(Applied) Cryptography

Tutorial #1

Manuel Barbosa (mbb@fc.up.pt) Rogério Reis (rvr@fc.up.pt)

MSI/MCC/MERSI – 2021/2022

Answer the following questions

1 - Consider the set of numbers in the range 0..250 and note $p = 251$ is a prime number. Now consider two different procedures for generating a value in this range:

- Ganerate a 8-bit word b uniformly at random and then reduce $b \pmod p$, i.e. compute the remainder of b divided by p.
- Generate a 64-bit word w uniformly at random and then reduce $w \pmod p$.

Compute probability of each value in the range 0..250 occurring at the output of each of the above procedures. Are these distributions uniform? If not, can you think of a way to quantify how distant they are from uniform?

2 - Repeat the exercise for $p = 2^8$, i.e., a power of 2.

3 - Use Sage to compute the entropy of the two distributions referred in the first question above.

- HINT: The entropy value in this case is

$$\sum_{i=0}^{n-1} -\Pr[i] \cdot \log_2(\Pr[i])$$

- E.g. The entropy associated with a perfect coin flip is $-\frac{1}{2} \cdot \log_2(\frac{1}{2}) + (-\frac{1}{2} \cdot \log_2(\frac{1}{2})) = 1$.

4 - Implement in sage the following sampling procedure for a (quasi) uniform value in the range 0..250:

- sample $x \in [0..2^k]$
- output $x \pmod{251}$.
- HINT: `randrange(10)` produces a random value between 1 and 10.

Write a function in Sage to compute the entropy of this distribution for any k.

How large must k be for Sage not to tell the difference to uniform?

5 - hexdump can be used to extract randomness from /dev/urandom. Explain what the following command is doing.

```
$ hexdump -n 32 -e '1/4 "%0X" 1 "\n"' /dev/urandom!!
```

Implement an alternative command that uses /dev/urandom to create a file with random bytes.

- HINT: use the shell dd command.

Use openSSL to do exactly the same.

- HINT: look at command rand.

6 - Use openSSL to generate a key pair where private key is protected with a password.

```
openssl genrsa -aes128 4096
```

See what happens when you increase/decrease the key size.

Investigate how openSSL converts the passphrase into a cryptography key for encryption/wrapping.

7 - Use openSSL to generate random Diffie-Hellman parameters.

```
openssl gendh -aes128 2048
```

See what happens when you increase/decrease the key size. Compare to the previous case.