Fırat Üniversitesi Müh. Bil. Dergisi
32(1), 287-294, 2020

Görüntü Şifreleme için Trivium-Doğrusal Eşlenik Üreteci Tabanlı Bit Üretimi

Taha ETEM[1*], Turgay KAYA[2]
[1] Elektrik-Elektronik Mühendisliği, Muş Alparslan Üniversitesi, Mühendislik Fakültesi, Muş, Türkiye
[2] Elektrik-Elektronik Mühendisliği, Fırat Üniversitesi, Mühendislik Fakültesi, Elazığ, Türkiye
[*1] t.etem@alparslan.edu.tr, [2] tkaya@firat.edu.tr

(Geliş/Received: 18/10/2019; Kabul/Accepted: 14/02/2020)

Öz: Sözde rastgele sayı üreteçleri uygulama kolaylığı sebebiyle tercih nedeni olmaktadır. Herhangi bir donanım olmaksızın rastgele sayı dizilerini hızlı bir şekilde üretebilmek önemli bir avantajdır. Üretilen sayıların kriptolojik uygulamalar gibi hassas alanlarda kullanılması durumunda bazı istatistiksel gerekliliklerin karşılanması gerekmektedir. Bu amaçla uygulanan sözde rastgele sayı üreteci tasarımı LCG algoritmasıyla oluşturulduktan sonra Trivium algoritmasıyla son işlemden geçirilerek istatistiksel testlerde başarılı bir rastgele sayı dizi elde edilmiştir. Üretilen sayılar DES, 3DES ve AES şifreleme algoritmaları için anahtar olarak kullanılarak örnek görüntü şifreleme uygulamaları yapılmıştır. Oluşturulan görüntü şifreleme sisteminin her bir algoritma için işlenme hızları karşılaştırılmıştır. AES şifreleme standardı hem en güncel algoritma olması hem de şifreleme ve şifre çözme sürelerindeki başarısıyla öne çıkmıştır. Ayrıca incelenen algoritmalar içerisinde 256 bit'e kadar en uzun şifreleme anahtarlarını desteklemesi sebebiyle daha güvenli bir şifreleme işlemi gerçekleştirdiği söylenebilir.

Anahtar kelimeler: Sözde Rastgele Sayı Üreteci, Doğrusal Eşlenik Üreteci, Trivium Algoritması, Görüntü Şifreleme.

Trivium-Linear Congruential Generator Based Bit Generation For Image Encryption

Abstract: Pseudo random number generators are preferred because of the easy application. Without any hardware, quickly generating random number streams is an important advantage. Applying generated numbers on delicate fields such as cryptographic applications is required some statistical necessities. For this purpose, pseudo random number generator application design is created by LCG algorithm then post-processing is made by Trivium algorithm for obtaining statistically successful random number stream. Generated numbers are used as a key for DES, 3DES and AES cryptography algorithms for applications of sample image encryption. The processing speeds of the created image encryption system for each algorithm were compared. The AES encryption standard stands out with its being the most up-to-date algorithm as well as its success in encryption and decryption durations. Also, it can be said that it performs a more secure encryption process since it supports the longest encryption keys up to 256 bits among the examined algorithms.

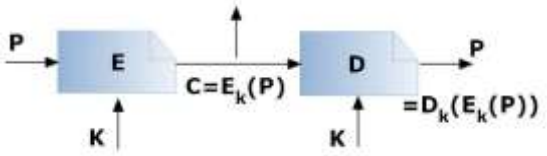
Key words: Pseudo Random Number Generator, Linear Congruential Generator, Trivium Algorithm, Image Encryption.

1. Giriş

Önemli verileri gizlemek, başkaları tarafından anlaşılamayacak bir forma sokmak için günümüzde birçok farklı teknik kullanılmaktadır. Şifreleme işlemi yapılırken genelden mantıksal elemanlardan ve matematiksel işlemlerden faydalanılarak oluşturulan bir model kullanılır [1]. Birçok sistem şifreleme işlemi için anahtar diye tabir edilen bir değer kullanır. Anahtar uzunluklarının artması, genellikle şifreli verinin istenmeyen kişilerce çözülmesini zorlaştırsa da şifreleme işlemi için gerekli kaynak gereksinimi ve zamanı da arttırmaktadır [2]. Anahtar uzunluğunun artması tahmin edilmesi gereken sayı olasılığını arttırmış olacak böylece üçüncü şahısların anahtarı elde etmesi zorlaşarak zaman alacaktır. Bu tarz sistemlerde anahtarın uzunluğu arttığında hem donanımsal ve yazılımsal gerçeklemelerde sorun oluşabilir. Bazı kaotik devre tabanlı şifreleme sistemleri bu sorunları azaltsa da uygulama zorluğundan kaynaklanan sebeplerle bu sistemlerin kullanımı yaygınlaşmamıştır [3]. Şifreleme teknikleri genel olarak Şekil 1'de gösterilen orijinal data (P), şifrelenmiş data (C), şifreleme anahtarı (K), şifreleme algoritması (E) ve şifre çözme algoritmasından (D) meydana gelmektedir [4].

Güvenli olmayan haberleşme ortamlarında şifrelenmiş bir şekilde veri grupları elde etmek zor bir iş değildir. Ancak bu veri kümelerini anlamlandırabilmek için kullanılan şifre çözme uygulamasını ve şifreleme anahtarını elde etmek gerekir. Belirli protokollerle standart hale getirilen bu işlemler verinin istenmeyen ortamlarda elde edilememesini, istenilen ortamda ise kolay bir şekilde elde edilebilir olmasını hedefler [5].

* Sorumlu yazar: t.etem@alparslan.edu.tr. Yazarların ORCID Numarası: [1] 0000-0003-1419-5008, [2] 0000-0002-7732-6194



Şekil 1. Şifreleme ve şifre çözme uygulamalarının genel işleyişi[4]

Şifreleme işlemleri için anahtarın durumuna bağlı olarak simetrik anahtarlı şifreleme ve asimetrik anahtarlı şifreleme olarak iki yapı mevcuttur. Burada kullanılan anahtarın dağıtımı özel bir kanalla ya da yine veri iletiminde kullanıldığı gibi gizli bir kanalla yapılabilir.

Şifreleme anahtarı kriptolojik uygulamaların temelini oluşturduğu için seçilmesi büyük önem arz etmektedir [6]. Bu ihtiyacı giderebilmek amacıyla literatürde birçok rastgele sayı üreteci tasarımı bulunmaktadır [7]. Elde edilen her yeni rastgele sayı üreteci tasarımıyla kullanılabilirlik arttırılmıştır [8-9]. Ayrıca rastgele sayı üreteçlerinin şans oyunları, istatistiksel uygulamalar gibi farklı kullanım alanları da mevcuttur [10].

Rastgele sayı üreteçleri sözde ve gerçek rastgele sayı üreteçleri olmak üzere iki ana başlıkta toplanmıştır. Bunların her iki bileşenini de içeren hibrit rastgele sayı üreteçleri ise sonradan türetilmiştir [11]. Gerçek rastgele sayı üreteçleri genellikle ölçülebilen veriler ve devre tabanlı olarak karşımıza çıkarken sözde rastgele sayı üreteçleri genelde matematiksel işlemler içeren algoritmalarla elde edilir [12].

Sözde rastgele sayı üreteçleri farklı algoritmalarla elde edilebilirler. Şifreleme uygulamalarında sıklıkla kullanılmaları harici bir donanıma sahip olmaksızın oluşturulup yüksek hızlarda çalışmaya imkan vermelerinden kaynaklanmaktadır [13]. Literatürde farklı tasarımlara sahip çok sayıda sözde rastgele sayı üreteci bulunmaktadır [14].

Rastgele sayı üreteçleri çıktısında elde edilen sayıların genelde farklı yapılarda olması rastgele sayı üreteci için kullanımlarında bir ön işlem ihtiyacını doğurmaktadır [15]. Öncelikle bu sayıların normalizasyon işlemleri yapılmalı ve sayılar binary (ikili) sayı formatına dönüştürülmelidir. İkili sayı koduna dönüştürülmüş sayıların istatistiksel analizleri hassas bir şekilde yapılarak rastgele sayı üreteci içerisinde kullanımları değerlendirilmelidir. Sonuç olarak elde edilen sayıların analizleri için literatür tarafından sıklıkla kullanılan NIST-800-22 veya FIPS-140-1 gibi istatistiksel testler uygulanmalıdır. Test sonuçlarına göre tasarlanan sistem gözden geçirilmelidir. Bir testten bile başarısız olan bir sayı dizisi rastgele olarak kabul edilmeyecektir. Bazı istatistiksel özellikleri arttırmak için diziye son işlem algoritmaları uygulanabilir [16].

Güncel olarak kullanımda olan birçok şifreleme algoritması mevcuttur. Bu algoritmalar genel olarak veriyi farklı döngülerle şifrelemeye çalışır [17]. Temel amaçları çok uzun anahtarlar kullanmadan şifreleme algoritmasının kendi oluşturduğu entropi yardımıyla verinin sadece istenilen kişilerce elde edilebilir olmasını sağlamaktır [18]. Bu algoritmalardan en çok kabul göreni geçmişte DES, güncellenen versiyonuyla 3DES ve günümüzdeki en güncel versiyonu AES olmuştur. Bu algoritmaların tercih edilmesinde veri güvenliğinin yanı sıra uygulama kolaylığı ve çabukluğu da öne çıkmıştır. Ayrıca bu algoritmalar görüntü şifreleme için kullanıma uygun algoritmalardır [19].

Bu çalışmada Linear Congruential Generator (LCG) tabanlı Trivium algoritmasıyla bir sözde rastgele sayı üreteci tasarımı oluşturulmuştur. Oluşturulan sistemin istatistiksel NIST testi yardımıyla rastgeleliliği incelenmiştir. Oluşturulan rastgele sayı üretecinin çıktıları DES, 3DES ve AES algoritmaları için anahtar olarak kullanılarak görüntü şifreleme uygulamalarında kullanılmıştır.

2. Doğrusal Eşlenik Üreteci

Temel anlamda rastgele sayılar üretmek için kullanılan LCG algoritması oldukça temel matematiksel işlemlere dayanır.

$$X_i = [(A \times X_i) + B] \bmod C \tag{1}$$

Denklem 1'de görüldüğü üzere üretilen verilen bir X_i başlangıç değeriyle rastgele sayı üretimi başlar. Üretilen sayıların maksimum değeri C değişkeni ile belirlenir. Diğer değişkenler yardımıyla da farklı rastgele sayı dizileri üretmek mümkün olmaktadır [20].

LCG'nin en büyük avantajı minimum oranda sistem kaynağını kullanarak rastgele sayı üretmesidir. Ancak oldukça basit bir algoritma yardımıyla üretilmesi kriptolojik uygulamalarda kullanımını kısıtlamaktadır. Sayı

üretim kapasitesi yüksek tutulduğunda istatistiksel olarak iyi sonuçlar verse de düşük kapasiteli sayı üreteci tasarımlarında istatistiksel zaaflar bulunabilmektedir [21].

Algoritma 1. LCG için Python kodları

```
def lcg(modindisi, a, b, ilkdeger):
    while True:
        seed = (a * ilkdeger + b) % modindisi
        yield ilkdeger
```

Algoritma 1 içerisinde LCG için örnek Python kodu verilmiştir. LCG birçok programlama dilinde kolayca elde edilebilmektedir. Şifreleme uygulamaları için de uygun olması sebebiyle LCG için Python tercih edilmiştir.

3. Trivium Algoritması

Üretilen rastgele sayıların istatistiksel özelliklerini iyileştirmek ve kriptolojik uygulamalarda kullanılabilir hale getirmek için Trivium algoritması son-işlem olarak uygulanmıştır. Trivium algoritması oldukça hızlı bir şekilde çalışırken rastgele sayı dizisinin çıktılarından çok önemli istatistiksel değişikliklere yol açar. Esnek bir uygulama alanına sahip olması sebebiyle sıklıkla tercih edilen bir algoritma olmuştur [22].

Tablo 1. Trivium Algoritması Parametreleri

Parametreler	**Boyut**
Anahtar Boyutu	80 bit
Başlangıç Değeri Boyutu	80 bit
İç Durum Boyutu	288 bit

Tablo 1'de Trivium algoritmasının parametreleri gösterilmiştir. Trivium algoritmasının işleyiş şeması ise aşağıdaki algoritmada verilmiştir. 288 bitlik iç durum değişkenleri $s_1, s_2, \ldots, s_{288}$ ile ifade edilmiştir. Bu algoritma ile hızlı bir şekilde rastgele sayılar üretmek mümkündür. Ayrıca Trivium algoritması yapılan çalışmalarla saldırılara karşı dayanıklı olduğunu göstermiştir [22].

Algoritma 2. Trivium Algoritması İşleyişi

for $i=1$ *to* n

$t_1 \leftarrow s_{66} \oplus s_{93}$

$t_2 \leftarrow s_{162} \oplus s_{177}$

$t_3 \leftarrow s_{243} \oplus s_{288}$

$z_i \leftarrow t_1 \oplus t_2 \oplus t_3$

$t_1 \leftarrow t_1 \oplus s_{91} \cdot s_{92} \oplus s_{171}$

$t_2 \leftarrow t_2 \oplus s_{175} \cdot s_{176} \oplus s_{264}$

$t_3 \leftarrow t_3 \oplus s_{286} \cdot s_{287} \oplus s_{69}$

$(s_1, s_2, \ldots\ldots, s_{93}) \leftarrow (t_3, s_1, \ldots\ldots, s_{92})$

$(s_{94}, s_{95}, \ldots\ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots\ldots, s_{176})$

$(s_{178}, s_{179}, \ldots\ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots\ldots, s_{287})$

end for

4. İstatistiksel NIST 800-22 Testi

Literatürde çok sayıda istatistiksel test bulunmasına karşın en çok kabul gören istatistiksel test NIST (National Institue of Standards and Technology) tarafından önerilmiştir. Toplamda 15 istatistiksel test bulunan NIST-800-22 tüm otoriteler tarafından istatistiksel rastgelelik testleri arasında en çok kabul görendir [23].

NIST-800-22 testi, kendi içerisinde bulunan 15 farklı bit serisini rastgelelik açısından incelemek için genellikle yeterlidir. Son iki test minimum bir milyon adet örnek gerektirdiğinden genellikle test için bir milyondan fazla örnek kullanılır. NIST-800-22 testi için, çıktılar P-değerleri ile değerlendirilmektedir. P-değerinin genellikle 0.01'den büyük olması şartı aranır [24]. Böylece elde edilen değerler 0.01<P-değeri<1 aralığında olmalıdır. P-değerinin 1'den büyük olarak hesaplanması zaten matematiksel olarak mümkün değildir. Genel kabul olarak görülen 0.01'den küçük olarak P-değeri hesaplandığında ise rastgelelik testinin başarısız olduğu savunulabilir [25].

Tablo 2'de LCG çıktıları ham bir şekilde ve Trivium algoritmasıyla son-işlem uygulanmış bit dizilerinin istatistiksel test sonuçları verilmiştir.

Tablo 2. NIST 800-22 Test Sonuçları

NIST Tests	LCG Çıktısı Başarısı	LCG Çıktısı P-Değerleri	Trivium Algoritması Başarısı	Trivium Algoritması P-Değerleri
Frequency Monobit Test	Başarısız	0	Başarılı	0.79
Frequency Test within a Block	Başarısız	0	Başarılı	0.27
Runs Test	Başarısız	0	Başarılı	0.25
Longest Run of Ones in a Block Test	Başarısız	0	Başarılı	0.34
Binary Matrix Rank Test	Başarılı	0.21	Başarılı	0.07
Discrete Fourier Transform Test	Başarısız	0	Başarılı	0.41
Non Overlapping Template Matching Test	Başarısız	0	Başarılı	0.03
Overlapping Template Matching Test	Başarısız	0	Başarılı	0.32
Universal Test	Başarısız	0	Başarılı	0.20
Linear Complexity Test	Başarısız	0	Başarılı	0.91
Serial Test	Başarısız	0/0	Başarılı	0.95/0.94
Approximate Entropy Test	Başarısız	0	Başarılı	0.13
Cumulative Sums Test	Başarısız	0	Başarılı	0.54
Random Excursions Test	Başarısız	0	Başarılı	0.30
Random Excursions Variant Test	Başarısız	0	Başarılı	0.31

Tabloda görüldüğü üzere LCG çıktısı sonucunda başarısız olan tüm testler Trivium algoritmasıyla uygulanan son işlem sonrasında başarılı olmuştur. Trivium algoritmasının etkinliği bu şekilde açıklanabilir.

5.Şifreleme Uygulamaları

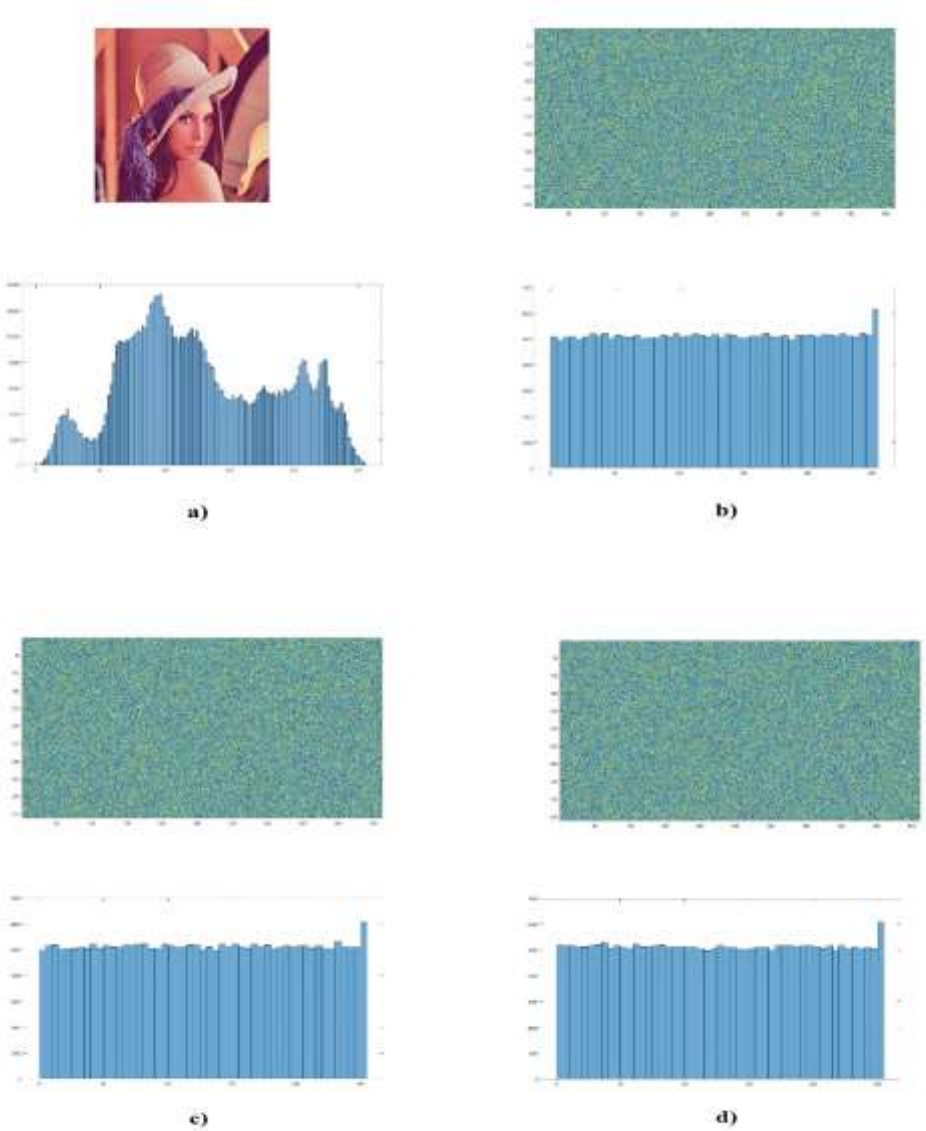
Şifreleme işlemleri için DES, 3DES ve AES algoritmaları ayrı ayrı örnek resimler üzerine uygulanmıştır. Şifreleme uygulamaları Python platformunda gerçekleştirilmiştir.

Tablo 3. Şifreleme Algoritmalarının Anahtar Uzunlukları

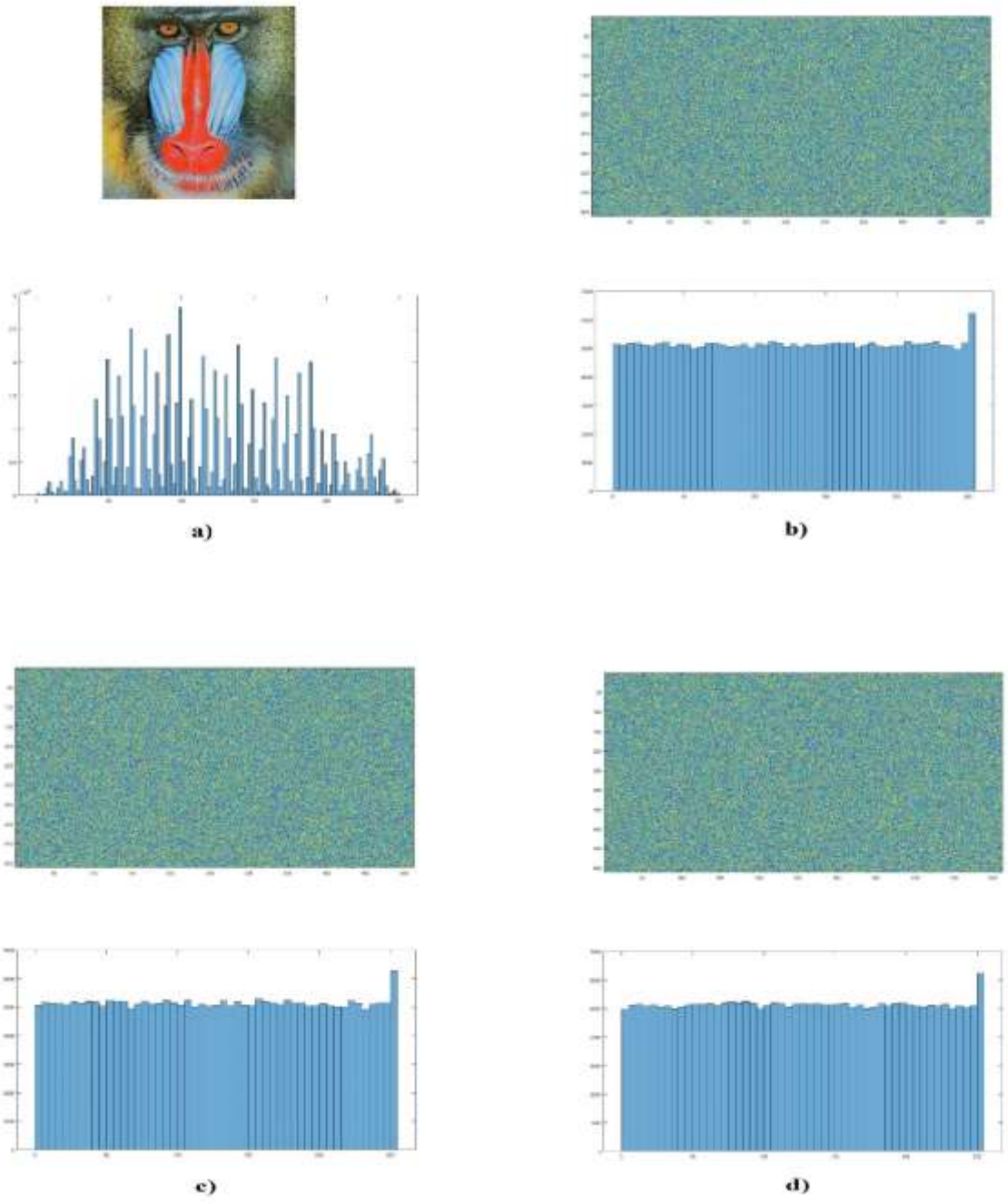
Şifreleme Algoritması	Anahtar Uzunluğu
DES	64 bit
3DES	192 bit
AES	128/192/256 bit

Kullanılacak olan şifreleme algoritmalarının anahtar uzunlukları Tablo 3'te verilmiştir. AES için 128 bit, 192 bit ya da 256 bitlik anahtarların kullanılması uygundur.

Şifreleme işleminin uygulandığı iki örnek 512x512 boyutunda renkli görüntü işleme standart resmi aşağıda verilmiştir.



Şekil 2. Görüntü Şifreleme I: **a)** Orijinal resim ve histogramı **b)** DES ile şifrelenmiş görüntü ve histogramı **c)** 3DES ile şifrelenmiş görüntü ve histogramı **d)** AES ile şifrelenmiş görüntü ve histogramı



Şekil 3. Görüntü Şifreleme I: **a)** Orijinal resim ve histogramı **b)** DES ile şifrelenmiş görüntü ve histogramı **c)** 3DES ile şifrelenmiş görüntü ve histogramı **d)** AES ile şifrelenmiş görüntü ve histogramı

6. Sonuçlar

Örnek görüntü işleme resimlerinden iki tanesine uygulanan DES, 3DES ve AES şifreleme algoritmaları resimleri başarıyla şifrelemiş ve sonrasında yine başarılı bir şekilde herhangi bir veri kaybı olmadan orijinal resimleri geri döndürmüştür.

Tablo 5. Şifreleme Algoritmalarının Çalışma Süreleri

Örnek Resim	Şifreleme Algoritması	Şifreleme Süresi (ms)
Lena 512x512	DES	Şifreleme: 19.68 ms Şifre Çözme: 14.11 ms
	3DES	Şifreleme: 42.05 ms Şifre Çözme: 36.63 ms
	AES	Şifreleme: 13.71 ms Şifre Çözme: 6.49 ms
Mandrill 512x512	DES	Şifreleme: 10.58 ms Şifre Çözme: 5.83 ms
	3DES	Şifreleme: 18.03 ms Şifre Çözme: 11.94 ms
	AES	Şifreleme: 8.03 ms Şifre Çözme: 3.27 ms

Tablo 5'te tüm algoritmalar için şifreleme ve şifre çözme süreleri gösterilmiştir. Kodların işlenme süreleri bilgisayar kaynaklarına ve anlık kullanım miktarına bağlı olarak kodların farklı zamanlarda çalıştırılması sonucunda %1'den daha az değişkenlik gösterdiği tespit edilmiştir.

7.Değerlendirme

Bu çalışmada görüntü şifreleme uygulamalarında kullanılmak üzere LCG ve Trivium algoritması tabanlı bir sözde rastgele sayı üreteci tasarımı yapılmıştır. Elde edilen rastgele sayı üreteci tasarımı istatistiksel NIST 800-22 testiyle değerlendirilerek rastgele değerler elde edildiği ispatlanmıştır. Tasarlanan LCG çıkışında istatistiksel testlerde başarısız olan sayı dizisi Trivium algoritmasıyla uygulanan son işlem sayesinde tüm testlerden başarıyla geçmeyi başarmıştır.

Elde edilen rastgele sayılar şifreleme algoritmaları için farklı uzunluklardaki anahtarlar olarak kullanılarak şifreleme işlemi yapılmıştır. DES, 3DES ve AES için yapılan uygulamalar sonucunda uygulanan sistem için en hızlı şifreleme ve şifre çözme süreleri sırasıyla AES, DES ve 3DES olarak tespit edilmiştir.

Her iki görüntünün farklı histogram grafikleri olmasına karşın şifrelenmiş tüm verilerin uniform dağılımlı histogramlara sahip olduğu görülmüştür. Böylece sadece şifrelenmiş verileri analiz ederek uygun şifre çözme algoritmasını ve şifreleme anahtarını bilmeden gizli verilere ulaşılmasının mümkün olmadığı belirlenmiştir.

Teşekkür

Bu çalışma Taha Etem'in doktora tezinden hazırlanmıştır.

Kaynaklar

[1] Coskun S, Pehlivan I, Akgul A, GÜREVİN B. A new computer-controlled platform for ADC-based true random number generator and its applications. Turkish J. Electr. Eng. Comput. Sci. 2019. pp. 847–860,.

[2] Kaya T. A true random number generator based on a Chua and RO-PUF: design, implementation and statistical analysis. Analog Integr. Circuits Signal Process. 2019. 2.

[3] Tuna M, Karthikeyan A, Rajagopal K, Alcin M, Koyuncu I. Hyperjerk multiscroll oscillators with megastability: Analysis, FPGA implementation and a novel ANN-ring-based True Random Number Generator. AEU - Int. J. Electron. Commun. 2019. 112.

[4] Akgul A. Yeni Kaotik Sistemler ile Rasgele Sayi Üreteci Tasarimi Ve Çoklu-Ortam Verilerinin Yüksek Güvenlikli Şifrelenmesi. Sakarya Üniversitesi Fen Bilimleri Enstitüsü. 2015.

[5] Arslan Tuncer S. Real-Time Random Number Generation with Ring Oscillator Based Double Physically Unclonable Function. *J. Microelectron. Electron. Components Mater.* 2018. 48(2). pp. 121–128.

[6] Akgul A, Kacar S, Pehlivan I. An Audio Data Encryption with Single and Double Dimension Discrete-Time Chaotic Systems. Tojsat. 2015. 5(3). pp. 14–23.

[7] Koyuncu I, Ozcerit A, Pehlivan I, Avaroglu E. Design and implementation of chaos based true random number generator on FPGA. IEEE Signal Processing and Communications Applications Conference, 2014, pp. 236–239.

[8] Tuncer T. Implementation of duplicate TRNG on FPGA by using two different randomness source. Elektron. ir Elektrotechnika 2015. 21(4). pp. 35–39.

[9] Buchovecká S, Lórencz R, Kodýtek F, Buček J.True random number generator based on ring oscillator PUF circuit, Microprocess. Microsyst. 2017. 53. pp. 33–41.
[10] Petchlert B, Hasegawa H. Using a low-cost electroencephalogram (EEG) directly as random number generator. Proc. - 2014 IIAI 3rd Int. Conf. Adv. Appl. Informatics. 2014. pp. 470–474.
[11] Avaroglu E, Koyuncu I, Özer AB, Turk M. Hybrid pseudo-random number generator for cryptographic systems. Nonlinear Dyn. 2015. 82(1). pp. 239–248.
[12] Chen X. Modeling Random Telegraph Noise as a Randomness Source and its Application in True Random Number Generation. IEEE Trans. Comput. Des. Integr. Circuits Syst. 2016. 35(9). pp. 1435–1448.
[13] Neugebauer F, Polian I, Hayes JP. S-box-based random number generation for stochastic computing. Microprocess. Microsyst. 2018. 61(1). pp. 316–326.
[14] Avaroglu E. Pseudorandom number generator based on Arnold cat map and statistical analysis. Turkish J. Electr. Eng. Comput. Sci. 2017. 25(1). pp. 633–643.
[15] Arslan Tuncer S, Kaya T. True Random Number Generation from Bioelectrical and Physical Signals. Comput. Math. Methods Med. 2018. 2018. pp. 1–11.
[16] Akgul A, Arslan C, Aricioglu B. Desıgn Of An Interface For Random Number. 2017. 1(1). pp. 1–18.
[17] Moosavi SR, Nigussie E, Virtanen S, Isoaho J. Cryptographic key generation using ECG signal. 2017 14th IEEE Annu. Consum. Commun. Netw. Conf. CCNC. 2017. pp. 1024–1031.
[18] Etem T, Kaya T. A novel True Random Bit Generator design for image encryption. Phys. A Stat. Mech. its Appl. 2020. 540.
[19] Devi A, Sharma A, Rangra A. A Review on DES, AES and Blowfish for Image Encryption & Decryption. Int. J. Eng. Comput. Sci. 2015. 4(6). pp. 12646–12651.
[20] O'neill ME. A PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation. ACM Trans. Math. 2017. 217. pp. 1–46.
[21] Bhattacharjee B. Maity K. Das K. A Search for Good Pseudo-random Number Generators : Survey and Empirical Studies. 2018.
[22] Kaya T. Memristor and Trivium-based true random number generator. Phys. A Stat. Mech. its Appl. 2020. 124071.
[23] Chen X, Zhang Y, Zhang G, Zhang Y. Evaluation of ECG random number generator for wireless body sensor networks security. 2012 5th Int. Conf. Biomed. Eng. Informatics. 2012. pp. 1308–1311.
[24] Sathiyamurthi P, Ramakrishnan S. Speech encryption using chaotic shift keying for secured speech communication. Eurasip J. Audio, Speech, Music Process. 2017. 2017.
[25] Akgul A, Moroz I, Pehlivan I, Vaidyanathan S. A new four-scroll chaotic attractor and its engineering applications. Optik. 2016. 127. pp. 5491–5499.