Solutions to Selected Problems

Guide to Internet Cryptography

Companion Material

February 24, 2026

Preface This document provides solutions to selected problems from the book *Guide to Internet Cryptography: Security Protocols and Real-World Attack Implications*. The material is intended for educational use in courses and self-study.

Book website: https://link.springer.com/book/10.1007/978-3-031-19439-9

1 Chapter 21: Cryptographic Data Formats

Problem 21.1 Data formats

You have the task of specifying a data format for the given purpose. Would you choose TLV encoding or character-based encoding?

(a) For a new IoT protocol, your task is to specify headers that are used to switch IoT packets.

(b) For a cloud interface, your task is to specify a query language to ask for certain datasets.

Solution

(a) Here I would use TLV encoding, to save bandwidth, and for easier parsing of the headers. Also, I wouldn't want a complex, error-prone parser like XML or JSON at this level. Since arbitary byte values can be expected at this network level, escaping mechanisms for bytes representing protected characters (e.g., { and <) would have to be in place.

(b) Here I would choose a character-based encoding, to enjoy the greater flexibility in creating new data formats for new requests.

Problem 21.2 XML

(a) Why is the order of opening and closing tags important? Is the same strict syntax required for HTML?

(b) Can you sketch an XML document that contains the same information as Listing 21.1, but does not contain any text nodes?

Solution

(a) The strict order on opening an dclosing tags in XML ensures that the resulting data structure can be represented as a tree, and that the corresponding tree-based algorithms

can be applied to manipulate the data structure.
HTML parsers are much more lenient in parsing, which may result in hidden attack surfaces.
(b) Yes.

```
<?xml version="1.0" standalone="yes"?>
<conversation>
  <greeting text="Hello, world!" />
  <response text="Stop the planet, I want to get off!" />
</conversation>
```

Problem 21.3 XPath

Can you specify at least 3 different XPath expressions which select all `<title>` nodes from Listing 21.2?

Solution

```
//title
/book_list/book/title
/descendant-or-self::node()/child::title
```

Problem 21.4 XSLT

Which line in Listinf 21.6 indicates that XSLT may be Turing-complete?

Solution

Line 12, because there is a FOR loop.

Problem 21.5 XML Signature

(a) In an enveloped XML signature, the signature value is part of the signed XML tree. Now, if a signature is computed and inserted into the `<SignatureValue>` element, the hash value of the signed XML tree changes, and the signature becomes invalid. How can you solve this hen-and-egg problem?

(b) In Listing 21.7, please check if the length of the hash value in `<DigestValue>` matches the selected hash function.

(c) In the XML file from Listing 21.1, only the `<author>` elements shall be signed, but not the whole `<book_list>` element. Can you think of a transform that can achieve this?

Solution

(a) By excluding the signature value itself from the computation of the hash value.

(b) A SHA-1 hash value has 160 bits, which equals 20 bytes. In `<DigestValue>`, this value is base64 encoded. The 18 first bytes result in $18/3 \cdot 4 = 24$ bytes. Ther remaining 2 bytes are also encoded in 3 bytes, which gets us to 27 bytes. The last ASCII character `=` is for padding, so the length of 28 byte is correct.

(c) Yes, we can use a XSLT transform with the XPath `/book_list/book/author` to select only the `<author>` elements and their content, so only the concatenation of these elements will be signed.

Problem 21.6 XML Encryption

Please sketch the resulting XML file if the credit card number in Listing 21.8 is encrypted.

Solution

```
<?xml version="1.0"?>
<PaymentInfo xmlns="http://example.org/paymentv2">
  <Name>John Smith</Name>
  <CreditCard Limit="5,000" Currency="USD">
  <EncryptedData
Type="http://www.w3.org/2001/04/xmlenc#Element"
  xmlns="http://www.w3.org/2001/04/xmlenc#">
    <CipherData>
      <CipherValue>A23B45C56...</CipherValue>
    </CipherData>
  </EncryptedData>
    <Issuer>Example Bank</Issuer>
    <expiration>04/02</expiration>
  </CreditCard>
</PaymentInfo>
```

Problem 21.7 JSON

(a) Please have a look at Listing 21.11. Why must the book entries have different names, while there are each two entries for the identical names `Id`, `title`, `author` and `year`?

(b) Please sketch how you would verify the JSON web signature fromFigure 21.5. In which order would you evaluate the different, dot-separated parts?

(c) Please sketch how you would decrypt the plaintext from the JSON web encryption token given in Figure 21.6.

Solution

(a) The concept behind JSON is (nested) *associative arrays*, where the array entries are not indexed by integers, but by strings. These strings must be different is they should point to defferent entries in the associative array.

(b) I would proceed as follows:

1. Decode the JOSE header to learn the required hash and signature algorithms.
2. Use the first two labels (the signature input: encoded JOSE header.encoded JWS payload) as input to the hash algorithm.
3. Decode the last label (the encoded signature value).
4. Use the computed hash value, the signature value, and the public key of the signer

to validate the signature.

5. Iff the signature is valid, decode the middle label (the encoded JWS payload) and use it in further computations.

(c) I would proceed as follows:

1. Decode the JOSE header to learn the required decryption algorithm.
2. Retrieve the private key and decrypt the symmetric key contained in the second label.
3. Decode the IV.
4. Retrieve the associated data AAD.
5. Use the symmetric key, the IV and the ciphertext to check the authentication tag and to decrypt the ciphertext.