*Avrupa Bilim ve Teknoloji Dergisi*
*Özel Sayı 28, S. 493-499, Kasım 2021*

***Araştırma Makalesi***

www.ejosat.com ISSN:2148-2683

*European Journal of Science and Technology*
*Special Issue 28, pp. 493-499, November 2021*

***Research Article***

# Son İşlem Algoritmaları İçin Web Tabanlı Yazılım Suiti Geliştirilmesi

Didem Yosunlu[1], Erdinç Avaroğlu[2*]

[1] Mersin University, Faculty of Engineering, Departmant of Computer Engineering, Mersin, Turkey, (ORCID: 0000 – 0001 – 6917 – 4912), didemoz80@gmail.com
[2*] Mersin University, Faculty of Engineering, Departmant of Computer Engineering, Mersin, Turkey, (ORCID: 0000-0003-1976-2526), eavaroglu@mersin.edu.tr

(1st International Conference on Applied Engineering and Natural Sciences ICAENS 2021, November 1-3, 2021)

(**DOI:** 10.31590/ejosat.1008063)

**ATIF/REFERENCE:** Yosunlu, D. & Avaroğlu, E. (2021). Son İşlem Algoritmaları İçin Web Tabanlı Yazılım Suiti Geliştirilmesi. *Avrupa Bilim ve Teknoloji Dergisi*, (28), 493-499.

**Öz**

Son işlem algoritmaları üretilen rasgele sayıların istatistiksel özelliklerini geliştirmek amacıyla uygulanan ve birçok rasgele sayı üretecinin ihtiyaç duyduğu bir aşamadır. Rasgele sayılara ihtiyaç duyan kullanıcılar rasgele sayı üreteçlerine erişebilseler bile son işlem yöntemlerini uygulayabilmek için kullanabilecekleri hazır bir kaynak bulunmamaktadır. Uygulama geliştirme aşamalarını bilseler bile son işlem yönteminin yapısını incelemeleri ve sonrasında bu yöntemi koda dökerek geliştirmeleri hayli vakit alacaktır. Bu sıkıntıları gidermek ve ihtiyaç halinde kullanıcıların son işlem yöntemlerine hızlı erişimlerini sağlayabilmek adına seçili bazı algoritmaların bulunduğu, literatürde olmayan bir yazılım süiti geliştirilmiştir. Kolay bir arayüz ile kullanıcı dostu olarak geliştirilen web uygulaması C# ile yazılmış ve tüm kullanıcıların ulaşabileceği şekilde yayınlanmıştır. Bu nedenle gerçek rasgele sayı üzerine çalışanlar için önemli bir kaynak olacaktır. Geliştirilen web tabanlı bu yazılıma son kullanıcılar postprocess.mersin.edu.tr adresinden erişebilir. Son işlem yazılımının sonuçlarını görebilmek amacıyla örnek saf bit dizileri yazılımdan geçirilerek üretilen son işlem sonuçları NIST (National Institute of Standards and Technology) testlerine tabi tutulmuştur ve tasarlanan yazılımın başarılı bir şekilde çalıştığı doğrulanmıştır.

**Anahtar Kelimeler:** Rasgele sayılar, Rasgele sayı üreteçleri, Son işlem algoritmaları, Web tabanlı uygulama

# Web-based Software Suit Development For Post Procesing Algorithms

**Abstract**

Post-processing algorithms are a step that is applied to improve the statistical properties of the generated random numbers and is needed by many random number generators. Even if users who need random numbers have access to random number generators, there is no readily available resource they can use to apply post-processing methods. Even if they know the application development stages, it will take a lot of time to examine the structure of the post-processing method and then code this method and develop it. A software suite, which does not exist in the literature, has been developed with some selected algorithms in order to eliminate these problems and to provide users with fast access to finishing methods when needed. The web application developed in a user-friendly way with an easy interface was written in C # and published in a way that all users can access. For this reason, it will be an important resource for those studying on true random numbers. The web-based software can be accessed by end users at postprocess.mersin.edu.tr. In order to see the results of the post-processing software, the results produced by passing the sample raw bit strings through the software were subjected to NIST (National Institute of Standards and Technology) tests and it was confirmed that the designed software works successfully.

**Keywords:** Random numbers, Random number generators, Post processing algorithms, Web-based application.

[*] Corresponding Author: eavaroglu@mersin.edu.tr

# 1. Giriş

Rasgelelik temelde tahmin edilemeyen anlamına gelmektedir. Rasgele sayılar ise belirli aralıkta tanımlanan, üretilme olasılığı eşit olan ve birbirinden bağımsız olan sayılardır. Bilgisayar bilimlerinde sıkça ihtiyaç duyulan bu sayılar özellikle kriptografi alanı için oldukça önemli bir yere sahiptir. Kriptolojik uygulamalarda anahtar üretimi ve kimlik doğrulama gibi güvenlik tarafı için kritik öneme sahip rasgele sayılar sistemin güvenliğini doğrudan etkilediği için tahmin edilememeli ve tekrar üretilememelidir. Bu şekilde iyi istatistiksel özelliklere sahip rasgele sayılar üretilmesi amacıyla rasgele sayı üreteçleri geliştirilmeye başlanmıştır.

Rasgele sayı üreteçleri (RSÜ) üç ana sınıda ayrılmaktadır. Bunlar gerçek rasgele sayı üreteçleri(GRSÜ), sözde rasgele sayı üreteçleri (SRSÜ) ve hibrit rasgele sayı üreteçleridir.(HRSÜ)

Sözde rasgele sayı üreteçleri bir tohum değerini deterministik algoritmalara tabi tutarak rasgele bit dizileri üretmektedirler. Fakat periyodik olabilmekle beraber periyodun açığa çıkması ile üretilen sayıların tamamen tahmin edilebilir olma durumu oluşmaktadır. Ayrıca tohum değerinin korunamaması da aynı duruma sebep olabilmektedir. Ancak hızlı ve ucuz olmalarından dolayı diğer birçok uygulamada tercih edilmektedir.

Gerçek rasgele sayı üreteçleri ise entropi kaynağı olarak faz gürültüsü, termal gürültü, titreşim, rastgele telgraf gürültüsü, fotoelektrik etki gibi deterministik olmayan fiziksel olaylar kullanılarak gerçek rasgele diziler oluştururlar. Bu kaynaklardan elde edilen sinyaller sayısallaştırılıp sonrasında örnekleme yapılarak rasgele sayılar elde edilir.

Hibrit rasgele sayı üreteçleri de yukarıda bahsedilen SRSÜ ve GRSÜ'nin beraber kullanıldığı sistemlerdir. Genel olarak GRSÜ'den elde edilen rasgele sayı SRSÜ'de tohum değeri olarak kullanılmaktadır (Avaroğlu ve Türk, 2013).

Üretilen rasgele sayıların rasgeleliği ve üretecin kalitesi hakkında bilgi veren bazı testler bulunmaktadır. RSÜ'ler tarafından üretilen dizilerin rasgelelik kalitesinin değerlendirilmesinde İstatistiksel test paketleri çok önemli rol oynar. Üretilen rasgele sayılar her zaman iyi istatistiksel özellikler göstermemektedir. GRSÜ'de bu zayıflıkların giderilmesi amacıyla elde edilen bit dizileri son işleme tabi tutularak güçlendirilmesi amaçlanmaktadır. Son işlem algoritmaları çıkış bit oranında azalmaya sebep olabilmesine rağmen üretecin güvenliğini arttırdığından dolayı birçok sistemde tercih edilmektedir. Son işlemin rasgele sayılar için bu denli önemli olmasından dolayı çok eskilerden beri bu konuda araştırmalar yapılmakta ve yeni yöntemler duyurulmaktadır. Fakat bu yöntemler çoğunlukla ya yazılı olarak açıklanmış ya da sözde kod olarak sunulmuştur. Bu sebeple kullanmak isteyenlerin kendilerinin bu yöntemleri uygulanabilir hale getirmesi gerekmektedir.

Bu makalede, sık kullanılan son işlem algoritmalarından bazıları seçilerek ASP.NET C# programlama dili ile geliştirilen, herkesin her ortamdan ulaşabileceği ve kolaylıkla kullanabileceği bir web uygulaması sunulmaktadır. Uygulamanın kolay erişilebilir, kolay uygulanabiliyor olması ve literatürde benzer bir örneğinin olmaması sebebiyle önemli bir kaynak olacaktır.

(Avaroğlu ve ark., 2015)'de halka osilatörün donanımsal uygulaması sonucu elde edilen bit dizisi ile (Avaroğlu ve ark., 2014)'te kaotik çekerler kullanılarak elde edilen bit dizileri geliştirilen yazılım üzerinde kullanılarak yazılımın başarılı bir şekilde çalıştığı gözterilmiştir. Elde edilen sonuçlar NIST testlerine tabi tutulmuştur.

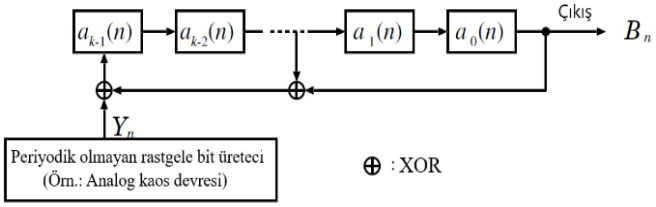
Bölüm 2'de literatür taraması yapılarak son yıllarda önerilmiş olan son işlem yöntemleri açıklanmaktadır. Bölüm 3'te uygulamaya dahil edilen son işlem algoritmalarından detaylı olarak bahsedilirken Bölüm 4'te geliştirilen yazılımın detayları yer almaktadır. Bölüm 5'te ise sonuç bölümü verilmektedir.

# 2. Literatür Taraması

TRNG sistemlerinde oluşturulan rastgele sayıların istatistiksel özelliklerini iyileştirmek için günümüze kadar çeşitli son işlem algoritmaları önerildi. Bunlardan bazıları resilient fonksiyonu, Doğrusal Beslemeli Kayan Yazmaç, Lojistik Harita, SHA-256, Trevisan's extractor, Toeplitz hashing, Keccak algoritması, Trivium algoritması.

Sunar ve arkadaşları tarafından yapılan bir çalışmada halka osilatörlerinin kullanıldığı bir GRSÜ'de son işlem algoritması olarak resilient fonksiyon kullanılmıştır. Resilient fonksiyon için bilinen bir genişletilmiş BCH kodu olan [256, 16, 113] kodu kullanılmaktadır. 114 halka osilatörden alınan çıkış bit dizileri 256 bitlik bloklara ayrılarak resilient fonksiyona verilir ve 16 bit rasgele sayı dizisini çıkış olarak verir. Saldırılara karşı güçlü olmasına karşın veri kaybı 1/16 oranındadır (Sunar ve ark., 2007).

Bir diğer son işlem yöntemi Doğrusal Beslemeli Kayan Yazmaç (Linear Feedback Shift Register). Uygulamasının kolay ve kullanışlı olması açısından, iyi istatistiksel özelliklere sahip diziler üretmesi ve geniş tekrarlama periyoduna sahip olmasından dolayı birçok uygulamada son işlem yöntemi olarak sıkça tercih edilmektedir. (Tsuneda ve ark., 2008)'de k hafıza hücresine sahip doğrusal beslemeli kayan yazmaç ve periyodik olmayan bir rasgele sayı üreteci kullanılarak geliştirilen bir rasgele sayı üreteci önerilmiştir. Şekil 1'deki gibi bir yapı kullanılarak rasgele sayıların periyodikliğinin giderilmesi amaçlanmıştır. (Tsuneda ve Morikawa, 2013) ve (Kyaw ve Tsuneda, 2017)'de de tasarım üzerine çalışılarak üreteç geliştirilmiştir. Yapılan çalışmalarda önerilen son işlem algoritmasının k büyük olduğunda çok etkili olduğu gözlemlenmiştir.




Şekil 1. DBKY kullanılan RSÜ'nün genel yapısı

Lojistik harita ise donanım üzerinde uygulaması kolay olan, bir başlatma koşulu ve bir kontrol parametresi gerektiren birinci dereceden bir denklemdir (Avaroğlu ve ark., 2015). Makalede Sunar ve arkadaşı tarafından geliştirilmiş olan TRNG kullanılmıştır (Sunar ve ark., 2007). Fakat makalede kullanılan resilient fonksiyon 1/16 oranında azalmaya sebep olduğundan dolayı yeni bir son işlem algoritması önerilmiştir. Lojistik haritanın son işlem olarak kullanılması, saldırılara karşı daha güvenli bir sistem sağlar. Literatürdeki birçok işlem sonrası yöntem veri hızını düşürürken, önerilen son işlem veri hızını azaltmamaktadır. Bu sistemin bir diğer faydası da rasgelelik kaynağından kaynaklanan korelasyonun lojistik harita kullanımı

ile ortadan kaldırılabilmesidir. Fakat sistemin yüksek güç tüketimine sahip olması bir dezavantajdır.

Önerilen bir diğer son işlem yöntemi ise SHA-256'dır (Loza ve Matuszewski, 2014). Bu makalede GRSÜ'nün entropi kaynağı olarak halka osilatörler kullanılmıştır. Üretilen bitler 8 bitlik bloklara ayrılmıştır ve her blok 64 byte büyüklüğünde FIFO içerisinde saklanmıştır. Önerilen son işlem sonucunda 8 tane 32 bitlik kelime (256 bit toplam) üretilir.

Güvenlik konusunda oldukça başarılı olan extractor üzerine ise birçok çalışma bulunmaktadır. Kısaca tanımlamak gerekirse extractor, katalizör olarak az sayıda ek rastgele bit kullanarak zayıf bir rastgele kaynaktan gerçek rastgele bitleri(neredeyse) çıkaran bir fonksiyondur. Trevisan, sözde rasgele sayı üreteçlerine dayalı rasgelelik çıkarıcıları oluşturmak için bir yaklaşım önerdi (Trevisan, 2001). Daha sonra Raz ve arkadaşları (Raz ve ark., 2002) tarafından bu sistem daha da geliştirilmiştir.

Bir başka extractor ise Wegman'ın önerdiği Toeplitz hashing matris (Wegman ve Carter, 1981). Toeplitz Hash Algoritması, anahtarın matris çarpımı yoluyla uygun bir Toeplitz matrisi ile hash değerlerini hesaplayan hash fonksiyonlarını açıklar. Fakat Toeplitz hash rasgelelik çıkarıcısı, bilgisayar seri hesaplamalarında uygulandığında çok fazla zaman harcayacağından büyük miktarda hesaplama gerektirir. Cihaz ve veri toplama tekniklerinin gelişmesiyle birlikte ham rastgele sayı üretme hızı yükkseltilirken mevcut rasgelelik çıkarma hızı talebin büyümesini karşılayamaz. Bu nedenle (Zhang ve ark., 2016)'da extractor FBGA üzerinde uygulanarak hızını arttırmak amaçlanmıştır.

Bir diğer çalışmada ise, özetleme algoritmaları sınıfından olan keccak algoritması kullanılmıştır. Rasgele üretimi için keccak algoritması kullanılırken güvenliği arttırmak için ek girdi olarak halka osilatörler kullanılmıştır (Yakut ve ark., 2019). Önerilen sistemde, 1024 bitlik rasgele sayı dizisi elde edebilemek amacıyla ek girdi olarak 512 bit ham gerçek rasgele sayı alınmıştır.

Yakut ve arkadaşları tarafından yapılan bir diğer makalede ise 1600 bit sayı dizisi üretebilmek amacıyla yine keccak algoritması kullanılmıştır.

(Kaya, 2020) Trivium algoritmasını son işlem olark kullanmıştır. Çok hızlı bir şekilde çalışan Trivium algoritması üretilen sayı dizisi üzerinde çok önemli istatistiksel değişikliklere yol açar. Esnek bir uygulama alanına sahip olması sebebiyle sıklıkla tercih edilen bir algoritma olmuştur. Trivium algoritması yapılan çalışmalarla saldırılara karşı dayanıklı olduğunu gösterirken veri kaybının olmaması da önemli bir avantajdır.

# 3. Son İşlem Algoritmaları

Geliştirilen rasgele sayı üreteçleri ile elde edilen ham bit dizilerinin bazıları zayıf istatistiki özellikler gösterebilmektedirler. Bu zayıflığın giderilmesi amacı ile üretilen bu bit dizilerine son işlem algoritmaları uygulanmaktadır. Uygulanan son işlem algoritmasına bağlı olarak bu sistemler, çevresel değişikliklere ve saldırganlara karşı dirençli hale gelirken üretecin güvenliği de buna bağlı olarak artmaktadır. Uygulanan bazı son işlem algoritmaları çıkış bit oranında azalmaya sebep olabilmesine rağmen üretecin entropisini arttırdığından dolayı birçok sistemde kullanılmaktadır.

Bölüm 2'de de bahsedildiği gibi son işlem algoritmaları üzerine birçok çalışma yapılmıştır ve yapılmaya devam edilmektedir. Fakat önerilen yöntemlerin bazıları teoride kalırken bazılarının ise uygulamaya dökülmesi kodlama bilgisi olmayan ya da uygulama geliştirmek için vakti bulunmayan araştırmacılar için oldukça zordur. Bu sebeple bu çalışmada en sık kullanılan son işlem yöntemleri arasından sekiz tanesini seçerek hepsinin bir arada bulunduğu bir yazılım süiti geliştirildi. Seçilen algoritmalar aşağıda başlıklar halinde verilmiştir.

## 3.1. XOR Algoritması

Uygulama kolaylığı bakımından en sık kullanılan son işlem yöntemi olan XOR algoritması, rasgelelik kaynağından elde edilen bit dizisinin n bitlik (n=2) bloklara ayrılarak her bloğun kendi içerisinde XOR işlemine tabi tutularak 1 bitlik çıkış vermesi işlemidir (Davies, 2002). Çıkış bit verimini 1/n oranında azaltan bu yöntem üreteç tarafından üretilen bitlerden kaynaklı biası azaltmak için yaygın olarak kullanılmaktadır.

Tablo 1'de XOR algoritmasının son işlem çıkış durumları verilmiştir.

*Tablo 1. XOR Son işlem*

| Bit Dizisi | XOR Çıkışı |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 1 |
| 11 | 0 |

## 3.2. Von Neumann Algoritması

Bit dizilerindeki korelasyonu gidermek için Von Neumann tarafından önerilmiş ilk ve en eski son işlem algoritmasıdır. (Suresh ve Burleson, 2010). Bu son işlem algoritmasında da tablo 2'de gösterildiği gibi üretilen bit dizileri 2 bit olarak alınır ve bit dizisi (1,0) ise çıkış biti 1, eğer bit dizisi (0,1) ise çıkış biti 0 olarak çıkışa verilir. (0,0) ve (1,1) durumlarında ise çıkış verilmez.

*Tablo 2. Von Neumann Son işlem*

| Bit Dizisi | Von Neumann Çıkışı |
|---|---|
| 00 | Çıkış yok |
| 01 | 0 |
| 10 | 1 |
| 11 | Çıkış yok |

## 3.3. H Fonksiyonu

Bir diğer son işlem yöntemi ise Dichtl tarafından önerilen, Quasigroup dizi dönüşümüne dayalı olan H son işlem algoritmasıdır (Dichtl, 2007). 16 bit giriş bitinden 8 bit çıkış elde edilir. Son işlem için giriş bitleri $a_0, a_1, a_2, \dots a_{15}$ şeklinde tanımlanmaktadır. $b_0, b_1, \dots b_7$ ise $b_i = a_i \oplus a_{(i+1)} mod 8$ ile tanımlanmaktadır. Çıkış bit dizisi olan $c_0, c_1, \dots c_7$ ise $c_i = b_i \oplus a_{(i+8)}$ ile tanımlanır. 16 bit giriş dizisi a1 ve a2 olarak ayrılır ve Eş. 1 uygulanarak çıkış elde edilir.

$$H(a1, a2) = XOR\ (XOR\ (a1, rotateleft\ (a1, 1)\ ),\ a2\ ) \quad (1)$$

Daha sonra H fonksiyonu geliştirilerek Eş. 2' te gösterilen H2 ve Eş. 3'te gösterilen H4 fonksiyonları tanımlanmıştır.

H2(a1, a2) = XOR(XOR(XOR(a1, rotateleft(a1,1)), rotateleft (a1,2)), a2)

H4(a1, a2) = XOR (XOR (XOR (XOR (a1, rotateleft (a1, 1)), rotateleft (a1, 2)), rotateleft (a1, 4)), a2)

Uygulamaya H, H2 ve H4 fonksiyonları dahil edilmiştir.

### 3.4. SBOX

SBOX, DES şifreleme algoritmasında kullanılmaktadır. (Avaroğlu ve Tuncer, 2020) tarafından yapılan çalışmada SBOX son işlem algoritması olarak önerilmiştir. SBOX son işlem algoritmasında tablo 3 'te verilen değerler kullanılmaktadır. DES 'teki SBOX kendisine gelen 6 biti alır ve 4 bit çıkış verir. Bu işlemin nasıl yapıldığı aşağıda bir örnek üzerinden açıklanmıştır.

*Örnek Bit Dizisi:*

*01101010011010111010101110101011100000101100001*

*1*

1. Adım: İlk 6 bitlik bloğu alıyoruz ve 1. bit ile 6. biti alarak satırımızı buluyoruz. (00->0)
2. Adım: Arada kalan bitler ile sütun belirliyoruz. (1101->13)
3. Adım: S1 kutusunda 1. ve 2. adımda denk gelen değeri buluyoruz. Tabloda karşılık gelen değerin ikili gösterimi bizim çıkış dizimizi oluşturmaktadır. (9->1001)
4. Adım: Tüm blokları bu şekilde işleyene kadar bu işlemleri tekrar ediyoruz ve 48 bitlik giriş dizisi için 32 bit çıkış dizisi üretiyoruz.

*Çıkış Bit Dizisi:*

*10011000101110010110001100101111*

*Tablo 3. S1 Kutusu*

| S1 | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 01 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 10 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 11 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

### 3.5. Mixing Bits in Steps and XORing of Adjacent Bits

Önerilen bir diğer son işlem yöntemi de 2016'da Nikolic ve Veinovic tarafından üretilen bit dizisinin entropisini arttırmak için ve dizideki düzeni sağlamak amacıyla Mixing Bits in Steps and XORing of Adjacent Bits algoritmasını önermişlerdir. (Nikolic ve Veinovic, 2016). Kurulan sistemin 3 aşaması bulunmaktadır.

İlk olarak entropi kaynağından gelen analog sinyaller örneklenmektedir. Daha sonra elde edilen düzensiz bit dizilerinin düzeltilmesi amacıyla son işlem algoritmasından geçirilmiştir. Algoritmanın adımları aşağıda verilmiştir:

1.adım: Karıştırma algoritması girişe gelen ilk 2 biti alır.

2.adım: İlk 2 bitin arasına 3. bit yerleşir.

3.adım: 3 ve 2. Bit arasına 5.bit konulur. Bu durumda görünüm x1; x4; x3; x5; x2 olacaktır.

4.adım: 1 ve 4. Bit arasına 6.bit;

2 ve 5 arasına 7. Bit;

4 ve 3 arasına 8.bit;

5 ve 3 arasına 9.bit yerleştirilir.

5.adım: x1; x6; x4; x8; x3; x9; x5; x7; x2 olarak son düzen ortaya çıkar.

İşlemler tamamlandıktan sonra komşu bitlere XOR işlemi uygulanarak son bit dizisi elde edilir. İstatistiki açıdan iyi sonuçlar verdiği gösterilmiştir.

$$x1$$
$$x1x2$$
$$x1x3x2$$
$$x1x4x3x5x2$$
$$x1x6x4x8x3x9x5x7x2$$
$$x1x10x6x12x4x14x8x16x3x17x9x15x5x13x7x11x2$$
$$\vdots$$
$$\vdots$$
$$\vdots$$
$$x_{n-1}1, x_n\mathbf{1}, x_{n-1}2, x_n\mathbf{3}, \ldots, x_{n-1}\mathbf{r\text{-}1}, x_n\mathbf{m\text{-}1}, x_{n-1}r, x_n\mathbf{m}, x_{n-1}r{+}1, \ldots, x_n\mathbf{4}, x_{n-1}\mathbf{s\text{-}1}, x_n\mathbf{2}, x_{n-1}s$$
$$\underbrace{x_{n-1}1 \oplus x_n1, x_{n-1}2 \oplus x_n3, x_{n-1}3 \oplus x_n5, \ldots, x_n4 \oplus x_{n-1}s\text{-}1, x_n2 \oplus x_{n-1}s}$$
$$p_1 \quad , \quad p_2 \quad , \quad p_3 \quad , \ldots , \quad p_{n-1} \quad , \quad p_n$$

Şekil 2. Mixing Bits in Steps and XORing of Adjacent Bits yöntemi gösterimi

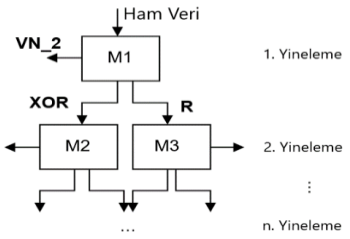
### 3.6. Iterating Von Neumann

1992'de Iterating Von Neumann (IVN) yöntemi ortaya konulmuştur (Peres, 1992). Bu yöntemde IVN, atılan bitleri toplayıp onları bir sonraki aşamaya yollayarak tekrardan kullanmaktadır. IVN modülünün 3 bloğu aşağıda gösterilmiştir.

1. VN_2 (giriş çifti "01" veya "10" ise, ardından "0" veya "1" çıkışı verir, diğerleri çıkış olmaz)

2. XOR işlemine tabi tutulur.
3. R ('00' ise '0', '11' ise '1' çıktı, diğerleri çıktı olmaz)

XOR ve R işlemleri tamamlandıktan sonra elde edilen veri tekrar işlenmemiş veri olarak bir sonraki yinelemeye verilir.




Şekil 3. Iterating Von Neumann

## 4. Geliştirilen Yazılım

RSÜ ve son işlem algoritmaları son yıllarda birçok araştırmacı tarafından kullanılan bir alandır. Son işlem algoritmaları konusunda birçok makalede yeni öneriler sunulmuştur. Ancak bu son işlem algoritmalarının yazılım ortamında bir arada sunulması sağlanmamıştır ve yazılımı geliştirme konusuna uzak olan kişiler için oldukça zor olacaktır.

Yukarıda açıklanan sebepler nedeniyle kullanıcı dostu ve rasgele sayı üreteçleri konusunda çalışan araştırmacıların rahatlıkla kullanabilecekleri bir yazılım ortamı geliştirilmiştir. Uygulama güncel,kolay kullanıma sahip ve etkili olamsından kaynaklı ASP.NET web geliştirme ortamı ile birlikte C# dili ve Bootstrap tasarım aracı ve MVC yazılım geliştirme mimarisi kullanılmıştır. Herhangi ek bir kuruluma ihtiyaç duymayan bu uygulamaya bir browser aracılığı ile internet bağlantısı olan tüm cihazlardan erişilebilmektedir. Uygulamanın arayüz tasarımı Şekil 4'te verildiği gibidir.


Şekil 4. Arayüz tasarımı

Kullanıcıların rahatlıkla kullanabilmesi açısından oldukça kullanıcı dostu ve sade bir arayüz tasarlanmıştır. Yazılım arayüzünde gerçekleştirilmesi istenen işlem doğrultusunda, 8 tane buton bulunmaktadır. Bu butonlarda son işlem yöntemleri bulunmaktadır. Bunlar; H, H2 ve H4 algoritmaları, XOR algoritması, Von Neumann algoritması, , SBOX, Mixing Bits in Steps and XORing ve Iterating Von Neumann algoritmalarıdır.

Arayüzde kullanıcının yapması gereken içinde rasgale sayı kaynağından elde edilmiş bit dizisi bulunan .txt formatında bir dosyayı sisteme yüklemek, istenen son işlem algoritmasını seçmek ve dönüştürme işlemini başlatmaktır. Sistem dosya içerisindeki bit dizisine seçilen son işlem yöntemini uygulayarak algoritma ile aynı isimde ve yine .txt formatında bir dosya üretir.

Tasarlanan yazılımı test etmek amacı ile (Avaroğlu ve ark., 2015) ve (Avaroğlu ve ark., 2014) makalelerinden alınan saf bit dizileri yazılım üzerinde son işlem algoritmalarına tabi tutulmuştur. Bu işlem sonrası elde edilen bit dizileri NIST testine tabi tutularak sonuçlar Tablo 4 ve 5 'de verilmiştir.

*Tablo 4. (Avaroğlu ve ark., 2015) makalesi kullanılarak üretilen bit dizisine uygulanılan test sonuçları*

| | **Saf bit** | **Xor** | **Vonneumann** | **H** | **H2** | **H4** | **Sbox** | **İtervanneu** | **Mixing** |
|---|---|---|---|---|---|---|---|---|---|
| Frekans Testi | - | - | 0,359 | 0,345 | 0,634 | 0,651 | 0,649 | 0,314 | - |
| Blok Frekans Testi | - | - | 0,589 | 0,932 | 0,244 | 0,812 | 0,128 | 0,341 | 0,044 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Akış Testi | - | - | 0,079 | 0,392 | 0,328 | 0,735 | 0.190 | 0,569 | 0,828 |
| En Uzun Birler Testi | - | - | 0,379 | 0,046 | 0,269 | 0,579 | 0,143 | 0,325 | 0,434 |
| İkili Matris Rankı Test | 0,454 | 0,785 | 0,990 | 0,280 | 0,128 | 0,148 | 0,574 | 0,545 | 0,272 |
| Ayrık Fourier Testi | 0,028 | 0,284 | 0,088 | 0,969 | 0,613 | 0,184 | 0,326 | 0,976 | 0,189 |
| Örtüşmeyen Şablon Eşleştirme | - | 0,016 | 0,023 | 0,752 | 0,076 | 0,328 | 0,134 | 0,336 | 0,261 |
| Örtüşen Şablon Eşleştirme Testi | - | 0,282 | 0,332 | 0,166 | 0,519 | 0,558 | 0,443 | 0,224 | 0,749 |
| Maurer Testi | - | 0,217 | 0,150 | 0,563 | 0,164 | 0,464 | 0,698 | 0,219 | 0,142 |
| Doğrusal Karmaşıklık Testi | 0,625 | 0,577 | 0,980 | 0,347 | 0,782 | 0,689 | 0,285 | 0,979 | 0,064 |
| Seri Testi | - | 0,354 | 0,765 | 0,163 | 0,522 | 0,372 | 0,438 | 0,424 | 0,185 |
| Yaklaşık Entropi Testi | - | 0,022 | 0,174 | 0,135 | 0,132 | 0,654 | 0,130 | 0,898 | 0,095 |
| Kümülatif Toplamlar Testi | - | - | 0,290 | 0,195 | 0,794 | 0,423 | 0,315 | 0,166 | - |

*Tablo 5.* (Avaroğlu ve ark., 2014) *makalesi kullanılarak üretilen bit dizisine uygulanılan test sonuçları*

| | **Saf bit** | **Xor** | **Vonneumann** | **H** | **H2** | **H4** | **Sbox** | **İtervanneu** | **Mixing** |
|---|---|---|---|---|---|---|---|---|---|
| Frekans Testi | 0,645 | - | 0,423 | 0,614 | 0,595 | 0,483 | 0,030 | 0,733 | 0,065 |
| Blok Frekans Testi | - | - | 0,451 | 0,829 | 0,087 | 0,987 | 0,059 | 0,463 | 0,071 |
| Akış Testi | - | - | - | 0,058 | 0,072 | 0,655 | 0,062 | - | 0,121 |
| En Uzun Birler Testi | - | - | - | 0,929 | 0,679 | 0,532 | 0,121 | - | 0,093 |
| İkili Matris Rankı Test | 0,684 | 0,584 | 0,231 | 0,631 | 0,587 | 0,963 | 0,832 | 0,680 | 0,267 |
| Ayrık Fourier Testi | - | - | 0,088 | 0,035 | 0,343 | 0,556 | 0,168 | 0,758 | 0,681 |
| Örtüşmeyen Şablon Eşleştirme | - | - | - | 0,563 | 0,796 | 0,878 | 0,158 | 0,133 | 0,072 |
| Örtüşen Şablon Eşleştirme | - | - | 0,975 | 0,870 | 0,989 | 0,160 | 0,054 | - | 0,383 |
| Maurer Testi | - | - | 0,243 | 0,624 | 0,342 | 0,444 | 0,852 | 0,697 | 0,658 |
| Doğrusal Karmaşıklık Testi | 0,678 | 0,547 | 0,254 | 0,867 | 0,862 | 0,826 | 0,591 | 0,327 | 0,589 |
| Seri Testi | - | - | 0,126 | 0,185 | 0,991 | 0,930 | 0,667 | 0,182 | 0,360 |
| Yaklaşık Entropi Testi | - | - | - | 0,133 | 0,974 | 0,816 | 0,541 | 0,063 | 0,436 |
| Kümülatif Toplamlar Testi | 0,721 | - | 0,178 | 0,944 | 0,625 | 0,679 | 0,033 | 0,781 | 0,071 |

# 5. Sonuç

Rasgele sayılarda son işlem algoritmaları alanında eksik gördüğümüz bir kısım olan bu uygulama bu alanda çalışan araştırmacılar için oldukça faydalı olacaktır. Henüz bir benzeri geliştirilmeyen bu uygulama kullanıcının bir rasgelelik kaynağından elde ettiği ham rasgele bitleri seçtiği son işlem algoritmasından geçirerek daha iyi istatistiksel özelliklere sahip rasgele sayılar elde etmesini sağlamaktadır. Uygulamanın herhangi bir kurulum ya da ek programlar gerektirmemesi, sadece bir browser üzerinden erişebilir ve basit bir arayüze sahip olması kullanıcılar için oldukça büyük kolaylık sağlamaktadır. Gerçekleştirilen web tabanlı yazılıma son kullanıcılar tarafından "postprocess.mersin.edu.tr" adresinden ulaşılarak, gerçek rasgele sayı üreteçlerinden elde edilen saf bit dizilerinin son işlem sonuçları alınabilecektir.

# Kaynakça


Avaroğlu E., Türk M., (2013). Son İşlemin Gerçek Rasgele Sayı Üreteçleri Üzerindeki Etkisinin İncelenmesi. *6th International Information Security and Cryptology Conference*, ISCTURKEY 2013, 290–294.

Avaroğlu E., Tuncer T., Özer A.B., Ergen B., Türk M., (2015). A novel chaos-based post-processing for TRNG. *Nonlinear Dynamics*, 81, 189–199.

Avaroğlu E., Tuncer T., Özer A.B., Türk M., (2014). A new method for hybrid pseudo random number generator. *J. Microelectron. Electron. Compon. Mater*, 4(4), 303–311.

Sunar B., Martin W. J., Stinson D. R., (2007). A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks. *IEEE Transactions on Computers 2007*, 56 (1), 109–119.

Tsuneda A., Mitsuishi S., Inoue, T., (2008). A Study on Generation of Random Bit Sequences with Post-Processing by Linear Feedback Shift Registers. *International Journal of Innovative Computing, Information & Control*, 4(10), 2631–2638.

Tsuneda, A., Morikawa, K., (2013). A Study on Random Bit Sequences with Prescribed Auto-Correlations by Post-Processing Using Linear Feedback Shift Registers. *2013 European Conference on Circuit Theory and Design (ECCTD)*.

Kyaw T.N.N., Tsuneda A., (2017). Generation of chaos-based random bit sequences with prescribed auto-correlations by post-processing using linear feedback shift registers. *NOLTA 2017*, 8, 224–234.

Loza S., Matuszewski L., (2014). A True Random Number Generator Using Ring Oscillators and SHA-256 as Post-Processings. *In International Conference on Signals and Electronic Systems (ICSES) 2014*, 1–4.

Trevisan L., (2001). Extractors and pseudorandom generators. *Journal of the ACM*, 48(4), 860–879.

Raz R., Reingold O., Vadhan S., (2002). Extracting all the Randomness and Reducing the Error in Trevisan's Extractors. *Journal of Computer and System Sciences,* 65(1), 97–128.

Wegman M. N., Carter, J. L., (1981). New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3), 265–279, 1981.

Zhang X., Nie Y., Liang H., Zhang J., (2016). FPGA implementation of Toeplitz hashing extractor for real time post-processing of raw random numbers. *2016 IEEE-NPSS Real Time Conference (RT),* Padua 2016, 1-5, doi: 10.1109/RTC.2016.7543094.

Yakut S., Tuncer T., Özer A. B., (2019). Secure and Efficient Hybrid Random Number Generator Based on Sponge Constructions for Cryptographic Applications. *Elektronika Ir Elektrotechnika*, 25(4), 40–46.

Yakut S., Tuncer T., Özer A. B., (2020). A New Secure and Efficient Approach for TRNG and Its Post-Processing Algorithms. *Journal of Circuits, Systems and Computers,* 29(15).

Kaya T., (2020). Memristor and Trivium-based true random number generator. *Phys. A Stat. Mech. its Appl.,* 124071.

Davies R. B., (2002). Exclusive OR (XOR) and hardware random number generators. http://www.robertnz.net/pdf/xor2.pdf.

Suresh V. B., Burleson W. P., (2010). Entropy extraction in metastability-based TRNG. *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST),* 135–140.

Dichtl M., (2007) Bad and Good Ways of Post-processing Biased Physical Random Numbers. *Proceedings of International Workshop on Fast Software Encryption* (Luxembourg, Luxembourg, Mar. 26-28, 2007), FSE '07. Lecture Notes in Computer Science, 4593, Springer, Berlin, Germany, 137–152.

Avaroğlu E., Tuncer T., (2020). A novel S-box-based postprocessing method for true random number generation. *Turk. J. Elec. Eng. & Comp. Sci.,* 28, 288–301.

Nikolic S., Veinovic M. D., (2016). Advancement of True Random Number Generators Based on Sound Cards Through Utilization of a New Post-processing Method. *Wireless Personal Communications*, 91(2), 603–622.

Peres Y., (1992). Iterating Von Neumann's Procedure for Extracting Random Bits. *Annals Statistics*, 20(1), 590–597.