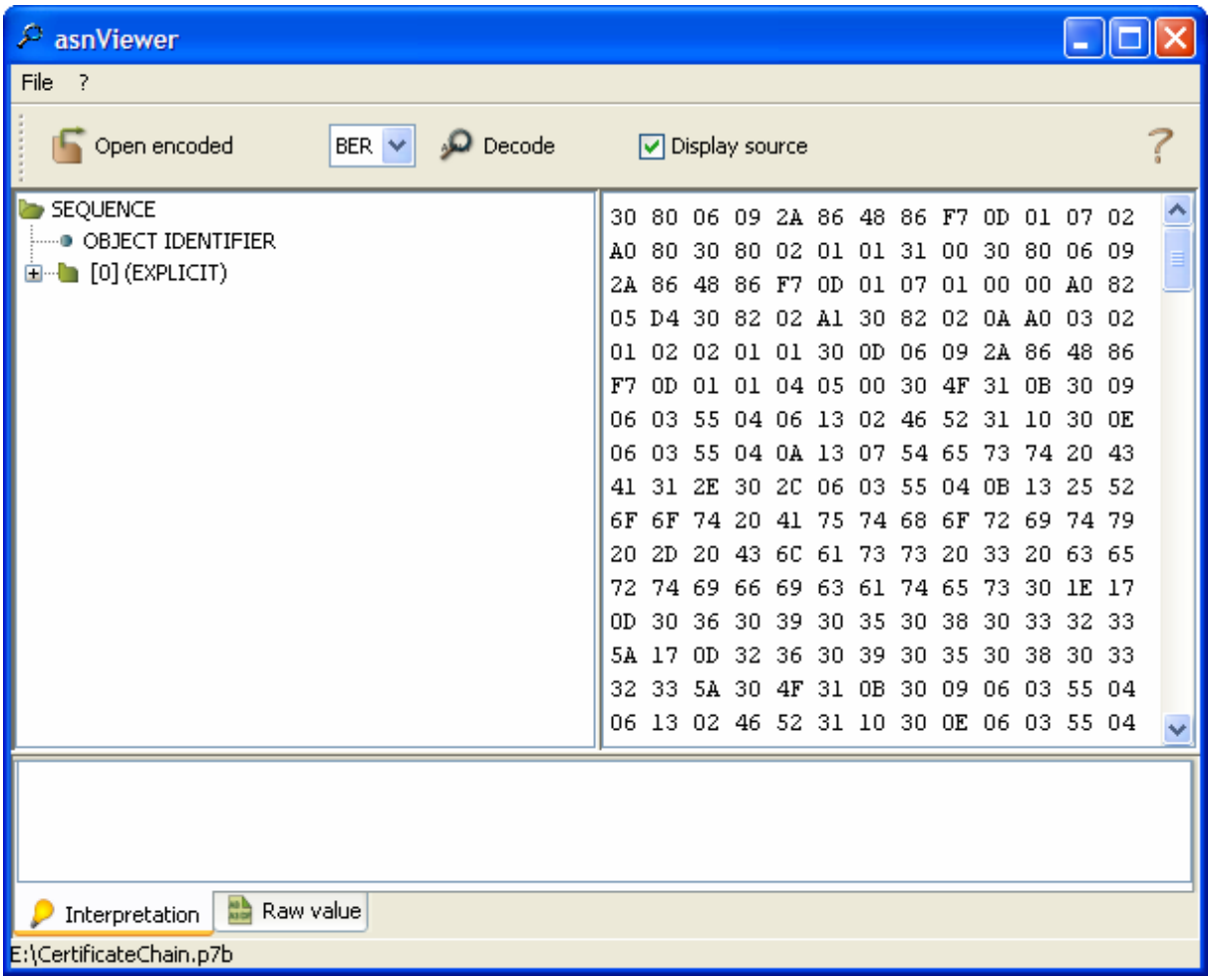

Figure 3

Changing current decoding rules to DER leads to an error in the tree view creation process (fig. 4) as DER is a subset of BER.

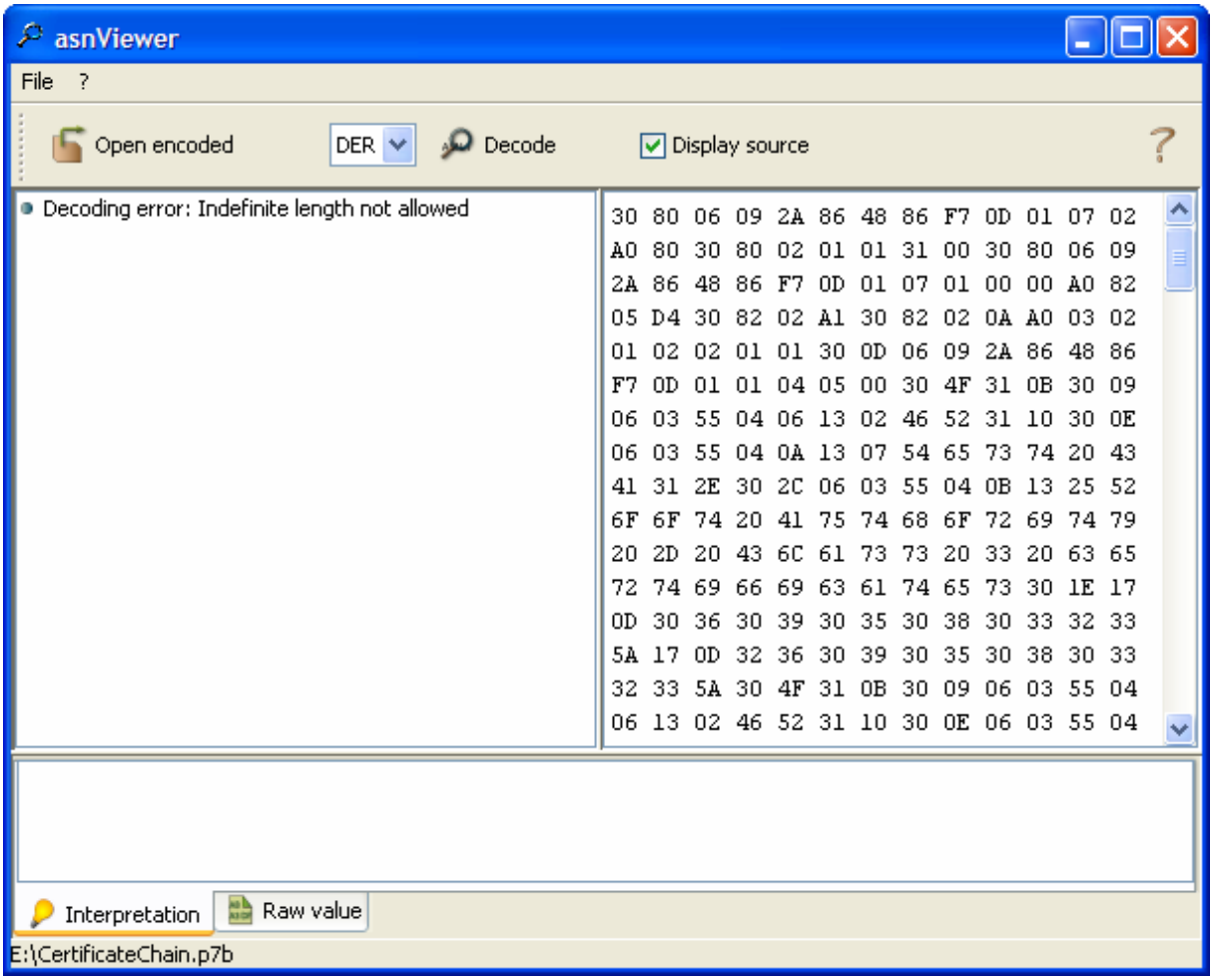

Figure 4

Decoding rules can be selected before loading a file to avoid decoding errors.

Tree view display

The tree view presents a readable version of the encoded file. asnViewer tries to interpret tags when possible:

- Untagged primitive types are directly recognized
- Untagged constructed types are displayed depending on their inner elements to make a difference between `SEQUENCE` and `SEQUENCE OF` objects as well as `SET` and `SET OF` objects. If a constructed object comprises two different inner elements tags (an `INTEGER` and a `BOOLEAN` for example), asnViewer displays the appropriate type (`SEQUENCE` or `SET`) instead of the corresponding collection (`SEQUENCE OF` or `SET OF`). The `OF` suffix is displayed between parenthesis as asnViewer cannot be sure that the decoded element is a collection of similar inner elements (`SEQUENCE OF` or `SET OF`)
- Tagged objects are displayed according to their ASN.1 specification: class and tag value are enclosed between squares (`[APPLICATION 0]` if class is `APPLICATION` or `[0]` if `CONTEXT` for example). Explicit tagging is proposed when the tagged object is encoded in constructed form and it encloses a single inner (tagged or untagged) object (fig. 5).

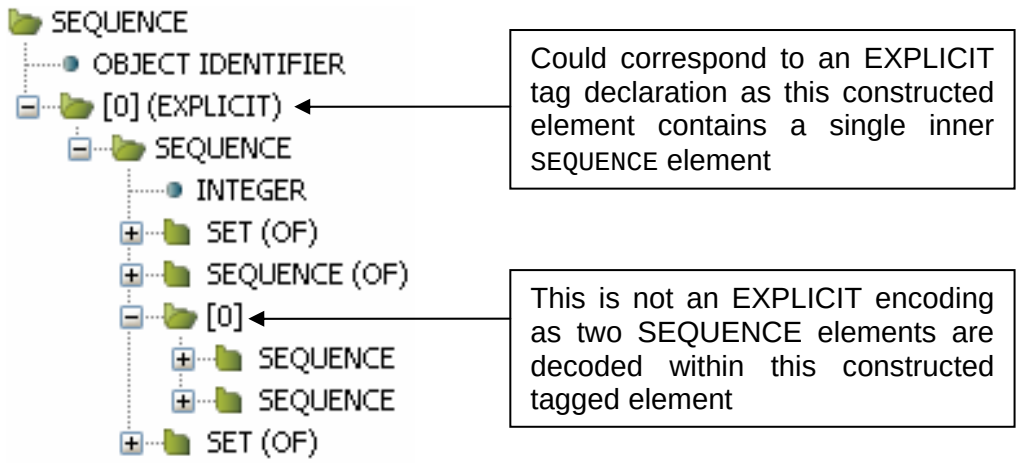

Figure 5

A message replaces the tree view decoding in the following cases:

- The selected rules are not able to decode the encoded file (decoding error shown in fig. 4)
- The file contains extra information, no matter where this information is located (security error). Please refer to PowerASN for Java security for more details.

Browsing the tree view

Clicking on a ⊞ sign (or alternatively, double-clicking on a constructed element) displays or hides its inner elements.

Selecting a tree view element highlights its corresponding hexadecimal part in the source file (fig. 6).

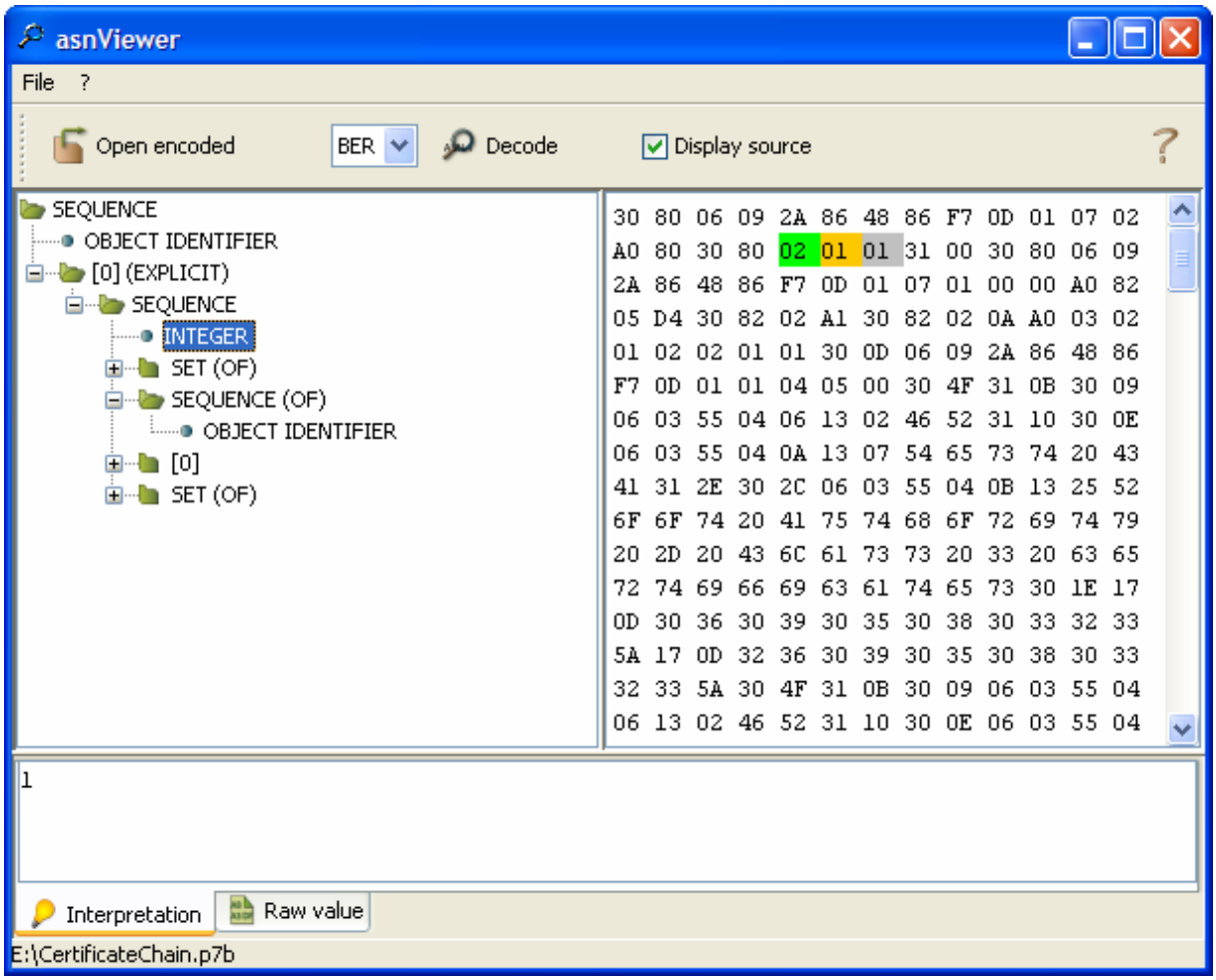

Figure 6

Tag-Length-Value components are depicted using different colors:

- Tag (or type) is shown in green
- Length is highlighted in orange (definite length, see fig. 6) or in rose (indefinite length). In case of indefinite length, end flags (double zero) are also displayed in rose (fig. 7)

- Value is always displayed is light gray, independently from the ASN.1 object form.

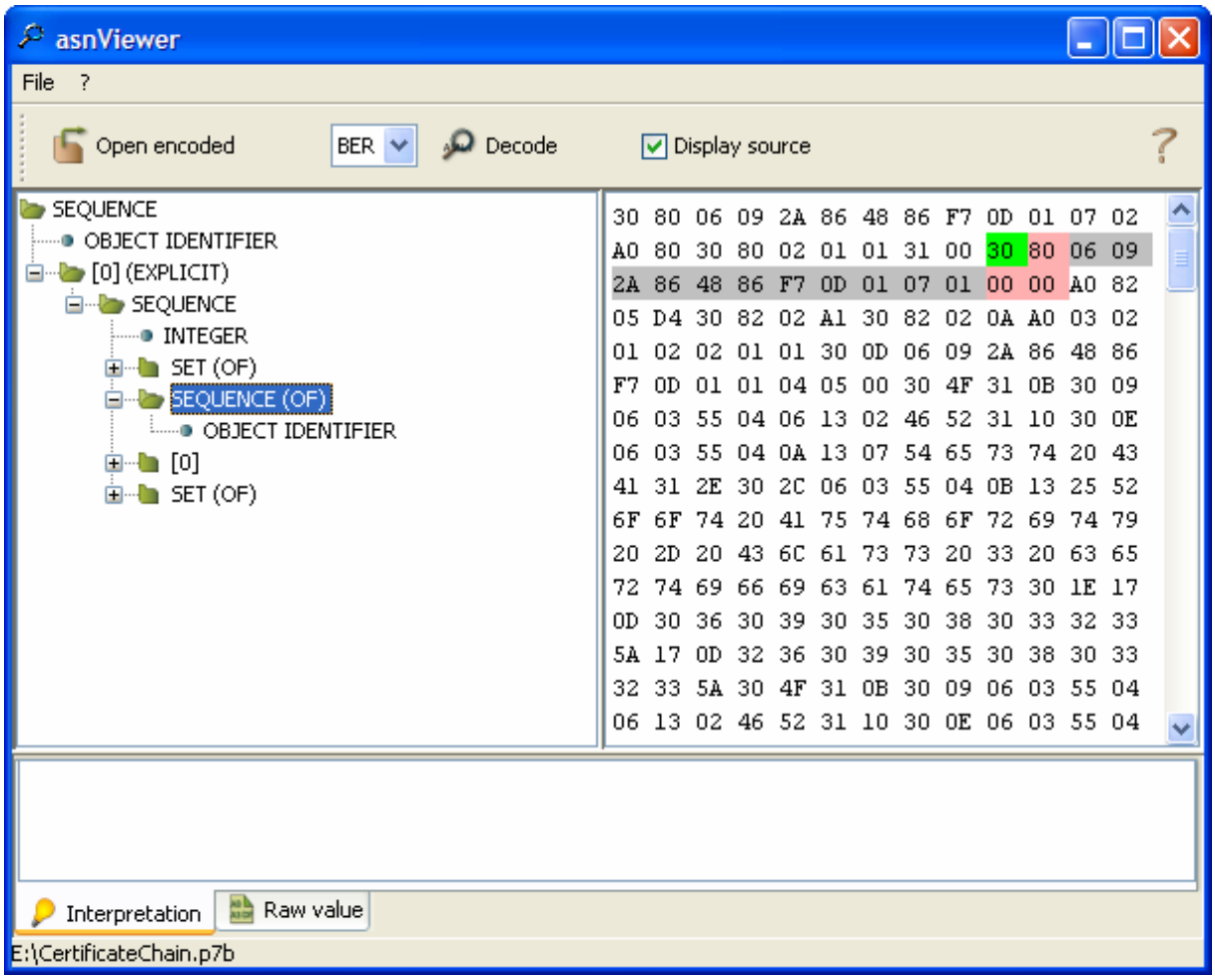

Figure 7

Hiding / showing the source file

It is possible to show or hide the source file in the hexadecimal view by respectively checking or unchecking the “Display source” box. Hiding the source file hexadecimal view (fig. 8) is mainly used when the tree view needs more horizontal space or when the file loading process raises an out of memory error (in case of huge files only).

Unchecking the “Display source” box before loading a file prevents from loading this file into the hexadecimal view. The file is not loaded in the hexadecimal view until this box is checked.

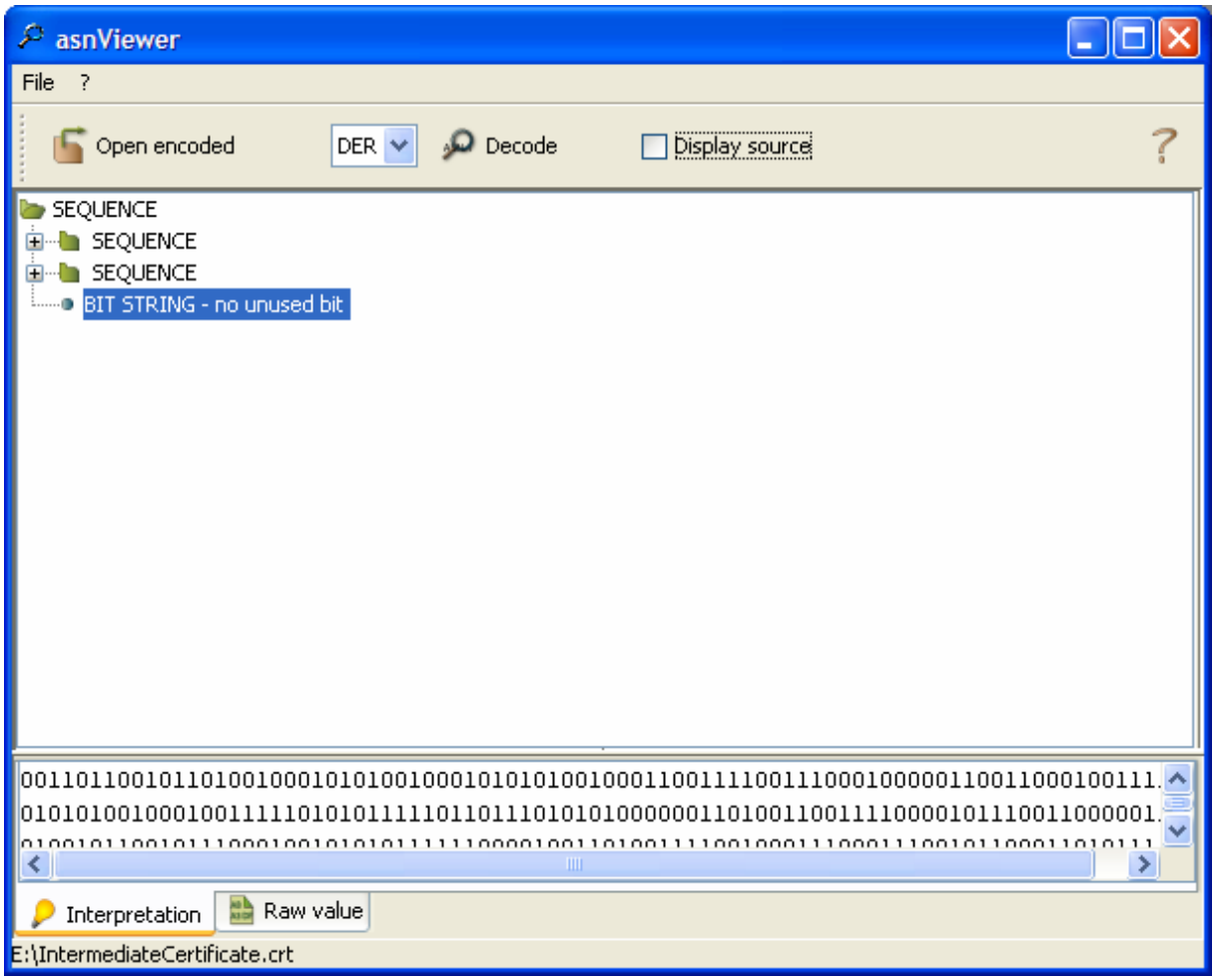

Figure 8

When displaying huge files, it is recommended to uncheck the “Display source” checkbox.

Changing the selected component is the tree view (fig. 9) is reflected to the hexadecimal view (fig. 110) when the “Display source” box is checked. The selected file is loaded into the hexadecimal view if not already loaded. Thus it may take some time to display the source file.

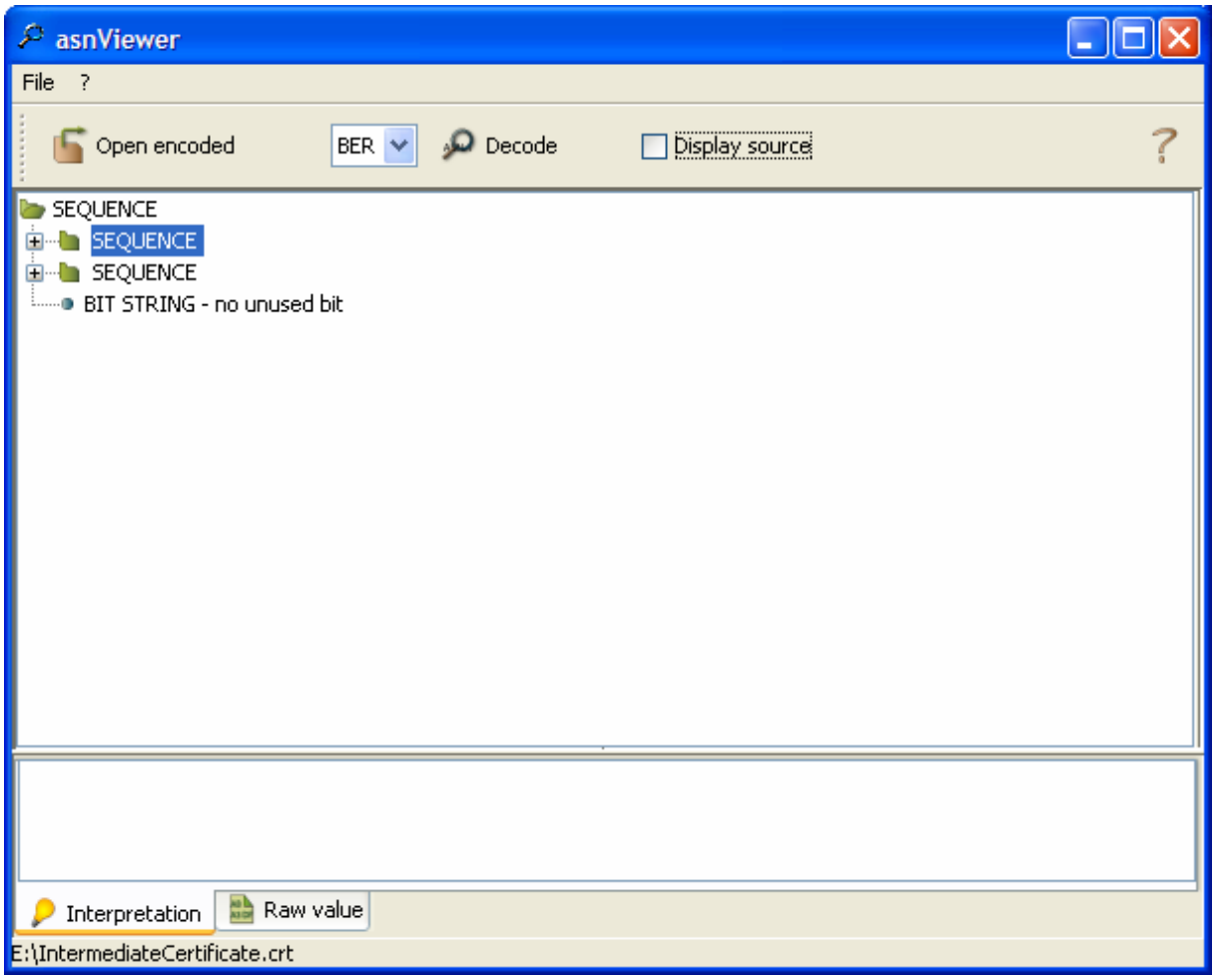

Figure 9

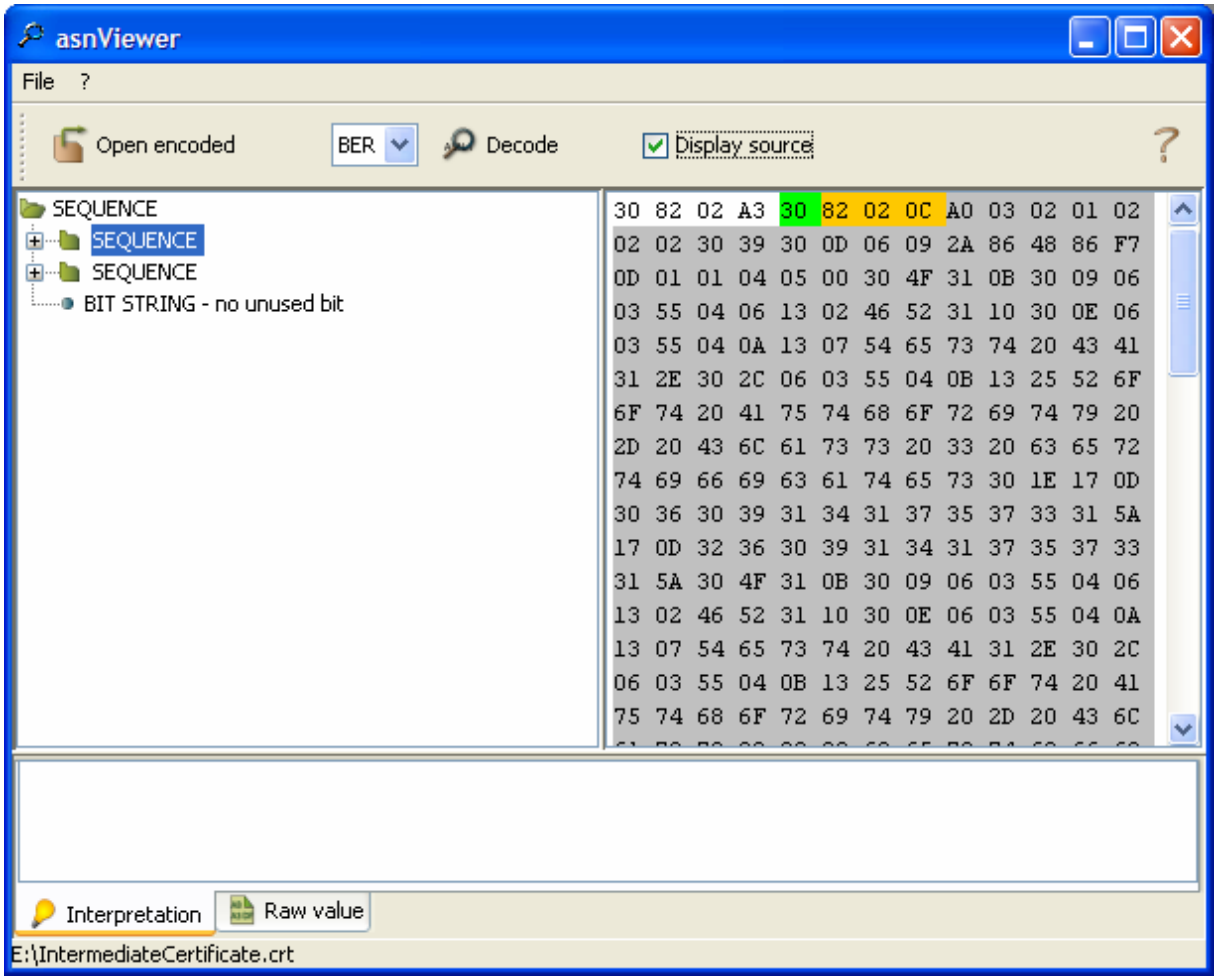
asnViewer
File ?
Open encoded
BER
Decode
Display source
SEQUENCE
SEQUENCE
SEQUENCE
BIT STRING - no unused bit
30 82 02 A3 30 82 02 0C A0 03 02 01 02
02 02 30 39 30 0D 06 09 2A 86 48 86 F7
0D 01 01 04 05 00 30 4F 31 0B 30 09 06
03 55 04 06 13 02 46 52 31 10 30 0E 06
03 55 04 0A 13 07 54 65 73 74 20 43 41
31 2E 30 2C 06 03 55 04 0B 13 25 52 6F
6F 74 20 41 75 74 68 6F 72 69 74 79 20
2D 20 43 6C 61 73 73 20 33 20 63 65 72
74 69 66 69 63 61 74 65 73 30 1E 17 0D
30 36 30 39 31 34 31 37 35 37 33 31 5A
17 0D 32 36 30 39 31 34 31 37 35 37 33
31 5A 30 4F 31 0B 30 09 06 03 55 04 06
13 02 46 52 31 10 30 0E 06 03 55 04 0A
13 07 54 65 73 74 20 43 41 31 2E 30 2C
06 03 55 04 0B 13 25 52 6F 6F 74 20 41
75 74 68 6F 72 69 74 79 20 2D 20 43 6C
Interpretation
Raw value
E:\IntermediateCertificate.crt

Figure 10

Interpretation and raw tabs

All primitive objects (independently from their tagging) are evaluated in the interpretation and raw section. The interpretation tab (fig. 11) shows the human-readable value of the ASN.1 primitive object (except `OCTET STRING` ASN.1 objects which may enclose non-printable characters).

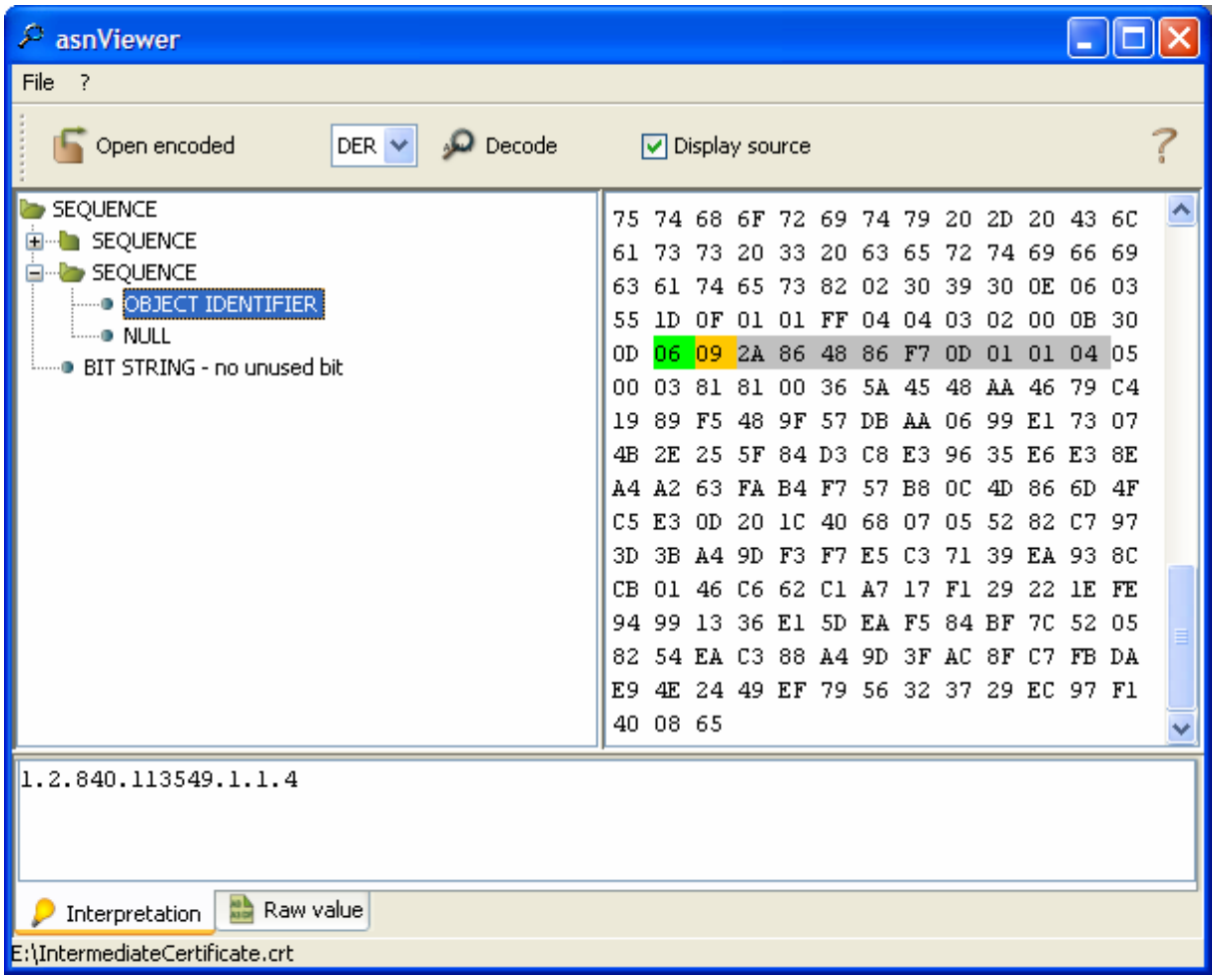

Figure 11

Each interpreted values is deduced from the source raw value (fig. 12) of the considered ASN.1 primitive object. The raw value is also displayed in the source file hexadecimal view. This tab is of great help when the file could not be loaded in the hexadecimal view (or when “display source” is not checked).

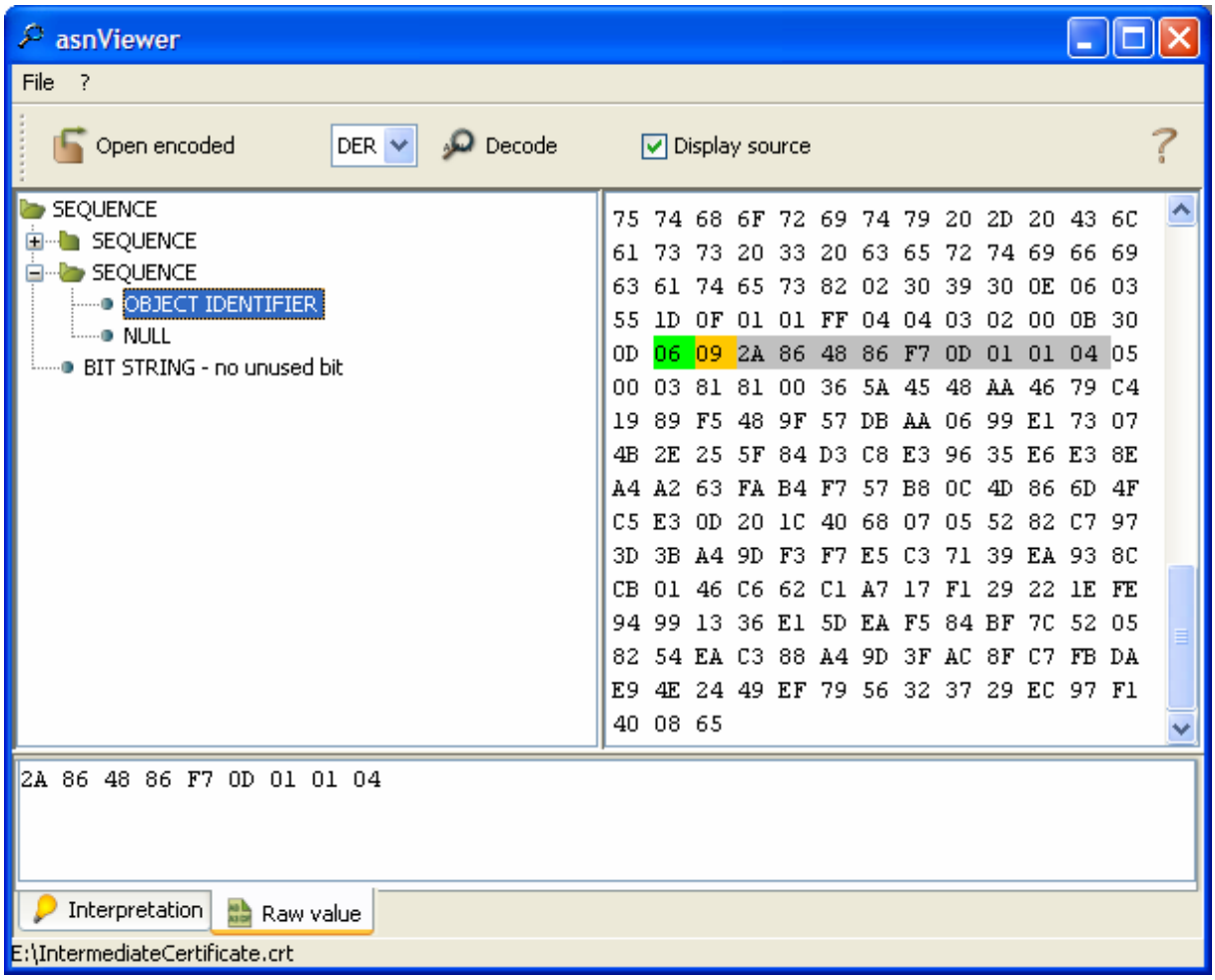

Figure 12

BIT STRING values are interpreted as `0`, `1` sequences. The number of unused bits is displayed in the tree view. However, the “raw” tab indicates the number of unused bits as its first hexadecimal couple.

Please note that “Interpretation” and “Raw” tabs only refer to primitive objects.

Known limitations

Available rules

asnViewer supports BER and DER rules only. XER is not scheduled as this encoding is intrinsically user-friendly. Other encoding rules will be available depending on PowerASN for Java encoding rules support.

Opening huge files

Huge files may not be correctly displayed in the hexadecimal view. In this case, the hex view is replaced by `<Out of memory>`. Moreover, the cursor does not change to a waiting cursor during the opening process, although the corresponding pieces of code are present.

If any idea to replace the current hexadecimal view with another able to display large files, please let me know.

Contact

For any comment, suggestion or bug fix related to current version of this software, please refer to PowerASN website (http://www.powerasn.com) to get asnViewer appropriate email adress.