Academic Year: Winter Semester 2021/22

6090: Security of Computer and Embedded Systems

Problem Sheet 1

Preliminaries & Embedded Systems Security

General Remarks

The purpose of problem sheets is to help you get to grips with the course materials. You do not have to hand them in and they will not be marked. The material may, however, be covered in the exam, and the lectures will assume that you have completed these problem sheets.

Overall, these problem sheets serve three purposes:

1. Help you to extend and deepen your knowledge of the subject as well as to apply what you learned to concrete problems
2. The exercises in the problem sheets are similar to the exercises that you might be asked in the exam.
3. They are lecture notes that include detailed references to further readings, provide additional material for catching up on preliminaries that you might not remember that well anymore, and, last but not least, repeat the most important material from the lecture.

In this problem sheet, we will deepen our knowledge of mathematical preliminaries on set theory, modular arithmetic and exponentiation. Furthermore, we will revisit embedded security topics from the lecture.

1. Repetition

In this section, we will repeat a few basic laws and notations of set theory. We will use this notation in this sheet for discussing certain aspects of access control. Later we will use the same notation in the analysis of security protocols.

Exercise 1: *Basic Set Notation*

A *set* is a collection of distinct objects. For example, the numbers 0, 3, and 48 are distinct objects. We write

$$M = \{0, 3, 48\}$$

for the set M containing the numbers 0, 3, and 48. Alternatively, we use the notation

$$X = \{x | p(x)\}$$

to describe the set containing all x that fulfil the predicate p. This is called *set comprehension* and the predicate p might be described using the usual logical notation, e.g.

- The logical connectives: ¬_ (not, negation), _ ∧ _ (and), _ ∨ _(or)
- The quantifiers: ∃_. _ (exists) and ∀_. _ (for all)

For example, we can describe the set of all even natural numbers as follows:

$$X = \{x \mid x \in \mathbb{N} \ \wedge \ x \bmod 2 = 0\}$$

where $x \in \mathbb{N}$ denotes that x is contained in the set of natural numbers (usually written as $\mathbb{N}$), _ ∧ _ is the logical connective "and", and $x \bmod 2 = 0$ denotes that x divided by two has the remainder 0, i.e., x is even.

Note that sets neither impose an order on the elements nor do sets contain the same element multiple times, i.e.

$$M = \{0, 3, 48\} = \{3, 0, 48\} = \{48, 0, 3\} = \{3, 0, 48, 3\}$$

Moreover, we write

- $\emptyset$ (or $\{\}$) for the empty set, i.e., the set that does not contain any elements
- $x \in M$ for the fact that the set M contains the element x
- $x \notin M$ for the fact that the set M does not contain the element x
- $X \subseteq Y$ for the fact that X is a subset of Y, i.e., Y contains at least all elements contained in X
  Note that X and Y might be the same set, if we want to exclude this case, we write $X \subset Y$
- $M = M_1 \cup M_2$ for the fact that M contains all elements that are contained in M_1 or M_2, i.e., the union of the both sets. Formally, we define
  $$M_1 \cup M_2 = \{x \mid x \in M_1 \vee \ x \in M_2\}$$
  (where _ ∨ _ is the logical notation for "or")
- $M = M_1 \cap M_2$ for the fact that M contains all elements that are contained in M_1 and M_2, i.e., the intersection of the both sets. Formally, we define
  $$M_1 \cap M_2 = \{x \mid x \in M_1 \wedge \ x \in M_2\}$$
  (where _ ∧ _ is the logical notation for "and")

Let $X = \{1, 3, 8\}, Y = \{0, 4, 7\}$, and $Z = \{x \mid x \text{ is a prime number smaller than } 20\}$. Answer the following questions:

1. Which of the following statements is true?
   - $3 \in Y$
   - $8 \notin X$
   - $5 \in Z$
   - $X \subseteq Y$
2. Compute the following sets:
   - $X \cup Y$
   - $X \cap Z$
3. Convince yourself that the following laws hold:
   - $\emptyset \cap A = \emptyset$
   - $A \cup B = B \cup A$
   - $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Exercise 2: ***Modular Arithmetic***

We all know modulo computations from our daily life: Talking about the 12-hour clock where we calculate module 12. For example, we are used that three hours after 11 am, it is 2 pm. More formally:

$$(11 + 3) \bmod 12 = 14 \bmod 12 = 2 \bmod 12$$

where $\bmod$ is the modulo relation. Informally, we can understand the result of the modulo operation as the remainder of an integer division, i.e.,

$$14 \div 12 = 1 \quad \text{remainder: } 2$$

In modulo computations, we can make use of the following laws:

- $(a + b) \bmod n = \big((a \bmod n) + (b \bmod n)\big) \bmod n$
- $(a \cdot b) \bmod n = \big((a \bmod n) \cdot (b \bmod n)\big) \bmod n$

Compute solutions x for following equations:

1. $x = 10 \bmod 3$
2. $x = 6 \bmod 2$
3. $2 = 7 \bmod x$
4. $1 = 5 \bmod x$
5. $5 = x \bmod 11$
6. $x = (8 \bmod 7 + 9 \bmod 7) \bmod 7$
7. $x = 17 \bmod 7$
8. $x = (8 \bmod 7 \cdot 9 \bmod 7) \bmod 7$
9. $x = 72 \bmod 7$

Exercise 3: *Exponentiation*

Exponentiation b^n(sometimes also written as $\exp(b, n)$) is a mathematical operation that corresponds to repeated multiplication:

$$b^n = \underbrace{b \cdot b \cdot \cdots \cdot b}_{n \text{ times}}$$

We call b the basis and n the exponent. For example:

$$2^8 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 256$$

Note that the result of an exponentiation grows very quickly, e.g.,

$$8^{15} = 35184372088832$$

is, on the one hand, already too large for most pocket calculators. On the other hand, for exponentiation that occur frequently in encryption examples, this is very small. Luckily, many encryption schemes use modular arithmetic.

In particular, the following equivalences are helpful in computing large exponentiation:

- $a^k \bmod n = (a \bmod n)^k \bmod n$
- $a^{k^j} = a^{(k \cdot j)}$
- $a^{(k+j)} = a^k \cdot a^j$

1. Use the previous equivalences to compute $8^{15} \bmod 13$ without using a calculator.
2. Use the rule $a^{(k+j)} = a^k \cdot a^j$ for constructing a fast (i.e., minimise the number of multiplications) exponentiation algorithm.

Hint: Consider the case where the exponent n is even, i.e., $n = 2 \cdot k = k + k$. Moreover, you might want to use recursion.

2. Embedded Systems Security

This section contains a couple of questions covering the basic concepts of embedded systems and their security that we covered in the lecture. You might want to review the lecture slides before working on these questions.

Exercise 4: *Embedded Systems Requirements*

1. What are basic requirements for embedded systems?
2. Which metrics are relevant for the dependability requirement?
3. What metric in security requirement of embedded systems is not covered by the dependability requirement as well?

Exercise 5: *Attacks Vectors in Embedded Systems*

In the lecture, we discussed the possible attack vectors in modern car systems. Please recall these and list the potential attack vectors.

References

1. Ross J. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001. The complete book is available at: http://www.cl.cam.ac.uk/~rja14/book.html

2. Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 5th edition, 2001. The complete book is available at: http://cacr.uwaterloo.ca/hac/

3. D. Elliott Bell and Leonard J. LaPadula. Secure Computer Systems: A Mathematical Model, volume II. In Journal of Computer Security 4, pages 229–263, 1996. An Electronic Reconstruction of Secure Computer Systems: Mathematical Foundations, 1973.

4. Christof Paar, Jan Pelzl. Understanding Cryptography: A Textbook for Students and Practitioners, Springer, 2010.