Secure and Dependable Systems
Jacobs University Bremen
Dr. Florian Rabe

Homework 7
given: 2017-05-10

Homework 7

You have to submit your solutions as announced in the lecture.
Unless mentioned otherwise, all problems are due 2017-05-19, before the lecture.
There will be no deadline extensions unless mentioned otherwise in the lecture.

Problem 7.1 *Asymmetric Encryption*

Points: 4

Implement an asymmetric encryption scheme based on RSA.

It should have the following

- a key generation function that, given $n \in \mathbb{N}$, randomly chooses primes p, q such that $p \cdot q \geq 2^n$, and then picks a random e for which d can be found,
- encryption and decryption functions that use RSA.

Write a unit test that checks the inversion condition: pick an n and an n-bit message, encrypt and decrypt it, and compare the result for equality.

Problem 7.2 *Hash Collisions*

Points: 4

Consider the following (weak) hash function $hash : \{0,1\}^* \to \mathbb{Z}_N$ for $N = 9993201131$: $hash(x)$ is obtained as follows

1. append 0s to x such that its length is a multiple of 32, and split the result into 32-bit blocks $w_1, \ldots, w_n$
2. put $h := 0$
3. [1] for each $i = 1, \ldots, n$, put $h := (h + 2 + w_i)^{1234567} \bmod 9993201131$
4. return h

Using theory and/or brute force, find a collision of $hash$. Show your work (theory and/or program).

Problem 7.3 *Password Hashing*

Points: 4

Implement $hash$ from the previous problem as a function that hashes strings by using the ASCII codes of the characters as the values $w_1, \ldots, w_n$.

Assume $hash$ is (foolishly) used to hash passwords without any salting or stretching, and we expect to have access to some hashes in the future. In order to prepare a break-in, build a table for pairs $(hash(s), s)$ for as many strings s as you can so that you can lookup passwords once you have obtained the hashes.

You may work in groups to build larger tables.

Problem 7.4 *Bonus Problem*

Points: depending on effort, at most 5% of grade

No deadline for this problem.

Using $hash$ from above, find a meaningful English word/sentence that hashes to 0.

I have not checked how difficult this is. If it is very easy, you have to find a very nice long sentence. If it is very difficult, you may also look for pronounceable words that hash to small numbers.

[1] The version I showed in the lecture used $h + w_i$ instead of $h + 2 + w_i$. I changed it to protect against attacks involving $w_1 \in \{0, 1\}$.