Fırat Üniversitesi Müh. Bil. Dergisi
34(2), 627-634, 2022

Araştırma Makalesi
https://doi.org/10.35234/fumbd.1107291

Java Temelli Rasgele Sayı Dizisi Test Ortamı Uygulaması

Kenan İNCE[1]
[1]Yazılım Mühendisliği Bölümü, Mühendislik Fakültesi, İnönü Üniversitesi, Malatya, Turkey
[1]kenanince@gmail.com

(Geliş/Received: 21/04/2022; Kabul/Accepted: 07/09/2022)

Öz: Bu çalışmada kriptografinin temel taşlarından olan rasgele sayıların rasgelelik analizlerinin Java ortamında arayüz aracılığı ile kolay yapılabilmesi hedeflenmiştir. Rasgele sayı üreteçleri (RSÜ) siber güvenlik çalışmaların önemli bir alanıdır. Farklı kategorilerdeki RSÜ'ler kullanılarak üretilmiş olan sayı dizilerinin tahmin, taklit ve tekrar edilemez olması için güçlü istatistiki özellikler göstermesi gerekmektedir. Üretilen sayıların bu şartlara uyumluluğu farklı istatistiksel testlerden oluşan test ortamları kullanılarak yapılmaktadır. Rasgele sayı üreteçleriyle üretilen sayıların rasgelelik analizlerini yapan farklı programlama dilleri ile geliştirilmiş test ortamları mevcuttur. Fakat bu test ortamlarının çalıştırılabilmesi için gerekli kütüphanelerin yüklenmesi gerekmektedir. Son zamanlarda geliştirilen birkaç örnek dışında rassallık analizleri için hala terminal arayüzeri kullanılmaktadır. Bu çalışmada NIST SP 800-22 Rev.1a testleri ile analiz yapan, Java Swing tabanlı bir masaüstü uygulaması geliştirilmiştir. Uygulamada üretilmiş olan sayı dizileri test edilebileceği gibi aynı zamanda işletim sistemine göre Java SecureRandom kütüphanesinin sunmuş olduğu algoritmaları kullanarak rasgele sayı üretimi ve ardından üretilen bit dizilerin testleri de yapılabilmektedir.

Anahtar kelimeler: Java SecureRandom, NIST Test Suit, RSÜ (Rasgele SayıÜreteçleri), SHA (Secure Hash Algorithm)

Java Based Random Number Sequence Test Suite

Abstract: In this study, it is aimed that the randomness analysis of random numbers, which are the cornerstones of cryptography, can be done easily through the interface in the Java environment. Random number generators (RSU) are an important area of cyber security studies. Sequences of numbers produced using RNGs of different categories must show strong statistical properties in order to be unpredictable, inimitable and unrepeatable. The compatibility of the generated numbers with these conditions is carried out using test environments consisting of different statistical tests. There are test environments developed with different programming languages that analyze the randomness of numbers produced by random number generators. However, in order to run these test environments, the necessary libraries must be loaded. Terminal interfaces are still used for randomness analysis, except for a few recent examples. In this study, a Java Swing-based desktop application was developed that analyzes with NIST SP 800-22 Rev.1a tests. The number sequences produced in the application can be tested, as well as the random number generation and then the tests of the generated bit sequences by using the algorithms offered by the Java SecureRandom library according to the operating system.

Key words: Java SecureRandom, NIST Test Suit, RNG, SHA (Secure Hash Algorithm)

1. Giriş

Kriptolojinin amacı, bilginin mahremiyetini gerekli kişiler için gerekli süre boyunca sağlamaktır. Bilginin çoğalması, erişim yollarının ve hızının artması ile birlikte kriptoloji çalışmaları önem kazanmıştır. Özellikle bütün dünyada kişisel verilerin korunması kapsamında kanunlar çıkarılmıştır [1].

Veri yetkilendirmesi en temelde seçilen bir şifre ile ya da parmak izi, retina verisi gibi kişiye özel taklit edilemez kaynaklar kullanılarak sağlanır. Kullanıcının erişim tercihi her ne ise bu verinin de şifrelenerek saklanması ve buna göre yetkilendirme yapılması gerekir. Günümüzde bir verinin türüne bakılmaksızın aktarılırken, yetkilendirme yaparken veya kaydedilirken bütün aşamalarında bir şekilde şifrelemeden geçmesi gerekmektedir.

Şifreleme için geliştirilmiş birçok yöntem mevcuttur [2-4]. Bütün şifreleme algoritmalarının kesişim noktası Rasgele Sayı Üreteçleridir (RSÜ) denilebilir [5-7]. Akış şifreleme gibi şifreleme yöntemleri RSÜ üzerine inşa edilirken, bir kısmı ise güvenli anahtar üretimi aşamasında kullanırlar.

RSÜ iki ana kategoride incelenir, Gerçek Rasgele Sayı Üreteçleri (GRSÜ) ve Sözde Rasgele Sayı Üreteçleri (SRSÜ). GRSÜ genel olarak fiziksel olaylara dayanan üreteçler, SRSÜ ise yazılımsal olarak çeşitli karıştırma teknikleri kullanan deterministik üreteçlerdir. GRSÜ kriptolojik olarak daha güvenlidir ancak daha maliyetli ve zahmetli yöntemlerdir. SRSÜ ise daha ulaşılabilir fakat güvenlik zafiyetleri olan yöntemlerdir. Günümüzde yaygın kullanılan bütün programlama dillerinde rasgele sayı üretim kütüphaneleri mevcuttur. Fakat genel olarak bu üreteçler kriptografik olarak güvenli sayılmazlar.

Sorumlu yazar: kenanince@gmail.com. Yazarların ORCID Numarası: [1]0000-0003-4709-9557

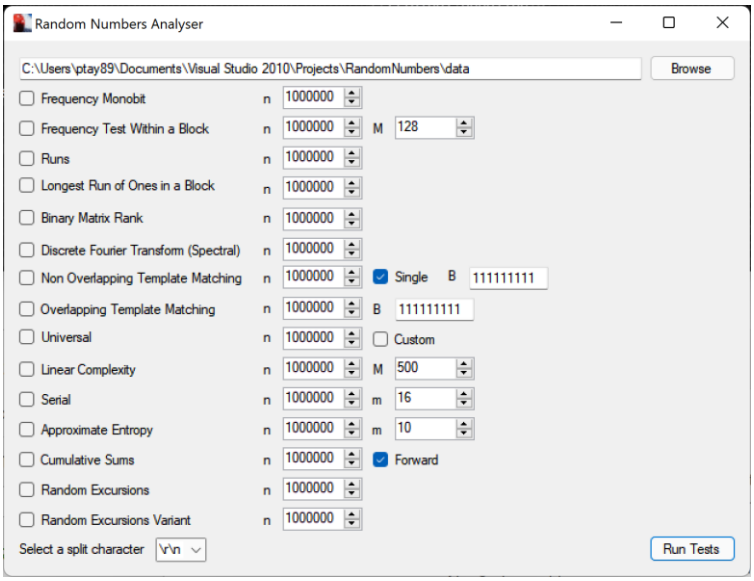
Bir RSÜ'nün kriptografik olarak güvenli sayılabilmesi için iyi istatistiksel özelliklere sahip olmalıdır. Yani tahmin, taklit ve tekrar edilemez olduğunun sunulabilmesi gerekir. Bu noktada rassallık test ortamları bu istatistiksel özellikleri test etmektedirler. Yaygın kullanılan rassallık test ortamları NIST SP 800-22 Rev.1a [8], Diehard [9], TESTU01 [10], ENT [11] ve CryptX [12] uygulamalarıdır. Bu test ortamlarının ortak yönü, RSÜ'ler tarafından üretilmiş olan sayı dizilerini istatistiksel testlerden geçirerek beklenen özellikleri sağlayıp sağlamadığını bir temel olarak ortaya koymaktır. Bu anlamda RSÜ'ler için temel doğru niteliğindedirler. Bu test ortamları arasında ortak testler olmakla birlikte farklılıkları da vardır.

Bütün RSÜ çalışmalarında, önerilen RSÜ'nin güvenilirliğini göstermek adına rassallık analiz test sonuçları sunulmalıdır. Araştırmacılar bir şekilde geliştirilmiş olan bu araçları kullanarak önerdikleri yöntemlerin güvenilirliğini göstermektedir.

Literatürde yapılan araştırma sonucu, görsel arayüz kullanan, kullanıcı dostu uygulamaar araştırılmıştır. Fakat ulaşılan uygulamaların ihtiyacı yeterli ölçüde sağlamadığı düşünülmüştür. Bu sebeple bu uygulamanın geliştirilmesine karar verilmiştir.

NIST STS'nin dökümantasyonunda K.1 bölümünde grafiksel kullanıcı arayüzünden bahsedilmektedir. Bu uygulama çalıştırılarak sonuç alınamamış fakat ilgili dökümanda bahsedilen şekli ile sonuçların text dosyası olarak kaydedildiği saptanmıştır. Girdi olarak ikili diziyi yine sadece text olarak almaktadır. Bu durum RSÜ çalışmaları için çalışma dışı zaman kaybı anlamına gelmektedir.

Araştırmamızda karşılaşılan bir diğer uygulama Şekil 1'de gösterilen, ulaşılan sayfada belirtildiği üzere C# dili ile implemente edilmiştir. Ancak bu uygulamanın geliştirici sayfası veya dokümantasyonuna ulaşılamamıştır. Bu sebeple istenilen amaçla kullanılmaya uygun bulunmamıştır. Öncelikle rasgele sayı üretim işlevi bulunmamaktadır. Ayrıca bütün testlere uzunluk değerinin tek tek atanması şeklinde bir yol izlenmiştir. Oysa bir RSÜ test edilirken üretmiş olduğu dizinin bütün testlerden elde edilen p değerlerinin sunulması gerekir [13].



Şekil 1. Araştırma sonucu karşılaşılan bir RSÜ test ortamı

Bir diğer karşılaşılan test ortamı ise Owlet RNG Analyzer uygulamasıdır [14]. Fakat bu uygulama lisanslı bir uygulamadır. Owlet RNG Analyzer uygulamasının geliştirilen bu çalışma gibi detaylı bir uygulama olduğu düşünülmektedir. Sadece NIST STS ile değil, diğer test ortamlarını da kullanarak test yapılabilmektedir. Ancak lisanslı bir uygulama olması ve açık kaynak olmaması dezavantajlarıdır.

Yapılan araştırma sonucu bulunan uygulamaların dezavantajlarından ötürü bu çalışma gerçekleştirilmiştir. Çalışmanın diğer çalışmalardan ayrılan en büyük özellikleri:

1. Açık kaynak olarak geliştirilmiş ve çevrimiçi olarak kaynak kodları ile kullanıma sunulmuştur.
2. Java ile geliştirildiği için, diğer uygulamalar gibi işletim sistemi bağımlılığı yoktur.

Bu çalışmanın literatüre katkısı, ara yüz yardımı ile, herhangi bir teknik donanıma ihtiyaç duymadan geliştirilen RSÜ mimarisini hızlıca test edilebilmesini sağlayan, platformdan bağımsız bir Java uygulamasının geliştirilmiş olmasıdır.

2. Materyal ve Yöntem

Bu çalışmada Java programlama dili, JDK 11 (LTS) üzerinde geliştirilmiştir. Geliştirme ortamı olarak Netbeans 12.x ve arayüz kütüphanesi olarak Swing tercih edilmiştir.

2.1. NIST Testleri

Geliştirilen uygulama NIST istatistiksel testlerini içermektedir. NIST 15 adet testten oluşmaktadır. Bu testler üretilen akışların farklı istatistiksel özelliklerini test etmektedir. Bu sayede kriptografik olarak güvenilir olup olmadığı ortaya konulabilir. Tabiki gerçek bir rassal olaylar dizisi NIST testlerinden rassal değil olarak değerlendirilebilir. Bunun sebebi rasgelelik kavramının doğasından ötürüdür. Ancak SRSÜ ile üretilen akışlar arka planlarında deterministik bir sistem barındırdıklarından ötürü bu istatistiksel testlerden başarılı sonuç almadığı takdirde kriptografik olarak güvenli sayılamazlar.

NIST aşağıda kısaca açıklanan 15 istatistiksel testten oluşmaktadır.

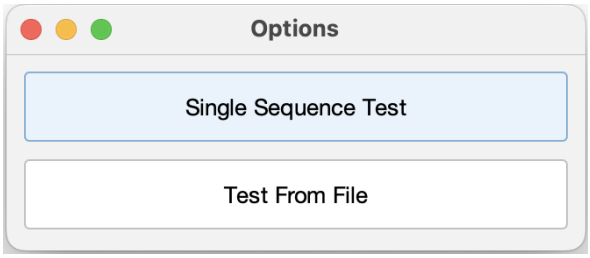
- **Frekans testi (Frequency-Monobit test):** Bu testin amacı test edilen akıştaki 1 ve 0 değerlerinin kıyasi oranlarını ortaya koymaktır. Hilesiz bir para atışı rassal olayında tekrar sayısı arttıkça yazı ve tura sayıları birbirine yaklaşır. Bu testin gerçekleştirdiği analiz bu para atma olayındaki gibi, akıştaki 1 ve 0 değerlerinin sayılarının birbirine belli bir aralık dahilinde yakın ya da eşit olması gerekmektedir.
- **Bir blokdaki frekans testi (Frequency test within a block):** Bu test frekans testi gibidir ancak frekans testinin M bit blok içerisinde gerçekleştirilmesi şeklindedir. M=1 alındığında frekans testi ile aynı sonucu verir. Bu sebeple M değerinin 2 ve üzeri seçilmesi gerekir. NIST dökümanına göre M değerinin 20 ve üzeri seçilmesi tavsiye edilmiştir.
- **Koşu testi (Runs test):** Bu testin amacı akıştaki ardışık aynı bitlerin uzunluklarını analiz etmektir. k uzunluğundaki bir gezintide k tane aynı bit olmalıdır. Bu test, bu tür sıfırlar ve birler arasındaki salınımın çok hızlı veya çok yavaş olup olmadığını belirler.
- **Bir bloktaki en uzun süreli koşu testi (Test for the longest run of ones in a block):** Bu testin amacı, test edilen dizideki en uzun koşunun uzunluğunun, rastgele bir dizide beklenen en uzun koşunun uzunluğuyla tutarlı olup olmadığını belirlemektir.
- **İkili matris derece testi (Binary matrix rank test):** Bu testin amacı, orijinal dizinin sabit uzunluktaki alt dizileri arasındaki doğrusal bağımlılığı kontrol etmektir.
- **Ayrık Fourier dönüşüm testi (Discrete Fourier transform test):** Bu testin odak noktası, dizinin Ayrık Fourier Dönüşümündeki tepe yükseklikleridir. Bu testin amacı, test edilen dizideki rastgelelik varsayımından bir sapmayı gösterecek olan periyodik özellikleri (yani birbirine yakın tekrarlayan modeller) tespit etmektir.
- **Örtüşmeyen şablon eşleştirme testi (Non-overlapping template matching test):** Bu testin odak noktası, önceden belirlenmiş hedef dizilerin oluşum sayısıdır. Belirli bir periyodik olmayan şablonun çok fazla tekrarını üreten oluşturucuları tespit etmektir. Test belirli bir m-bit şablonunu aramak için bir m-bit penceresi kullanılır. Desen bulunamazsa, pencere bir bit ilerletilir. Model bulunursa, pencere bulunan modelden sonraki bit'e sıfırlanır ve arama devam eder. M-bit bloklar GF^m alanındaki indirgenemez polinomlardır.
- **Örtüşen şablon eşleştirme testi (Overlapping template matching test):** Örtüşmeyen şablon eşleştirme testinde olduğu gibi m-bit pencerelerde m-bit blokları eşleştirir. Aralarındaki fark şablon eşleştiğinde aramaya devam etmeden önce pencerenin sadece 1 bit kaydırılmasıdır.
- **Evrensel istatistiksel test (Universtal statistical test):** Bu testin odak noktası, eşleşen desenler arasındaki bit sayısıdır. Testin amacı, dizinin bilgi kaybı olmadan önemli ölçüde sıkıştırılıp sıkıştırılamayacağını tespit etmektir. Önemli ölçüde sıkıştırılabilir bir dizinin rastgele olmadığı kabul edilir.
- **Doğrusal karmaşıklık testi (Linear complexity test):** Bu testin temelini LFSR çalışma yapısı oluşturmaktadır. Akışın rasgele kabul edilecek kadar karmaşık olup olmadığını belirler. Rasgele diziler uzun LFSR'ler ile ifade edilebilir. Bu sebeple çok kısa LFSR bulunması akışın rasgele olmadığı anlamına gelir.
- **Serilik testi (Serial test):** Bu testin odak noktası, tüm dizi boyunca olası tüm örtüşen m-bit modellerinin frekansıdır. Bu testte de block boyu m, 1 olarak alındığında frekans testi ile aynı karakteristikte olmaktadır.

- **Yaklaşık entropi testi (Approximate entropy test):** Testin amacı, iki ardışık uzunluktaki (m ve m+1) örtüşen blokların sıklığını rastgele bir dizi için beklenen sonuçla karşılaştırmaktır.
- **Kümülatif toplamlar testi (Cumulative sums test):** Bu testin odak noktası, akıştaki 1 sayısı 1 ile ve 0 sayısı -1 ile ifade edildiğinde ardışık toplamların maksimumdan (0) sapma miktarını analiz etmektir. Testin amacı, test edilen dizide meydana gelen kısmi dizilerin kümülatif toplamının, rastgele diziler için bu kümülatif toplamın beklenen davranışına göre çok büyük veya çok küçük olup olmadığını belirlemektir.
- **Rasgele geziler testi (Random excursions test):** Rastgele yürüyüş döngüsü, başlangıç noktasından başlayıp orijine geri dönen, rasgele alınan birim uzunluktaki bir dizi adımdan oluşur. Bu testin amacı, bir döngü içindeki belirli bir duruma yapılan ziyaretlerin sayısının, rastgele bir dizi için beklenenden farklı olup olmadığını belirlemektir.
- **Rasgele geziler varyant testi (Random excursions variant test):** Bu testin amacı, rastgele yürüyüşte çeşitli durumlara beklenen ziyaret sayısından sapmaları tespit etmektir.

2.2. Uygulama

2.2.1. Karşılama formu

Uygulamanın karşılama ekranında iki seçenek bulunmaktadır. Bunlar “Single Sequence Test” ve “Test From File” seçenekleridir. Birinci seçenek panoya alınan veya uygulamanın sunmuş olduğu oluşturucu algoritmalar yardımı ile üretilen tek bir ikili akışın testi için geliştirilen sayfaya yöntendirme yapmaktadır. İkinci seçenekte ise, ikili tabanda farklı ayıraçlar ile dosya formatında kaydedilmiş akışların test edilmesi için tasarlanmış arayüze yönlendirmektedir. Bu karşılama sayfasının ekran görüntüsü Şekil 2’de verilmiştir.



Şekil 2. Uygulama karşılama arayüzü

2.2.2. Test arayüzü

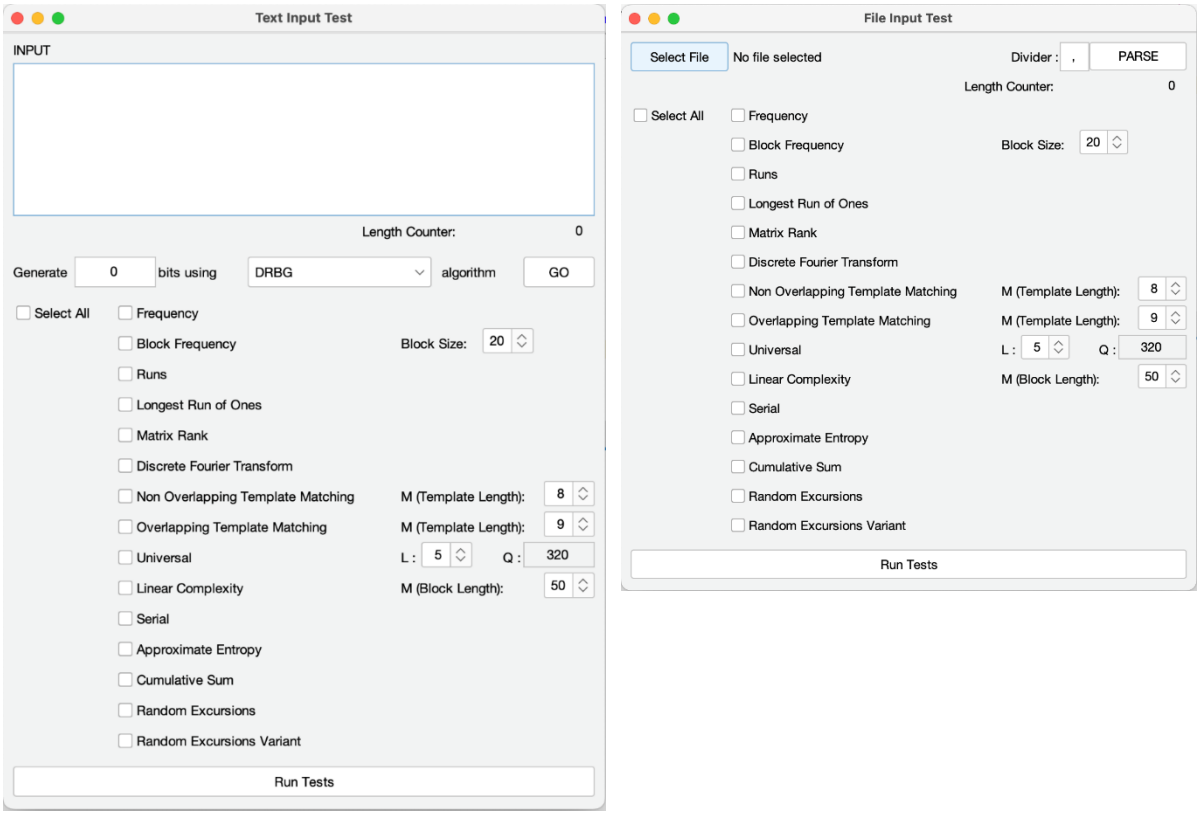
Karşılama ekranında “Single Sequence Test” butonuna tıklayarak Şekil 3.a, “Test From File” butonuna tıklanarak Şekil 3.b’de sunulmuş olan arayüzlere geçiş yapılır. Şekil 3.a’da verilmiş olaran arayüz iki farklı şekilde kullanılabilir. İlk olarak herhangi bir RSÜ tarafından üretilmiş olan akış panoya alınıp INPUT alanına yapıştırılarak test gerçekleştirilebilir. İkinci seçenek olarak ise, ileride detaylandırılacak olan algoritmalardan birini seçerek, belirtilen uzunlukta ikili bir akış oluşturarak test yapılabilir.

Akış oluşturma aşamasında temel olarak Java SecureRandom kütüphanesi kullanılmıştır. SecureRandom kütüphanesi programlama dillerinde standart olarak sunulan “random()” veya “rand()” fonksiyonları gibi kriptografik olarak zayıf bir PRNG olmadığı yapılan testlerde görülmüştür [17]. SecureRandom kütüphanesi içerisinde sunulan rasgele sayı üretme algoritmaları Tablo 1’de verilmiştir. Ayrıca ilgili seçim arayüzü Şekil 4’de sunulmuştur. Tablodan görülebileceği gibi bütün algoritmalar bütün işletim sistemlerin mevcut değildir. Ancak uygulama bütün işletim sistemlerinde sorunsuz çalışmakta, fakat işletim sisteminde var olan algoritmalar seçenek olarak gelmektedir. Ayrıca sunulan algoritmalar arasında Trivium yapısı [15] ile rasgele ikili akış üretme seçeneği eklenmiş durumdadır. Zaten açık kaynak olması sebebi ile farklı geliştiriciler tarafından bu algoritmaların sayısı arttırılabilir durumdadır. Trivium seçeneği ve eklenecek diğer algoritmalar implemete edileceğinden bütün işletim sistemlerinde çalışacak halde olacaktır.

Şekil 3 b’de verilmiş olarna arayüzde öncelikle akışın yer aldığı dosyanın seçilmesi gerekmektedir. Daha sonra dosyada bitler arası ayıraç ne ise bunun verilmesi beklenmekte, varsayılan ayıraç olarak “,” tercih edilmiştir. Bu işlemlerden sonra “PARSE” butonu ile dosya taranarak, ilgili ayıraç yardımı ile bit akışına dönüştürülmekte ve testler gerçekleştirilebilmektedir.

Şekil 3'te görülebileceği gibi kullanıcı NIST içerisinde yer alan bütün testleri uygulayabileceği gibi bazı testleri hariç tutarak akışı test edebilmektedir. Ayrıca NIST dökümantasyonuna göre kullanıcının değişiklik yapabileceği parametreleri değiştirerek testi kişiselleştirebilmektedir.

SecureRandom ile bit dizisi üretilerek veya panodan yapıştırılarak girdi penceresi doldurulduktan sonra hangi istatistiksel testlerin uygulanacağı seçilir. Aslında bütün NIST STS testlerinden geçmeyen bir RSÜ başarılı kabul edilmez. Ancak geliştirme esnasında özel olarak bazı testler tekrarlanarak üretecin karakteristiği analiz edilmek istenebilir. Bu sebeple bu testler seçimli halde uygulamaya aktarılmıştır.

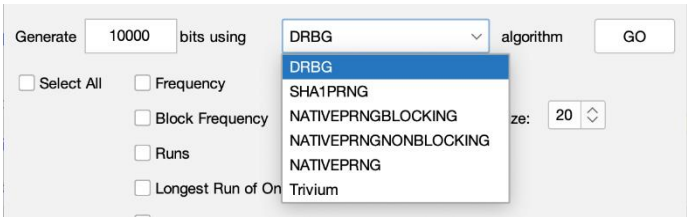


(a) (b)

Şekil 3. Test arayüzü a) Tek bir akış arayüzü b) Dosya taranarak yapılacak test arayüzü

Tablo 1. İşletim sistemlerine göre SecureRandom tohum algoritmaları

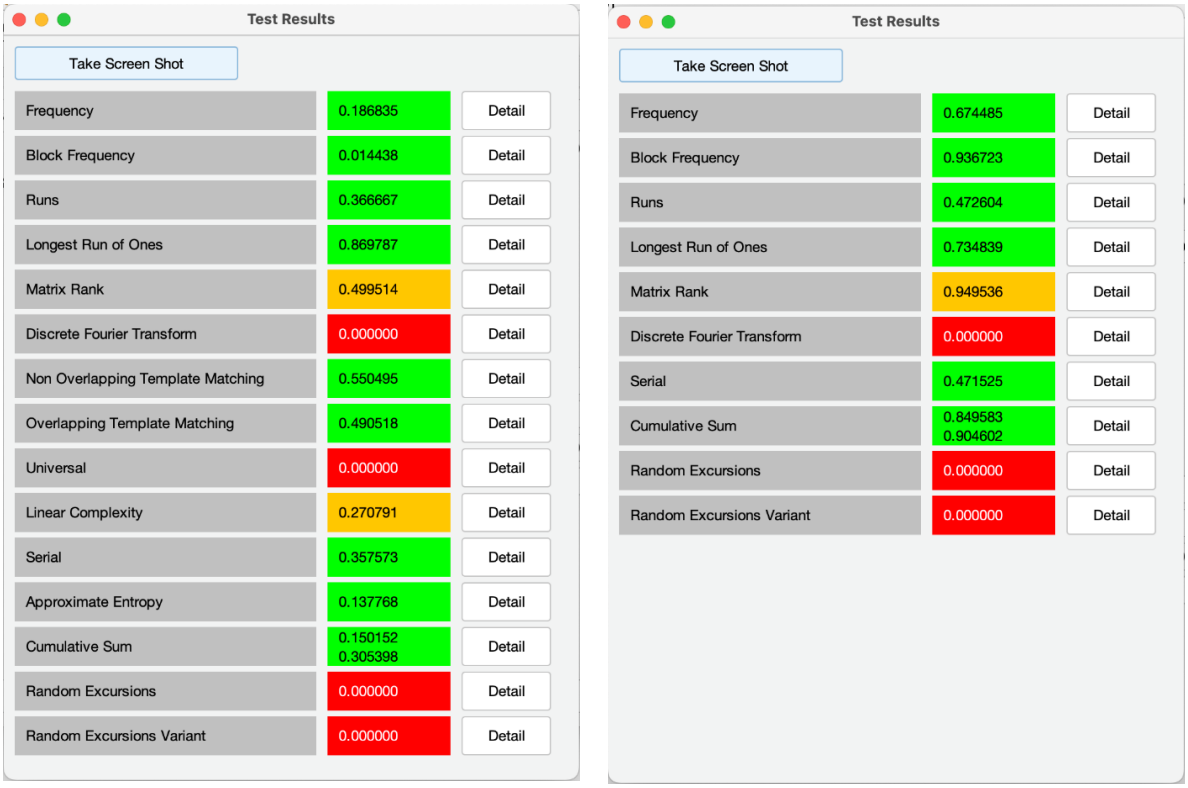
Algoritma	İşletim Sistemi
NativePRNG	Linux, Mac
NativePRNGBlocking	Linux, Mac
NativePRNGNonBlocking	Linux, Mac
PKCS11	-
SHA1PRNG	Linux, Mac, Windows
Windows-PRNG	Windows



Şekil 4. Windows işletim sisteminde tohum seçim algoritmaları

2.2.3. Sonuç Arayüzü

Bütün ayarlar yapıldıktan sonra test gerçekleştirilebilir. Testler gerçekleştirildiğinde Şekil 5'deki gibi bir sonuç ekranı üretilmektedir. Bu ekran seçilmiş olan testlerin sonucunu gösterecektir. Ayrıca bu ekranda çalışmalarda kullanılabilecek ekran görüntüsü alınabilmektedir.

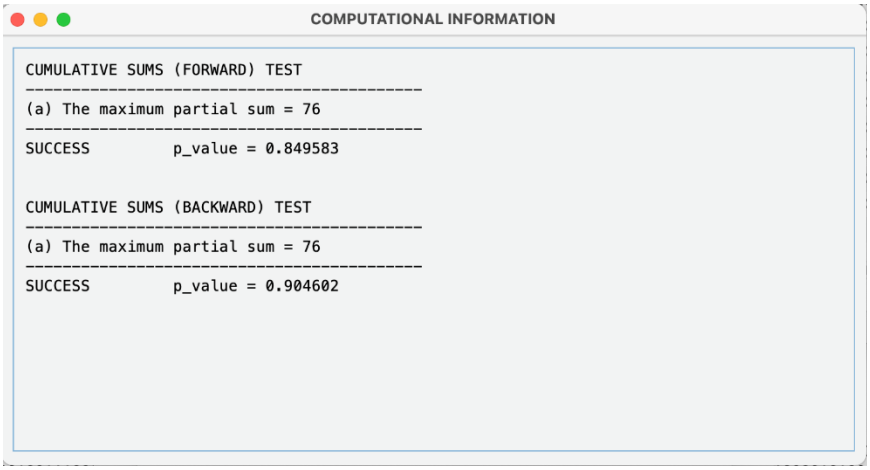


(a) (b)

Şekil 5. Test Sonuç Ekranı a) Bütün testler seçilmiş sonuç b) Bir kısım testler seçilmiş sonuç

Şekil 5'deki sonuç ekranında test edilmiş olan ikili dizinin seçilen istatistiksel testlerdeki "p_value" değerleri sunulur. Yeşil dizinin ilgili testten başarılı olduğunu, kırmızı ise başarısız olduğunu görsel olarak vurgulamaktadır. Turuncu ise dizinin başarılı "p_value" değerinin kabul edilen aralıkta olduğunu ancak testin bazı isterlerini (en düşük uzunluk gereksinimi gibi) sağlamadığını gösterir.

NIST STS test sonuçlarını uygulamada tasarlanan sonuç ekranından daha detaylı sunmaktadır. Özellikle "Overlapping Template Matching Test" gibi bazı testlerde incelenmesi ve analiz edilmesi gereken farklı hesaplama detayları bulunmaktadır. Bu hesaplama detaylarını Şekil 5'de sunulmuş olan sonuç penceresinde her testin yanında "Detail" butonu ile sağlanmıştır. Örnek bir test için detay sayfası Şekil 6'da verilmiştir.



Şekil 6. Rasgele seçilen bir test için detay sayfası

3. Tartışma

RSÜ'ler kriptolojik çalışmalarda büyük önem taşır. Bu sebeple ister GRSÜ olsun ister SRSÜ olsun, RSÜ'ler ile üretilen bit dizileri kullanılırken uygulama hassasiyeti göz önünde bulundurulmalıdır. Güçlü istatistiksel özellikler barındırmayan üreteçler kullanılmamalıdır. Bu bağlamda, RSÜ alanında çalışan araştırmacılar için mevcut test ortamlarının kullanımı zor olmayabilir. Ancak çalışma alanı bu olmayan, ya da sadece veri güvenliğini sağlamaya çalışan bir kullanıcı için bu testlerin kullanımı çok basit olmayacaktır.

Geliştirilmiş olan uygulama gerek akademik çalışmalarda gerekse bilgi güvenliği, kriptografi, siber güvenlik vb. gibi lisans derslerinde de kullanıma uygundur. Özellikle RSÜ konusu anlatılırken bazı kavramların görsel olarak anlatımında faydalı olacaktır.

4. Sonuçlar

Bu uygulama sonucunda, bütün işletim sistemlerinde çalışacak, kullanımı kolay bir NIST STS test ortamı geliştirilmiştir. Bu uygulama sayesinde RSÜ çalışmalarındaki ard arda gerçekleştirilen test aşamasının hızlı ve kolay olması hedeflenmiştir.

Uygulama açık kaynak olarak GitHub üzerinde paylaşılmıştır. Bu sayede geliştirilmeye ve açık bir şekilde kullanılmaya devam edeceği değerlendirilmektedir [16].

Teşekkür

Bu çalışma İnönü Üniversitesi Bilimsel Araştırma Projeleri Daire Başkanlığı'nın (İnönü BAP) FBG-2020-2143 numaralı projesi ile desteklenmiştir. Değerli destekleri için İnönü Üniversitesi İnönü BAP birimine teşekkürlerimi sunarım.

Kaynaklar

[1] https://www.mevzuat.gov.tr/mevzuat?MevzuatNo=6698&MevzuatTur=1&MevzuatTertip=5, Erişim Tarihi: 01.03.2022
[2] Iqra Basharat, Farooque Azam and Abdul Wahab Muzaffar. Article: Database Security and Encryption: A Survey Study. International Journal of Computer Applications 47(12):28-34, June 2012. doi: 10.5120/7242-0218
[3] Kumari, M., Gupta, S., and Sardana, P., "A Survey of Image Encryption Algorithms", 3D Research, vol. 8, no. 4, 2017. doi:10.1007/s13319-017-0148-5.
[4] John Justin M, Manimurugan S, A Survey on Various Encryption Techniques, International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-2 Issue-1, March 2012.
[5] F.J. Farsana, K. Gopakumar, A Novel Approach for Speech Encryption: Zaslavsky Map as Pseudo Random Number Generator, Procedia Computer Science, Volume 93, 2016, Pages 816-823, ISSN 1877-0509, doi: 10.1016/j.procs.2016.07.302.
[6] Hui Xu, Xiaojun Tong, Xianwen Meng, An efficient chaos pseudo-random number generator applied to video encryption, Optik, Volume 127, Issue 20, 2016, Pages 9305-9319, ISSN 0030-4026, doi: 10.1016/j.ijleo.2016.07.024.

[7] D. Liu, Z. Liu, L. Li and X. Zou, "A Low-Cost Low-Power Ring Oscillator-Based Truly Random Number Generator for Encryption on Smart Cards," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 63, no. 6, pp. 608-612, June 2016, doi: 10.1109/TCSII.2016.2530800.
[8] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo: A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications, Version STS-2.1, NIST Special Publication 800-22rev1a, April, 2010. http://csrc.nist.gov/publications/nistpubs/800-22rev1a/SP800-22rev1a.pdf.
[9] G. Marsaglia: The Marsaglia random number CDROM including the DIEHARD battery of tests of randomness. See http://stat.fsu.edu/pub/diehard, 1996
[10] P. L'Ecuyer, R. Simard: TestU01: A C library for empirical testing of random number generators, ACM Trans. Math. Softw., vol 33, 2007.
[11] J. Walker: ENT – A pseudorandom number sequence test program. 1993, http://www.fourmilab.ch/random/.
[12] W. Caelli et. al.: Crypt X Package Documentation, Information Security Research Centre and School of Mathematics, Queensland University of Technology, 1992. Crypt-X: http://www.isrc.qut.edu.au/resource/cryptx/
[13] https://code.google.com/archive/p/randomnumbertestsuite-nist/ erişim tarihi: 01.03.2022
[14] Owlet RNG Analyzer, https://www.bertendsp.com/products/owlet-rng-analyzer/#1509975186771-d75f9380-e39f, erişim zaman: 01.03.2022
[15] De Cannière, C. (2006). Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds) Information Security. ISC 2006. Lecture Notes in Computer Science, vol 4176. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11836810_13
[16] RNG Analyzer Github URL: https://github.com/kenan-ince/RTS
[17] K. İnce, "Security Analysis of Java SecureRandom Library", Avrupa Bilim ve Teknoloji Dergisi, no. 24, pp. 157-160, Apr. 2021, doi:10.31590/ejosat.900956