CS 409M : Introduction to Cryptography Fall 2024

Assignment 3
Echoes of Randomness

Instructor: Sruthi Sekar *TAs:* Giriraj Singh, Gyara Pragathi, Nilabha Saha

Problem 1: PRF, or Not PRF, That is the Question

Let F be a length-preserving pseudorandom function. For the following constructions of a keyed function $F' : \{0,1\}^n \times \{0,1\}^{n-1} \to \{0,1\}^{2n}$, state whether F' is a pseudorandom function. If yes, prove it; if not, show an attack.

a) $F'_k(x) := F_k(0||x)||F_k(1||x)$.

b) $F'_k(x) := F_k(0||x)||F_k(x||1)$.

Let $F^{(1)}$ and $F^{(2)}$ be length-preserving pseudorandom functions. For the following constructions of a keyed function F', state whether F' is a pseudorandom function. If yes, prove it; if not, show an attack.

c) $F' : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ where $F'_{k_1||k_2}(x) = F^{(1)}_{k_1}(x)||F^{(2)}_{k_2}(x)$.

d) $F' : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ where $F'_k(x_1||x_2) := F^{(1)}_k(x_1)F^{(2)}_k(x_2)$.

e) $F' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ where $F'_k(x) = F^{(1)}_k(F^{(2)}_k(x))$.

Problem 2: The Randomness Trials

Let F be a length-preserving pseudorandom function. For each of the following constructions, state its expansion factor and if it is a pseudorandom generator or not. If you claim it is a pseudorandom generator, prove the same. If you claim it is not a pseudorandom generator, provide a distinguisher with a non-negligible advantage.

a) $G(s) := F_s(1)||F_s(2)||\cdots||F_s(\ell-1)||F_s(\ell)$

b) $G(s) := F_s(2^0)||F_s(2^1)||\cdots||F_s(2^{\ell-1})||F_s(2^\ell)$

c) $G(s) := F_1(s)||F_2(s)||\cdots||F_{\ell-1}(s)||F_\ell(s)$

Problem 3: Gap b/w Multiple-EAV-Secure and CPA-Secure

Assuming the existence of pseudorandom functions, prove that there is an encryption scheme that has indistinguishable multiple encryptions in the presence of an eavesdropper, but is not CPA-secure.
Hint: The scheme need not be "natural". You will need to use the fact that in a chosen-plaintext attack, the adversary can choose its queries to the encryption oracle adaptively.

Problem 4: The Illusion of Randomness

Consider the following keyed function F: For security parameter n, the key is an $n \times n$ boolean matrix A and an n-bit boolean vector b. Define $F_{A,b} : \{0,1\}^n \to \{0,1\}^n$ by $F_{A,b}(x) := Ax + b$, where all operations are done modulo 2. Show that F is not a pseudorandom function.

Problem 5: EAV, CPA, or Both?

Let F be a pseudorandom function and G be a pseudorandom generator with expansion factor $\ell(n) = n + 1$. For each of the following encryption schemes, state whether the scheme has indistinguishable encryptions in the presence of an eavesdropper and whether it is CPA-secure. (In each case, the shared key is a uniform $k \in \{0,1\}^n$.) Explain your answer.

a) To encrypted $m \in \{0,1\}^{n+1}$, choose uniform $r \in \{0,1\}^n$ and output the ciphertext $\langle r, G(r) \oplus m \rangle$.

b) To encrypt $m \in \{0,1\}^n$, output the ciphertext $m \oplus F_k(0^n)$.

c) To encrypt $m \in \{0,1\}^{2n}$, parse m as $m_1 || m_2$ with $|m_1| = |m_2|$, then choose uniform $r \in \{0,1\}^n$ and send $\langle r, m_1 \oplus F_k(r), m_2 \oplus F_k(r+1) \rangle$.

Problem 6: Custom Counter Mode

Let F be a pseudorandom permutation. Consider the mode of operation in which a uniform value $\texttt{ctr} \in \{0,1\}^n$ is chosen, and the i^{th} ciphertext block c_i is computed as $c_i := F_k(\texttt{ctr} + i + m_i)$. Show that this scheme does not have indistinguishable encryptions in the presence of an eavesdropper.

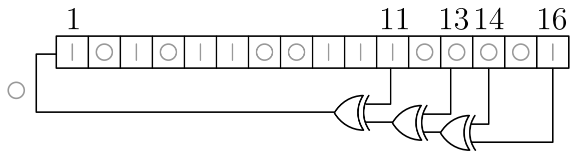
Problem 7: Conditional Composition (Hard Problem)

Let $\Pi_1 = (\texttt{Enc}_1, \texttt{Dec}_1)$ and $\Pi_2 = (\texttt{Enc}_2, \texttt{Dec}_2)$ be two encryption schemes for which it is known that at least one is CPA-secure (but you don't know which one). Show how to construct an encryption scheme Π that is guaranteed to be CPA-secure as long as at least Π_1 or Π_2 is CPA-secure. Provide a full proof of your solution.
Hint: Generate two plaintext messages from the original plaintext so that knowledge of either one reveals nothing about the original plaintext, but knowledge of both enables the original plaintext to be computed.

Advanced Problem: Attack on LFSRs

Linear Feedback Shift Registers (LFSRs) are a type of shift register where the input bit is a linear function of its previous state, typically involving XOR operations on specific bits of the register. Due to their simplicity and efficiency, LFSRs are widely used in cryptography, particularly in stream ciphers, pseudo-random number generation, and scrambling of data for error detection and correction. Their deterministic nature allows for the generation of sequences with desirable properties like long periods and good statistical characteristics, making them suitable for generating key streams in stream ciphers such as A5/1, used in GSM mobile communications. However, the linearity of LFSRs also makes them vulnerable to certain cryptanalytic attacks, leading to the development of more complex, non-linear variants in modern cryptographic applications.

In this problem, you will investigate you can mount an attack on a simple LFSR model. Let us take the example of a simple (Fibonacci) LFSR:



The symbols in the boxes represent the current state of the LFSR; in particular, the current state of the LFSR above is 1010110011100001. Since the state consists of 16 bits, this is called a 16-bit LFSR. We index the positions on the state from left to right with indices ranging from 1 to 16. Certain positions are said to be "tapped"; in the above example, the positions at indices $11, 13, 14$, and 16 are tapped. This LFSR can be used as a "PRNG" as follows:

- `Init`: It takes in a 16-bit seed s and initialises the state $\texttt{st}_0$ be the bits of the seed written left-to-right on the LFSR body. For instance, if the above is the starting state of the LFSR, then $\texttt{st}_0 = 1010110011100001$.

- `GetBits`: The bit y output is the bit at the rightmost position of the LFSR, i.e., the bit at index 16. For instance, in the above example, the output bit would be 1. If the current state is $\texttt{st}_t$, then the next state, $\texttt{st}_{t+1}$ would be obtained by shifting each bit one position to the right (the rightmost bit would "fall off" the LFSR as output), and replacing the first (leftmost) bit with the XOR of all the bits which were at the tapped position in $\texttt{st}_0$. In the above example, you can see that the bit that would get replaced in the first position in the next state would be given by $1 \oplus 0 \oplus 0 \oplus 1 = 0$. Thus, the next state of the LFSR would be 0101011001110000.

Now assume that someone uses a n-bit LFSR (where $n \geq 2$) with an unknown seed (initial state).

a) You know the taps to be at positions marked by the distinct indices $[I_1, I_2, \ldots, I_T]$ where $2 \leq T \leq n$ and $1 \leq I_j \leq n \ \forall j \in \{1, 2, \ldots, T\}$. Suppose you are given the first n output bits of the LFSR in order. Design an attack that would enable you to predict all the subsequent output bits of the LFSR with 100% accuracy.

b) Now assume that you do not know where the taps are (however, you do know there are ≥ 2 tapped positions). Suppose you are given the first $2n$ output bits of the LFSR in order. Design an attack that would enable you to predict all the subsequent output bits of the LFSR with 100% accuracy.

Historical Note: The Content Scrambling System (CSS) is a digital rights management (DRM) and encryption system used to protect content on DVD-Video discs. Introduced in 1996, CSS was designed to prevent unauthorized copying and distribution of DVD content by scrambling the data on the disc, making it accessible only to authorized devices with the appropriate decryption keys. CSS employs a pair of LFSRs to generate the pseudo-random keystream that is XORed with the data on the DVD to encrypt it. The LFSRs used in CSS are relatively simple: each LFSR is a 17-bit register with a specific set of taps. The two LFSRs work together, with one controlling the operation of the other, to produce the final keystream. However, the simplicity of these LFSRs and the overall design of CSS made it vulnerable to cryptanalysis, and CSS was eventually broken, leading to the development of more secure content protection systems.