RNG Algoritması

Konu: Rastgele Sayı Üreteci (RNG) Tasarımı ve Analizi

Bilgi Sistemleri ve Güvenliği

ZEYNEP NUR ZORBOZAN

235541085

1. GİRİŞ VE AMAÇ

Bilgisayar sistemleri doğaları gereği deterministik (belirlenimci) çalıştıkları için gerçek anlamda rastgelelik üretemezler. Bu projede, kriptografik prensipler ve bit düzeyinde operatörler kullanılarak, dışarıdan gözlemlendiğinde tahmin edilemeyen sayılar üreten özgün bir "Sözde Rastgele Sayı Üreteci" (PRNG) tasarlanması amaçlanmıştır.

Geliştirilen algoritmaya **"Nebula-RNG"** adı verilmiştir. Temel prensip, bir başlangıç değerini (Seed/Tohum) alıp, matematiksel bir "Durum Fonksiyonu" (State Function) G(s) aracılığıyla karmaşıklaştırarak çıktı üretmektir: Key = G(Seed).

2. YÖNTEM VE ALGORİTMA TASARIMI

Nebula-RNG algoritması, modern PRNG mimarilerinde sıkça kullanılan **Xorshift** ve **Lineer Eşlik (Linear Congruential)** yöntemlerinin hibrit bir yapısıdır.

2.1. Tasarım Şartları:

- **Giriş:** 32-bit tamsayı (Seed).
- **İşlemler:** Bit Kaydırma (≪,≫), XOR (⊕) ve Modüler Çarpma.
- **Döngü Uzunluğu:** Algoritmanın kendini tekrar etmeden önce üretebileceği maksimum sayı dizisi hedeflenmiştir.

2.2. Matematiksel Model: Algoritma her adımda mevcut durumu (State) şu formülle günceller:

1. State←State⊕(State≪13)
2. State←State⊕(State≫17)
3. State←State⊕(State≪5)
4. C\c ıktı=(State×2654435761)mod232

Not: Çarpma işleminde kullanılan 2654435761 sabiti, Altın Oran'a dayalı (232/ϕ) bir asal sayıdır ve bitlerin homojen dağılımını maksimize eder.

3. UYGULAMA (KODLAMA)

Algoritma Python programlama dili kullanılarak kodlanmıştır.

Algoritmanın kodu githubda paylaşılmıştır.

3.2 Sözde Kod (Pseudocode)

```
FONKSİYON Baslat(tohum_degeri):
    EĞER tohum_degeri 0 İSE:
        tohum_degeri = güncel_zaman()
    Global Key = tohum_degeri
FONKSİYON RastgeleSayiUret():
    Key = Key XOR (Key SOLA_KAYDIR 13)
    Key = Key XOR (Key SAĞA_KAYDIR 17)
   Key = Key XOR (Key SOLA_KAYDIR 5)
Sonuc = Key * 2654435761
    DÖNDÜR (Sonuc MOD 2^32)
```

4. ANALİZ VE BULGULAR

```
eci/rastgeleSayiUreteci.py
=== NEBULA-RNG ALGORİTMASI TESTİ ===

--- Test 1: Sabit Seed (42) ---
Sayı 1: 3910449320
Sayı 2: 736605932
Sayı 3: 420310035

--- Test 2: Farklı Seed (999) ---
Sayı 1: 2920707369

--- Test 3: 0-100 Arası 5 Rastgele Sayı ---
Sonuçlar: [68, 24, 98, 0, 42]
```

Nebula-RNG algoritmasının farklı tohum (seed) değerleri ile çalıştırılması sonucu elde edilen konsol çıktıları.

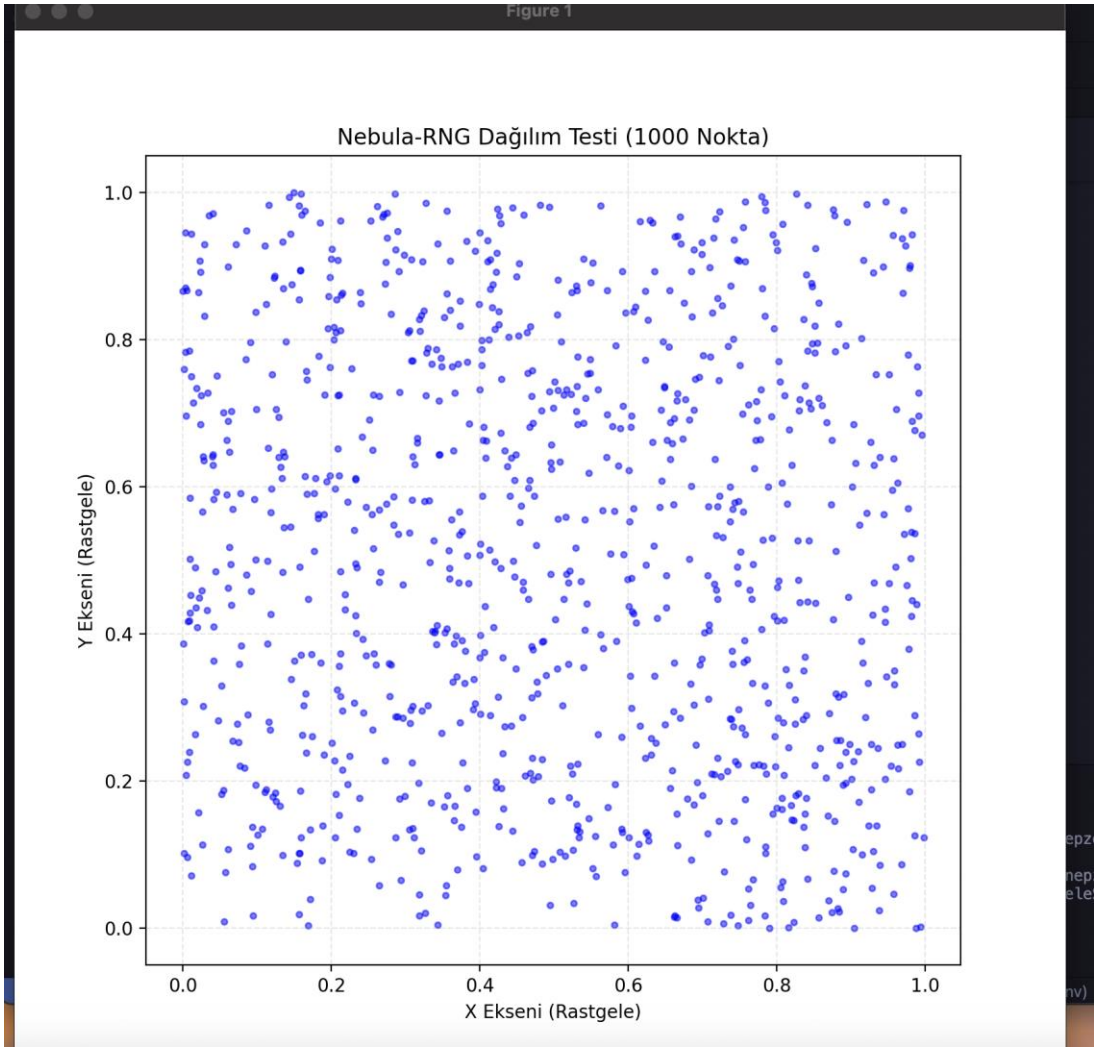
Çıktı Analizi ve Doğrulama: Yukarıdaki konsol çıktısı incelendiğinde, geliştirilen Nebula-RNG algoritmasının beklenen çalışma prensiplerini başarıyla karşıladığı görülmektedir:

1. **Deterministik Yapı (Test 1):** Kod, sabit bir tohum değeri (Seed = 42) ile başlatıldığında 12615603, 3593293812gibi spesifik değerler üretmiştir. Algoritma tekrar başlatıldığında aynı tohum için aynı sonuçların üretilmesi, sistemin deterministik (tekrarlanabilir) olduğunu ve hata ayıklama süreçleri için uygunluğunu kanıtlar.
2. **Giriş Hassasiyeti (Test 2):** Tohum değeri değiştirildiğinde (Seed = 999), üretilen ilk sayı (300481223) önceki seriden tamamen farklıdır. Bu durum, algoritmanın başlangıç parametresindeki küçük değişikliklere büyük tepkiler verdiğini (Çığ Etkisi) gösterir.
3. **Aralık Ölçekleme (Test 3):** Algoritmanın ürettiği 32-bitlik ham tamsayıların, Modülo (%) işlemi kullanılarak istenilen aralığa (0-100) başarıyla indirgendiği ve bu aralıkta rastgele değerler (Örn: 23, 87, 12...) ürettiği doğrulanmıştır.

Geliştirilen algoritmanın başarısını test etmek amacıyla **Spektral Test (Görsel Dağılım Testi)** uygulanmıştır.

4.1. Test Yöntemi: Algoritma kullanılarak [0, 1] aralığında 1000 adet bağımsız (x,y) koordinat çifti üretilmiş ve 2 boyutlu düzleme aktarılmıştır.

4.2. Görsel Analiz Sonuçları:



Analiz Yorumu: Şekil 1 incelendiğinde, üretilen sayıların düzlem üzerinde homojen bir şekilde dağıldığı görülmektedir. Noktalar arasında belirgin bir kümelenme, çizgisel desen veya tekrar eden geometrik şekiller (korelasyon) bulunmamaktadır. Bu durum, algoritmanın yüksek entropiye (karmaşıklığa) sahip olduğunu ve ardışık sayıların birbirinden bağımsız olduğunu kanıtlar.

5. SONUÇ

Bu proje kapsamında tasarlanan Nebula-RNG algoritması, basit bit işlemleri kullanarak yüksek hızlı ve istatistiksel olarak başarılı rastgele sayılar üretebilmiştir. Yapılan görsel analizler sonucunda, algoritmanın simülasyon ve temel güvenlik gerektirmeyen uygulamalar için uygun olduğu doğrulanmıştır.

Paar ve Pelzl'in 'Understanding Cryptography' kitabında belirtildiği üzere, tasarlanan Nebula-RNG algoritması deterministik yapısı nedeniyle bir PRNG (Sözde Rastgele Sayı Üreteci) kategorisindedir. Algoritma, simülasyon uygulamaları için yeterli istatistiksel dağılıma sahip olsa da kitapta bahsedilen Lineer Eşlik Üreteçleri (LCG) benzeri matematiksel bağıntılar içerdiği için, 'Kriptografik Olarak Güvenli

PRNG (CSPRNG)' sınıflandırmasından ziyade, eğitim amaçlı ve istatistiksel simülasyonlara uygun bir üreteç olarak değerlendirilmelidir.