Araştırma Makalesi

Son İşlem Algoritmalarının İncelenmesi

Didem Yosunlu*[1], Erdinç Avaroğlu[1]

[1]Mersin Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Mersin, Türkiye

Anahtar Kelimeler:
Rasgele Sayı Üreteçleri
Son İşlem
Son İşlem Algoritmaları

ÖZ

Rasgele sayılar, belirli bir aralık için tanımlanmış, oluşma olasılıkları birbirine eşit ve bu sayılar arasında belirli bir ilişki olmayan sayılar olarak tanımlanabilir. Rasgele sayı üreteçleri de rasgele sayıları üretmek için kullanılan araçlardır. Bu rasgele sayı üreteçlerinden elde edilen bit dizilerinin bazıları zayıf istatistiki özellikler göstermektedir. Bu nedenle üretilen dizilere bu zayıflığın giderilmesi amacıyla çeşitli son işlem algoritmaları uygulanmaktadır. Bu makalede şimdiye dek önerilen belli başlı son işlem algoritmaları incelenerek, karşılaştırılmaları yapılmıştır.

Examination of Post Processing Algorithms

Keywords:
Random Number Generator
Post Processing
Post Processing Algorithms

ABSTRACT

Random numbers can be defined as numbers that are defined for a certain range, their probability of occurrence is equal to each other, and there is no specific relationship between these numbers. Random number generators are tools used to generate random numbers. Some of the bit sequences obtained from these random number generators show poor statistical properties. For this reason, various post-processing algorithms are applied to the produced sequences in order to eliminate this weakness. In this article, major post-processing algorithms proposed so far have been examined and compared.

***Sorumlu Yazar**

*(didemoz80@gmail.com) ORCID ID 0000 – 0001 – 6917 – 4912
(eavaroglu@mersin.edu.tr) ORCID ID 0000-0003-1976-2526

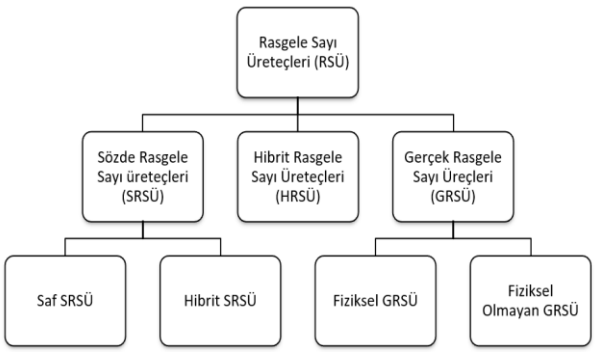
e-ISSN: 2717-8579
Geliş Tarihi: 04/11/2020; Kabul Tarihi: 06/12/2020

1. GİRİŞ

Rasgelelik en basit şekliyle deterministik olmayan yani belirli olmayandır. Rasgele sayılar ise, verilen iki koşulun yerine getirildiği bir dizide bulunan sayılardır; değerler belirli bir aralıkta eşit dağılmış veya ayarlanmış olmalı ve geçmiş değerleri veya şimdiki değerleri temel alarak gelecekteki değerleri tahmin etmek mümkün olmamalı.

Rasgele sayı üretimi, elektronik iletişimin gizliliğini korumak için temel bir süreçtir (Yadav, 2013). Günümüzde, rasgele sayı üreteçleri telekomünikasyon, bilgi güvenliği, oyun endüstrisi ve simülasyon başta olmak üzere birçok alan için çok önemli hale gelmektedir. Özellikle kriptografi, anahtar üretimi, kimlik doğrulama gibi telekomünikasyon uygulamaları için oldukça kritik bir öneme sahiptir (Anikin ve Alnajjar, 2016). Bu tarz uygulamalarda sistemin güvenliği üretilen bu sayıların gerçek rasgeleliliğine dayanmaktadır. Bundan dolayı kullanılan rasgele sayılar tahmin edilememe, tekrar üretilememe ve iyi istatistiksel özellikler gibi sıkı gereksinimleri sağlamaları gerekmektedir (Avaroğlu ve Türk, 2013). Bu sebeple de rasgele sayı üreteçlerine ihtiyaç doğmuştur.

Rasgele sayı üreteçleri (RSÜ), sözde rasgele sayı üreteçleri (SRSÜ), gerçek rasgele sayı üreteçleri (GRSÜ) ve hibrit rasgele sayı üreteçleri olmak üzere üç ana sınıfa ayrılır. SRSÜ'leri kendi içinde Saf ve Hibrit olarak ikiye ayrılırken GRSÜ'leri Fiziksel ve Fiziksel olmayan sayı üreteçleri olarak iki alt sınıfa ayrılmıştır. Şekil 1'de sınıflar gösterilmiştir.



Şekil 1. Rasgele Sayı Üreteçlerinin Sınıflandırılması

Sözde rasgele sayı üreteçleri tohum adı verilen bir başlangıç değerine bağlıdır. Belirlenen tohum değeri deterministik algoritmalara tabi tutularak uzun rasgele bit dizileri üretilir. Ucuz ve hızlı olmalarına karşın SRSÜ'ler periyodik olabilmektedir ve periyodun açığa çıkması ile üretilen rasgele sayıların tamamen tahmin edilebilir olmasını mümkün kılacaktır. Ayrıca tohum değerinin korunamaması da bu duruma neden olacaktır. Bu sebepten dolayı kriptografik uygulamalarda kullanımı uygun değildir.

Gerçek rasgele sayı üreteçleri entropi kaynağı olarak deterministik olmayan fiziksel olaylar kullanır. Faz gürültüsü, termal gürültü, titreşim, rastgele telgraf gürültüsü, fotoelektrik etki ve doğada istatistiksel olarak rasgele olan diğer fenomenler gibi fiziksel süreçlere dayanan yüksek hızlı ve gerçek rastgele diziler oluştururlar (Tehranipoor ve ark., 2016). Entropi kaynağı tarafından elde edilen sinyaller sayısala dönüştürülerek örnekleme yapılır. Örnekleme sürecinden sonra üretilen sayılar bazen zayıf istatistiki özellikler gösterebilmektedir. Bunun giderilmesi amacı ile üretilen bu bit dizileri son işleme tabi tutulmaktadır.

Son işlem algoritmaları zayıflıkları gidermeyi amaçlamakla beraber aynı zamanda sistemi saldırganlara ve çevresel değişikliklere karşı dirençli hale getirmeyi de amaçlamaktadır. Son işlem algoritmasına bağlı olarak üretecin güvenliği artmaktadır. Ancak, son işlem uygulamaları zayıflıkları giderirken uygulamaya bağlı olarak bit oranında azalmaya neden olabilirler. Buna rağmen kriptografik uygulamalar için kritik olan tahmin edilememe, tekrar üretilememe ve iyi istatistikî özellikleri sağlaması sebebiyle kriptolojide birçok uygulamada kullanılmaktadır.

Hibrit rasgele sayı üreteçleri GRSÜ'den elde edilen rasgele sayının SRSÜ'de tohum değeri olarak kullanılmasıyla her iki sistemin birlikte çalıştığı rasgele sayı üretecidir (Avaroğlu ve Türk, 2013).

Bu makalede amacımız şimdiye kadar önerilen ve sık kullanılan son işlem algoritmalarından bazılarının incelenerek karşılaştırılmalarının yapılmasıdır.

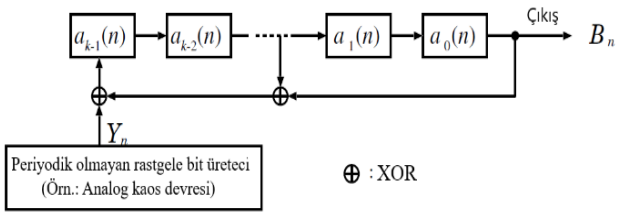
Bölüm 2'de literatür taraması yapılarak son yıllarda önerilen son işlem algoritmalarından bahsedilmektedir. Bölüm 3'te seçilen son işlem algoritmaları açıklanırken Bölüm 4'te ise bu algoritmaların karşılaştırmaları yer almaktadır.

2. LİTERATÜR TARAMASI

Günümüze kadar gerçek rasgele sayı üreteçlerindeki zayıflıkları gidermek ve istatistiki özellikleri güçlendirmek amacı ile birçok son işlem algoritması önerilmiştir. Bu algoritmalardan bazıları; Doğrusal Geri Beslemeli Kayan Yazmaç (DBKY), Lojistik Harita, SHA-256, Mixing Bits in Steps and XORing, High-Throughput Von Neumann, Keccak Algoritması ve SBOX.

Doğrusal Geri Beslemeli Kayan Yazmaç birçok uygulamada son işlem algoritması olarak kullanılmıştır (Tsuneda ve ark., 2008; Tsuneda ve Morikawa, 2013). Bunun sebepleri ise uygulamasının kolay ama kullanışlı olması, iyi istatistiksel özelliklere sahip diziler üretmesi ve aynı zamanda geniş tekrarlama periyodudur.

Örneğin (Tsuneda ve ark., 2008)'te üretilen rasgele sayıların periyodikliğini gidermek ve iyi istatistiksel özellikler göstermesi sağlamak için aşağıdaki şekilde bir yapı kullanılmıştır;



Şekil 2. DBKY kullanılan RSÜ'nün genel yapısı

Şekilde görüleceği üzere sistem bir rasgele sayı üreteci, k hafıza hücresi ve XOR dan oluşmaktadır. Her bir zaman biriminde her hafıza hücresi bir sağa kaydırılır, ayrıca $a_{k-1}(n)$ Denklem (1)'deki gibi güncellenir.

Denklem (1): DBKY hücre kaydırma

$$a_{k-1}(n+1) = a_0(n) \oplus a_1(n) \oplus Y_n$$

Yapılan çalışma göstermektedir ki önerilen son işlem algoritması k büyük olduğunda çok etkili olmaktadır.

Lojistik harita; donanım üzerinde uygulaması kolay olan, bir başlatma koşulu ve bir kontrol parametresi gerektiren birinci dereceden bir denklemdi (Avaroğlu ve ark., 2015). Lojistik haritanın yapısı en basit hali ile Denklem (2)'deki gibidir.

Denklem (2): Lojistik Harita

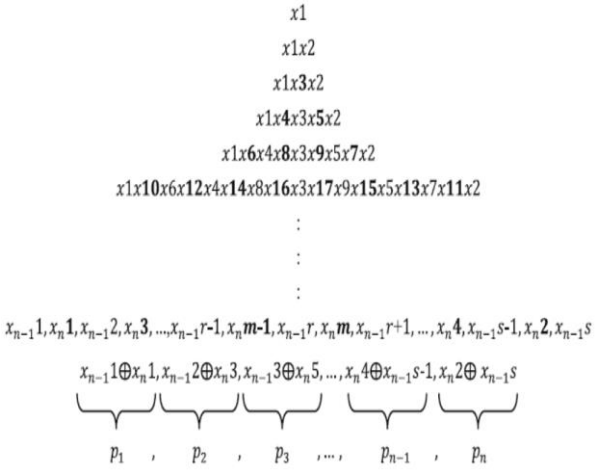
$$x(n+1) = \mathrm{r\,x(n)(1-x(n))}$$

Burada r sistem parametresi n ise tekrarlama sayısıdır. Aynı zamanda $0 \leq x(n) \leq 1$ koşulunun sağlanması gerekmektedir. $x(n)$ değeri $x(0)$ başlangıç değerinin seçilmesi ile hesaplanır. Sistem r parametresinin değerine göre farklı davranışlar sergileyebilmektedir. Aynı zamanda lojistik haritada başlangıç koşulunun değiştirilmesi ile farklı sayı dizileri üretilebilmektedir. Bu sebeple lojistik harita kullanan sistemler saldırılara karşı daha güvenlidir. Yüksek güç tüketimi sebebi ile dezavantajlı olmasına rağmen birçok son işlem algoritması veri çıkış oranını azaltırken lojistik harita veri çıkış oranını değiştirmemektedir.

Kaynak (Loza ve Matuszewski 2014)'te ise halka osilatörler ile tasarlanmış bir GRSÜ ile birlikte son işlem algoritması olarak SHA-2 özet fonksiyonları ailesinden olan SHA-256 kullanımı önerilmiştir. Kurulan tasarıma göre 512 rasgele biti giriş olarak alan SHA-256 algoritmasına yollanmak üzere GRSÜ'den gelen rasgele bitler 8 bitlik bloklara ayrılarak, her blok 64 byte genişliğinde olan FIFO buffer içerisinde saklanır. Bu algoritmanın sonucunda ise 8 tane 32 bitlik kelime (256 bit toplam) üretilir.

Önerilen bir diğer son işlem algoritmasında rasgele sayı üretmek için mikrofon aracılığı ile standart bir bilgisayarın, yeni nesil bir dizüstü bilgisayarın, tabletlerin ya da akıllı telefonların ses kartı kullanılarak bir rasgele gürültü sinyali sağlanmıştır (Nikolic ve Veinovic, 2016). Son işlem algoritması olarak da otokorelasyonu azaltmanın ve çıktı bit dizisinin toplam entropisini artırmanın yenilikçi bir yolu olduğu söylenen Mixing Bits in Steps and XORing of Adjacent Bits yöntemi kullanılmıştır. Kurulan sistemin 3 aşaması bulunmaktadır. İlk aşama, gürültü kaynağından elde edilen analog sinyalin sayısallaştırılmasıdır. İkinci aşamada bu sinyaller zayıflığın azaltılması amacı ile son işlem algoritmasından geçirilerek dahili rasgele sayılar üretilir. Üçüncü aşamada ise harici rasgele sayılar elde edilir.

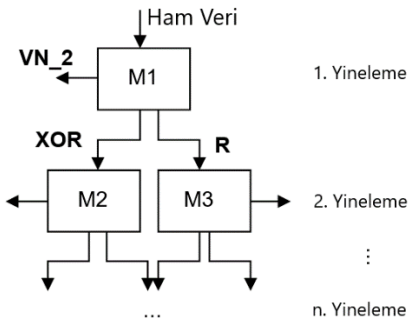
Karıştırma algoritması gelen ilk iki biti alır. İkinci aşamada üçüncü bit alınır ve ilk iki bitin arasına yerleştirilir. Üçüncü aşamada ise dördüncü bit 1. ve 3. bitin arasına yerleştirilir, beşinci bit ise 3. ve 2. bit arasına yerleştirilir. Bu aşamada görünüm x1; x4; x3; x5; x2 şeklindedir. Dördüncü adımda, altıncı bit 1. ve 4., yedinci 2. ve 5., sekizinci 4. ve 3. ve dokuzuncusu 5. ve 3. arasına yerleştirilir. Bu, dördüncü adımı tamamlar ve ortaya çıkan düzen x1; x6; x4; x8; x3; x9; x5; x7; x2 şeklinde olur. Tüm yerleştirmeler yapıldıktan sonra komşu olan bitler XOR işlemine tabi tutularak rasgele bit dizisi son halini almış olur. Bu uygulama yavaş olmasına rağmen istatistiki açıdan oldukça iyi sonuçlar vermiştir. Makalede aynı zamanda yavaşlığı azaltmak için de farklı bir yöntem önerilmektedir.



Şekil 3. Mixing Bits in Steps and XORing of Adjacent Bits yöntemi gösterimi

İlk olarak J.von Neumann tarafından 1951'de önerilen Von Neumann algoritması (Von Neumann, 1951) ise hala birçok rasgele sayı üretecinde son işlem algoritması olarak kullanılmaktadır. Veri çıkış oranını düşürdüğünden dolayı geliştirilmeye devam edilmektedir. Örneğin kaynak (Peres, 1992)'de Iterating Von Neumann (IVN) yöntemi sunulmuştur. Bu yöntemde IVN atılan bitleri toplayıp onları bir sonraki aşamaya yollayarak tekrardan kullanmaktadır. Aşağıdaki şekilde de görüleceği gibi her IVN modülü üç blok içerir: VN_2 (giriş çifti "01" veya "10" ise, ardından "0" veya "1" çıkışı verir, diğerleri çıkış olmaz), XOR ("00" veya "11" çifti

oluşursa çıkış '0', '01' veya '10' çiftleri daha sonra '1' çıktısı oluşur), R ('00' ise '0', '11' ise '1' çıktı, diğerleri çıktı olmaz). XOR ve R işleminden sonra çıkış verisi tekrardan işlenmemiş veri olarak bir sonraki yinelemeye verilir.



Şekil 4. Iterating Von Neumann (IVN)

1972'de ise Elias N bits Von Neumann (Elias, 1972) algoritmasını önermiştir. Orijinal Von Neumann (VN_2) algoritmasında 2 bit ile işlem yapılırken VN_N algoritmasında N bit ile işlem yapılmaktadır. Burada N'nin değerini artırarak, VN_N, yalnızca 1 kez işleme ile Shannon entropisine yakın yüksek çıktı oranı elde edebilir. Bununla birlikte, teorik değer ile gerçekçi değer arasında bir çıktı oranı farkı vardır. Teorik çıktı oranı N = 4, 8, 16 olduğunda sırasıyla %49,2, %68,2, %81,0'dır. Gerçekçi çıktı oranı sırasıyla %40,6, %55,2 ve %73,1'dir. Bu arada, N = 17'nin ötesinde verimlilik doymuş hale gelir. Bu yüzden uygun N değerlerinin seçilmesi gerekiyor.

2018'de hazırlanan makalede (Zhang ve ark., 2018) ise VN_N algoritması geliştirilerek teorik değer ile gerçekçi değerin birbirine yakınlaştırılması amaçlanmıştır. Burada herhangi bir N (N ≥ 2) değeri için bir eşleşme algoritması hazırlanmıştır. Böylece örneğin VN_2 algoritması 25.0% veri çıkış oranına sahipken VN_4 40.6% oranında çıkış vermiştir. Makalenin devamında VN_N+waiting adı verilen bir bekleme stratejisi de önerilerek VN_4+waiting için çıkış oranı 46.9% 'a kadar çıkarılmıştır. Böylece teorik çıktı oranı olan 49.2% a oldukça yaklaşılmıştır.

Kriptolojik uygulamalar için güvenliği arttırmak amaçlı kullanılan bir diğer son işlem yöntemi ise özet fonksiyonlar için son standart olan Keccak Algoritması. Selman Yakut, Taner Tuncer ve Ahmet Bedri Özer'in yayınladığı makalede (Yakut ve ark., 2019) güvenli rasgele sayılar üreten Keccak algoritması ve tahmin edilebilirliği ve yeniden üretimi engelleyen ek girdiler üreten halka osilatörler beraber kullanılmıştır. Çalışmada üreteci daha verimli hale getirmek için Keccak algoritması yeniden düzenlenip kullanılmıştır. Önerilen üreteçte, ek girdi olarak 512 bit ham gerçek rasgele sayı alınıp 1024 bitlik gerçek rasgele sayı dizisi üretilmektedir.

Aynı yazarların yayınladığı bir diğer makalede ise yine Keccak algoritması kullanılarak 1600 bit giriş için sıfır veri kaybı ile 1600 bitlik rasgele sayı dizisi üretilmiştir (Yakut ve ark., 2020).

Önerilen bir diğer son işlem yöntemi ise S-box (Avaroğlu ve Tuncer, 2020). SBOX DES, AES gibi şifreleme sistemlerindeki doğrusal olmayan tek bileşendir. Bu yüzden güçlü şifreleme sistemlerinin tasarlanması için kriptolojik özellikleri iyi olan S-Box'ların tasarlanması gerekmektedir. (Avaroğlu ve Tuncer, 2020)'de önerilen algoritmada RO tabanlı bir GRSÜ tanımlanmış ve son işlem algoritması olarak da DES için tasarlanmış olan S-box kullanılmıştır. Kullanılan S-box 6 bit giriş için 4 bit çıkış üretmektedir. 6 bit giriş dizisinin en önemli biti (MSB) ve en az anlamlı biti (LSB) satırı belirlerken, diğerleri sütunu belirlemektedir. Bu satır ve sütundaki sayının ikili sistemdeki karşılığı çıkış bit dizisi olarak kabul edilir. Veri çıkış oranını 2/3 oranında azaltmaktadır.

3. SON İŞLEM ALGORİTMALARI

Rasgele sayı üreteçlerinden elde edilen bit dizilerinden bazıları zayıf zayıf istatistiki özellikler göstermektedir. Bu nedenle üretilen dizilere bu zayıflığın giderilmesi amacıyla çeşitli son işlem algoritmaları uygulanmaktadır. Son işlem algoritmasına bağlı olarak üreteçler, saldırganlara ve çevresel değişikliklere karşı dirençli hale gelirken üretecin güvenliği de artmaktadır.

3.1. Von Neumann Son İşlem

Von Neumann (Suresh ve Burleson 2010) algoritması bit dizisindeki düzensizlikleri gidermeyi amaçlayan en basit ve en eski son işlem yöntemlerindendir. Algoritma giriş olarak üreteçten gelen eşzamanlı bitleri alır. Tabloda gösterildiği gibi bit dizisi (1,0) ise çıkış biti 1, eğer bit dizisi (0,1) ise çıkış biti 0 olarak üretilir. (0,0) ve (1,1) bit dizileri dikkate alınmaz.

Bu dikkate alınmayan bit dizilerinden dolayı sistemin çıkış bit hızı sabit değildir.

Tablo 1. Von Neumann son işlem çıkışı

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

3.2. XOR Son İşlem

XOR (Davies, 2002) doğrulama, elde edilen bit dizisinin n bit(n=2) bloklara ayrılıp bu blokların kendi içerisinde XOR işlemine tabii tutulması ile 1 çıkış biti üretilmesi işlemidir. Çıkış bit verimini 1/n kez azaltmaktadır.

Tablo 2. XOR son işlem çıkışı

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

3.3. H Fonksiyonu Son İşlem

Bir diğer son işlem yöntemi ise Dichtl tarafından önerilen, Quasigroup dizi dönüşümüne dayalı olan H son işlem algoritmasıdır (Dichtl, 2007). Şekil 1.1'de de görüleceği gibi 16 bit giriş bitinden 8 bit çıkış elde edilir. Son işlem için giriş bitleri $a0$, $a1$, $a2$, … $a15$ şeklinde tanımlanmıştır. $bi = ai \oplus a(i+1) mod 8$ ile $b0$, $b1$,..., $b7$ 8 bit tanımlanır. $c0$, $c1$, … , $c7$ çıkışı $ci = bi \oplus a(i+8)$ ile tanımlanır.

16 bit giriş dizisini $a1$ ve $a2$ olarak ikiye ayırdığımızda Denklem (2) deki gibi olmaktadır. sözde kod olarak gösterimi aşağıdaki şekilde olmaktadır;

Denklem (3): H fonksiyonu sözde kod gösterimi

H (a1, a2) = XOR (XOR (a1, rotateleft (a1, 1)), a2)

Daha sonra H fonksiyonu geliştirilerek Denklem (4)' te gösterilen H2 ve Denklem (5)'te gösterilen H4 fonksiyonları tanımlanmıştır.

Denklem (4): H2 fonksiyonu sözde kod gösterimi

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

Denklem (5): H4 fonksiyonu sözde kod gösterimi

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

3.4. SBOX Son İşlem

SBOX ile amaç bir tablo vasıtası ile kimin kiminle yer değiştireceğinin belirlenmesidir. Oluşturulan S-kutusunun tipine veya kategorisine bakılmaksızın, herhangi bir S-kutusunun etkili olabilmesi için belirli özellikler göstermesi gerekir. İstediğiniz herhangi bir yer değiştirme şemasını bir araya getirip iyi bir S-kutusu oluşturamazsınız. Bir S-kutusu karışıklık (confusion) ve difüzyon (diffusion) sağlamalıdır. S-kutularında bu girişleri üretmenin bir yolu, n giriş bitlerini m çıkış bitlerine eşleyen doğrusal olmayan bir Boole fonksiyonu (Boolean function) oluşturmaktır.

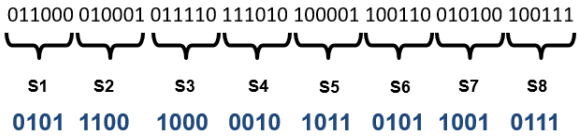
(Avaroğlu ve Tuncer, 2020)'de önerilen son işlem yönteminde DES için hazırlanan 8 SBOX kullanılmıştır.

DES S-Kutuları 6 bit giriş alır ve 4 bit çıkış verir. Görüldüğü üzere 6 bitin dışta olanları (1. ve 6. bit) satırları, iç tarafta kalan bitler (2.,3.,4.,5.) sütunları oluşturmaktadır. Buna göre seçilen satır ve sütundaki sayı da çıkışı verir. Şekil 5'te gösterilen örnekle açıklayacak olursak;

Şekil 5. Örnek bit dizisi

S1 için ilk 6 bitlik diziye baktığımızda MSB ve LSB yani 00 bize satırı verecektir. Bu durumda bakmamız gereken satır 0. Arada kalan bitler ise sütunu verecektir; 1100 yani; 12. Bu durumda bakmamız gereken yer S-box 1'de 0. Satır 12. Sütun. Karşılığında denk gelen sayı 5. Bunun 2li sistemdeki karşılığı ise bize çıkış bit dizimizi verecek yani; 0101.

Bu şekilde tüm dizi için SBOX üzerindeki karşılıklarını bularak Şekil 6'da gösterilen yeni çıktı dizimizi üretiyoruz.

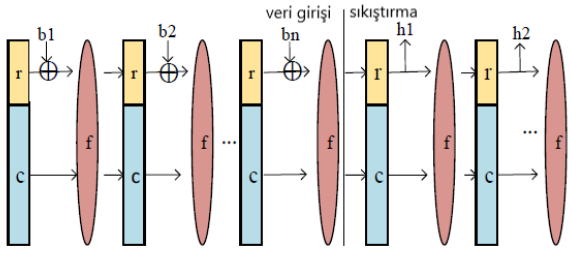


Şekil 6. SBOX son işlem algoritması uygulandıktan sonra oluşan bit dizisi

3.5. Keccak Algoritması

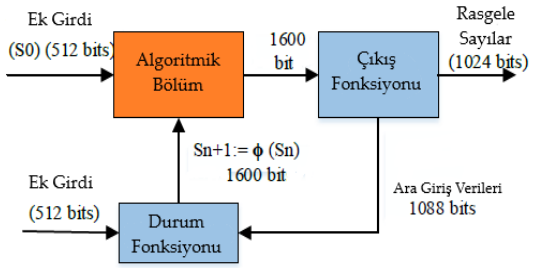
Özet algoritmalarının en son standardı olan Keccak, sünger yapısına dayandığı için herhangi bir uzunlukta bir girdi dizisi alabilir ve çeşitli uzunluklarda özet değeri üretebilir (Yakut ve ark., 2019).

Şekil 7'de gösterildiği gibi bir yapıya sahip olan Keccak algoritması 2 temel parçadan oluşmaktadır. İlk parçada giriş mesaj blokları işlenirken ikinci bölümde özet değerleri üretilmektedir.



Şekil 7. Keccak algoritmasının genel yapısı

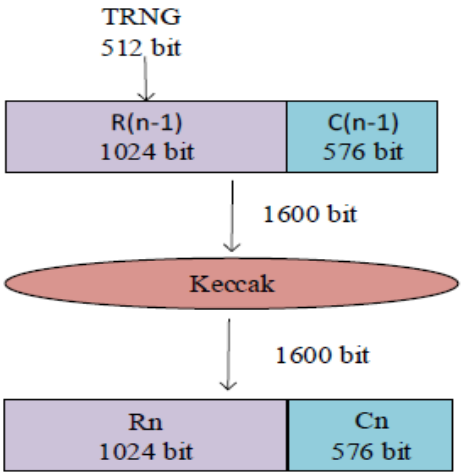
(Yakut ve ark., 2019)'da Keccak algoritması yeniden düzenlenerek kullanılmıştır ve 512 bit giriş dizisi ek girdi olarak alınıp 2014 bit gerçek rasgele ayı üretimi sağlanmıştır. Kullanılan yapı Şekil 8'de gösterilmiştir. 512 bit ham gerçek rasgele sayı ek girdi olarak kullanılmıştır. Üretecin ilk çalışmasında ek girdi(S0) Keccak fonksiyonuna verilirken, sonraki Durum Fonksiyonuna verilir. Keccak fonksiyonunda algoritmik işleme tabi tutulan saf veriden 1600 bitlik özet değeri üretilir. Bu algoritmik işlemlerde 1600 bitlik giriş alınır ve 24 tur işlendikten sonra 1600 bitlik çıktı oluşur. Çıkış fonksiyonu kullanılarak alınan 1600 bit özet değerinden 1024 bit gerçek rasgele sayı ve 1088 bit ara giriş verisi üretilir. Ara giriş verisi ve ek girdi kullanılarak bir sonraki çalışmada Keccak algoritması için 1600 bitlik giriş verisi oluşturulur.



Şekil 8. (Yakut ve ark., 2019)'da önerilen rasgele sayı üretecinin genel yapısı

Yine aynı yazarların bir diğer çalışmasında ise yine Keccak algoritması kullanılarak 1024 bit ham gerçek rasgele sayı girişinden, 1024 bit gerçek rasgele sayı çıkışı üretilmiştir (Yakut ve ark., 2020).

Şekil 9'da gösterildiği üzere oluşturulan yapı R giriş vektörü ve C ara vektöründen oluşur. Her çalışmada 1024 bit ham gerçek rasgele sayı R, bir önceki çalıştırmadan gelen 576 bit vektör kısmı C olarak alınır. Bu 1600 bitlik giriş vektörü, algoritma tarafından işlenir ve 1600 bitlik bir çıktı vektörü oluşturulur. Bu çıkış vektörü, R ve C bloklarından oluşur.



Şekil 9. (Yakut ve ark., 2020)'de önerilen rasgele sayı üretecinin genel yapısı

3.6. Resilient Fonksiyon

(n, m, t)-resilient fonksiyonu $F(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m)$ şeklinde, $Z_2^n \rightarrow Z_2^m$ olan ve $i_1, \ldots, i_t$ koordine eden herhangi bir t için Z_2'den herhangi bir $a_1, \ldots, a_t$ sabiti için ve ortak etki alanının herhangi bir y öğesi için Prob $[F(x) = y | x_{i1} = a_1, \ldots x_{it} = a_t] = \frac{1}{2^m}$ özelliğini sağlayan bir fonksiyondur.

Daha basit bir ifade ile açıklayacak olursak; fonksiyon girdi olarak verilen herhangi bir t değerinin bilinmesi kişinin çıktıda rasgele bir tahminden fazlasını yapmasına izin vermez. Bu yönüyle oldukça güvenlidir ve birçok kriptografik uygulamada kullanılmaktadır.

Sunar, Martin ve Stinson tarafından yapılan bir çalışmada da resilient fonksiyonlar deterministik bitlerin filtrelenmesi için kullanılmıştır (Sunar ve ark., 2007). Burada hata düzeltme kodlarından resilient fonksiyon oluşturulmuştur. Bunun için de basit bir teknik verilmiştir: G, [n, m, d] doğrusal kod için bir üretici matrisi olsun. $f(x) = xG^T$ kuralına göre bir $f : \{0,1\}^n \mapsto \{0,1\}^m$ fonksiyonu tanımlandığında, bu fonksiyon f(n, m, d-1)-resilient fonksiyonudur. Uygulama için kullanılan kod ise H_m Hamming kodunun ikilisi olan, tek yönlü $[2^m - 1,\ m,\ 2^{m-1}]$ doğrusal kod $H_m^{\perp}$ 'dir.

Makalede 114 halka osilatörden alınan çıktı resilient fonksiyonuna verilir. Resilient fonksiyon için bilinen bir genişletilmiş BCH kodu olan [256, 16, 113] kodu kullanılmaktadır. Yani 256 bitlik bloklara ayrılan osilatör çıktıları resilient fonksiyona verilir ve fonksiyondan yalnızca 16 bitlik çıkış alınır.

4. SONUÇ

Tablo 3. Son işlem algoritmaları ve bit dizisi azalma oranları

Son İşlem Algoritmaları	**Bit Dizisi Azalma Oranı**
Von Neumann	3/4
XOR	1/2
H-Fonksiyonu	1/2
SBOX	2/3
Keccak Algoritması	0/1
Resilient Fonksiyon	1/16

XOR son işlem algoritması basit ve sabit bit çıkış hızı sağlamasından dolayı tercih edilse de sistemin güvenliğini sağlama konusunda zayıf kalmaktadır. Von Neumann algoritması ise entropiyi ideal değer olan 1'e yaklaştırırken bazı giriş bitleri atıldığından dolayı sistemin bit çıkış hızı sabit değildir ve 3/4 oranında azalmaya sebep olur.

H fonksiyonu 15/16 oranında bir azalmaya sebep olurken entropi değerinin 1 olarak elde edilmesi en büyük avantajıdır.

SBOX son işlem algoritması ise NIST testlerinde başarılı sonuçlar elde etmesinin yanı sıra otokorelasyon ve karmaşıklık ölçümlerinde de

başarılı sonuçlar elde etmiştir. Ayrıca H fonksiyonu ve Von Neumann algoritmalarında olduğu gibi entropi değeri 1 olarak elde edilmektedir. Bu başarılı sonuçlardan dolayı kriptografi gibi birçok alan için kullanıma uygun olduğu görünmektedir.

Resilient fonksiyonu ise saldırılara karşı güçlü olmasına karşın H fonksiyonu gibi 15/16 oranında bir veri kaybına sebep olmaktadır.

Keccak algoritmasında ise ek girdiler kullanılarak rasgele sayıların yeniden üretilmesi ve tahmin edilebilirlik önlenmiştir. Yapılan testler göstermektedir ki kriptografik uygulamalar için gerekli olan güvenlik gereksinimleri karşılanmaktadır. Algoritmanın en önemli avantajı ise veri kaybına sebep olmamasıdır.

KAYNAKÇA

Yadav, A. (2013). Design and Analysis of Digital True Random Number Generator, M.Sc. *Thesis Virginia Commonwealth University, Virginia.*

Anikin, I. V. & Alnajjar, K. (2016). Psseudo-Random Number Generator Based on Fuzzy Logic. In *2016 International Siberian Conference on Control and Communications (SIBCON), 2016 International Siberian Conference, 1-4.*

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

Tehranipoor, F., Yan, W. & Chandy, J. A. (2016). Robust Hardware True Random Number Generators using DRAM Remanence Effects. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), 79–84.*

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. In *2013 European Conference on Circuit Theory and Design (ECCTD).*

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.*

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.*

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.*

Von Neumann, J. (1951). Various Techniques Used in Connection With Random Digits. *National Bureau of Standards Applied Math Series 12, 36–38.*

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

Elias, P. (1972). The efficient construction of an unbiased random sequence*. Ann. Math. Statist, 43(3), 864–870.*

Zhang, R., Chen, S., Wan, C. & Shinohara, H. (2018). High-Throughput Von Neumann Post-Processing for Random Number Generator. In *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), 1-4.*

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*

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

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

Davies, R. B. (2002). Exclusive OR (XOR) and hardware random number generators. *1-11. http://www.robertnz.net/pdf/xor2.pdf*

Dichtl, M. (2007). Bad and Good Ways of Post-processing Biased Physical Random Numbers. In *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.*

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.*