Set four: Hashing, protocols and privacy. Deadline: Friday October 27, 23:59 (11:59 PM)

Please consult the 'Practical assignments instructions and rules' document before proceeding with the exercises.

Exercise 1. [1 point]
Purpose of this exercise: learn to manipulate CRC computations.

Solve exercises 8 and 9 chapter 5, section 5.11: Stamp (2011, p. 155).

Exercise 2. [2.5 points]
Purpose: implement the Tiger hash.

For this exercise, you will implement the Tiger hash function. The input of your program will be a message in binary. The output should be the hash corresponding to this text.

If you are using Python to implement this program, due to Themis limitations the filename of the file containing the main function should be tigerHash.py.

Some notes about common problems and pitfalls regarding the Tiger hash:

1. **The inner round input schedule**
   Mark Stamp gives an input order in the book for the inputs to $f_{m,i}$, but this input order gives a misleading picture. Take the second input (b, c, a) in $f_{m,i}(b, c, a)$. The inner round schedule is such that the order of the output values is (b, c, a) as well! You need to make sure that you first transform this output back to (a, b, c), and only then do you transform it into the next input order (c, a, b). Forgetting to do this makes your output incorrect. An alternative implementation is to save the state of (a, b, c) away from the function, and only record the order in which they are input into the function. This is something like: pass a '1' on the first input to indicate that the first value is stored in position b, a '2' on the second input to indicate that the second value is stored in position c, and a '0' on the third input for an analogous interpretation. In such a scheme, you have a handle on which value to put in as the 'a', 'b', and 'c' value, but you don't have to worry about the order of a, b, c because that is not dependent on the function you are using.
2. **Error in figure 5.1, page 137**
   Table 5.1 on page 137 (Wiley, 2nd edition) contains a small error: the value x2 in the line w2 = x2 + w1 should of course be w2.
3. **Modulo 2^{64} operations**
   Operations in the Tiger hash are always modulo 2^{64}. Some languages can

deal with this very easily, because of their behaviour regarding integer overflow. However, other languages might have more trouble with this.

4. **Operations use unsigned values**
   Calculations in the Tiger hash use unsigned values, as opposed to signed ones. In a language like C, this makes the use of an unsigned long long as your data type obvious, but other languages do not support this natively. Additionally, C's unsigned long long is not guaranteed to be limited to 64 bits. Unsigned operations can be simulated using a mask - specifically the mask $0xFFFFFFFFFFFFFFFF$ **- on each operation**.

5. **The padding scheme**
   The padding scheme is consistent but also curious. It works as follows: first, we append an 0x01 byte (not bit!) to the input text. If the length modulo 64 $<$ 56 bytes, then we add 0x00 bytes until the length modulo 64 equals 56. The same is done if length modulo 64 $>$ 56 - we just go on until the next time that length modulo 64 equals 56. The last step of the padding is that we take the length of the original message (before padding anything on to it!), multiply it by 8 (this is the weird bit) and append it to the end of the input, such that we get a message with a length that is divisible by 64 (in terms of bytes). The multiply by 8 part is strange and it is very difficult to trace the exact specification where this is indicated, but it is the correct definition for the standard padding scheme.

Exercise 3. [0.5 points] Purpose: implement HMAC.

Create a program that reads a key and a message, then applies the HMAC algorithm to it and shows the output. The hash function you should use is the Tiger hash you implemented in exercise 2. The input of the program is given in binary form: first the key, followed by the separator byte $0xFF$, followed by the message. The key is never longer than 64 bytes. The output should also be in binary.

Note: What isn't stated very clearly in many places is that the key in the HMAC algorithm should be padded with consecutive 0x00 bytes until it has a length equal to 64 bytes. This is in fact necessary in order to do the XOR with the ipad and opad.

If you are using Python to implement this program, due to Themis limitations the filename of the file containing the main function should be HMAC.py.

Exercise 4. [1 point]
Purpose of this exercise: investigate what the probability is of obtaining PGP keys with equal fingerprints.

Determine the probability that at least two people (assuming there are 7.5 billion people (7.5 * 10^9) on earth) use a PGP key having the same public key fingerprint.

As an illustration, the fingerprint of a certain GPG key is (in blocks of 4 hexadecimal digits):

DF32 13DE B156 7732 E65E 3B4D 7DB2 A8BE EAE4 D8AA

Hint:

To compute the exact probability is rather difficult, but a very good approximation is possible, where a 'worst case' approach is used. For that approximation the following result can be used:

```
(1 - x) ^ n >= (1 - n * x) for x in [0,1]
```

Exercise 5. [1 point]
Purpose: Understand the Chinese Wall Policy.
Assume that there is a large consultancy company called GronConsultGroup that is active in several sectors {pharmaceutical, finance, media} and has six clients $\{C_1,C_2,C_3,C_4,C_5,C_6\}$ with possible conflict of interests. The clients C_1 and C_3 are from finance sector, the clients C_2, C_4 and C_6 are from the pharmaceutical sector, and C_5 is from media sector. GronConsultGroup stores the following documents from each client on which employees work on a project basis:

- $C_1 = \{D_1, D_2, D_3\}$
- $C_2 = \{D_4, D_6\}$
- $C_3 = \{D_5, D_7\}$
- $C_4 = \{D_9\}$
- $C_5 = \{D_8\}$
- $C_6 = \{D_{10}, D_{11}\}$

GronConsultGroup wants to employ Chinese Wall policy in order to prevent conflicts of interests while its employees E_1 and E_2 access to sensitive documents of its clients.
Hint: Google search: "Chinese wall access control model" provides examples of such diagrams however feel free to be creative.

a. Identify conflict of interest classes with explanations of how you derived them.

b. Sketch a tree-like diagram that classifies all documents according to conflict of interest classes you identified.

Exercise 6. [1 point]
Purpose: Get familiar with XACML.
NOTE: To answer this question, consult the XACML specification that can be found online.
http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

The security administrator of an organization A defines the XACML policy below. Given the policy:

a. Describe its semantics in plain English.

b. Discuss what will happen if you send the following request (in simplified format) to Policy Decision Point (PDP) and why.
`Request : [subject(role, Manager), subject(division, Finance), resource(resource-id, Report1), resource(confidential, true) action(action-id, read)]`

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
     xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
       http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
     PolicyId="urn:oasis:names:tc:xacml:3.0:policy1"
     RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
    <Description/>
    <Target>
        <AnyOf>
           <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Manager</AttributeValue>
                    <AttributeDesignator MustBePresent="true"
                             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
                             AttributeId="urn:oasis:names:tc:xacml:1.0:subject:role"
                             DataType="http://www.w3.org/2001/XMLSchema#string"/>
               </Match>
            </AllOf>
         </AnyOf>
         <AnyOf>
           <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Report1</AttributeValue>
                    <AttributeDesignator
                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                             AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                             DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </Match>
         </AllOf>
      </AnyOf>
    </Target>
    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:rule1" Effect="Deny">
        <Target>
           <AnyOf>
             <AllOf>
```

```
               <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">C</AttributeValue>
                    <AttributeDesignator
                             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
                             AttributeId="urn:oasis:names:tc:xacml:1.0:subject:client"
                             DataType="http://www.w3.org/2001/XMLSchema#string"/>
               </Match>
           </AllOf>
          </AnyOf>
          <AnyOf>
             <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
                    <AttributeDesignator
                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                             AttributeId="urn:oasis:names:tc:xacml:1.0:resource:confidential"
                             DataType="http://www.w3.org/2001/XMLSchema#boolean"/>
                </Match>
             </AllOf>
          </AnyOf>
       </Target>
    </Rule>
    <Rule RuleId="allowRule" Effect="Permit"/>
</Policy>
```

Exercise 7. [1 point]
Purpose: Become familiar with the Kerberos protocol.

Explain how the protocol ensures that the authentication is completed within the authentication's allowed lifetime. Also explain why the client, in message H, receives the encrypted timestamp.

Exercise 8. [1 point]
Purpose: Verify SSL certificates.

Https sites use ssl connections to ensure encrypted communication. Ssl does not imply authentication.

- What is required to do proper authentication?
- What is the meaning of 'doing authentication'?
- Is https://www.icce.rug.nl properly authenticating itself? How and why if so? What organization verified icce.rug.nl's certificate? What organization is the final CA?
- What's the validity period of www.icce.rug.nl's certificate?

All the RUG https sites should properly authenticate themselves. Can you find a rug web-site whose https connection does not properly authenticate itself?

(RUG sites not properly authenticating themselves? Over the past years usually one or two were reported. Let's see whether you guys can find more. . .)

Exercise 9. [1 point]
Purpose: Become familiar with k-anonymity.

Consider that the following table contains medical records of individuals from different backgrounds and ages, and has been released according to a certain anonymity level (i.e. k-anonymity has been applied). The Race and Age columns are quasi-identifiers and Disease column is sensitive.

Race	**Age**	**Disease**
Afro-American	34	Flu
Caucasian	21	Viral
Caucasian	21	Viral
Hispanic	18	Cancer
Afro-American	34	Flu
Hispanic	18	Cancer
Afro-American	34	Flu
Hispanic	18	Cancer

a. What is the k-anonymity level in the released table? Explain how you obtained it.

b. Do you see an anonymity problem in the released table? If yes, explain in detail with the reasons and provide a possible solution with (distinct) l-diversity. If no, justify why you do not have any problem..

Exercise 10. [1 point]
Purpose: Get a sense for differential privacy.

Explain the intuition behind ϵ-differential privacy from the following points:

a. What type of problem does it try to solve and how does it work in general terms?

b. Given the following table that summarizes the results of a survey whether certain people committed a crime X, show several query results after applying Laplace noise with $\epsilon = 2$ on the results of the query, and summarize your observations.

Crime X Committed	**Number of people**
Yes	10
No	40