Introduction to Cryptography – Exercise no. 3

Submit in Pairs/Single to mailbox 19 by 27/12/12, 1:00 p.m.

1. Let $X$ be a set of $N$ different values. Let $S_1, S_2$ be two subsets chosen randomly from $X$ (i.e., each element of $X$ has the same probability to be a member of $S_1$, and likewise for $S_2$).

   (a) Prove that when $|S_1| \cdot |S_2| = N$ (when $|S_1|, |S_2| > N^\epsilon$), the probability that $S_1 \cap S_2 \neq \phi$ is 63.2%.

   (b) Find the minimal value $t$, such that $|S_1| = |S_2| = t$ such that the probability that $S_1 \cap S_2 \neq \phi$ is 50%.

2. Prof. Menjir has suggested the following compression function: The function accepts a 128-bit chaining value $CV$ and a 1408-bit message block $M$. The 1408-bit message block $M$ is treated as eleven 128-bit subkeys, i.e., $M = K_0 || K_1 || \ldots || K_{10}$ (the first 128-bit of the message being the first subkey, etc.). To compress the inputs, the value $CV$ is encrypted using 10 rounds of AES, where the 11 subkeys are $K_0, K_1, \ldots, K_{10}$, with an additional feed-forward.

   Formally:
   
   $$CV' = AES_{K_0, K_1, \ldots, K_{10}}(CV) \oplus CV.$$ 

   (a) Show how to construct collisions in this compression function as efficiently as you can.

   (b) Show how to find a preimage of a given chaining value in this hash function as efficiently as you can.

3. Calculate by using the (Extended) Euclidean Algorithm:

   (a) $\gcd(891, 673)$

   (b) $\gcd(975, 124)$

   (c) The inverse of 48 modulo 101.

   (d) The inverse of 20 modulo 101.

   Provide the steps of the computation in your answers.

4. Solve the following two systems:

   $$\begin{align*}
   x &\equiv 4 \pmod{5} \\
   x &\equiv 3 \pmod{7} \\
   x &\equiv 1 \pmod{11}
   \end{align*}$$

   and

   $$\begin{align*}
   x &\equiv 3 \pmod{4} \\
   x &\equiv 5 \pmod{6} \\
   x &\equiv 2 \pmod{13}
   \end{align*}$$
5. Let \( p \) be a prime number and assume that the decomposition of \( p - 1 \) into its prime factors is known.

(a) Show an algorithm that given a number \( g \in \mathbb{Z}_p^* \) can determine whether \( g \) is a generator of \( \mathbb{Z}_p^* \). Analyze the time complexity of the algorithm you suggested, and prove its correctness (Algorithms that require more than \( O(\log^3(p)) \) modular multiplications will be disregarded).

(b) Show an algorithm that given a number \( g \in \mathbb{Z}_p^* \) can find the order of \( g \) in \( \mathbb{Z}_p^* \). Your algorithm should not perform more than \( O(\log_3(p)) \) modular multiplications. Analyze the time complexity of the algorithm you suggested.

6. This question deals with the complexity of the Euclidean Algorithm. Assume that we start the algorithm with \( a > b > 0 \) and we get:

\[
\begin{align*}
a &= q_1b + r_1 & 0 < r_1 < b \\
b &= q_2r_1 + r_2 & 0 < r_2 < r_1 \\
r_1 &= q_3r_2 + r_3 & 0 < r_3 < r_3 \\
&\vdots \\
r_n &= q_{n+1}r_n & 0 < r_n < r_{n-1} \\
r_1 &= q_nr_{n-1} + r_n & \text{for all } 0 \leq i < n
\end{align*}
\]

Denote \( \alpha = \frac{1 + \sqrt{5}}{2} \approx 1.618 \).

(a) Prove that \( \alpha^2 = \alpha + 1 \).

(b) Prove that \( r_{n-i} \geq \alpha^i \) for all \( 0 \leq i < n \).

(c) Prove that the number of divisions in the Euclidean Algorithm \( \leq \log_\alpha a \).

(d) Is this bound tight?