Cryptology complementary
Exercices#1

2018-W06

Exercise 1: One-time pad

Q.1: Let $x \in \mathbf{F}_2$ and r be an element drawn uniformly at random from $\mathbf{F}_2$ (i.e. $\Pr[r = 0] = \Pr[r = 1] = 0.5$. What is:

- $\Pr[x + r = 0]$?
- $\Pr[x + r = 1]$?
- $\Pr[x = 0 | x + r = 0]$?
- $\Pr[x = 0 | x + r = 1]$?
- $\Pr[x = 1 | x + r = 0]$?
- $\Pr[x = 1 | x + r = 1]$?

Hint. You may use Bayes' formula: $\Pr[A|B] = \Pr[B|A] \Pr[A] / \Pr[B]$.

Q.2: Let $\mathbf{x} \in \mathbf{F}_2^n$ and r be an element drawn uniformly at random from $\mathbf{F}_2^n$. For each element $\mathbf{y}$, $\mathbf{z}$ of $\mathbf{F}_2^n$, what is $\Pr[\mathbf{x} + \mathbf{r} = \mathbf{y}]$? $\Pr[\mathbf{x} = \mathbf{z} | \mathbf{x} + \mathbf{r} = \mathbf{y}]$?

Q.3: Assume that $x \in \{0,1\}^n$ is a message written as binary data. Assume that $r \in \{0,1\}^n$ is drawn uniformly at random among all binary strings of length n. Explain why observing $x \oplus r$ (the bitwise XOR of x and r) does not reveal any information about x.

Q.4: We will informally say that a cipher $\mathscr{C}$ is *perfectly secure* if observing the *ciphertext* $c = \mathscr{C}(p)$ does not reveal any new information about the *plaintext* p.

Let p and$k$ be n-bit strings. Under what condition on k is the cipher $\mathscr{C} : p \mapsto p \oplus k$ perfectly secure?

Q.5: Let $\mathscr{C}$ be a perfectly secure cipher as above. Is its concatenation $\mathscr{C}^2 : p||p' \mapsto p \oplus k||p' \oplus k$ perfectly secure?

Exercise 2: Binary vectors

Q. 1: Write a small "naïve" C function that computes the scalar product of two vectors of $\mathbf{F}_2^{32}$. This function must have the following prototype:

```
uint32_t scalar32_naive(uint32_t x, uint32_t y).
```

Q. 2: Write another implementation of the same function, of prototype

```
uint32_t scalar32_popcnt(uint32_t x, uint32_t y),
```

that uses a *bitwise and* instruction "`&`" and the *population count* function for 32-bit words "`__builtin_popcount()`".

Q. 3: Write a function that computes a matrix-vector product $\mathbf{x}M$ for $M \in \mathscr{M}_{32}(\mathbf{F}_2)$, using a scalar product as a sub-routine. This function must have the following prototype:

```
uint32_t mul32_scalar(uint32_t m[32], uint32_t x).
```

Q. 4: Write another such function using a *table* implementation. You may assume that all of the linear combinations of eight consecutive rows of the matrix have been precomputed and stored in a table `uint32_t m[4][256]`. That is, `m[0][x]` is equal to $\sum_{i \in \texttt{nz(x)}} M_i$, `m[1][x]` is equal to $\sum_{i \in \texttt{nz(x)}} M_{i+8}$, etc., where `nz(x)` is the set of the indices of the non-zero bits of `x`. This function must have the following prototype:

```
uint32_t mul32_table(uint32_t m[4][256], uint32_t x).
```

Q. 5: Write a test function that computes a large number (e.g. 2^{24}) of matrix-vector multiplications. Time the execution of the resulting programn, in function of the chosen implementation (including different implementations for the scalar product used in `mul32_scalar`).

Q. 6: If possible, redo the previous question with another compiler.