Exercise Sheet 3

Introduction to IT-Security

Submit your solutions via Gitlab. There are no CI tests for this assignment.
Deadline: *Wednesday, December 3rd, 09:00 a.m. CET 2025*

Attacks on Asymmetric-Key Cryptography

1. Alice wants to trick Bob into signing a message m. She knows that Bob is using plain RSA. Alice prepares two messages $m_1, m_2 \in \mathbb{Z}_N^*$ such that $m_1 \cdot m_2 \mod N = m$, and asks Bob to sign m_1 and m_2 using his private key d, which Bob does.

   (a) (2 points) How does the attack continue and why does it work?

   (b) (5 points) **Master:** Bob has decided to use a publicly known hash function H to prevent Alice from forging signatures. Instead of signing a message m directly Bob from now on will sign $H(m)$. Explain how this approach prevents forgeries. Name 3 required properties the function H should have to prevent forgeries in general.

2. Alice writes a lover letter to Bob. She signs the letter then encrypts it with Bob's public key and sends it to Bob. Unfortunately, Bob plays false. He decrypts the letter and encrypts it with Dave's public key. He then forwards the letter to Dave, such that Dave believes Alice loves him. This is called *surreptitious forwarding*.

   (a) (2 points) Does first encrypting and then signing the message solve Alices problem? Is there another problem?

   (b) (2 points) Describe a solution to improve both the sign-then-encrypt and encrypt-then-sign approach.

Weird Service

3. Bob is running a strange service on the host `weird.exercise.itsec.ias.tu-bs.de` at TCP port 3333. The service uses SSL. When connecting to the service, you can send and receive data. You will observe a strange protocol for mutual authentication. This protocol is symmetric, that is, sender and receiver authenticate in the same way.

   (a) (8 points) Using python, implement a man-in-the-middle attack by connecting twice to the service and exchanging messages.

   ```
   usage:  mitm.py [-h] [--out FILE] IP/DOMAIN PORT
   ```

   This tool should report *all* captured text messages and write them to the output file or, if not specified, to standard output (`stdout`). Note that the four-digit numbers are not text messages but part of the protocol.

   Hint: Simply echoing messages is not a sufficient.

   (b) (1 point) Additionally, provide the retrieved flag (you will know when you see it ☺) in a file called `src/flag.txt`

   (c) (3 points) Why did you succeed?