Cryptography

Academic Year 2025-2026

Homework 1

Angela Speranza, ID 0001190011

October 17, 2025

Exercise 1.
Claim. The scytale cipher, defined as a triple of algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, is not perfectly secure.

According to Shannon's definition of perfect secrecy, a scheme is *perfectly secret* if for all $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$ such that $\Pr[C = c] > 0$, it holds that

$$\Pr[M = m | C = c] = \Pr[M = m].$$

The key point of this definition is that tge knowledge of the ciphertext c does not give any information about the message m. In other words, the a posteriori probability of m given c is equal to the a priori probability of m. Equivalently, a scheme is perfectly secret if for all $m_0, m_1 \in \mathcal{M}$ and all $c \in \mathcal{C}$ such that $\Pr[C = c] > 0$, it holds that

$$\Pr[C = c | M = m_0] = \Pr[C = c | M = m_1].$$

This means that the probability of obtaining a certain ciphertext c is the same regardless of which message m_0 or m_1 was encrypted. We also have to recall a very important result, which can be seen also as a limitation of perfect secrecy: if a scheme is perfectly secret, then the size of the key space must be at least as large as the size of the message space, i.e. $|\mathcal{K}| \geq |\mathcal{M}|$. These are all the ingredients we need to prove that the scytale cipher is not perfectly secure.

Brief description of the scytale cipher. The scytale is a classical transposition cipher: a strip is wrapped around a cylinder. In this case, the key is the rod's diameter, namely the number of letters that fit "around" it. The message is written across the strip while it is being wrapped arounf the rod: unwinding the strip produces the ciphertext. The receiver, who knows the diameter of the rod, wraps the strip around a rod of the same diameter and reads the message across. If the text does not fit perfectly in the rectangle, it is usually padded with some extra characters (e.g. X's), appendend at the end of the message. For a message of length n, the scytale key space is defined as

$$\mathcal{K}_n = \{1, 2, \ldots, n\}$$

where each $s \in \mathcal{K}_n$ denotes the number of letters that fit around the rod. Let $r = \lfloor n/s \rfloor$ be the number of rows in the transposition.

- The key generation algorithm Gen chooses a key $s \in \mathcal{K}_n$ uniformly at random.
- The encryption algorithm Enc, given a plaintext m of length n, proceeds as follows:
  - Write m in a $r \times s$ matrix, row by row. If the last row is not complete, pad it with extra characters (e.g. X's).

- Read the matrix column by column to obtain the ciphertext c.

- The decryption algorithm Dec reverses the encryption process: it fills the same $r \times s$ matrix column by column with the ciphertext c and reads it row by row to recover the plaintext m, discarding any padding characters.

From this description, we easily notice that the encryption and decryption algorithms are deterministic, i.e. the same plaintext encrypted with the same key will always produce the same ciphertext. Also, the encryption is merely a *permutation* of the plaintext characters, preserving their frequency and the message length.

Proof. We can now prove that the scytale cipher is not perfectly secure. We will do so by showing two different arguments.

- ***1: Counting argument.*** Shannon's bound require that $|\mathcal{K}| \geq |\mathcal{M}|$ for perfect secrecy. However, in the scytale cipher, the key space is $\mathcal{K}_n = \{1, 2, \ldots, n\}$, which has size n. On the other hand, the message space $\mathcal{M}$ consists of all possible strings of length n over a given alphabet. Let Σ be the alphabet used. The size of the message space is then $|\mathcal{M}| = |\Sigma|^n$. This is exponentially larger than the key space for any alphabet of size greater than 1 (e.g., the English alphabet has 26 letters). Therefore, $|\mathcal{K}| < |\mathcal{M}|$, violating Shannon's requirement for perfect secrecy.

- ***2: Indistinguishability argument*** Consider two different messages $m_0 = a^n$ and $m_1 = b^n$, where a and b are distinct characters in the alphabet Σ. When we encrypt these messages using the scytale cipher, for every key $s \in \mathcal{K}_n$, we have thta:

$$\mathsf{Enc}_s(m_0) = a^n \quad \text{and} \quad \mathsf{Enc}_s(m_1) = b^n.$$

  Hence, if we observe a ciphertext $c^* = a^n$, we can be certain that the original message was m_0 and not m_1. This means that

$$\Pr[C = c^* | M = m_0] = 1 \quad \text{and} \quad \Pr[C = c^* | M = m_1] = 0$$

  To better formalize this argument, we can consider the indistinguishability experiment $\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi}$ for an adversary $\mathcal{A}$. In the standard perfect-secrecy experiment, the adversary chooses $(m_0, m_1) = (a^n, b^n)$ and receives $c = \mathsf{Enc}_s(m_b)$ for a random key s and bit b. If the adversary outputs 0 when $c = a^n$ and 1 otherwise, Eve always guesses the correct bit b. Her success probability is thus 1, whereas perfect secrecy requires success probability $\frac{1}{2}$. Hence, the scytale fails the indistinguishability definition as well. Perfect secrecy requires that these probabilities be equal for all ciphertexts, which is clearly not the case here. The scheme leaks information about the frequency distribution of the symbols, so any two messages with different simbols histograms can be distinguished.

Conclusion. The scytale cipher is not perfectly secret. Its keyspace has size n, while the message space has size $|\Sigma|^n$, violating Shannon's necessary bound $|\mathcal{K}| \geq |\mathcal{M}|$. Moreover, scytale encryption merely permutes message symbols, preserving their frequencies and length; hence, the ciphertext distribution depends on the plaintext. An adversary can distinguish messages such as a^n and b^n with probability 1. Therefore, the scytale cipher cannot satisfy Shannon's definition of perfect secrecy. Perfect secrecy can only be achieved with keys having entropy at least as large as that of the message, e.g., a one-time pad.

——————————— * ———————————

Exercise 2.

Claim. Consider a pseudorandom generator G with expansion factor ℓ, and let ℓ' be any polynomial such that $\ell(n) > \ell'(n) > n$. Define

$$H(x) = G(x)|_{\ell'(|x|)}.$$

Prove that H is still a pseudorandom generator.

Let's first recall the requirements for a pseudorandom generator (PRG). An algorithm G must be:

- **Deterministic:** it does not rely on any kind od randomness.
- **Efficient:** it runs in polynomial time.
- **Expanding:** it outputs a longer string, i.e. $\ell(n) > n$.
- **Pseudorandom:** its output cannot be distinguished from a truly random string of the same length by any efficient adversary.

Immediate observations:

- H is deterministic and PPT if G is too, since H just runs G and takes a prefix.
- H is expanding because $|H(x)| = \ell'(|x|) > |x|$.

Algorithm 1 Computation of $H(x)$

```
Input: $x \in \{0,1\}^n$
Output: $H(x) \in \{0,1\}^{\ell'(n)}$
$y \leftarrow G(x)$ ;                                   // Compute the output of $G$
return prefix($y, \ell'(n)$)
```

Proof. Suppose there exists a PPT distinguisher D that distinguishes the output of H from a truly random string of length $\ell'(n)$ with non-negligible advantage $\epsilon(n)$.

We build a distinguisher D^* for G as follows.

Algorithm 2 Construction of D^* from D

```
Input: a string $z$ of length $\ell(n)$
Output: $D^*(z) \in \{0,1\}$
Find $n$ such that $|z| = \ell(n)$ ;                    // $\ell$ is polynomial and monotone
$p \leftarrow$ prefix($z, \ell'(n)$)
return $D(p)$
```

Now consider two cases withthe corresponding probabilities.

Let X be a uniformly random seed in $\{0,1\}^n$, let r be a uniformly random string in $\{0,1\}^{\ell(n)}$, and let r' be a uniformly random string in $\{0,1\}^{\ell'(n)}$.

- When the input to D^* is $G(X)$ (the pseudorandom output of G on a random seed X):

$$\Pr[D^*(G(X)) = 1] \;=\; \Pr[D(\text{prefix}(G(X), \ell'(n))) = 1] \;=\; \Pr[D(H(X)) = 1].$$

- When the input to D^* is a truly random string r of length $\ell(n)$:

$$\Pr[D^*(r) = 1] \;=\; \Pr[D(\text{prefix}(r, \ell'(n))) = 1] \;=\; \Pr[D(r') = 1],$$

because the prefix of a uniform string is uniform of the shorter length.

Therefore the distinguishing advantage of D^* against G equals the advantage of D against H:

$$|\Pr[D^*(G(X)) = 1] - \Pr[D^*(r) = 1]| = |\Pr[D(H(X)) = 1] - \Pr[D(r') = 1]| \geq \epsilon(n).$$

This contradicts the pseudorandomness hypothesis on G. Hence no such D exists and H is a pseudorandom generator.

Conclusion. Intuitively, taking only the first $\ell'(n)$ bits of $G(x)$ does not make its output any easier to distinguish from random, because a prefix of a random string is still random. Formally, we built a distinguisher D^* for G from any distinguisher D for H with the same advantage, contradicting the security of G. Thus, H is also a pseudorandom generator.

*

Exercise 3.

Pseudorandom Generator 1. Prove that the following function is a pseudorandom generator:

$$G_1(x) = x \cdot 010$$

The function G_1 is **not a pseudorandom generator**. G_1 simply appends the constant 3-bit suffix "010" to the input string x. This means that the last three bits of the output are always the same, regardless of the input.

Consider an efficient distinguisher D that on input $y \in \{,1\}^{n+3}$ outputs 1 iff the last three bits of y are "010", and 0 otherwise. If $y = G_1(x)$ with uniform $x \in \{0,1\}^n$, then $D(y) = 1$ with probability 1. If, on the other hand, y is uniform in $\{0,1\}^{n+3}$, then $D(y) = 1$ with probability 2^{-3}. Thus, the distunguishing advantage of D is is $1 - 2^{-3}$, which is non-negligible. Therefore, G_1 is not a pseudorandom generator.

Pseudorandom Generator 2. Prove that the following function is a pseudorandom generator:

$$G_2(x) = F(x, 0^{|x|})$$

Here, F is a pseudorandom function.

The function G_2 is **not a pseudorandom generator**. By definition, G_2 outputs $|x|$ bits, meaning that the pseudorandom function returns an n-bit string when its second argument has length n. Thus, G_2 does not expand its input, violating a key requirement for pseudorandom generators: $|G_2(x)| = |x|$. Since expansion is one of the necessary conditions, G_2 cannot be a pseudorandom generator.

Pseudorandom Generator 3. Prove that the following function is a pseudorandom generator:

$$G_3(x) = \begin{cases} x \cdot x & \text{if } |x| \leq 2 \\ F(x, 1|x|) \cdot F(x, 0|x|) & \text{otherwise} \end{cases}$$

The function G_3 **is a pseudorandom generator** for sufficiently large input lengths. It respects all the requirements for being a pseudorandom generator: it is deterministic and poly-time since it makes at most two evaluations of F. In addition, for all n, $|G_3(x)| = 2n > n$, hence the expansion requirement is satisfied.

We can prove it by reduction: we build an adversary D_F from a distinguisher D for G_3. D_F has access to an oracle $\mathcal{O}$ that is either $F(k, \cdot)$ for a uniform key k or a truly random function.

Algorithm 3 Function $D_f(1^n)$

```
Input: Security parameter - $1^n$
Output: Output of $D$ on constructed input
$y \leftarrow \mathcal{O}(1^n)$ ;            // Query the oracle on input $1^n$
$z \leftarrow \mathcal{O}(0^n)$ ;            // Query the oracle on input $0^n$
return $D(y \| z)$ ;   // Run $D$ on the concatenation of the two outputs
```

After answering all of D's queries (by using thr same oracle responses), D_F outputs whatever D outputs. Now consider the two cases:

- If the oracle is $F_k(\cdot)$, then each sample returned to D is distributed as $G_3(k)$ and thus D precisely sees G_3 (random input).
- If the oracle is a truly random function, then the concatenated output are two independent uniform strings of length n, which is itself a uniform string of length $2n$. Thus, D sees a uniform string of length $2n$.

Hence the adversary's distinguishing advantage equals to D's, which, by assumption, is non-negligible. This contradicts the pseudorandomness of F. Therefore, G_3 is a pseudorandom generator.

The special-case behavior for $|x| \leq 2$ affects only finitely many input lengths; security notions are asymptotic, so these finitely many exceptions do not preclude G_3 being a PRG for large n.

Pseudorandom Function 1. Prove that the following binary function is a pseudorandom function:

$$F_1(k, x) = x \oplus k$$

We show that it is **not a PRF** by constructing a distinguisher.

Consider a distinguisher D with oracle access to a function $H(\cdot)$ that behaves as follows:

- Query H on two distinct inputs x_1, x_2 (for example $x_1 = 0^n$ and $x_2 = 1^n$), obtaining outputs y_1, y_2.
- Compute $y_1 \oplus y_2$ and output 1 if $y_1 \oplus y_2 = x_1 \oplus x_2$, otherwise output 0.

If $H(\cdot) = F_1(k, \cdot) = \cdot \oplus k$, then

$$y_1 \oplus y_2 = (x_1 \oplus k) \oplus (x_2 \oplus k) = x_1 \oplus x_2,$$

so D outputs 1 with probability 1:

$$\Pr[D^{F_1(k,\cdot)} = 1] = 1.$$

If instead H is a truly random function, then y_1 and y_2 are independent uniformly random n-bit strings. In this case,

$$\Pr[y_1 \oplus y_2 = x_1 \oplus x_2] = 2^{-n},$$

since $y_1 \oplus y_2$ is uniformly distributed over $\{0,1\}^n$. Therefore,

$$\Pr[D^{R(\cdot)} = 1] = 2^{-n}.$$

The distinguishing advantage of D is thus

$$\text{Adv}(D) = \left|1 - 2^{-n}\right| = 1 - 2^{-n},$$

which is non-negligible.

Hence, there exists an efficient distinguisher with non-negligible advantage, and we conclude that $F_1(k, x) = x \oplus k$ is **not** a pseudorandom function.

Pseudorandom Function 2. Prove that the following binary function is a pseudorandom function:

$$F_2(k, m) = G(k)\big|_{|k|} \oplus m$$

where G is a pseudorandom generator.

We show that it is **not a PRF** by constructing a distinguisher.

Consider a distinguisher D with oracle access to a function $H(\cdot)$ that behaves as follows:

- Query H on two distinct inputs m_1, m_2 (for example $m_1 = 0^{|k|}$ and $m_2 = 1^{|k|}$), obtaining outputs y_1, y_2.
- Compare the left $|G(k)|$-bit blocks of y_1 and y_2, and output 1 if they are equal, otherwise output 0.

If $H(\cdot) = F_2(k, \cdot) = G(k)\big|_{|k|} \oplus \cdot$, then the left $|G(k)|$-bit blocks of y_1 and y_2 are identical, since both are derived from the same $G(k)$. Therefore, D outputs 1 with probability 1:

$$\Pr[D^{F_2(k,\cdot)} = 1] = 1.$$

If instead H is a truly random function, then the left $|G(k)|$-bit blocks of y_1 and y_2 are independent uniformly random strings. Thus, they are equal only with probability $2^{-|G(k)|}$:

$$\Pr[y_1^{\text{left}} = y_2^{\text{left}}] = 2^{-|G(k)|},$$

and therefore,

$$\Pr[D^{R(\cdot)} = 1] = 2^{-|G(k)|}.$$

The distinguishing advantage of D is thus

$$\mathrm{Adv}(D) = \left|1 - 2^{-|G(k)|}\right| = 1 - 2^{-|G(k)|},$$

which is non-negligible.

Hence, there exists an efficient distinguisher with non-negligible advantage, and we conclude that $F_2(k, m) = G(k)\big|_{|k|} \oplus m$ is **not** a pseudorandom function.