PART 1

Exercise 1.1

Consider a shift cipher in which each letter of the plaintext is encrypted under a fixed key. Given the ciphertext FHRR, what is the probability that the plaintext is, respectively, WOOD or WIND? What happens to the probabilities if the key is changed (at random) any time we encrypt a letter? Justify your answers.

Exercise 1.2

Consider the following protocol that authenticates A to B:

$$A \quad \rightarrow \quad B \quad : \; B, Sign_A(t_A)$$

where t_A is a time-stamp from A. Time-stamps are accepted, as usual, if they are received inside an acceptance window and are cached by the recipient in order to detect replays.

1. Show a replay attack that allows an intruder to impersonate A with another user C;
2. Illustrate a simple modification of the protocol that fixes the previous problem and show how the attack is prevented.

Exercise 1.3

A flawed RSA setup initializes the cipher with $n = 11 * p$, where p is a large prime (> 1024 bits). Describe in detail an attack that computes the private exponent a given n and the public exponent b.

PART 2

Exercise 2.1

Program `pwd` checks the password given on the command line. If we try to increase the length of the password we observe the following output:

```
$ ./pwd AAAAAAAAAAAA
ACCESS DENIED!
$ ./pwd AAAAAAAAAAAAAAAA
ACCESS DENIED!
$ ./pwd AAAAAAAAAAAAAAAAAA
*** stack smashing detected ***: ./pwd terminated
Aborted (core dumped)
```

1. Explain in detail what is going on and what kind of attack is prevented by the observed security mechanism.
2. What happens if the mechanism is turned off?

Exercise 2.2

Consider a web site which is vulnerable to a reflected XSS such as:

```
http://mysite.com/login.php?name=<script>alert("Hi there!");</script>
```

What happens when the page is loaded? Describe an attack that leaks the victim cookies and submits them to the attacker web site.

Exercise 2.3

Program `name` reads a name from the standard input and prints a greetings message:

```
$ ./name
Please insert your name: r1x
Hello r1x
```

An attacker discover the following behavior:

```
$ python -c "print 'AAAA%7\$08x'" | ./name
Please insert your name: Hello AAAA41414141
$ python -c "print '\xd0\x85\x04\x08%7\$08x'" | ./name
Please insert your name: Hello 080485d0
$ python -c "print '\xd0\x85\x04\x08%7\$s'" | ./name
Please insert your name: Hello PWDw3lld0ne
$
```

What is going on? What is probably the cause of the problem and what is a possible fix?