# **DENEY FÖYÜ**

# **Deney Adı: Sözde Rastgele Sayı Üretimi**

# **Deney No: 01**

# **Deneyin Yapıldığı Tarih: 05.03.2025**

# **Deneyi Yapanlar: Abdulsamet Duran 22054189 Rastgele Sayı Üreteçleri (PRNG) ve Uygulama Açıklamaları**

## **1. Giriş**

Bu doküman, dört farklı sözde rastgele sayı üreteci (PRNG - Pseudo Random Number Generator) algoritmasını açıklamakta ve bunların Python ile uygulanmasını anlatmaktadır. Kullanılan PRNG algoritmaları şunlardır:

* **Mersenne Twister**
* **Xorshift**
* **PCG (Permuted Congruential Generator)**
* **Blum Blum Shub (BBS)**

Her bir algoritmanın mantığı, kullanılan formüller ve kodun açıklamaları aşağıda verilmiştir.

## **2. Mersenne Twister Algoritması**

**Mersenne Twister**, yüksek kalite ve uzun periyotlu rastgele sayı üretimi için geliştirilmiş bir PRNG algoritmasıdır. 1997 yılında Matsumoto ve Nishimura tarafından geliştirilmiştir.

### **Çalışma Prensibi**

* **624 uzunluğunda bir durum dizisi** kullanır.
* **Özyinelemeli formüllerle** yeni sayılar üretir.
* **Bit kaydırma ve XOR işlemleri** ile karıştırma yapılır.
* **Periyot uzunluğu:** 2^19937 - 1 (çok büyük bir periyot sağlar).

### **Kullanılan Formüller**

1. **Durum Dizisinin Başlatılması:** si=f ⋅ (si−1⊕(si−1≫(w−2)))+is\_i = f \,\cdot\, (s\_{i-1} \oplus (s\_{i-1} \gg (w-2))) + i
2. **Yeni Sayı Üretimi:**
   * xx değeri, önceki sayıların belirli bitlerinin kombinasyonudur.
   * xA=x≫1xA = x \gg 1 ve eğer xx tekse özel bir sabitle XOR yapılır.
   * Son olarak, çeşitli bit kaydırma ve maskeleme işlemleri uygulanır.

### **Kod Açıklaması**

* **initialize\_state(seed)** fonksiyonu, başlangıç tohumunu alarak durumu başlatır.
* **random\_uint32()** fonksiyonu, yukarıdaki özyinelemeli formüllerle yeni rastgele sayılar üretir.

## **3. Xorshift Algoritması**

**Xorshift**, hızlı ve hafif bir rastgele sayı üreteçidir. 2003 yılında George Marsaglia tarafından önerilmiştir.

### **Çalışma Prensibi**

* **Basit bit kaydırma ve XOR işlemleriyle** rastgelelik sağlar.
* **Çok hızlıdır ve az bellek tüketir**.

### **Kullanılan Formüller**

x=x⊕(x≪a)x = x \oplus (x \ll a) x=x⊕(x≫b)x = x \oplus (x \gg b) x=x⊕(x≪c)x = x \oplus (x \ll c)

Burada:

* a,b,ca, b, c sabit kaydırma değerleridir.
* **XOR ve bit kaydırma işlemleri** sayıyı karıştırarak rastgeleliği sağlar.

### **Kod Açıklaması**

* **Xorshift sınıfı**, bir tohum alır ve **next()** fonksiyonuyla yeni sayılar üretir.
* Bit kaydırma işlemleri ile XOR kullanılarak rastgelelik sağlanır.

## **4. PCG Algoritması**

**PCG (Permuted Congruential Generator)**, modern ve kaliteli bir PRNG algoritmasıdır.

### **Çalışma Prensibi**

* **Lineer Kongruanslı Üreteç (LCG)** ile durum güncellenir.
* **Bit karıştırma ve döndürme işlemleri** ile daha yüksek rastgelelik sağlanır.

### **Kullanılan Formüller**

1. **Durum Güncellemesi:** s=s×m+cmod  264s = s \times m + c \mod 2^{64}  
     
    Burada:  
   * mm çarpan (6364136223846793005)
   * cc sabit artış (1442695040888963407)
2. **Çıktının Karıştırılması:** y=((s≫18)⊕s)≫27y = ((s \gg 18) \oplus s) \gg 27  
   * Ardından döndürme işlemi uygulanarak nihai sayı elde edilir.

### **Kod Açıklaması**

* **PCG sınıfı**, bir başlangıç değeri alarak rastgele sayılar üretir.
* **next() fonksiyonu**, yukarıdaki formülleri kullanarak rastgele sayılar oluşturur.

## **5. Blum Blum Shub Algoritması**

**Blum Blum Shub (BBS)**, kriptografik olarak güvenli bir PRNG’dir.

### **Çalışma Prensibi**

* **Modüler aritmetik kullanarak** rastgele sayı üretir.
* **Kriptografik güvenliği** oldukça yüksektir, ancak diğer PRNG’lere göre daha yavaştır.

### **Kullanılan Formüller**

1. **Modüler Çarpım:** sn+1=sn2mod  ns\_{n+1} = s\_n^2 \mod n
2. **Çıktının Alınması:**
   * Sayının belirli bitleri seçilerek rastgele sayı elde edilir.

Burada:

* **n = p \* q** (p ve q büyük asal sayılar).
* **Durum değeri, modüler karesel işlem ile güncellenir**.

### **Kod Açıklaması**

* **BlumBlumShub sınıfı**, asal sayılarla çarpılmış bir modül üzerinden çalışır.
* **next() fonksiyonu**, yukarıdaki formülü kullanarak yeni rastgele sayı üretir.

## **6. Sonuç**

Bu çalışmada dört farklı PRNG algoritmasının uygulaması gerçekleştirilmiştir. Algoritmaların temel prensipleri ve matematiksel formülleri açıklanmıştır. Kodlar, her algoritma için 100 rastgele sayı üretip dosyalara kaydedecek şekilde düzenlenmiştir.

* **Mersenne Twister**, uzun periyotlu ve güçlü bir rastgelelik sunar.
* **Xorshift**, hafif ve hızlı bir algoritmadır.
* **PCG**, modern ve yüksek kaliteli bir PRNG’dir.
* **Blum Blum Shub**, kriptografik olarak güvenli ancak görece yavaş bir yöntemdir.

Her algoritma, farklı kullanım senaryolarına göre tercih edilebilir. Örneğin, şifreleme uygulamalarında **Blum Blum Shub**, hız gerektiren yerlerde **Xorshift**, genel amaçlı rastgelelik için **Mersenne Twister** veya **PCG** kullanılabilir.