İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

AES UYGULAMASI'NIN FPGA GERÇEKLEMELERİNE KARŞI GÜÇ ANALİZİ SALDIRISI

YÜKSEK LİSANS TEZİ

Müh. Hakan KAYIŞ

Anabilim Dalı : ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ

Programı : ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ

Tez Danışmanı: Yard. Doç.Dr. Sıddıka Berna ÖRS YALÇIN

HAZİRAN 2006

ÖNSÖZ

Çağımızda bilgi, en önemli güç konumundadır, bilgiye sahip olanlar diğerlerinden daha üstün konumdadırlar, bu gerçek ise bilgiye sahip olma çabasını inanılmaz boyutlara çıkartmaktadır. Bilgi elde edimi , her türlü yoldan yapılmaktadır, kendiniz uğraşarak , bularak veya başkasının bulduğunu , bildiğini değişik yollardan elde ederek.

Bu nedenle bilgiye sahip olmak kadar onu korumak , istenilen yerlere ulaştırmak ve istenilen amaca yönelik kullanılabilmesini sağlamak da bilginin kendisi kadar önemli hale gelmektedir.

Bu çaba, neredeyse bilgi tarihi kadar eskidir ve kriptografi biliminin temellerini oluşturmaktadır. Tarihte Romalıların Sezar şifrelemesinden , günümüzün gelişmiş değişik şifreleme metod ve algoritmalarına kadar bu çaba sürmüştür, görünen o ki bilginin boyutu ile de katlanarak artacaktır. Bir taraf bilgiye sahip olacak ve saklayacak , birileri de saklananı elde etmeye, kendine göre değişik amaçlarla kullanmaya çalışacaktır. Öyle ki iyi ve kötü dahi burada subjektif kalacaktır. Örneğin, kendi ülkesi için başka bir ülkenin haberleşme şifrelerini çözmeye çalışan kişi , ülkesi için “iyi”, diğer taraf için “kötü” olacaktır.

Kısacası şifreleme ve şifreleri çözme çabası , her zaman devam edecektir, daha gelişmiş şifreleme teknikleri çıkacak , bunlar da farklı yöntemlerle kırılmaya çalışılacaktır; önemli olan bu yarışta kazanmak değil, daima güncel olabilmektir, çünkü “kırılamayacak algoritma yoktur”. Bu sanıyorum Hint destanlarındaki iyi ile kötünün mücadelesi gibidir, asla bitmeyecektir.

Bizim tez çalışmamız , bir standart olarak kabul gören AES algoritmasını ve akıllı kart gerçeklemesine yönelik bir saldırı olan farksal güç analizi saldırısını incelemektir. Gerçeklemede önlem alınmadığı takdirde AES algoritmasının ciddi bir

zayıflığı olduğu tez çalışması kapsamında ayrıntılı olarak incelenmiştir; tasarımcılar bir güvenlik açığı ile karşı karşıya kalmamak için önlem almalıdırlar.

Tez çalışmalarım sırasındaki özverili yardımlarından dolayı danışman hocam Yrd. Doç. Dr. Sıddıka Berna Örs Yalçın'a teşekkür ederim. Bu yoğun dönemde bana manevi desteğini esirgemeyen eşim Ayça Kayış'a şükran borçluyum.

Haziran 2006 **Hakan KAYIŞ**

İÇİNDEKİLER

ÖNSÖZ **II**
KISALTMALAR **VI**
TABLO LİSTESİ **VII**
ŞEKİL LİSTESİ **VIII**
SEMBOL LİSTESİ **IX**
ÖZET **X**
SUMMARY **XII**

1. GİRİŞ **1**
1.1. Giriş ve Çalışmanın Amacı 1

2. SONLU UZAY ARİTMETİĞİ **2**
2.1. Giriş 2
2.2. Bayt Gösterim Şekilleri 2
2.2.1. İkilik Gösterim 3
2.2.2. Onluk Gösterim 3
2.2.3. Onaltılık Gösterim 3
2.2.4. Polinom Gösterim 4
2.3. Polinom Toplama işlemi 5
2.4. Polinom Çarpım işlemi 6
2.5. Polinom Bölme işlemi 7

3. AES - GELİŞMİŞ ŞİFRELEME STANDARDI **10**
3.1. Giriş 10
3.2. Rijndael Algoritmasının Genel Yapısı 11
3.3. Durum Atanması 14
3.4. Bayt yer değiştirme 14
3.5. Satır Öteleme İşlemi 17
3.6. Sütun karıştırma işlemi 19
3.7. Tur sonu XOR işlemi 22
3.8. Anahtar liste Mekanizması 23
3.8.1. Kelime Döndürme işlemi 26
3.9. AES şifreleme adımları örneği 31

4. GÜÇ ANALİZİ SALDIRISI **33**
4.1. Giriş 33
4.2. Basit güç analizi saldırısı 34
4.3. Farksal güç analizi saldırısı 35
4.4. Saldırı için Öngörülen Algoritma 36
4.4.1 Saldırı Amaçlı Model 1 36
4.4.2 Saldırı Amaçlı Model 2 38
4.5. Teorik Altyapı 39

4.6. Ölçüm İşlemleri 40
4.7. Saldırı Öncesi Benzetim Çalışması 42
4.7.1. Giriş dosyaları 42
4.7.2. Yazılan Programlar 43
4.7.2.1 AES Şifreleme 43
4.7.2.2 Şifre Çözme 44
4.7.2.3 Bellek İçerik Gösterimi 44
4.7.2.4 Durum Geçişi Analizi 47
4.7.2.5 Saldırı Noktası Durum Geçiş Analizi Sonuç Çıktıları 49
4.7.2.6 Tüm Bellek Durum Geçiş Analizi Sonuç Çıktıları 50
4.7.2.7 Saldırı Noktası Durum Bit Analizi Sonuç Çıktıları 50
4.8. Farksal Güç analizi Saldırısı, Verilerin Kullanımı 52
4.9. Farksal Güç analizi Saldırısı, Ölçüm Verileri Kullanımı 60

5. SONUÇLAR VE TARTIŞMA 61

KAYNAKLAR 62

ÖZGEÇMİŞ 64

KISALTMALAR

AES	**:** Advanced Encryption Standart
DES	**:** Data Encryption Standart
NIST	**:** National Institute of Standards and Technology
SPN	**:** Substitution-Permutation Network
TDES	**:** Triple Data Encryption Standart
SPA	**:** Simple Power Analysis
DPA	**:** Differential Power Analysis
XOR	**:** Exclusive OR
GF	**:** Galois Field
CMOS	**:** Complementary Metal Oxide Semiconductor
RCON	**:** Round constant
FPGA	**:** Field Programmable Gate Array
NSA	**:** National Security Agency

TABLO LİSTESİ

		Sayfa No
Tablo 2.1 :	XOR işlemi sonuçları	5
Tablo 3.1 :	Anahtar uzunluğu – tur sayısı ilişkisi	11
Tablo 3.2 :	Byte Yer Değiştirme tablosu– S kutusu[xy] (onaltılık düzende)	16
Tablo 3.3 :	rcon değerleri	25

ŞEKİL LİSTESİ

		Sayfa No
Şekil 2.1	**:** ”Klasik” polinom toplama işlemi	5
Şekil 2.2	**:** İkilik polinom toplama işlemi	6
Şekil 2.3	**:** ”Klasik” polinom çarpımı	6
Şekil 2.4	**:** İkilik polinom çarpım işlemi	7
Şekil 2.5	**:** ”Klasik” polinom bölme işlemi	8
Şekil 2.6	**:** İkilik düzende bölme işlemi	9
Şekil 3.1	**:** AES turları	12
Şekil 3.2	**:** Tüm AES resmi	13
Şekil 3.3	**:** Durum Ataması işlemi	14
Şekil 3.4	**:** Bayt yer değiştirme işlemi	15
Şekil 3.5	**:** Satır öteleme işlemi	18
Şekil 3.6	**:** Sütun öteleme işlemi	20
Şekil 3.7	**:** Tur sonu değeri ile XOR işlemi	22
Şekil 3.8	**:** Anahtar üretme fonksiyonu	24
Şekil 3.9	**:** Anahtar Üretimi	25
Şekil 3.10	**:** Kelime döndürme işlemi	26
Şekil 3.11	**:** AES işlem değerleri	32
Şekil 4.1	**:** Yan kanal çıkışları	33
Şekil 4.2	**:** Saldırı amaçlı model-1’e ait algoritma	38
Şekil 4.3	**:** Saldırı amaçlı model-2’e ait algoritma	39
Şekil 4.4	**:** Ölçüm düzeneği	42
Şekil 4.5	**:** Devre yapısı için geliştirilen algoritma	45
Şekil 4.6	**:** AES algoritması için öngörülen register düzeneği resmi	53
Şekil 4.7	**:** R2, Model 1 için Anahtarın En yüksek Anlamlı Bitinin korelasyon değerleri	55
Şekil 4.8	**:** R2, Model 1 için Kullanılan mesaj sayısının korelasyon bağlantısı	56
Şekil 4.9	**:** R3, Model 1 için Anahtarın En yüksek Anlamlı Bitinin korelasyon değerleri	57
Şekil 4.10	**:** R3, Model 1 için kullanılan mesaj sayısının korelasyon değerleri	57
Şekil 4.11	**:** R2, Model 2 için Anahtarın En yüksek Anlamlı Bitinin korelasyon değerleri	58
Şekil 4.12	**:** R2, Model 2 için kullanılan mesaj sayısının korelasyon değerleri	58
Şekil 4.13	**:** R3, Model 2 için Anahtarın En yüksek Anlamlı Bitinin korelasyon değerleri	59
Şekil 4.14	**:** R2, Model 2 için kullanılan mesaj sayısının korelasyon değerleri	59
Şekil 4.15	**:** Benzetim Verisi ile gerçek ölçümün birleştirilmesi ve korelasyon eldesi	60

SEMBOL LİSTESİ

R1	**:** Register 1
R2	**:** Register 2
R3	**:** Register 3
R4	**:** Register 4
R5	**:** Register 5
S	**:** Durum
GF	**:** Galois Field, Galois Uzayı
F	**:** AES adımlarının herbiri
K1 ... 10	**:** AES Anahtar dizisinden tur değerleri
$\mathbf{S_1 ... \ S_{15}}$	**:** AES Durum bytelarının herbiri
$\mathbf{b_i}$	**:** işlem yapılan bit değerleri
$\mathbf{m_i}$	**:** Ölçüm matrisi değerleri
$\mathbf{c_{i,j}}$	**:** Tahmin için gerekli ölçüm miktarı değeri

AES UYGULAMASI'NIN FPGA GERÇEKLEMELERİNE KARŞI GÜÇ ANALİZİ SALDIRISI

ÖZET

AES algoritmasının FPGA kart gerçeklemeleri , çalışma sırasında yan kanal etkisi olarak değişik çıktılar üretirler; ısı, elektromanyetik radyasyon, güç harcaması gibi. Bu çıktılar önlem alınmadığı takdirde şifreleme algoritmasına ve anahtarına ait bilgiler içerebilirler.

Tez kapsamında 128 bit anahtar kullanılan AES algoritması ve bu algoritmanın akıllı kart gerçeklemesine yönelik farksal güç analizi incelenmiştir.Günümüzde AES algoritması ve donanım gerçeklemeleri yoğun olarak kullanılmaktadır. Farksal güç analizi ilk olarak Paul KOCHER tarafından 1998 yılında gündeme getirilmiştir.

Farksal güç analizinin temel prensibi , gerçeklemenin yapıldığı akıllı kartta kullanılan registerların durum değiştirme anında harcadıkları gücün AES algoritmasının temel dayanağı olan şifreleme anahtarına ait bilgi içermesidir.

Algoritmanın gerçeklemesi sırasında fonksiyonlar arası geçişler registerlarda tutulmaktadır. Registerlarda saklanan bit değerleri, değişimlerini 4 farklı şekilde yapabilirler ; 0 > 0, 0 > 1, 1 > 0, 1 > 1 şeklinde geçişler mümkündür. Güç harcaması 0 > 1 geçişinde diğer durum değiştirmelerine göre çok daha fazladır ve diğerleri bunun yanında ihmal edilebilir.

Bu işlem için 20000 adet giriş değeri alınmış ve yazılan C programları ile öngörülen model kapsamında güç harcaması analizi yapılmıştır. Belli tur ve saat dilimindeki güç harcaması , anahtarın sadece 1 byte değeri değiştirilerek incelenmiş ve bir korelasyon analizi yapılmıştır. Bu veriler gerçek ölçüm verileri ile beraber kullanılacaktır.

Güç harcamasının bu karakteristiği, bize yapılan ölçümler ile AES anahtarının korelasyon analizi yardımı ile eldesini mümkün kılar. Ölçüm ortamının kalitesine

göre belli sayıda güç ölçümü , bir bilgisayarda yapılan öngörülen model güç harcama verileri ile beraber kullanılarak AES anahtarına ait 8 bit elde edilir. Analiz, 8 bitlik anahtar parçasını elde etmeye yöneliktir ve öngörülü model için hazırlanan veri ile sırayla 128 bitlik anahtarın tamamı elde edilebilir.

DIFFERENTIAL POWER ANALYSIS ATTACK AGAINST FPGA IMPLEMENTATION OF AES ALGORITHM

SUMMARY

Implementations of the AES algorithm on FPGA cards produce side channel effects during runtime; like heat, electromagnetic radiation, power consumption .If not avoided in the design phase, these side channel effects may have information about the algorithm and the AES encryption key .

In our thesis work, AES with 128 bits encryption key and differential power analysis (DPA) against the FPGA implementation of this algorithm is studied. Nowadays, the AES algorithm and hardware implementations are mostly used. Differential power analysis against the hardware implementations are firstly pronounced by Paul KOCHER in 1998 .

The basic principal of the DPA depends on the power consumption of the toggle operation of the registers used in the implementation of FPGA; this side effect may have information about one of the main part of the algorithm, the AES key.

In the implementation, the transition results between the AES function blocks are stored in registers. The transition of the bits stored in registers may vary in 4 different states; as given 0 > 0, 0 > 1, 1 > 0, 1 > 1 . The power consumption is much more than the others in 0 > 1 transition and the others can be negligible according to this one.

For the analysis , 20000 input text is prepared and analyzers coded in C are used for DPA for the assumed models. The power consumption is calculated in the suggested round and clock cycle and correlation analysis is done. After all, the datas gathered in this phase will be used with the measured data from the FPGA card.

The characteristics of the power consumption makes it feasible to get the AES key with the correlation analysis. Depending on the quality of the measurement system,

an amount of the measurement data, an analysis of the DPA for the suggested model in a PC , the 8 bits of the AES key can be found. Analysis is for 8 bits and after all, by repeating this procedure , all 128 bits of the key can be found.

1. GİRİŞ

1.1 Giriş ve Çalışmanın Amacı

Tez kapsamında 128 bit anahtar kullanılarak 128 bitlik girişlerin şifrelendiği algoritma incelenmiştir. AES Algoritmasına Farksal Güç Analizi saldırısı incelenmesi 3 farklı bölümde gerçeklendi.

2. bölümde konuya ait matematiksel altyapı ve kullanılan aritmetik işlemleri örnekler de verilerek ayrıntılı olarak incelendi.

3. bölümde AES algoritmasının ayrıntılı olarak incelenmesi yapıldı. Algoritmaya ait anahtar üretme fonksiyonu, bayt yer değiştirme, satır öteleme, sütun karıştırma , tur sonu XOR işlemi fonksiyonları şekillerle anlatıldı. Bu fonksiyonların C gerçeklemesinin kod bölümleri de açıklamalar eklendi. Bölüm sonunda verilen bir şifre ve giriş için 10 tur boyunca tüm işlem sonuçlarını içeren örnek verildi.

Son bölümde ise yan kanal saldırılarına ait basit bir açıklama yapıldıktan sonra belirlediğimiz 2 tane gerçekleme modeline ait algoritmalar için analiz amaçlı yazılan programlar ayrıntılı olarak anlatıldı, analizler sonucunda oluşan korelasyon sonuçlarına ait grafikler verildi , inceleme sonucunda 1. turun 2. saat diliminde R2 belleği , yani bayt yer değiştirme işlem sonucu belleği ve 3. saat diliminde saldırı R3 belleği, yani satır öteleme belleği içerik değişimlerinin uygun olduğu görüldü.

Analiz programlarının öngörülen modeller için korumasız yapılacak akıllı kart gerçeklemelerinin saldırıya açık oldukları tespit edildi ve sonuç olarak gösterildi. Yazılan programlar ve programlar sonucunda oluşturulan analiz verileri, ekler altında bir CD’de verilmiştir.

2. SONLU UZAY ARİTMETİĞİ

2.1 Giriş

Sonlu uzay aritmetiği, sonlu sayıda eleman içeren sayı uzayı içerisinde, tanımlı tüm işlemlerin yine aynı uzayda sonuçlar ürettiği aritmetiktir. Bu özelliği ile normal aritmetikten ayrılmaktadır. Yine bu özelliğinden dolayı da birçok alanda uygulama bulmaktadır, kriptografi ve Rijndael şifreleme algoritması da bunların arasındadır.

Bu bölümde bir baytın değişik gösterilim şekilleri gösterilmiştir ve sonlu uzayda temel aritmetik işlemleri anlatılmaktadır. Sonlu uzay kavramı aynı zamanda Galois Uzayı olarak da tanımlanır. En basit ifade ile sonlu sayıda eleman içeren uzay şeklinde tanımlanabilir.Örnek Galois sonlu uzayı,

$GF(2^8)$, $2^8 = 256$

adet farklı sayı içermektedir. Bu sayılar bir bayt ile gösterilebilen (0 . . . 255) arasındaki sayılardır. Rijndael şifreleme algoritmasında karakteristiği 2 olan Galois Uzayı ($GF(2^8)$) kullanılmaktadır.

Devam eden bölümlerde anlatılan işlemlerde sonlu uzaya ait iki elemanın toplama veya çarpması anlatılmaktadır ve sonuç yine sonlu uzay içinde bir elemandır. Özel XOR ve kalan işlemleri, işlem sonucunun sonlu uzayda olmasını sağlamaktadır **[3]**.

2.2 Bayt Gösterilim Şekilleri

Aşağıdaki dört bölümde sonlu uzaydaki bir elemana ait farklı gösterilim şekilleri verilmiştir. Ayrıca birer örnekte her gösterilim için eklenmiştir.

2.2.1 İkilik Gösterilim

Bir bayt , 8 bit içermektedir. Aşağıda ikilik gösterilime ait örnek vardır.

10100011_2

2.2.2 Onluk Gösterilim

Onluk gösterilim, ikilik sayının tüm bitlerinin karşı düşen ikinin üs değeri ile çarpılıp toplanması ile elde edilen sayıdır. Ekteki örnekte ayrıntılı gösterilim mevcuttur:

$$1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 2^7 + 2^5 + 2^1 + 2^0 \quad \textbf{(1)}$$

$$= 128 + 32 + 2 + 1$$

$$= 163_{10}$$

2.2.3 Onaltılık Gösterilim

0 ile 15 arası rakamlar dört bit ile gösterilebilirler . Ancak 10 ile 15 arası rakamlar (0 . . . 9) arası rakamlarla gösterilemezler ve bu nedenle A ile F arası harfler onaltılık düzende bu rakamları göstermek için kullanılırlar.

$0000_2 = 0_{10} = 0_{16}$

$0001_2 = 1_{10} = 1_{16}$

$0010_2 = 2_{10} = 2_{16}$

$0011_2 = 3_{10} = 3_{16}$

$0100_2 = 4_{10} = 4_{16}$

$0101_2 = 5_{10} = 5_{16}$

$0110_2 = 6_{10} = 6_{16}$

$0111_2 = 7_{10} = 7_{16}$

$1000_2 = 8_{10} = 8_{16}$

$1001_2 = 9_{10} = 9_{16}$

$1010_2 = 10_{10} = \boldsymbol{A}_{16}$

$1011_2 = 11_{10} = \boldsymbol{B}_{16}$

$1100_2 = 12_{10} = \boldsymbol{C}_{16}$

$1101_2 = 13_{10} = \boldsymbol{D}_{16}$

$1110_2 = 14_{10} = \boldsymbol{E}_{16}$

$1111_2 = 15_{10} = \boldsymbol{F}_{16}$

İkilik gösterilimden onaltılık gösterilime geçmek için bir bayt iki adet dörtlüye bölünür ve her iki dörtlü, karşı düşen onaltılık karşılığı ile gösterilir. Aşağıdaki örnekte ayrıntılı gösterilmiştir:

$$10100011_b = \underbrace{1010}_{\boldsymbol{A}_h} \ \underbrace{0011}_{3_h} = \boldsymbol{A}3_h$$

Onaltılık düzenden onluk düzene geçmek için sol hane 16 sayısı ile çarpılır ve sağ hane 1 ile çarpılır ve her iki çarpım sonucu toplanır:

$$A3_h = A \cdot 2^4 + 3 \cdot 2^0 = 10 \cdot 16 + 3 \cdot 1 = 160 + 3 = 163_d$$

2.2.4 Polinom Gösterilim

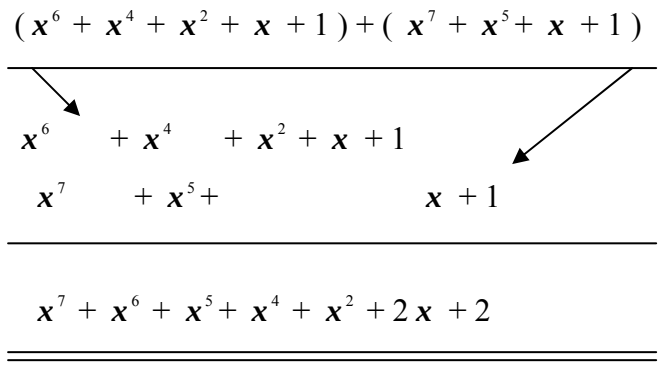
Bir baytın polinom gösterilimi ikilik sistemden onluk sisteme geçmek için kullanılan (1) denklemine çok benzer. Bu denklemde 2 sayısı yerine x konularak polinom gösterilim elde edilir:

$$1 \cdot \boldsymbol{x}^7 + 0 \cdot \boldsymbol{x}^6 + 1 \cdot \boldsymbol{x}^5 + 0 \cdot \boldsymbol{x}^4 + 0 \cdot \boldsymbol{x}^3 + 0 \cdot \boldsymbol{x}^2 + 1 \cdot \boldsymbol{x}^1 + 1 \cdot \boldsymbol{x}^0 = \boldsymbol{x}^7 + \boldsymbol{x}^5 + \boldsymbol{x} + 1$$

Unutulmaması gereken en önemli nokta polinom gösterilimde $GF(2^8)$ sonlu uzayında katsayıların 1 değerini alabileceğidir. (ve 0 değeri tabiki).

2.3 Polinom toplama işlemi

İki adet polinomun toplaması işlemi aynı üstel değere sahip x değerlerinin katsayıların toplanması şeklindedir ve Şekil 1'de gösterilmiştir.



Şekil 2.1: "Klasik" polinom toplama işlemi

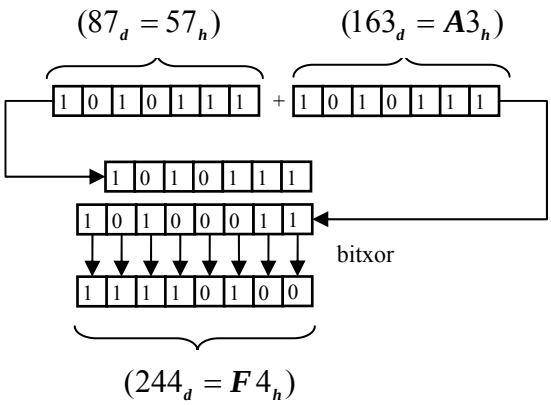
Bu işlem sonucunda sonuç polinomun bazı katsayıları 0 veya 1 olmayabilir. (Şekil2.1'de 2x ve 2 değerinde olduğu gibi), bu nedenle sonuç polinomu bir bayt gösterilimi olamayacaktır.

Toplama sonuç polinomunda sadece ikilik düzende katsayıların olmasını sağlamak için Tablo 2.1'de sonuç kümesi gösterilmiş olan XOR işlemi kullanılacaktır. İki adet 1 rakamının XOR işlemi 2 olmadığından sonuç polinomda 2 değerinde katsayı olmayacaktır.

Tablo 2.1: XOR işlemi sonuçları

X	y	x xor y
0	0	0
0	1	1
1	0	1
1	1	0

Şekil 2.2'de 2 adet ikilik düzende sayının toplama işlemi ve sonuç bayt değeri gösterilmiştir:



Şekil 2.2: İkilik polinom toplama işlemi

Sonuç polinomu

$$11110100_b \quad = \quad 244_d \quad = \quad F4_h \; = \; x^7 + x^6 + x^5 + x^4 + x^2$$

Şekil 2.2'deki değeri gösterir , burada 2x ve 2 değerleri çıkarılmıştır.

2.4 Polinom Çarpım İşlemi

İki adet polinomun çarpımı, aynı üstel değere sahip x'lerin katsayılarının toplanması ile gerçeklenir. Bu klasik polinom çarpımıdır.

$$(x^6 + x^4 + x^2 + x + 1) * (x^7 + x^5 + x + 1)$$

$$\begin{array}{rrrrrrrrrrrrrr}
 & & & & & x^7 & + x^5 & + & & & x + 1 & + \\
 & & & & x^8 & + x^6 & & & & + x^2 + x & & + \\
 & & & x^9 & + x^7 & & & & + x^3 + x^2 & & & + \\
 & & x^{11} & + x^9 & & & + x^5 + x^4 & & & & & + \\
 & x^{13} & + x^{11} & & & + x^7 + x^6 & & & & & & + \\
\hline
 & x^{13} & +2x^{11} & +2x^9 & +x^8 & +3x^7 & +2x^6 & +2x^5 & +x^4 & +x^3 & +2x^2 & +2x & +1
\end{array}$$

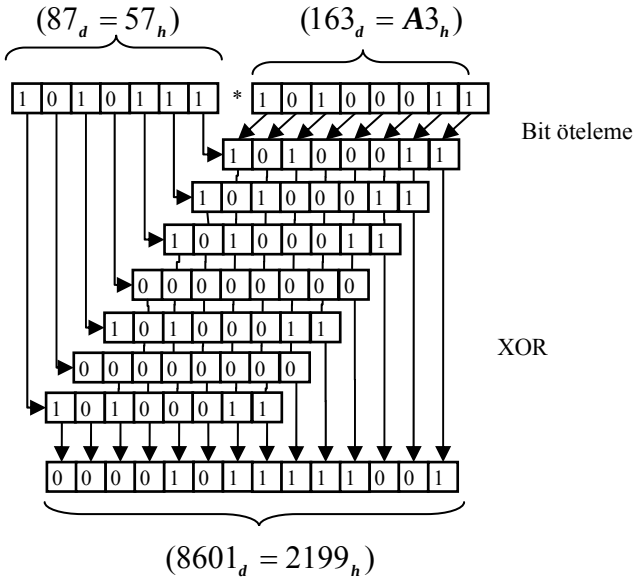
Şekil 2.3: "Klasik" polinom çarpımı

Tekrar klasik polinom çarpımında bazı katsayılar 1 veya 0 değerinden farklı değerler, (2 ve hatta 3 değerleri) almıştır. Bu katsayılar sonlu uzay işleminde bu değerleri almamalıdırlar ve bu nedenle düzenlenmesi gereklidir. Bu nedenle genelleştirilmiş

XOR işlemi her katsayı için gözönüne alınmalıdır ve her tek sayı 1 değerine çevrilmelidir.

$$x^{13} + x^{8} + x^{7} + x^{4} + x^{3} + 1$$

Şekil 2.4'de bit seviyesinde çarpma işlemi verilmiştir. İkilik sayılardan bir tanesi diğerinin bit değerine göre her bit için ötelenir. Eğer sıradaki sayı 1 değerinde ise diğer baytın hepsi, 0 ise sadece 0 değerleri konulur. Sırayla her bit için öteleme işlemi bittikten sonra aynı sıradaki tüm bitler xor işlemine tabi tutulurlar ve sonuç bit serisi elde edilir.



Şekil 2.4: İkilik polinom çarpım işlemi

Ancak sonuçta görüldüğü üzere polinomumuzun derecesi 7 değerinden büyüktür ve bayt olarak gösterilemez; sonuç polinomunun GF(2^8) içinde değildir. Sonucun Galois sonlu uzayında olması için indirgenmesi gerekmektedir.

2.5 Polinom Bölme işlemi

Klasik bölme işlemi Şekil 2.5'de gösterilmiştir:

$$(x^{13} + x^8 + x^7 + x^4 + x^3 + 1) : (x^8 + x^4 + x^3 + x + 1) = x^5 - x$$

$$- \quad (x^{13} + x^9 + x^8 + x^6 + x^5)$$

$$-x^9 + x^7 - x^6 - x^5 + x^4 + x^3 + 1$$

$$- \quad (-x^9 - x^5 - x^4 - x^2 - x)$$

$$x^7 - x^6 + 2x^4 + x^3 + x^2 + x + 1$$

Şekil 2.5: "Klasik" polinom bölme işlemi

Bölünen polinomun en büyük üstel değeri (x^{13}) , bölenin en yüksek üstel değeri (x^8) ile bölünür ve sonuç (x^5) olarak elde edilir. Daha sonra (x^5) bölenin tüm değerleri ile çarpılarak bölünen polinomunda çıkarılır ve yeni bir bölünen polinomu elde edilir:

$$(-x^9 + x^7 - x^6 - x^5 + x^4 + x^3 + 1)$$

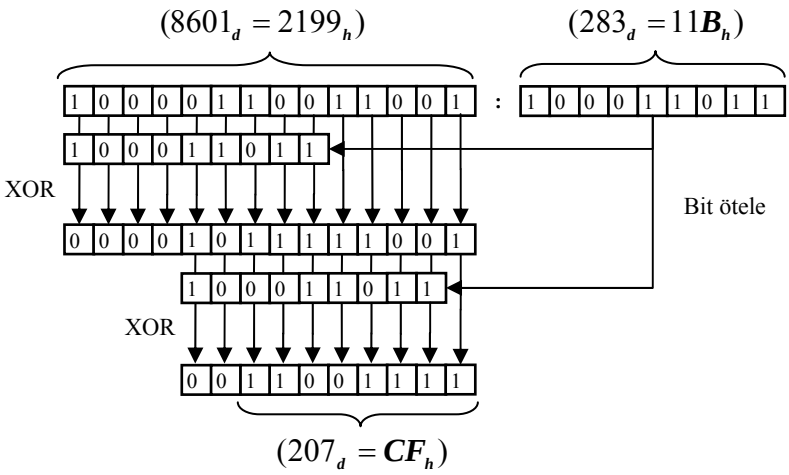
Daha sonra aynı işlemler tekrar yapılır ve bu işlem bölünen polinomunun en yüksek üstel değeri bölenin en yüksek üstel değerinden küçük değerde oluncaya kadar devam eder. En sonda kalan bölünen, işlem sonucunda kalan polinomunu oluşturur:

$$(x^7 - x^6 + 2x^4 + x^3 + x^2 + x + 1)$$

Sonuç polinomuna genel xor işlemi uygulanarak (tek katsayılar > 1 , çift katsayılar > 0 işlemi) bayt gösteriliminde sonuç elde edilir:

$$(x^7 + x^6 + x^3 + x^2 + x + 1)$$

Bit seviyesinde işlemler Şekil 2.6 'da gösterilmiştir:



Şekil 2.6: İkilik düzende bölme işlemi

3. AES - GELİŞMİŞ ŞİFRELEME STANDARDI

3.1 Giriş

Teknolojinin geliştiği ve gelişmeye çok hızlı bir şekilde devam edeceği bu bilgi çağında bilgisayarlar ve internet ortamı hayatımızın vazgeçilmez birer unsuru haline gelmiştir. Böyle bir ortamda bilginin korunması ve bir noktadan bir noktaya iletilmesi çok büyük önem kazanmıştır. Verilerin güvenli bir biçimde aktarımı ve elde edilmesi için, kriptografi bilimi aracılığı ile çeşitli şifreleme, anahtarlama ve çözümleme algoritmaları oluşturulmaktadır. Bu yeni algoritmalar gelişen teknolojiye uygun bir şekilde oluşturulmalı ve gelişen teknolojilere uygun tasarlanmalıdır.

Şifreleme algoritmalarını yapısal olarak iki ana gruba ayırabiliriz. Simetrik ve Asimetrik şifreleme algoritmaları. Simetrik şifreleme algoritmalarında verilerin şifrelenmesinde ve şifrenin çözülmesinde tek anahtar, Asimetrik şifreleme algoritmalarında şifreleme ve şifre çözmede ayrı iki anahtar kullanılmaktadır.

Bu yeni oluşturulan algoritmaların standartlaşması için birçok ülke kendi standartlaştırma enstitülerini kurmuştur. Bunların en önemlisi Amerika'da 1960'da kurulan NIST'tir. NIST 1977 yılında bir simetrik şifreleme algoritması olan DES'i bir standart olarak belirlemiştir. DES uzun yıllar güvenilir bir algoritma olarak kullanıldı. Kriptanalistler teknolojinin gelişimine paralel daha güçlü donanıma sahip bilgisayarlar sayesinde DES kırmak için yoğun bir şekilde uğraştılar. Bunun sonucu olarak DES kırılmıştır ve daha güvenli olan TDES kullanılmaya başlanmıştır. TDES arka arkaya DES şifreleme algoritmasını tekrarlayarak ortaya çıkmıştır. DES'in ve TDES' in güvenirliğini kaybetmesiyle NIST yeni şifreleme algoritmasına yönelmiştir.

1997'de NIST DES'in yerine AES'i seçmek ve geliştirmek için bir program duyurdu. Tek bir standart geliştirmek için kriptografi topluluğundan algoritmalar

istediler. 1998 yılında 15 algoritma önerildi ve NIST bunların içinden beş finalisti 1999 yılında seçti. NIST'in planı 2000 yılında standart olması için bir yada birkaç algoritma seçmekti.

DES'in yerine getirilen AES, daha hızlı daha güçlü ve daha ucuz olmalıydı. Yazılımda kullanıldığı zaman daha hızlı olmakla birlikte donanımda da kolay kullanılabilmeliydi (Akıllı kart vs). Uzun zaman kullanımda olan DES gibi o da saldırılara uzun süre karşı koymalıydı. Bir çok algoritma uzun süre incelendikten sonra beş tane finalistte karar kılındı. Bunlar Serpent, Rc6, Rijndael, Twofish ve Mars şifreleme algoritmalarıydı. Bu noktadan sonra bu algoritmalar arasından hangisinin standartlaşması gerektiği hakkında yoğun çalışmalar yapıldı. Bu beş finalist arasında şifreleme ve şifre çözme hızlarının yanında yazılım ve donanım uygunluğu, kolay uygulanması ve en önemli olarak güvenlik performansı açısından incelendi. Bu çalışmalardan sonra Rijndael şifreleme algoritması birinci olarak seçildi **[1]**.

3.2 Rijndael Algoritmasının Genel Yapısı

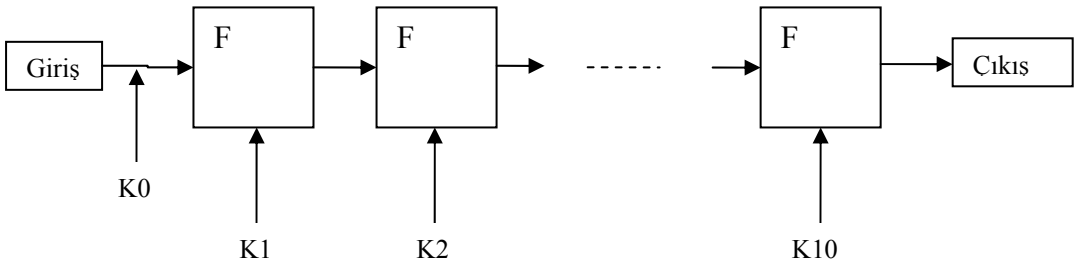
AES (Rijndael) algoritması 128 bit veri bloklarını 128, 192, 256 bit anahtar seçenekleri ile şifreleyen bir algoritmadır. Döngü sayısı anahtarın büyüklüğüne yani içerdiği bit sayısına göre değişmektedir. Tablo 3.1'de şifreleme anahtar uzunluğu ve bunlara karşı düşen şifreleme tur sayıları verilmiştir:

Tablo 3.1: Anahtar uzunluğu – tur sayısı ilişkisi

Anahtar uzunluğu (bit)	Tur sayısı
128	10
192	12
256	14

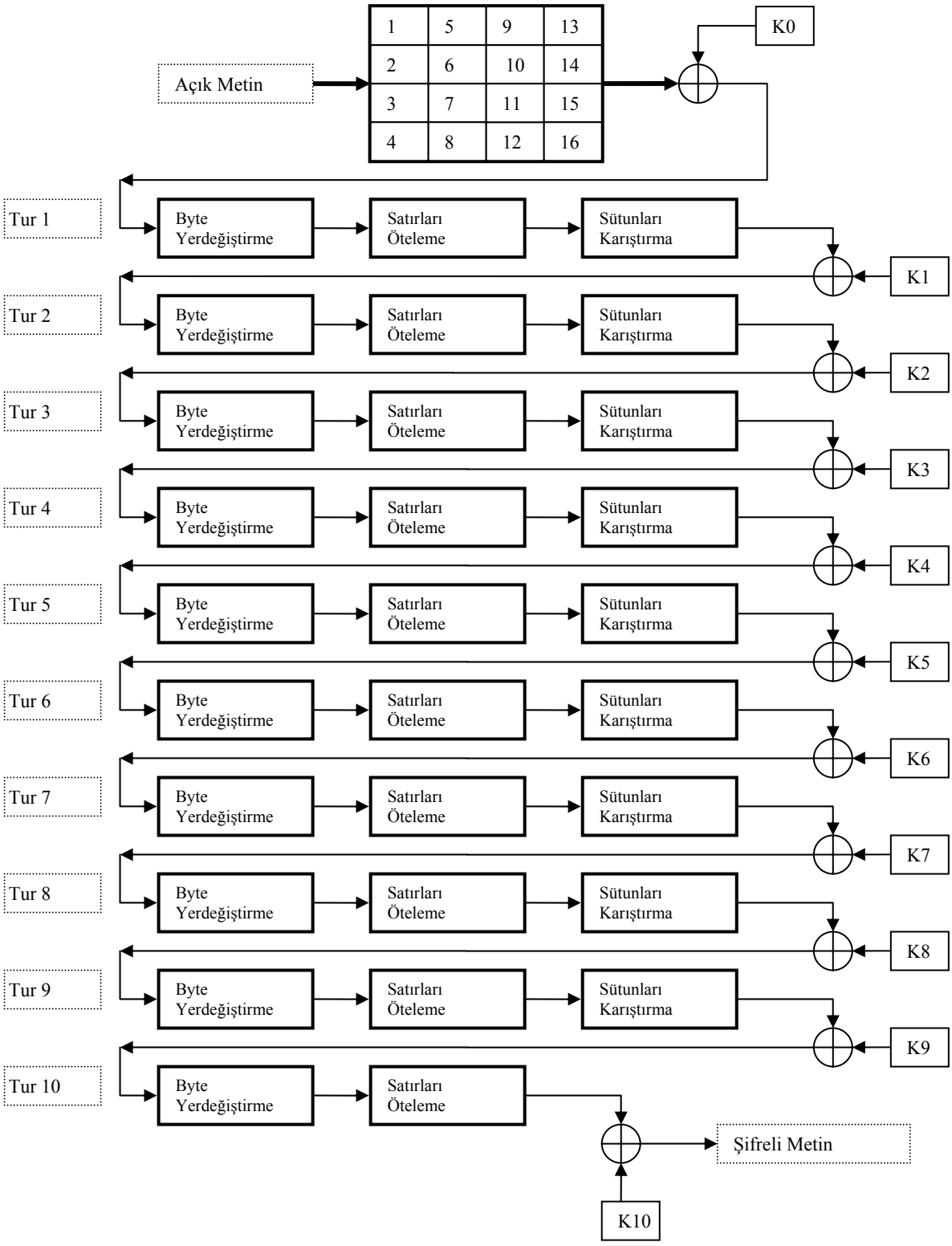
DES, 56 bitlik anahtar ile 64 bitlik bloğu şifrelemektedir. Triple DES (üçlü DES) ise

genellikle 112 bitlik anahtar ile 64 bitlik blokları şifrelemektedir.



Şekil 3.1: AES turları

Geliştirilmiş şifreleme standardının (AES) işlemsel yolu Şekil 3.1'de gösterilmiştir. AES algoritmasında her tur dört katmandan oluşur. İlk olarak 128 bit veri 4×4 bayt matrisine dönüştürülür. Daha sonra her döngüde sırasıyla baytların yerdeğiştirmesi, satırların ötelenmesi, sütunların karıştırılması ve anahtar planlamadan gelen o tur için belirlenen anahtar ile XOR'lama işlemleri yapılır. Baytların yerdeğiştirilmesinde 16 bayt değerinin her biri 8 bit girişli ve 8 bit çıkışlı S kutusuna sokulur. S kutusu değerleri, Galois alanı'nda (Galois Field - GF) GF(28), 8 bitlik polinom için çarpmaya göre ters alındıktan sonra doğrusal bir dönüşüme sokularak elde edilmiştir. Satırların ötelenmesi işleminde 4×4 bayt matrisinde satırlar ötelenmiş ve sütunların karıştırılması işleminde herhangi bir sütun için o sütundaki değerler karıştırılmıştır. Döngünün son katmanında ise o döngüye ait anahtar ile XOR işlemi yapılmaktadır. Şekil 3.2, 128 bit anahtar ile şifreleme için AES algoritmasını göstermektedir **[6]**.



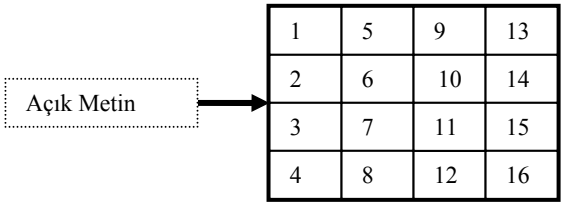
Şekil 3.2: Tüm AES resmi **[2]**.

AES'de baytlar üzerinde tanımlanan operasyonlar kullanılmasının yanında, 2 önemli avantaj sağlamaktadır:

- AES 'in katıksız yazılım uygulaması çok hızlıdır. Örneğin Pentium 200Mhz üzerinde çalışan bir C++ uygulaması, 70Mbit/s şifreleme performansı göstermektedir ;
- AES 'in farksal ve doğrusal şifreleme analizlerine direnci S kutusu seçimine bağlı değildir. DES için S kutuların NSA için açık içerdiğinden şüphelenilmektedir. Aslında bütün operasyonlar basittir.

3.3 Durum Atanması

16 baytlık giriş veri bloğu için işlemler 4x4 lük matris formuna atanarak yapılır. Bunun için veri bloğumuzu ilk olarak bu matrise çevirmemiz gerekmektedir. Şekil 3.3'de bu işlem gösterilmiştir.

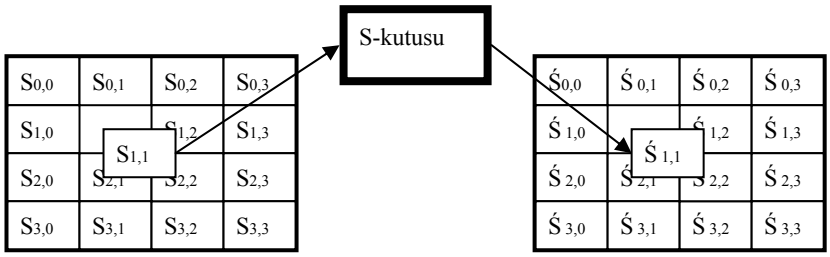


Şekil 3.3: Durum Ataması işlemi

3.4 Bayt yer değiştirme

Bayt yer değiştirmesi işlemi , 16 baytlık tüm veri bloğunun baytları için ayrı ayrı yeni bayt değerleri yaratılmasıdır. Bu dönüşüm işlemi için S kutusu (S-box) tabloları kullanılır **[4]**.

Şekil 3.4'de 16 baytlık bir blok için bayt yer değiştirme işlemi gösterilmiştir:



Şekil 3.4: Bayt yer değiştirme işlemi

Bayt Yer Değiştirme işlemi tersi olan bir işlemdir ve iki adet dönüşüm ile elde edilir:

1. İlk adımda çarpmaya göre ters işlemi uygulanır.
3. İkinci adımda GF(2) uzayında Ilgın dönüşüm uygulanır.

$$b_i^{'} = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

Matris formunda S kutusundaki ılgın dönüşümü (2) denklemi ile gösterilmiştir:

$$\begin{pmatrix} b'_7 \\ b'_6 \\ b'_5 \\ b'_4 \\ b'_3 \\ b'_2 \\ b'_1 \\ b'_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad \textbf{(2)}$$

S kutusu için yer değiştirme işleminin olası tüm sonuçları Tablo 3.2'de gösterilmiştir.

Tablo 3.2: Bayt Yer Değiştirme tablosu– S kutusu[xy] (onaltılık düzende)

Hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
	4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

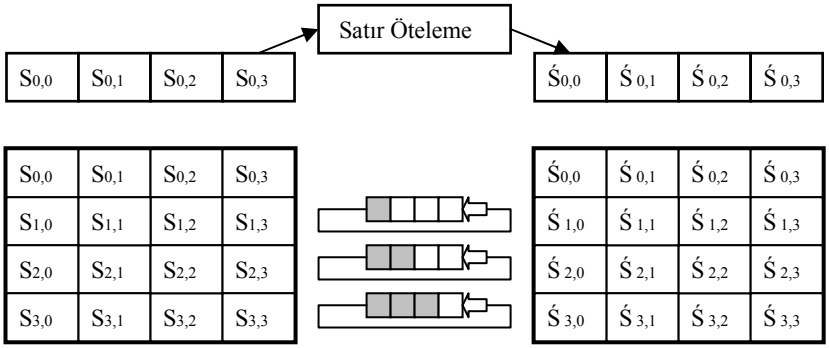
Yazılımda ise yer değiştirme fonksiyonu aşağıdaki kod ile gerçeklenmiştir:

```
k = 0;

while (k < Nk)
{
// buraya r1 ara degeri konacak.

  enc[k].key0 = r1_ara[4*k];
  enc[k].key1 = r1_ara[4*k+1];
  enc[k].key2 = r1_ara[4*k+2];
  enc[k].key3 = r1_ara[4*k+3];
```

```
  // subbytes
    a = enc[k].key0/16;     b = enc[k].key0%16;
    sb_output = s_box[a][b];
    enc[k].key0 = sb_output;
    a = enc[k].key1/16;     b = enc[k].key1%16;
    sb_output = s_box[a][b];
    enc[k].key1 = sb_output;
     a = enc[k].key2/16;     b = enc[k].key2%16;
    sb_output = s_box[a][b];
    enc[k].key2 = sb_output;
     a = enc[k].key3/16;     b = enc[k].key3%16;
    sb_output = s_box[a][b];
    enc[k].key3 = sb_output;
  k = k+1;
}  // register r2 dolduralim. after subbytes register
for (i=0; i<=3; i++)
{
  r2[n][4*i]   = enc[i].key0;
  r2[n][4*i+1] = enc[i].key1;
  r2[n][4*i+2] = enc[i].key2;
  r2[n][4*i+3] = enc[i].key3;
}
```

3.5 Satır Öteleme İşlemi

Satır öteleme işlemi son üç satır üstünde işlem yapar. Şekil 3.5'de gösterilmiştir **[4]**.



Şekil 3.5: Satır öteleme işlemi

Satır Öteleme işlemi yazılımda aşağıdaki kod ile gerçeklenmiştir:

```
k = 0;
while (k < Nk)

{
// buraya r2 ara degeri konacak.

  enc[k].key0 = r2_ara[4*k];
  enc[k].key1 = r2_ara[4*k+1];
  enc[k].key2 = r2_ara[4*k+2];
  enc[k].key3 = r2_ara[4*k+3];
  k = k+1;
}

temp.key0 = enc[0].key1;
  temp.key1 = enc[1].key1;
  temp.key2 = enc[2].key1;
  temp.key3 = enc[3].key1;
  enc[0].key1 = temp.key1;
  enc[1].key1 = temp.key2;
  enc[2].key1 = temp.key3;
  enc[3].key1 = temp.key0;
  temp.key0 = enc[0].key2;
```

```
        temp.key1 = enc[1].key2;
        temp.key2 = enc[2].key2;
        temp.key3 = enc[3].key2;
        enc[0].key2 = temp.key2;
        enc[1].key2 = temp.key3;
        enc[2].key2 = temp.key0;
        enc[3].key2 = temp.key1;
        temp.key0 = enc[0].key3;
        temp.key1 = enc[1].key3;
        temp.key2 = enc[2].key3;
        temp.key3 = enc[3].key3;
        enc[0].key3 = temp.key3;
        enc[1].key3 = temp.key0;
        enc[2].key3 = temp.key1;
        enc[3].key3 = temp.key2;

   // register r3 dolduralim. after shift rows register
      for (i=0; i<=3; i++)
      {
        r3[n][4*i]   = enc[i].key0;
        r3[n][4*i+1] = enc[i].key1;
        r3[n][4*i+2] = enc[i].key2;
        r3[n][4*i+3] = enc[i].key3;
      }
/////   r3 bitir   /////
```

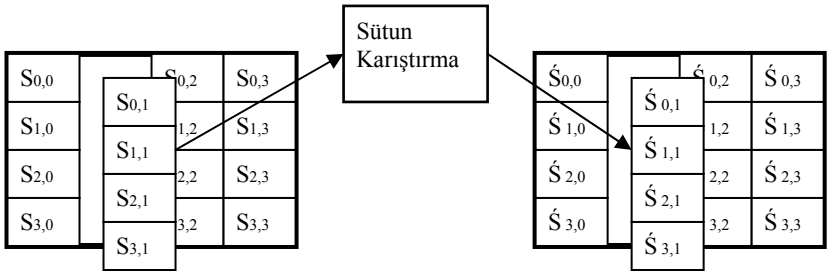
3.6 Sütun karıştırma işlemi

Sütun karıştırma işlemi girişine gelen 16 baytlık verinin sütunları üzerinde işlem yapar. Bölüm 2.3'de gösterildiği gibi her sütun 4. dereceden bir polinom gibi kullanılır.

Matris formunda yapılan işlem aşağıdaki gibidir:

$$\begin{pmatrix} S'_{3,c} \\ S'_{2,c} \\ S'_{1,c} \\ S'_{0,c} \end{pmatrix} = \begin{pmatrix} 02 & 01 & 01 & 03 \\ 03 & 02 & 01 & 01 \\ 01 & 03 & 02 & 01 \\ 01 & 01 & 03 & 02 \end{pmatrix} \begin{pmatrix} S'_{3,c} \\ S'_{2,c} \\ S'_{1,c} \\ S'_{0,c} \end{pmatrix}$$

Şekil 3.6'da sütun karıştırma işlemini nasıl çalıştığı gösterilmiştir **[4]**.



Şekil 3.6: Sütun karıştırma işlemi

Sütun karıştırma işlemi yazılımda aşağıdaki kod ile gerçeklenmiştir:

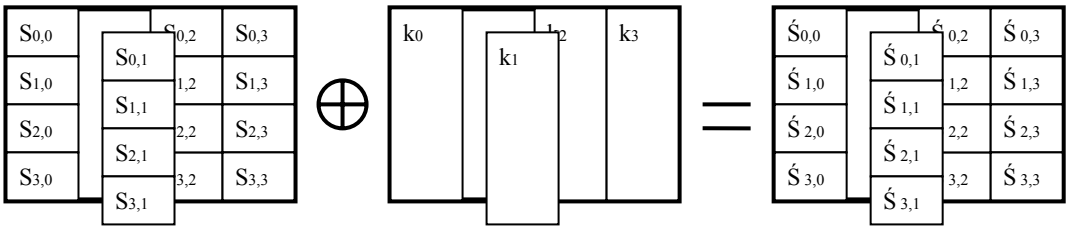
```
k = 0;

while (k < Nk)
{

// buraya r3 ara degeri konacak.
  enc[k].key0 = r3_ara[4*k];
  enc[k].key1 = r3_ara[4*k+1];
  enc[k].key2 = r3_ara[4*k+2];
  enc[k].key3 = r3_ara[4*k+3];
  k = k+1;

}

    k = 0;
```

```
    while (k < Nk)
    {
      temp.key0 = FFmul(0x02, enc[k].key0)^
        FFmul(0x03, enc[k].key1)^
              enc[k].key2^
              enc[k].key3 ;

      temp.key1 = FFmul(0x02, enc[k].key1)^
        FFmul(0x03, enc[k].key2)^
        enc[k].key3^
        enc[k].key0 ;

      temp.key2 = FFmul(0x02, enc[k].key2)^
        FFmul(0x03, enc[k].key3)^
        enc[k].key1^
        enc[k].key0 ;

      temp.key3 = FFmul(0x02, enc[k].key3)^
        FFmul(0x03, enc[k].key0)^
        enc[k].key2^
        enc[k].key1 ;

      enc[k].key0 = temp.key0;
      enc[k].key1 = temp.key1;
      enc[k].key2 = temp.key2;
      enc[k].key3 = temp.key3;
      k = k+1;

}   // register r4 dolduralim. after mix columns register

for (i=0; i<=3; i++)

{
  r4[n][4*i]   = enc[i].key0;
```

```
    r4[n][4*i+1] = enc[i].key1;
    r4[n][4*i+2] = enc[i].key2;
    r4[n][4*i+3] = enc[i].key3;
  }
```

3.7 Tur Anahtarı ile Toplama işlemi

AES algoritmasında her bir tur sonunda elde edilen matris, AES şifresinin ilgili tur için üretilmiş anahtarı ile XOR'a tabi tutulur. Şekil 3.7'de işlem gösterilmiştir**[4]**:



Şekil 3.7: Tur sonu değeri ile XOR işlemi

Yapılan işleme ait XOR işlem detayı Bölüm 2.2'de verilmiştir. Gösterilimler 128 bitlik bir anahtar ile, 128 bitlik bir bloğun AES şifrelemesi için verilmiştir.

Tur Anahtarı ile toplama işlemi yazılımda aşağıdaki kod ile gerçeklenmiştir:

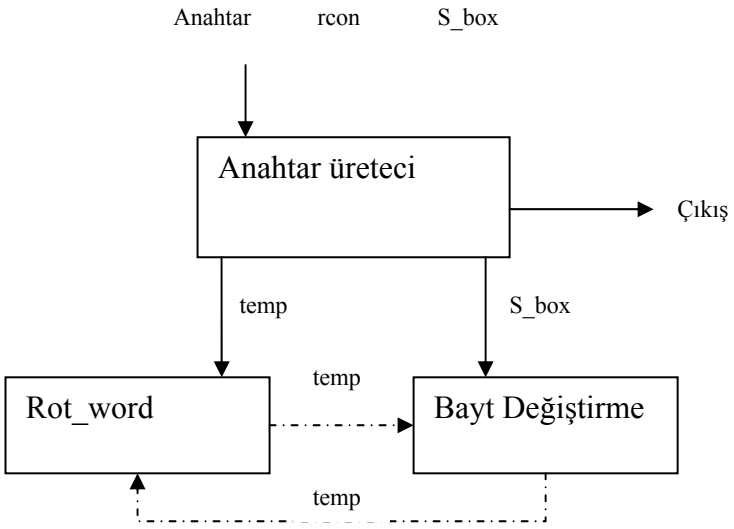
```
k = 0;

while (k < Nk)

{   // buraya r4 ara degeri konacak.
  enc[k].key0 = r4_ara[4*k];
  enc[k].key1 = r4_ara[4*k+1];
  enc[k].key2 = r4_ara[4*k+2];
  enc[k].key3 = r4_ara[4*k+3];

  k = k+1;
```

```
}
k = 0;
while (k < Nk)
{
    // key degerleri ile exor islemi
    enc[k].key0 = enc[k].key0^key[k].key0;
    enc[k].key1 = enc[k].key1^key[k].key1;
    enc[k].key2 = enc[k].key2^key[k].key2;
    enc[k].key3 = enc[k].key3^key[k].key3;
  aes_steps[n+1][4*k]   = enc[k].key0;
  aes_steps[n+1][4*k+1] = enc[k].key1;
  aes_steps[n+1][4*k+2] = enc[k].key2;
  aes_steps[n+1][4*k+3] = enc[k].key3;
  k = k+1;
}
// register r5 dolduralim. round sonu register

for (i=0; i<=3; i++)
{
  r5[n][4*i]   = enc[i].key0;
  r5[n][4*i+1] = enc[i].key1;
  r5[n][4*i+2] = enc[i].key2;
  r5[n][4*i+3] = enc[i].key3;
}
```

3.8 Anahtar Üreteci Oluşturma İşlemi

Anahtar üreteci, şifreleme ve şifreyi çözmede kullanılacak anahtar dizisini üretir. Bu işlem içinde "rcon" işlemi ve "s kutusu" işlemleri kullanılır. Şekil 3.8'de girişe gelen 16 bayt için bir adet anahtar üretme işlemi gösterilmiştir:

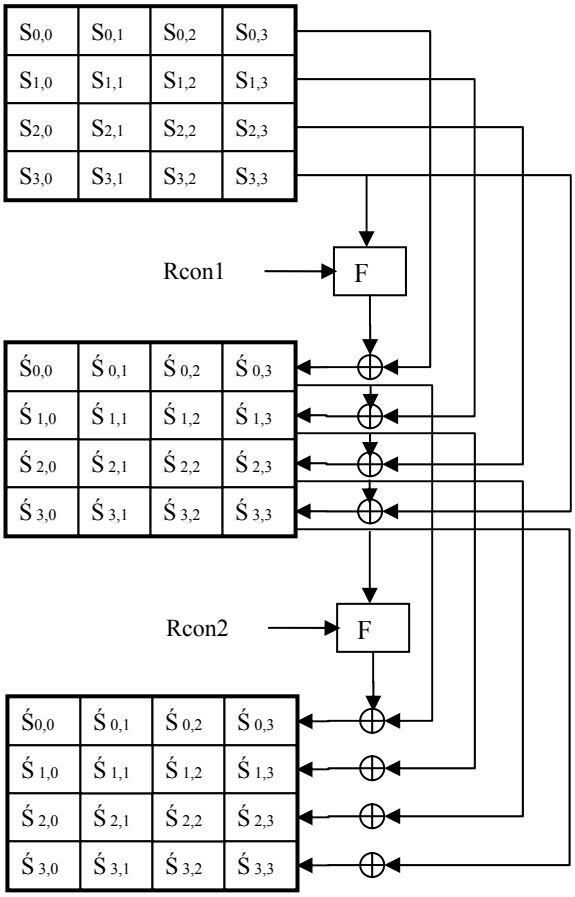
Kesik çizgili oklarla oluşturulmuş olan döngü, bayt çevrim ve bayt yer değiştirme fonksiyonlarının anahtar üreteci tarafından iteratif olarak çağırıldığını göstermektedir.



Şekil 3.8: Anahtar üretme fonksiyonu

Anahtar üretmenin temel prensibi bir satır ile dört önceki satırın bit - bit xor işlemine girmesi şeklindedir. Yedinci satır olan (k71 . . . k74) satırı, altıncı satır (k61 . . . k64) ve üçüncü satır (k31 . . . k34) ile XORlanması ile elde edilir. Şekil 3.9'da işlemin ayrıntısı gösterilmiştir **[3]**.

Farklı olarak dördün katının her bir satır fazlası (satır 5,9, ...)'dan farklı olarak üretilir. XOR işlemi uygulanmadan önce bir önceki satır, kendisine karşı düşen tur sabiti ile döngü işlemi , değiştirme işlemi ve XOR işlemine tabi tutulur. Her tur için tur sabitleri Tablo 3.3'te verilmiştir.



Döngü 10 defa tekrarlanarak devam eder.....

Şekil 3.9: Anahtar Üretimi

Tur sabiti değerleri Tablo 3.3'te verilmiştir. Aşağıda gerçekleştirilen kod içinde rcon değerlerini bulan kısım ayrı bir fonksiyon olarak gerçekleştirilmiş ve değerler bu fonksiyon yardımı ile bulunmuştur.

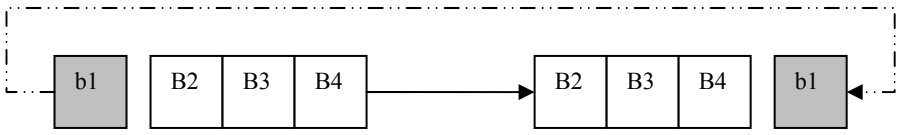
Tablo 3.3: rcon değerleri

tur değeri	tur sabiti (rcon) değeri
1	01 00 00 00
2	02 00 00 00
3	04 00 00 00
4	08 00 00 00
5	10 00 00 00
6	20 00 00 00

7	40 00 00 00
8	80 00 00 00
9	1b 00 00 00
10	36 00 00 00

3.8.1 Kelime Döndürme işlemi

Kelime Döndürme fonksiyonu kendi girişine gelen dört bayt bilgisini Şekil 3.10'da gösterildiği gibi çevirerek yeniden oluşturur ve çıkış değeri olarak iletir.



Şekil 3.10: Kelime döndürme işlemi

Şifre üretim işlemi yazılımda aşağıdaki kod ile gerçeklenmiştir:

```
void key_sch(void)
{
Nk = 4;
Nb = 4;
Nr = 10;

i = 0;

while (i < Nk)

{
  w[i].key0 = kx[0][4*i];
  w[i].key1 = kx[0][4*i+1];
  w[i].key2 = kx[0][4*i+2];
  w[i].key3 = kx[0][4*i+3];
```

```
  i = i+1;
}
i = Nk;

while (i < Nb * (Nr+1))
{
  temp.key0 = w[i-1].key0;
  temp.key1 = w[i-1].key1;
  temp.key2 = w[i-1].key2;
  temp.key3 = w[i-1].key3;

  if (i%Nk == 0)
  {
    rotated[0] = temp.key1;  rotated[1] = temp.key2;  rotated[2] = temp.key3;
rotated[3] = temp.key0;

    temp.key0 = rotated[0];
    temp.key1 = rotated[1];
    temp.key2 = rotated[2];
    temp.key3 = rotated[3];

    //  temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
    a = temp.key0/16;    b = temp.key0%16;
    sb_output = s_box[a][b];

    temp.key0 = sb_output;
    temp.key0 = temp.key0^rcon(i/Nk);
     a = temp.key1/16;    b = temp.key1%16;
    sb_output = s_box[a][b];
    temp.key1 = sb_output;

    a = temp.key2/16;    b = temp.key2%16;
    sb_output = s_box[a][b];
    temp.key2 = sb_output;
```

```
      a = temp.key3/16;     b = temp.key3%16;
      sb_output = s_box[a][b];
      temp.key3 = sb_output;
    }
   else if (Nk > 6 & i%Nk == 4)
   {
      a = temp.key0/16;     b = temp.key0%16;
      sb_output = s_box[a][b];
      temp.key0 = sb_output;
       a = temp.key1/16;     b = temp.key1%16;
      sb_output = s_box[a][b];
      temp.key1 = sb_output;
       a = temp.key2/16;     b = temp.key2%16;
      sb_output = s_box[a][b];
      temp.key2 = sb_output;
       a = temp.key3/16;     b = temp.key3%16;
      sb_output = s_box[a][b];
      temp.key3 = sb_output;
    }

//   w[i] = w[i-Nk] xor temp
   w[i].key0 = w[i-Nk].key0^temp.key0;
   w[i].key1 = w[i-Nk].key1^temp.key1;
   w[i].key2 = w[i-Nk].key2^temp.key2;
   w[i].key3 = w[i-Nk].key3^temp.key3;
   a = i/Nk;
   b = ((i-Nk)*Nk)%16 ;
   kx[a][b] = w[i].key0;
   kx[a][b+1] = w[i].key1;
   kx[a][b+2] = w[i].key2;
   kx[a][b+3] = w[i].key3;
   i = i + 1;
 }}
// rcon degerini return eden fonksiyon.
```

```
int rcon(int p_i)
{
   int y;
   int i;
   clrscr();
   y = 1;
   for (i=1; i<p_i; i++)
   {     y = xtime(y);}
   return y;}

int xtime(int a)
{
   int b;
   int sonuc;
   if (a == 0x80)
         b = 0x1B;
   else
         b = 0;
   sonuc = LShiftByte(a, 1);
   sonuc = sonuc^b;
   return sonuc;
}

int LShiftByte(int bytValue,int bytShiftBits)
{
  int m_bytOnBits[8];
  int m_byt2Power[8];
  int kayd;
  m_bytOnBits[0] = 1;
  m_bytOnBits[1] = 3;
  m_bytOnBits[2] = 7;
  m_bytOnBits[3] = 15;
  m_bytOnBits[4] = 31;
```

```
  m_bytOnBits[5] = 63;
  m_bytOnBits[6] = 127;
  m_bytOnBits[7] = 255;
  m_byt2Power[0] = 1;
  m_byt2Power[1] = 2;
  m_byt2Power[2] = 4;
  m_byt2Power[3] = 8;
  m_byt2Power[4] = 16;
  m_byt2Power[5] = 32;
  m_byt2Power[6] = 64;
  m_byt2Power[7] = 128;

  if (bytShiftBits == 0)
       return bytValue;
  else
  {
       if (bytShiftBits == 7)
       {
         if (bytValue && 1)
            kayd = 0x80;
         else
            kayd = 0;
         return kayd;
       }
  }

  kayd = ((bytValue & m_bytOnBits[7 - bytShiftBits]) *
m_byt2Power[bytShiftBits]);
  return kayd;
}
```

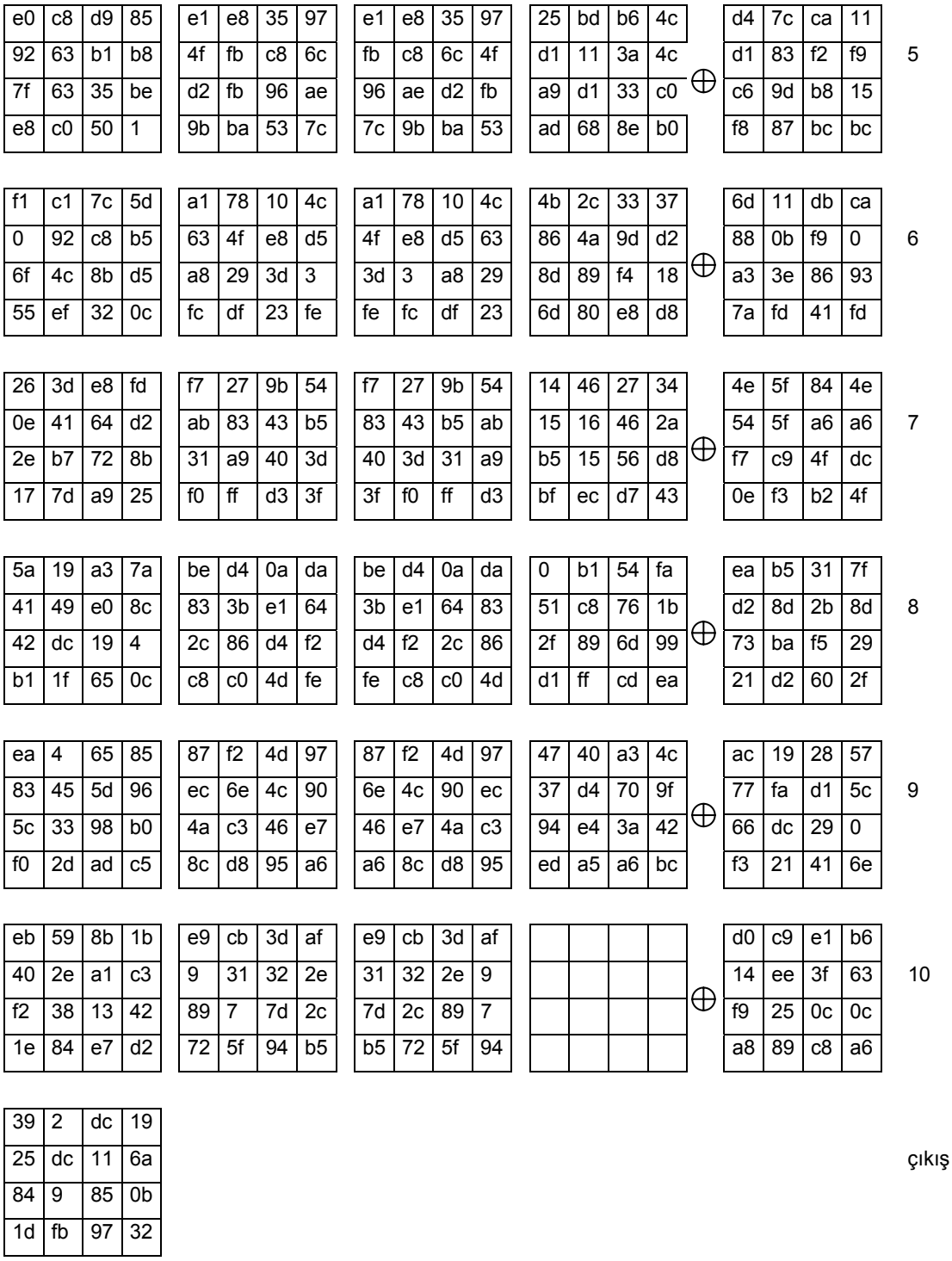
3.9 AES şifreleme adımları örneği

Şekil 3.11'de 128 bitlik bir blok ve 128 bitlik bir anahtar için, şifreleme adımları adım adım gösterilmiştir. Her işlemin sonunda aldığı değer onaltılık sistemde durum olarak incelenebilir **[5]**.

Giriş Bloğu = 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34

Anahtar = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

Tur Başlangıcı	Bayt Yer Değiştirme	Satır Öteleme	Sütun Karıştırma	Tura ait Anahtar dizisi	Tur
32 88 31 e0 43 5a 31 37 f6 30 98 7 a8 8d a2 34				2b 28 ab 9 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c	Giriş
19 a0 9a e9 3d f4 c6 f8 e3 e2 8d 48 be 2b 2a 8	d4 e0 b8 1e 27 bf b4 41 11 98 5d 52 ae f1 e5 30	d4 e0 b8 1e bf b4 41 27 5d 52 11 98 30 ae f1 e5	4 e0 48 28 66 cb f8 6 81 19 d3 26 e5 9a 7a 4c	a0 88 23 2a fa 54 a3 6c fe 2c 39 76 17 b1 39 5	1
a4 68 6b 2 9c 9f 5b 6a 7f 35 ea 50 f2 2b 43 49	49 45 7f 77 de db 39 2 d2 96 87 53 89 f1 1a 3b	49 45 7f 77 db 39 2 de 87 53 d2 96 3b 89 f1 1a	58 1b db 1b 4d 4b e7 6b ca 5a ca b0 f1 ac a8 e5	f2 7a 59 73 c2 96 35 59 95 b9 80 f6 f2 43 7a 7f	2
aa 61 82 68 8f dd d2 32 5f e3 4a 46 3 ef d2 9a	ac ef 13 45 73 c1 b5 23 cf 11 d6 5a 7b df b5 b8	ac ef 13 45 c1 b5 23 73 d6 5a cf 11 b8 7b df b5	75 20 53 bb ec 0b c0 25 9 63 cf d0 93 33 7c dc	3d 47 1e 6d 80 16 23 7a 47 fe 7e 88 7d 3e 44 3b	3
48 67 4d d6 6c 1d e3 5f 4e 9d b1 58 ee 0d 38 e7	52 85 e3 f6 50 a4 11 cf 2f 5e c8 6a 28 d7 7 94	52 85 e3 f6 a4 11 cf 50 c8 6a 2f 5e 94 28 d7 7	0f 60 6f 5e d6 31 c0 b3 da 38 10 13 a9 bf 6b 1	ef a8 b6 db 44 52 71 0b a5 5b 25 ad 41 7f 3b 0	4



Şekil 3.11: AES işlem değerleri

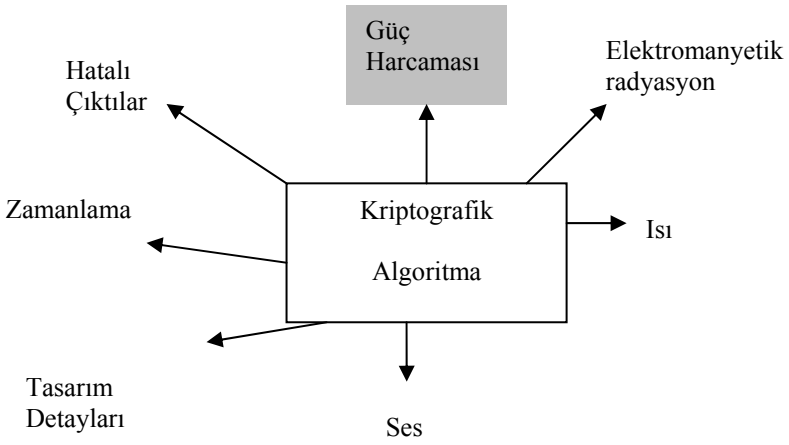
4. GÜÇ ANALİZİ SALDIRISI

4.1 Giriş

Genel olarak kriptografik donanımlar, bir şifreleme işlemini veya bir şifreleme algoritmasının tamamını içerirler. Uygulamalarda donanımda ayrıca saklanması gereken bir gizli veya özel anahtar olması gerekebilir. Bu nedenle bir kriptografik cihaz, bu özel bilginin elde edinimi veya çeşitli yollarla bulunmasını engelleyecek şekilde tasarlanmalıdır **[7]**.

Kriptografik sistemlere aktif saldırılar uzun bir dönemdir yapılagelmektedir. Pasif atakların ise ilk olarak 1996 yılında konu hakkında yayınlanan bir makale ile **[8]** büyük bir tehdit olduğunu gösterilmiştir. Pasif saldırıda, saldırıyı yapan kişi, kriptografi cihazının standart işlevlerini kullanır **[7]**.

Bu kullanım sırasında elde edilen fiziksel / elektriksel etkiler daha sonra saldırı amaçlı kullanılır.



Şekil 4.1: Yan kanal çıkışları

Şekil 4.1'de de gösterildiği şekilde değişik yan kanal bilgileri mevcuttur; örneğin işlem süresi, güç harcaması, elektromanyetik radyasyon, gibi. Eğer bu şekilde elde edilen çıkışlar, kriptografik anahtara ait bilgi içeriyorsa veya bulunması konusunda ipucu veriyorsa, yan kanal olarak adlandırılır. Günümüzde CMOS teknolojisi sayısal devrelerin gerçeklemesinde büyük oranda kullanılan teknolojidir. Bir CMOS devresinin güç harcamasında en önemli paya sahip olan kısmı dinamik güç harcamasıdır **[7]**.

İki tür bilgi mevcuttur: Geçiş sayısı sızıntısı ve Hamming ağırlığı sızıntısı.
Geçiş sayısı sızıntısı bize değişen bitlerin sayısı hakkında bilgi verirken, Hamming ağırlığı sızıntısı eşzamanlı olarak işlenen 1 bitlerinin sayısı hakkında bilgi verir.

İki tür güç analizi saldırısından bahsedilebilir:

- SPA: Basit güç analizi saldırısı
- DPA: Farksal güç analizi saldırısı.

4.2 Basit Güç Analizi Saldırısı

Saldırıyı yapan kişi, yan kanal saldırısından elde ettiği ölçüm ve bilgiyi, şifreleme anahtarını veya bir kısmını elde etmek için direkt olarak kullanır.

Özetleyecek olursak;

- Şifreleme algoritmasında kullanılan her bloğun kendi güç tüketim karakteristiği var ise yapılabilir.

- Saldırı güç harcamasından direkt şifreleme anahtarını elde eder. Güç harcaması dalga şekli gözle kontrol edilir. Bu nedenle tek bir ölçüm yapılması yeterlidir.

- Basit güç analizine karşı savunma geliştirmek kolaydır.

4.3 Farksal Güç Analizi Saldırısı

Farksal güç analizi ilk olarak Paul KOCHER tarafından 1998 yılında gündeme getirilmiş ve kriptografi dünyasında ses getirmiştir. Bir DES algoritması üzerinde güç analizi saldırısı ilk olarak **[9]** nolu makalede yapılmıştır. "Akıllı kart" üreticileri konu hakkında risk olduğunu görmüş ve bu konuya ciddi olarak eğilmişlerdir.

Farksal güç analizinde çok sayıda ölçüm yapılır ve elde edilen ölçümlerden gürültü filtrelenir. Bu şekilde elde edilen gerçek ölçüm sonuçlarına ek olarak saldırıyı yapan kişi(ler) tarafından bir model öngörülür. Bu modele göre ölçüm sonuçlarının bilgisayar ortamında benzetimi yapılır ve uygun veriler oluşturulur **[7]**.

Güç Harcaması Saldırıları sonuç alıcıdır; çünkü

- İstatistikseldir, sinyal işlenir
- Rastlansal mesaj gönderilir.
- Bilinen algoritmaya saldırılır.
- Tek bir akıllı kart yeterli olur.

Farksal güç analizi saldırısı için bileşenler şunlardır:

- Güvenilir Güç Ölçümleri
- Algoritma
- Güç Harcaması Öngörüleri
- Elde edilebilecek diğer tüm ek bilgiler

Saldırı 2 adımda yapılır:

- Gerçek ölçüm: akıllı kartta ölçüm yapılır.
- Analiz kısmı : Bir bilgisayarda öngörülü analiz gerçekleştirilir.

SPA yapılan işlemler ve güç harcaması arasındaki bağlantıyı kullanır, DPA ise işlenen veri ve güç harcaması arasındaki bağlantıdan yola çıkmaktadır.

4.4. Saldırı için Öngörülen Algoritma

Algoritma, saldırı yapılacak düzeneğin öngörülen, değişik şekilde elde edilen -ki bu her türlü elde etmeyi içerir, modeli anlamındadır. Biz saldırı yapılacak düzeneğin bir AES algoritması olduğunu biliyoruz, ancak bunun yanında elde edilecek her türlü veri bizim işimizi çok daha fazla kolaylaştıracaktır.

Algoritmaların en önemli özelliği, öngörülen AES gerçeklemesi çalışma şeklinin taklit edilmiş olmasıdır. Ayrıca ölçme ortamının getireceği etkiler de algoritma içinde yer almıştır. Ölçüm düzeneğinde, 20.000 adet giriş, sırasıyla kart gerçeklemesine gönderilecek ve ölçümler yapılacaktır. Bu durumda her giriş için ölçüm öncesinde devrede bellek içerikleri sıfırlanmamış olacaktır. Yani bir önceki ölçümün etkileri hala devam ediyor olacak ve bellek içeriklerine bu durum üzerinden işlemler yapılacaktır. Algoritmalar bu durumu gözönüne alacak şekilde gerçeklenmiştir.

Bu tezde saldırı yaptığımız düzeneğe ait iki adet model bilgisine sahip olduğumuzu varsaydık ve bunlara göre varsayım verisi oluşturduk. Aşağıdaki bölümlerde bu modeller incelenmiştir.

4.4.1. Saldırı Amaçlı Model 1

Şekil 4.2'de ilk model için genel algoritma verilmiştir. Şekilde görüldüğü üzere beş adet belleğimiz var ve bunların güç harcaması bizim algoritmalarımızın temel dayanağı olacaktır.

Algoritmadaki her bir işlem bloğu kendisine ait işlemi yapar, bu bloklar ekte verilmiş olan AES işlem bloklarıdır.

- Giriş değeri ile anahtarın XOR işlemi
- Bayt Yer Değiştirme
- Satır Öteleme
- Sütun Karıştırma

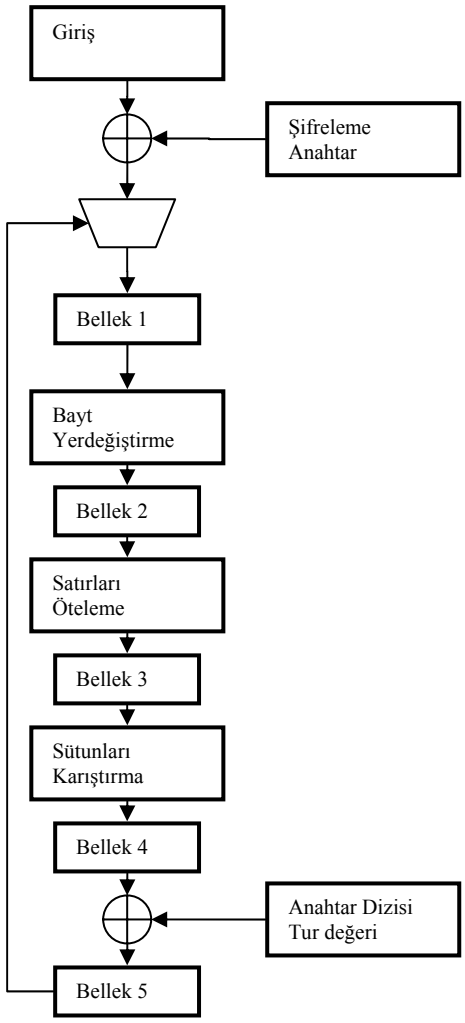
- Anahtarın turun durum değeri ile XOR işlemi

Her bir blok kendi girişindeki değeri, işlevine göre değiştirerek kendi bloğunu takip eden belleğe yazar. Bu nedenle, her turda, bellekler, sadece kendine ait saat diliminde anlamlı değer içerir. Onun dışındaki saat dilimlerinde anlamlı olmayan değerler içerebilirler. Ancak kesin olan, 5. saat diliminin sonunda R5 belleğinde olması gereken değer, tur sonu değeri olmalıdır. Buna göre anlamlı bellek değerleri için şunu diyebiliriz:

1. saat diliminde R1,
2. saat diliminde R2,
3. saat diliminde R3,
4. saat diliminde R4,
5. saat diliminde R5

bellekleri anlamlı değer içerecektir. 10 tur sonunda ve 10. turun 5. saat dilimi sonunda R5 belleğinde şifrelenmiş mesaj mevcut olacaktır.

Kullandığımız iki model arasındaki fark ise Anahtar dizisi ile ilgili davranış farkıdır. Şifreleme anahtar dizisinin tüm turlar için değerleri bellidir ve birinci modelde 10 adet AES turunun 5 adet saat dilimi boyunca anahtar dizisi çıkışlarının ideal tur değerinde olduğu varsayılacaktır. Sonuçta bu bir modeldir ve ilk modelimizin bu dezavantajlı noktasıdır. Ancak bu gerçek devrede böyle değildir.



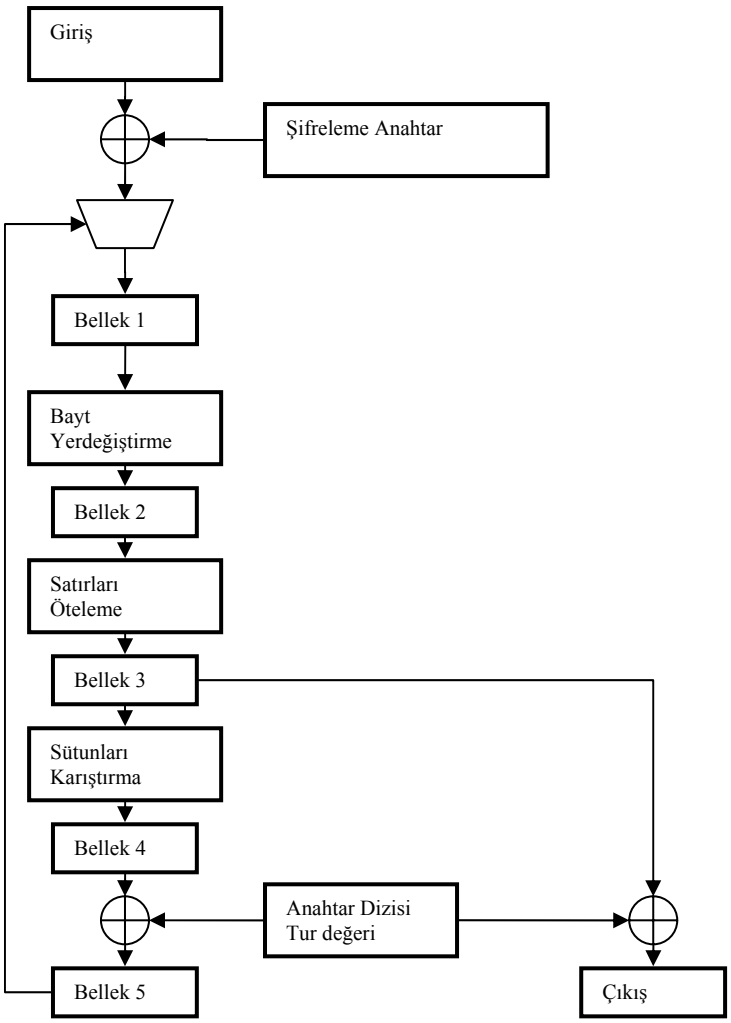
Şekil 4.2: Saldırı amaçlı model-1'e ait algoritma

4.4.1. Saldırı Amaçlı Model 2

Şekil 4.3'de model 2 için genel algoritma verilmiştir. İşlem bloklarının çalışması ve bellek içeriklerinin değiştirilmesi ilk model ile tamamen aynıdır. Yine 5 adet bellek mevcuttur ve işlem blokları her saat diliminde kendi girişlerindeki değerleri işlevlerine göre değiştirerek çıkışlarındaki belleklere iletirler.

İkinci modelimiz anahtar üretecinin tüm saat dilimlerinde karakteristiğini elde ettiğimiz ve kullandığımız model olacaktır. 10 tur ve her turda da 5 saat diliminde anahtar devresi çıkışlarını elde ederek kullandık. Bu değerler, "keysched_output.txt" dosyasında verilmiştir. Bu modelin çok daha gerçekçi olduğu tartışılmazdır.

Bu modelde elde ettiğimiz bir diğer bilgi de, onuncu tur boyunca R5 belleğinin içeriğinin 5 tur boyunca sabit olacağıdır. Bu nedenle 10. turda R5 içeriği değişmez ve daha sonraki işlemlere değişmeyen R5 içeriği yansıtılmıştır.



Şekil 4.3: Saldırı amaçlı model-2'e ait algoritma

4.5 Teorik Altyapı

DPA saldırısında saldırıyı yapan kişi, kriptografik cihaza ait bir modeli öngörmektedir. Bu modelin kalitesi, saldırıyı yapan kişinin cihaz hakkında elde edebildiği bilgi ile sınırlı olacaktır. Bu model yardımı ile cihaza ait çok sayıda yan kanal çıktısı kestirimi/öngörüsü elde edilir. Daha sonra elde edilen bu veriler gerçek cihazdan ölçülen yan kanal verileri ile karşılaştırılır. Bunun için istatistiksel fonksiyonlar kullanılır.

Bunların içinde en çok kullanılanları ortama farkı testi ve korelasyon analizidir. Biz bu tez kapsamında korelasyon analizi fonksiyonunu kullandık. Korelasyon analizi için öngördüğümüz model belli bir zaman dilimi içindeki çalışma için bize güç harcaması bilgisini verir. Daha sonra öngörülen modelin çıktıları ile gerçek ortamdaki cihazın aynı girişler için çalışması sırasında ölçülen güç harcaması ile korelasyonuna bakılır. Bu korelasyon ölçümü Pearson korelasyon katsayısı kullanılarak yapılabilir.

Bu noktada ekteki varsayımları yapalım:

$\boldsymbol{t_i}$, i. ölçüm datası olsun (i. ölçüm).

T ölçümler kümesi olsun.

$\boldsymbol{p_i}$, i. ölçüm için öngörülen model.

P öngörülen model kümesi.

T ve P'nin korelasyonu için aşağıdaki fonksiyon kullanılır.

$$\boldsymbol{C(T,P) = \frac{E(T.P) - E(T).E(P)}{\sqrt{Var(T).Var(P)}}}$$

Burada E(T) ölçüm kümesi T'nin beklenen değerini (ortalamasını) ve Var(T) bir ölçümün ortalamadan ne kadar saptığını gösterir.

Bu noktada eğer korelasyon yüksekse (1 veya -1'e yakınsa), öngörülen model ve ek olarak anahtar bulunması doğru demektir.

4.6 Ölçüm İşlemleri

Teknik olarak güç analizi saldırısı yapmak birçok zorluklar içermektedir. En önemli nokta, gürültü olmayan ve iyi durumda ölçüm yapılmasıdır. Gürültü miktarı ne kadar az olursa, o kadar az ölçüm ile istenilen sonuçlara ulaşılabilir.

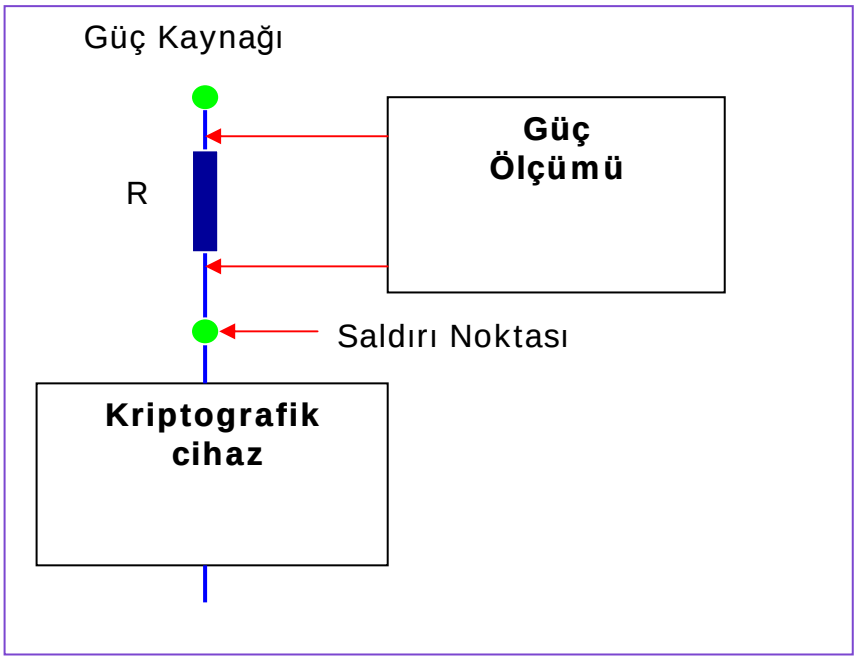
Ölçüm işleminde bir başka önemli noktada ölçüm düzeneğinin karmaşıklığıdır. Böyle bir düzenek genel olarak üç kısımdan oluşur:

- Saldırı yapılan cihaz
- bir adet ölçüm cihazı
- saldırıyı kontrol etmek için bir cihaz.

Bunlara ek olarak ölçüm sonuçlarının elde edilmesi ve saklanması amaçlı yazılım ve donanım gerekli olacaktır. Bu nedenle bir saldırı düzeneği hazırlanacağı zaman, saldırılan sistemin saldırıya karşı direncini ölçmek için ilk önce saldırının benzetiminin yapılması iyi olacaktır. Bu şekilde pratik saldırı düzeneğinde bir sorun olup olmadığı da görülmüş olacaktır. Bu şekilde benzetimi yapılan bir saldırıda, pratikte olabilecek gürültü olmayacağı için saldırının ne derece başarılı olabileceği de kestirilebilir.

Tez kapsamında saldırı yapılacak olan düzeneğe 20.000 adet giriş ile saldırı yapılacağı ve sonuçların elde edileceği bilgisi ile başladık. Ölçüm düzeneğinde önceden belirlenmiş girişler, sırayla kart gerçeklemesine gönderilir, ölçüm sonuçları bir bilgisayarda saklanır. İki adet ölçüm arasında kartta bir açma kapama işlemi yapılmaz; yani kartın içerisindeki bellek içerikleri bir önceki ölçüm işleminden kalan değerleri içerir.

Şekil 4.4'te temel ölçüm düzeneği basit gösterimle verilmiştir:



Şekil 4.4: Ölçüm düzeneği

4.7 Saldırı Öncesi Benzetim Çalışması

Güç analizi saldırısı için benzetim veri oluşturma öncesi önşart, bir adet anahtar belirlemek ve rastlansal olacak şekilde uygun sayıda giriş verisi oluşturmaktır. Bu iki veri, ekteki giriş dosyalarında tutulmuştur. Belirleme rastlantısal olarak yapılmıştır. Gerçek ortamda ölçüm amaçlı olarak aynı veriler kullanılacaktır. Herhangi bir durumda verilerde bir değişiklik olursa sadece bu dosyaların içeriği değiştirilerek benzetim verisi tekrar elde edilebilecektir.

4.7.1 Giriş dosyaları

“aes_key.txt” : Şifreleme amaçlı kullanılan 128 bitlik AES anahtarı bu dosyada tutulur. Veri onaltılık düzende girilmelidir; harflerin büyük küçük olması önemli değildir ancak fazla veya eksik girilmemesi gereklidir.

```
CB 49 06 49 C9 B1 B3 35 BA 4E 4D 6F 5F BC C5 AB
```

“plain.txt” : Şifrelenecek 128 bitlik blokların tutulduğu dosyadır. Her bir blok baytlar şeklinde ve düzgün olarak girilmelidir. Şifrelenecek blok sayısı önemli değildir. “Plain.txt” dosyası içinde bloklar sıra ile okunur ve şifreleme yapılarak çıkış dosyalarına yazılır. Programlar sırada bekleyen okunacak blok olduğu sürece amaçlanan analizi yaparak çıktılarını üretecektir. Dikkat edilecek noktalardan biri de her bir bloğun farklı bir satıra yazılmasıdır.
Ekte bir örnek dizi verilmiştir:

```
1D EA D0 FA 92 41 62 24 ED 3B B6 85 BF D2 0A 58
99 98 B6 B9 98 15 06 2A 4B 29 32 A5 00 73 D0 51
9D 06 3E 93 6E 5E 3D 23 00 91 5B 28 31 05 78 1A
B9 1A 6A 43 D8 DA 60 01 C5 40 EB C6 BC 2B 24 F8
0D 6B F0 71 36 D8 A2 7F B4 CA 71 3C 42 BC 50 4F
5A 06 B4 C7 3E 71 F9 42 84 2E BB 5D D8 5B 35 90
70 28 13 53 78 52 29 3D 2B 86 85 8E 81 22 26 97
A1 4A 0C 10 57 30 9B 0B B0 33 40 09 6A C1 FD E1
F8 E6 36 91 F9 C7 AA 81 36 BD 4C EB 05 A4 74 76
F3 DC 7C 2D 6A 42 55 8C 4E 4B 2D 9C AB FE 73 44
.......
```

4.7.2 Yazılan Programlar

4.7.2.1 AES Şifreleme

“Aes_bul.c” programı bu amaçla yazılmıştır. Verilen “plain.txt” karşılığında birebir çıkış dosyasının oluşturulması amaçlanmıştır. Temel şifreleme programıdır. Ölçüm düzeneği için input hazırlamak amacıyla yazılmıştır. Kodun algoritması sadece şifreleme amaçlı olarak yazılmıştır. Standart AES algoritmasını gerçekler. Anahtarı “aes_key.txt” dosyasından, şifrelenecek blokları ise “plain.txt” dosyasından okur; programın çalışması sonunda girişlere karşı düşen çıkışları içeren “register.txt” adlı dosyayı oluşturur.

4.7.2.2 Şifre Çözme

"Aes_desc.c" programı bu amaçla yazılmıştır. Daha sonraki çalışmalara yardımcı olması amacıyla yazılmıştır. Şifrelenmiş 128 bitlik blok için şifre ile giriş bloğunu oluşturur. Temel olarak şifre çözme işlemini gerçekleştirmektedir. Amaçlanan tez için gerekli değildir.

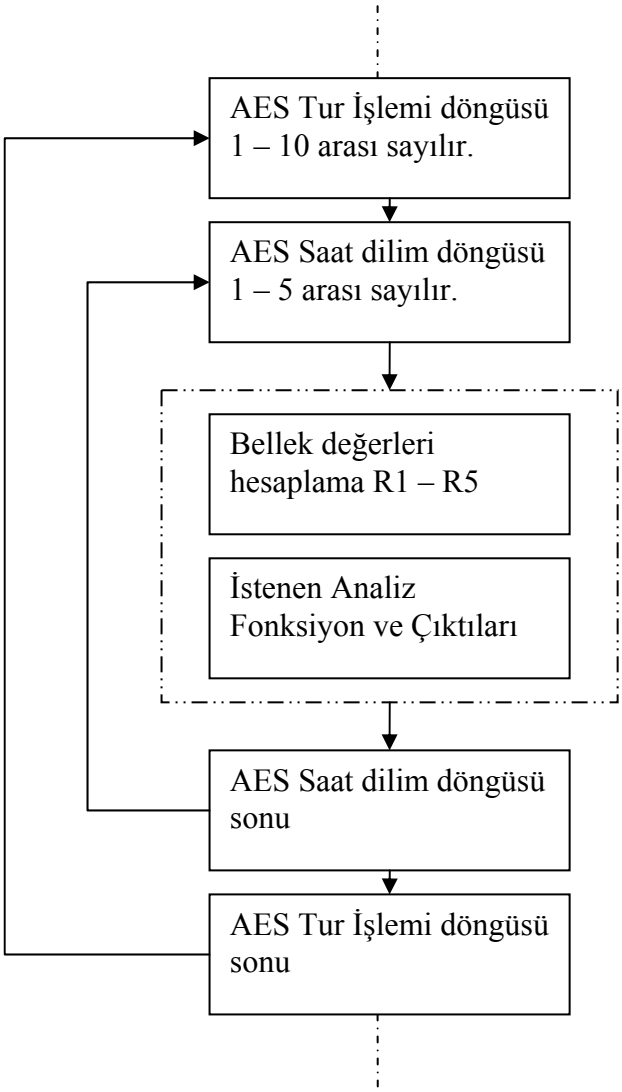
4.7.2.3 Bellek içerik gösterimi

Saldırı yapılacak modeller için sırasıyla "aes_rgv1.c", model 1 için, "aes_key1.c" ise model 2 için yazılmıştır. Programlar, "aes_key.txt" dosyasından şifreleme anahtarını alır, daha sonra sırası ile "plain.txt" dosyasından giriş değerlerini alarak şifreler.

Saldırı yapılacak olan AES algoritması gerçeklemesinin yapısı gözönüne alarak devre elemanlarının modellendiği programlardır. Sadece AES algoritmasının çıkışı olan şifre değil, aynı zamanda tur ve saat dilimi düşünülerek yazılımın algoritması oluşturulmuştur. Şekil 4.2 ve Şekil 4.3'de gösterilen R1, ... R5 belleklerinin her saat darbesinde değerleri bulunacak şekilde yazılım gerçeklenmiştir. Tur sayısı 10 ve her turdaki saat dilimi 5 olacak şekilde her tur ve saat diliminde bellek içerik hesaplamaları yapılmıştır. Her bir giriş için şifreleme yapıldıktan sonra, bir sonraki giriş değeri okunur ve şifreleme işlemi son giriş değerine kadar devam eder. Şifreleme işleminde bellek içerikleri bir sonraki işlemi etkileyecektir.

Şekil 4.5'de algoritma akışı verilmiştir. Programın akışında her bir işlem bloğu girişindeki bellek içeriğini alarak kendi işlevini yapacak şekilde çalışmakta ve çıkış belleğine sonucu aktarmaktadır. Bu nedenle program AES algoritmasının gerçeklemesi gibi çalışmaktadır. 10 tur ve 5 saat dilimi sonunda R5 belleğin içeriğinde şifrelenmiş çıkış değeri elde edilmektedir.

Model 2 için yazılan "aes_key1.c" programında ayrıca anahtar üreteci davranışı da eklenmiştir. 10. tur boyunca R5 içeriğinin sabit kaldığı bilinmektedir. Koda bu davranış da konularak R5 içeriğinin 10. tur boyunca sabit olması sağlanmıştır.



Şekil 4.5: Devre yapısı için geliştirilen algoritma

Bu programın en önemli çıktısı, tüm tur ve zaman dilimlerinde kontrol amaçlı olarak oluşturulan, belleklerin önceki ve sonraki değerleridir. Çıktılar, "sonuclar.txt" dosyasına yazılır. Bu şekilde tüm çalışma süresince devre içindeki belleklerin içerikleri görülmüştür. Aşağıda bu programın çıktısı verilmiştir:

round 0
time clock 1

r1 register 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
36 94 c5 ec ba ef b0 82 46 cc a3 d b6 1d 45 64

r2 register 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63

r3 register 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

r4 register 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

r5 register 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2b 7e 15 16 28 ae d2 a6 ab f7 15 88 9 cf 4f 3c

time clock 2

r1 register 36 94 c5 ec ba ef b0 82 46 cc a3 d b6 1d 45 64
36 94 c5 ec ba ef b0 82 46 cc a3 d b6 1d 45 64

r2 register 36 94 c5 ec ba ef b0 82 46 cc a3 d b6 1d 45 64
5 22 a6 ce f4 df e7 13 5a 4b a d7 4e a4 6e 43

r3 register 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63
63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63

r4 register 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
r5 register 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2b 7e 15 16 28 ae d2 a6 ab f7 15 88 9 cf 4f 3c

time clock 3

r1 register 36 94 c5 ec ba ef b0 82 46 cc a3 d b6 1d 45 64
36 94 c5 ec ba ef b0 82 46 cc a3 d b6 1d 45 64

r2 register 36 94 c5 ec ba ef b0 82 46 cc a3 d b6 1d 45 64
5 22 a6 ce f4 df e7 13 5a 4b a d7 4e a4 6e 43

r3 register 5 22 a6 ce f4 df e7 13 5a 4b a d7 4e a4 6e 43
5 df a 43 f4 4b 6e ce 5a a4 a6 13 4e 22 e7 d7

r4 register 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63
63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63

r5 register 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2b 7e 15 16 28 ae d2 a6 ab f7 15 88 9 cf 4f 3c
.....

Bu yapı daha sonraki çalışmaların da altyapısını oluşturacaktır.

4.7.2.4 Durum Geçişi Analizi

Bu amaçla "aes_rgv2.c" yazılmıştır. "Aes_rgv1.c" versiyonunda bulunmuş olan bellek değerleri bizim için altyapıyı oluşturan değerlerdir. "aes_rgv2.c" bu değerleri kullanarak saldırı amaçlı analiz kısımlarını eklediğimiz versiyondur. Her tur ve tüm saat dilimlerinde bellek değerleri bir önceki versiyonda anlatıldığı şekilde bulunmaktadır. Bu program ise her tur, saat diliminde çalışan ve belleklerin bir önceki değeri ile o anki değeri arasındaki geçiş sayılarını bularak uygun toplam tur değerini bulan versiyondur.
Belleklar arası geçişte 4 farklı durum sözkonusudur:

0 >> 0
0 >> 1
1 >> 0
1 >> 1

Tüm bu geçiş sayıları ayrı ayrı değişkenlerde bulunmuştur. Ancak bizim saldırı algoritmamızda anlamlı olan 0 >> 1 geçişidir. Diğer geçiş rakamları olası ileri çalışmalar için eklenmiş olup anlamlı olmaları durumunda kullanılmak üzere kodda gösterilmiştir. Ancak bizim saldırı algoritmamızda kulanılmayacaklardır.

Geçiş sayılarını bulmak için koda iki adet bayt giriş alan ve bunlar arasındaki geçiş sayılarını bulan fonksiyon eklenmiştir. Kodu aşağıdadır.

```
void gec_say(void)
{
mx = 0x80;
for (ix = 0; ix<8; ix++)
{
   if (gec_x & mx)
      xx[ix] = 1;
   else
      xx[ix] = 0;

   if (gec_y & mx)
      yy[ix] = 1;
   else
      yy[ix] = 0;
   mx = mx >> 1;
}
s10 =0;
s00 =0;
s01 =0;
s11 =0;

for (ix = 0; ix<8; ix++)

{
   if (xx[ix] == 0 & yy[ix] == 0)
      s00 ++;
   if (xx[ix] == 1 & yy[ix] == 0)
      s01 ++;
   if (xx[ix] == 0 & yy[ix] == 1)
      s10 ++;
   if (xx[ix] == 1 & yy[ix] == 1)
```

```
        s11 ++;
}
}
```

Her tur ve saat dilimi için çalışma sonrasında bellek içeriği ile bir önceki değeri arasındaki tüm değişim sayıları bu fonksiyon ile elde edilmektedir.

Bu versiyonda da AES anahtarı tur boyunca tüm saat dilimlerinde olması gereken tur değerinde olduğu varsayılmıştır. Programın en başında 10 tur için anahtar dizisi elde edilmektedir ve "key_sch.txt" adlı dosyaya kontrol amaçlı olarak yazılmaktadır. Anahtar dizisi, tüm tur ve saat dilimlerinde kullanılmaktadır.

4.7.2.5 Saldırı Noktası Durum Geçiş Analizi Sonuç Çıktıları

Saldırı yapılacak modeller için sırasıyla "aes_rgv4.c", model 1 için, "aes_key4.c" ise model 2 için yazılmıştır.

Bir önceki versiyonun kontrol çıktılarının olmadığı ve sadece kullanılacak olan "r2_ileri.txt" dosyasının oluşturulduğu versiyondur. Bu versiyon, saldırı amaçlı çıktılardan birini üreten versiyondur.

Burada en önemli nokta; saldırı noktamız olan 1. tur, 2. saat dilimine ait rakamları çıktı olarak ürettiğimizdir. Seçilen bu saldırı noktasının özelliği bu turda belleklerin içerdiği değerlerin değişimi incelendiğinde, sadece giriş anahtarının etkilediği R2 belleğinde değişiklik olmasıdır. Ayrıca bizim saldırı amaçlı olarak AES anahtarının sadece en yüksek anlamlı 8 bitini değiştirmemiz, 1. tur 2. saat dilimi sonunda sadece R2'nin 8 bitini etkileyecektir. Bu şekilde anahtar ile R2 belleğindeki değişim arasında ilişki olduğunu ve bu ilişkinin güç harcamasında da görülebileceği söyleyebiliriz.

Aslında tüm saat dilimleri için üretilen geçiş sayıları sadece bu saat dilimi için çıktı dosyasına işlenir. Bu sayede, başka bir saat dilimini görmek istersek, programda bunlar da hazır vaziyettedir. Tek yapılacak olan, bu değerlerin çıkış dosyasına "r2_ileri.txt" yazılacak şekilde analiz kısmı sonuna eklemektir.

“Aes_rgv4.c” ve “Aes_key4.c” programları kullanılarak, AES anahtarının en yüksek anlamlı baytı, 0x00 – 0xff değerleri arasında değiştirilerek çıktı değerleri hesaplanır. Bu hesaplama, programın beklediği giriş dosyalarından “aes_key.txt” dosyasının içeriğinin değiştirilmesi ve programın tekrar çalıştırılması ile elde edilir. Her anahtar için program çalışmasının çıktıları saklanarak korelasyon analizi için giriş verileri oluşturulur.

Aynı kod kullanılarak R3 bellek içeriğine 1. tur ve 3. saat dilimi içinde saldırı yapılabileceği görülmüştür. Kodlarda yapılan bir değişiklikle elde edilen verilerde bu noktanın da saldırıya açık olduğunu göstermiştir. Programların bu versiyonları ayrıca ekte verilmiştir.

Model 1 için “r3_gec.c” programı, 0>1 geçişlerini sayan programdır. Model 2 için “M2_r3_4.c” programı, 0>1 geçişlerini sayan programdır.

4.7.2.6 Tüm Bellek Durum Geçiş Analizi Sonuç Çıktıları

Saldırı yapılacak modeller için sırasıyla “aes_rgv5.c”, model 1 için, “aes_key5.c” ise model 2 için yazılmıştır.

Bu programlarda bir tam AES şifreleme döngüsü için hesaplamalar yaptırıldı. Sadece bir adet giriş bloğu için tüm tur ve saat dilimlerinde tüm belleklar için toplam sıfırdan bire (0 >> 1) geçişler sayılmıştır. Program giriş dosyası “plain.txt” dosyasında mevcut olan girşler için işlemleri yapacaktır, bu nedenle program çalıştırılmadan önce bu dosyada sadece bir adet giriş olduğu kontrol edilmelidir. Sonuçta elde edilen sayı bir bellek için değil, tüm bellekler için toplam olacaktır. Program çalışması sonucunda 50x1 boyutlarında bir matris elde edilir.

4.7.2.7 Saldırı Noktası Durum Bit Analizi Sonuç Çıktıları

Saldırı yapılacak modeller için sırasıyla “aes_rgv6.c”, model 1 için, “aes_key6.c” ise model 2 için yazılmıştır.

Şifreleme anahtarı “aes_key.tx”t dosyasından, giriş değerleri ise “plain.txt” dosyasından alınarak kullanılır. Bu programlarda R2 belleği için tur 1, saat dilimi 2'de sadece en yüksek anlamlı bayt içindeki 1 olan bitlerin sayısını bulmak amaçlanmıştır.

Tüm açık mesajlar için çalışır. Anahtara ait en yüksek anlamlı 8 bit, 0x00 – 0xff arasında değiştirilerek R2 belleğinin değişen 8 bitindeki 1 sayısı sayılır. Değiştirilen her anahtar değeri için 1 olan bit sayıları matrisi korelasyon analizi için girdi olacak şekilde saklanır.

Burada amaç, anahtarın sadece değişen 8 biti için, R2 belleğinin 8 bitinin değiştiğini bilmemizden dolayı bir korelasyon olabileceğini öngörmemizdir. Sadece 1 olan bitleri saymak yerine sıfırdan bire geçişleri saymak bizim için çok daha iyi bir korelasyon vereceğini söyleyebiliriz. Çünkü 1 olan bitleri saymak ve bu sayı ile güç harcaması arasında korelasyon aramak bize % 50 daha fazla ölçüm yapmak ihtiyacını doğuracaktır. 1 olan bitin bir önceki saat diliminde de değeri 1 olabilir. Yani burada bir güç harcaması olmamış demektir. Biz bu konuda daha fazla ölçüm yaparak bu sorunu aşmayı planlıyoruz. Bir bellek içerisindeki 1 olan bitlerin sayısını bulmak için aşağıdaki kod yazılmıştır:

```
void bir_say(void)
{
mx = 0x80;
for (ix = 0; ix<8; ix++)
{
   if (gec_x & mx)
      xx[ix] = 1;
   else
      xx[ix] = 0;
   mx = mx >> 1;
}
s10 =0;
s00 =0;
s01 =0;
```

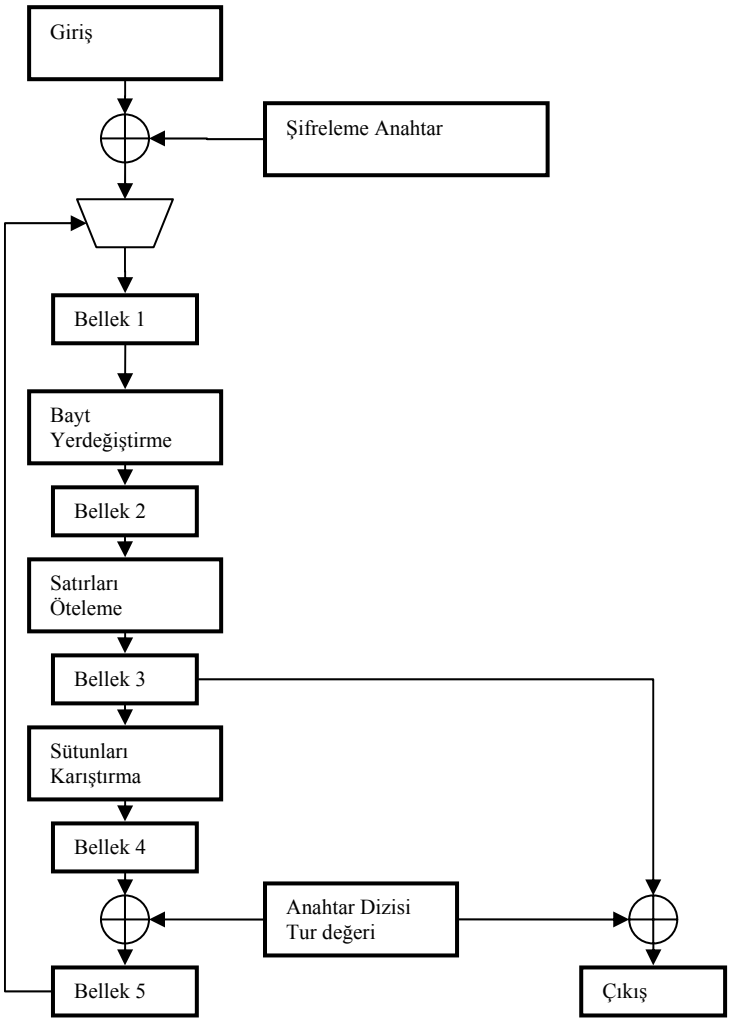
```
s11 =0;
for (ix = 0; ix<8; ix++)
{
   if (xx[ix] == 0)
      s00 ++;
   if (xx[ix] == 1)
      s11 ++;
}
}
```

Bu fonksiyon, 1. tur ve 2. saat diliminde şifreleme anahtarının değiştirilen 8 biti karşılığında, R2'nin değişen 8 bitindeki 1 sayılarını bulmak için kullanılır. Bulunan sayılar çıkış dosyası, "r2_ileri.txt" a yazılır.

Aynı kod kullanılarak R3 belleği içeriğine 1. tur ve 3. saat dilimi içinde saldırı yapılabileceği görülmüştür. Kodlarda yapılan bir değişiklikle elde edilen verilerde bu noktanın da saldırıya açık olduğunu göstermiştir. Model 1 için "A_r3t1c3.c" programı en yüksek anlamlı 8 bit içinde 1 değerindeki bit sayısını sayan programdır. Model 2 için "M2_r3_6.c" programı en yüksek anlamlı 8 bit içinde 1 değerindeki bit sayısını sayan programdır.

4.8 Farksal Güç analizi saldırısı, verilerin kullanımı

DPA saldırısı için hedef, Şekil 4.6'da gösterilen R2'nin en yüksek anlamlı 8 biti olacaktır.



Şekil 4.6: AES algoritması için öngörülen bellek düzeneği resmi

Bu algoritma için saldırı noktamızı belirlememizdeki en büyük etken R2'nin şifreleme için kullanılan ana anahtardan doğrudan etkilenmesidir. Giriş değeri ile XOR işlemine tabi tutulan anahtarın işlem sonucu R1 belleğinde saklanır. R2 ise bu değerin bayt yer değiştirme işlemi sonucunu içerir ve içerdiği değer anahtarın en yüksek anlamlı 8 bitine doğrudan bağlıdır. Bu amaçla bu noktada yani R2'deki güç harcamasının benzetiminin yapılmasına karar verildi. Saldırı öncesi benzetim verisi ile bu düzeneğin ve öngörünün çalışacağının; yani R2'ye yapılacak saldırının başarılı olacağının gösterilmesi amaçlanmaktadır.

Dinamik güç harcamasının benzetiminin yapılması için algoritmasının gerçeklemesinin yapısı biliniyor. Bu gerçeklemeye yönelik iki adet model öngörüldü. Modeller AES için benzetimi yapılan saldırı başarılı olursa; gerçek ortamda ölçülen

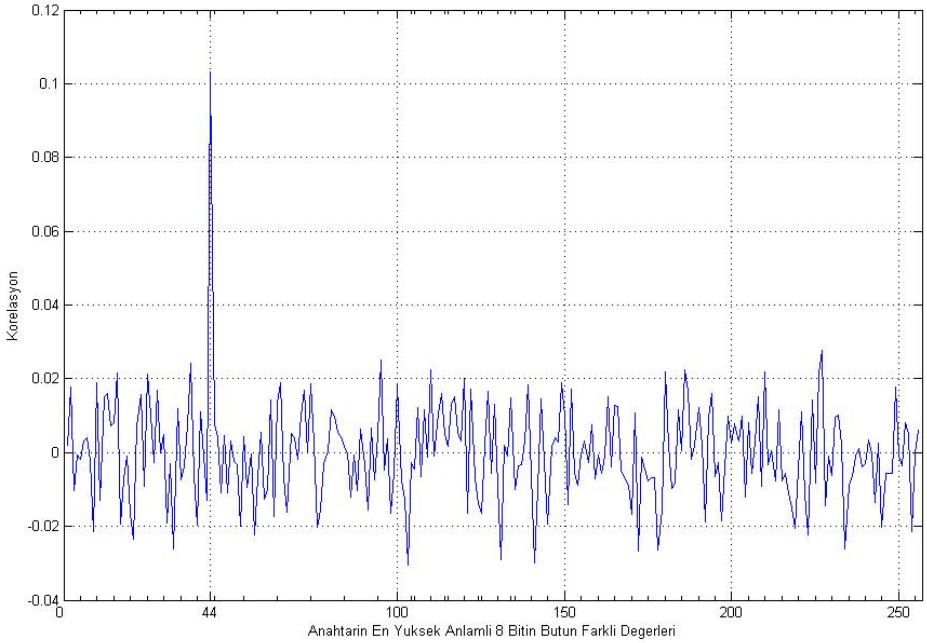
verilere yapılacak saldırının da başarılı olabileceğini söylemek ve gerçek saldırı için hazırlık yapmak anlamlı olur.

Benzetim verileri ile saldırı için oluşturulan ilk dosya güç harcamalarını içeren dosyadır. Bunun için N adet rastgele giriş (plain text) ve bir adet sabit anahtar (rastgele seçilmiş) oluşturulmuştur. Bunlar "aes_key.txt" ve "plain.txt" adlı dosyalarda saklanmaktadırlar. Analiz için yazılan kodlar bu dosyaları giriş olarak okuyacak şekilde yazılmıştır.

İlk olarak AES şifrelemesinde mevcut olan 10 tur için, donanımdaki beş adet belleğin önceki ve şimdiki durum geçişleri sayıldı. Bunlardan analizde kullanılmak üzere 1. tur ve 2. saat diliminde sıfırdan bire (0 > 1) geçiş sayıları sonuç dosyasına yazdırıldı. Bulunmak istenen anahtar için ve tüm girişler için yapılan hesaplamalar ile Nx1 boyutlu bir matris üretildi. Anahtarın en yüksek anlamlı sekiz biti için 0x00 ve 0xff değerleri arasındaki tüm değerler için bu hesaplamalar tekrarlandı ve Nx256 boyutlu bir matris üretildi. Bunun için yazılan kod, "aes_rgv4.c", "aes_key4.c" adlı C programlarıdır.

İkinci olarak oluşturulan analiz dosyasında, sadece R2 belleği için tur 1'de en yüksek anlamlı bayt içinde değeri 1 olan bitlerin sayısı bulundu. Tüm giriş değerleri (plain.txt) için çalıştırıldı. Ayrıca anahtarın en anlamlı 8 biti için 0x00 ve 0xff arasında değişecek şekilde hesaplamalar tekrarlandı.

Her iki dosyada da oluşturulan verilerle orijinal anahtarla oluşturulan veriler birbiriyle uyumlu olması gerekmektedir. Bu ikisi arasındaki korelasyon, diğer anahtar değerleri ile yapılan ölçümlerdeki korelasyondan daha anlamlıdır. Şekil 4.7'de deneme yoluyla en anlamlı 8 biti değiştirilmiş anahtarlar için korelasyon analizi görülmektedir.

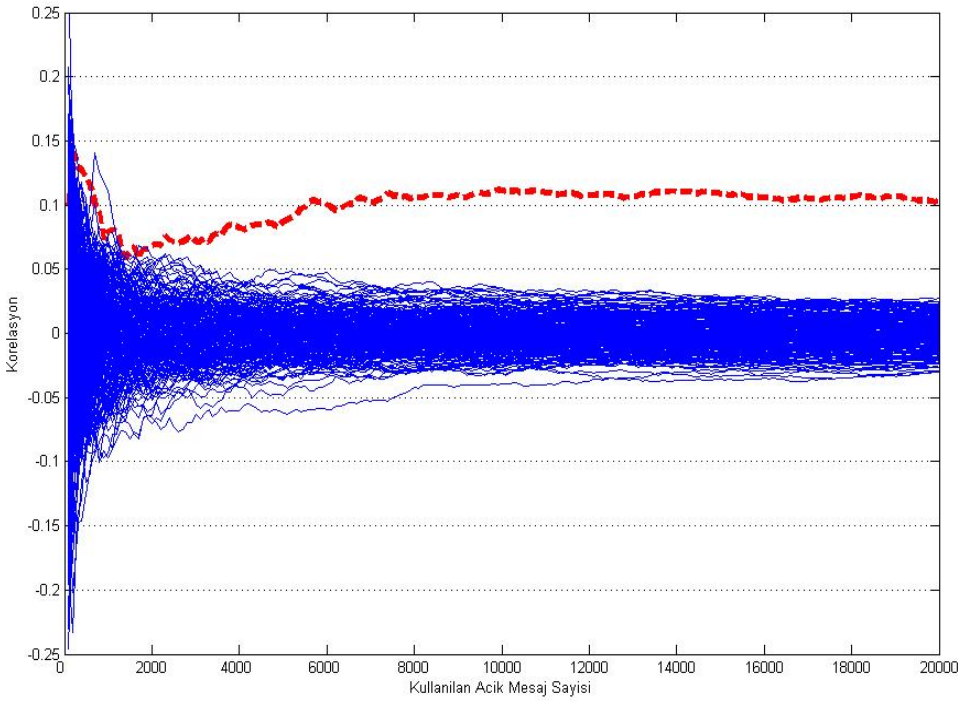


Şekil 4.7: R2 , Model 1 için Anahtarın En yüksek Anlamlı Bitinin korelasyon değerleri

Bu şekilde 0x00 – 0xff arası değişen anahtarlar ve korelasyonlarının grafiği Şekil 4.6'da verilmiştir. Beklenen, aranan anahtar ile korelasyonun diğerlerine göre daha yüksek değerde olmasıdır. Şekil 4.7'de 44 değerinde yani anahtarın 2B değerinde korelasyonun en yüksek olduğun gösterir. Bu sonuç şunu göstermektedir; saldırı amaçlı tasarlanan model, saldırı için düşünülen bellek, seçilen tur ve saat dilimi uygundur. Benzetim verileri ile olan saldırı, gerçek ortamda ölçülecek değerleri karşılayabilecektir.

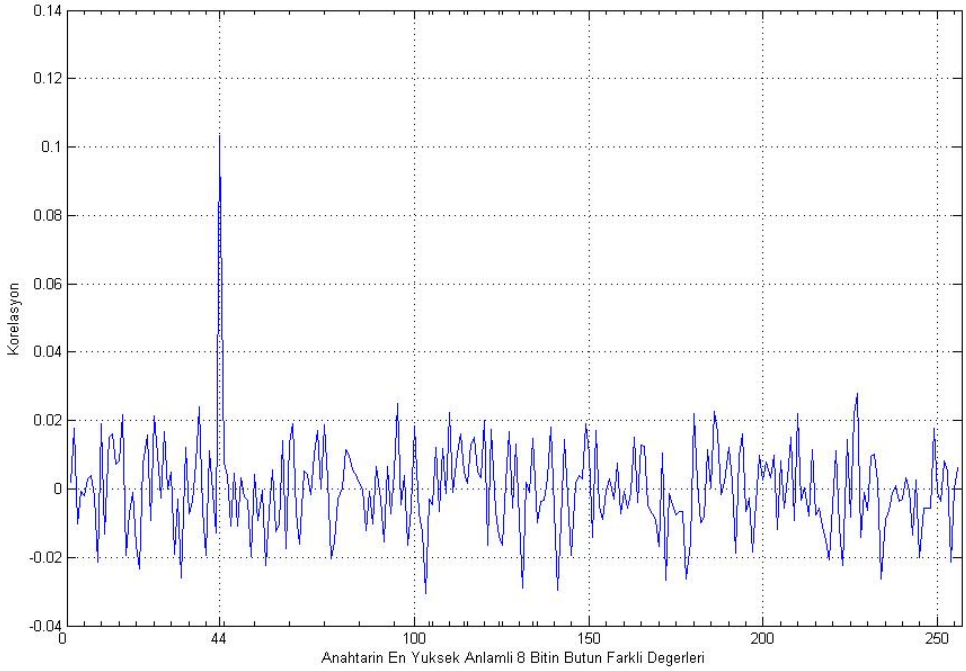
Bu aşamadan sonra karşılaşılacak sorun ne kadar deneme ile devre içinde saklanmış olan anahtarın bulabileceğidir.

Şekil 4.8'de farklı açık mesaj sayısı kullanılarak anahtarın mümkün bütün değerileri için korelasyon değişimi gösterilmiştir. Bu nedenle en anlamlı 8 biti bulmak için benzetim yapılmış saldırıda 4000 adet ölçüm yeterli olacaktır denebilir.



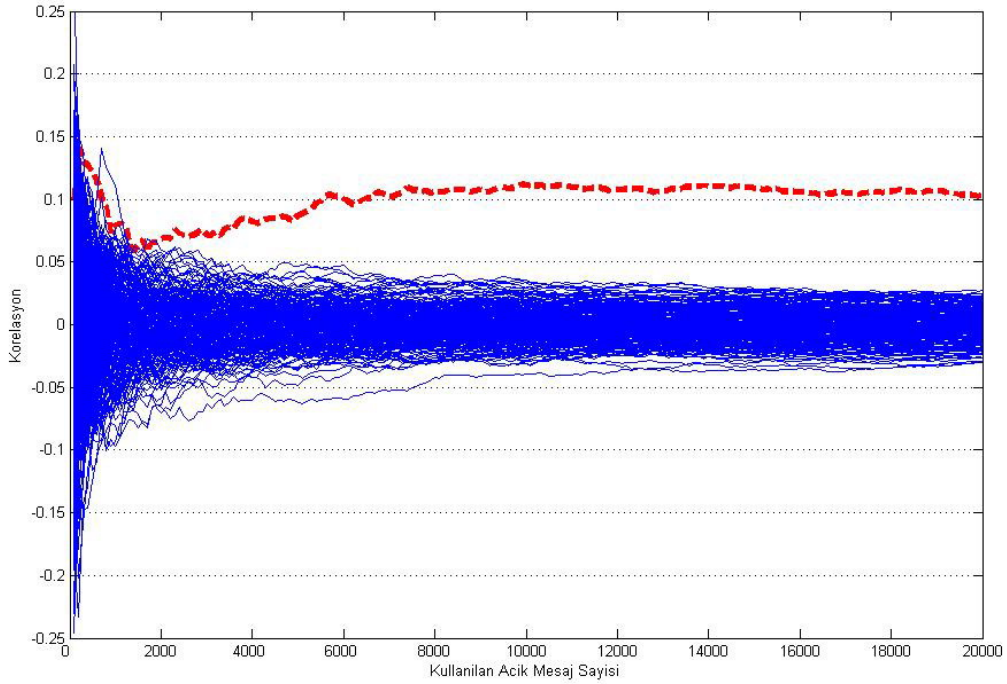
Şekil 4.8: R2 , Model 1 için Kullanılan mesaj sayısının korelasyon bağlantısı

Aynı sebeple Bellek 3 için de saldırı yapılabileceği görülmüştür. Saldırı zamanı olarak birinci turun 3. saat dilimi öngörülmüştür. Bu konuda analiz programları kullanılarak başka bir saldırı noktası arayışına girilmiş ve R2 için yapılan çalışma R3 için de yapılmıştır. Sonuç olarak oluşturulan verilerle korelasyon analizi yapılınca, bu noktada da saldırının mümkün olabildiği görülmüştür. Aşağıda Şekil 4.9'da korelasyon analizi sonuç grafikleri mevcuttur.



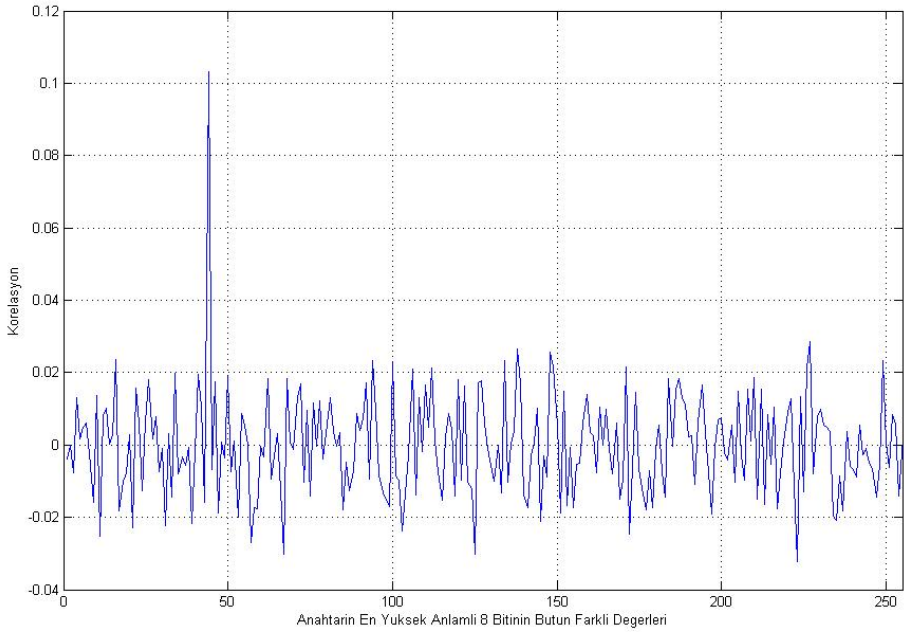
Şekil 4.9: R3 , Model 1 için Anahtarın En yüksek Anlamlı Bitinin korelasyon değerleri

Korelasyon analizi – plaintext sayısı arasındaki grafik Şekil 4.10'da verilmiştir.



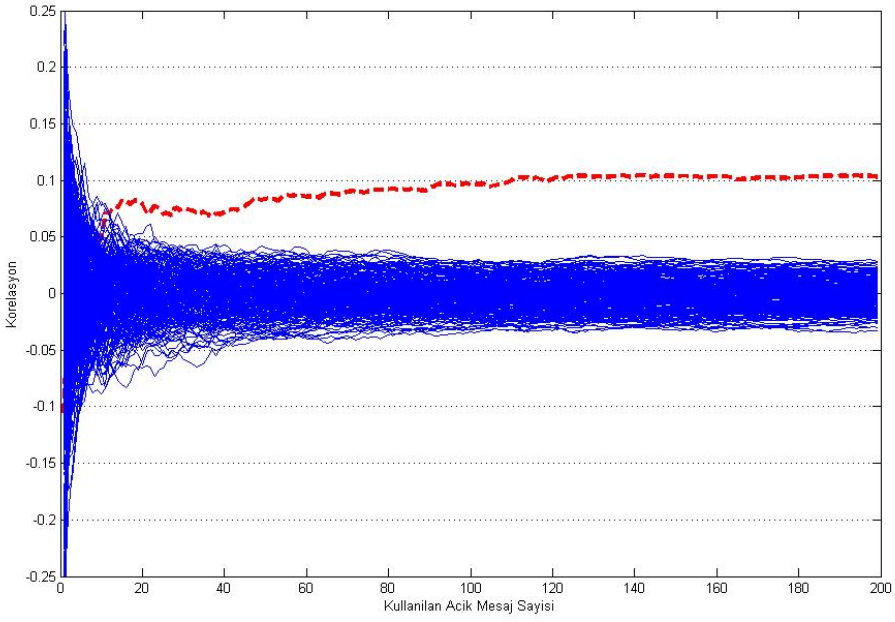
Şekil 4.10: R3 , Model 1 için kullanılan mesaj sayısının korelasyon değerleri

Model 2 için hesaplamalar neticesinde R2 belleği için ekteki Şekil 4.11'de korelasyon analizi sonuç grafikleri mevcuttur:



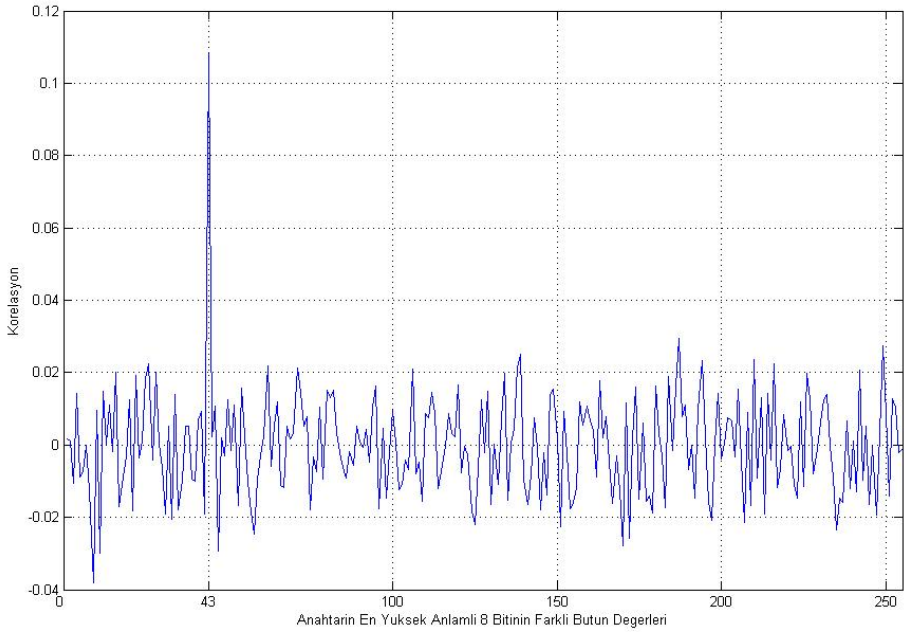
Şekil 4.11: R2 , Model 2 için Anahtarın En yüksek Anlamlı Bitinin korelasyon değerleri

Model 2'nin R2 için Korelasyon analizi–plaintext sayısı korelasyonu Şekil 4.12'dedir.



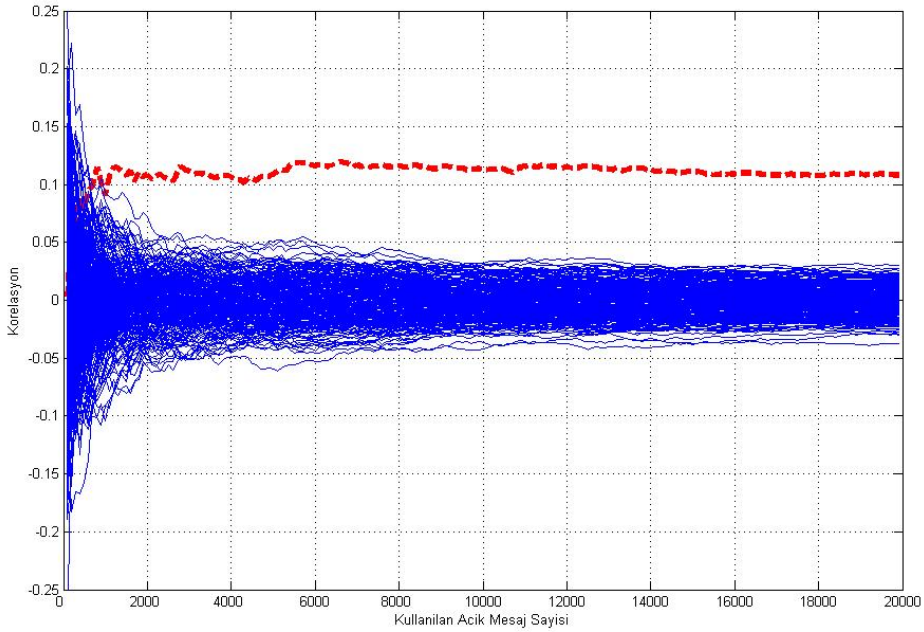
Şekil 4.12: R2 , Model 2 için kullanılan mesaj sayısının korelasyon değerleri

Model 2 için, 1 tur içinde 3. saat diliminde R3 belleğine saldırı da anlamlı değerler korelasyon içermektedir. Ekteki şekil 4.13'de bu görülebilir.



Şekil 4.13: R3 , Model 2 için Anahtarın En yüksek Anlamlı Bitinin korelasyon değerleri

Model 2'nin R3 için Korelasyon analizi–plaintext sayısı korelasyonu Şekil 4.14'dedir.



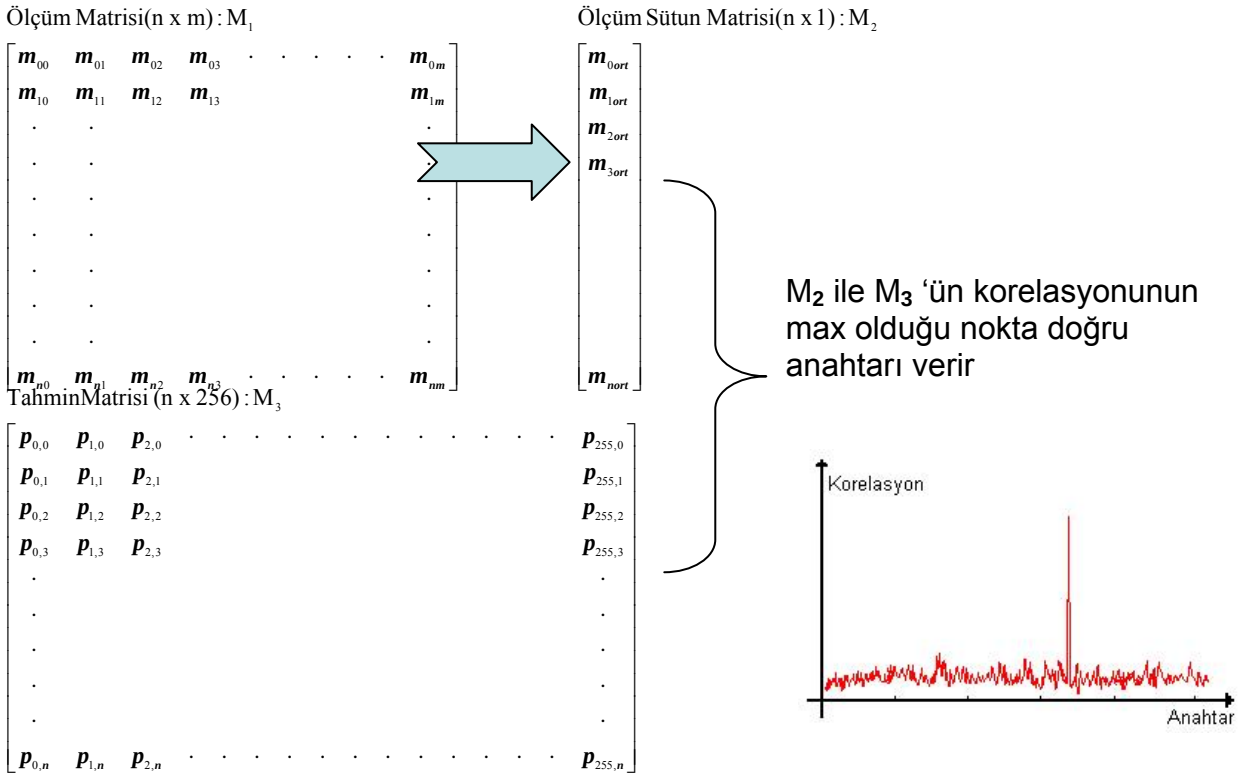
Şekil 4.14: R3 , Model 2 için kullanılan mesaj sayısının korelasyon değerleri

4.9 Farksal Güç Analizi saldırısı, ölçüm verileri kullanımı

Bu bölümde DPA saldırısı benzetim yaptığımız ve benzetim sonucu ölçüm verisiyle saldırı yapılabileceğini gördüğümüz modelin gerçek ortam ölçüm verileri ile tekrarlayacağız.

Bölüm 4 de benzetim amaçlı kullandığımız N adet plaintext için şifreleme işlemini gerçekleştiririz.

Ölçüm detayları sonucunda elde edilen matris, tahmin sonuçları ile korelasyona tabi tutulur, bu işlem sonucunda ölçüm sonuçları ile en yüksek korelasyonu veren bayt, aradığımız anahtar bayt bilgisidir sonucuna ulaşırız. Şekil 4.15'da bu konudaki ilişki verilmiştir.



Şekil 4.15: Benzetim Verisi ile gerçek ölçümün birleştirilmesi ve korelasyon eldesi

5. SONUÇLAR VE TARTIŞMA

Konu ile ilgili olarak en önemli nokta, Türkiye'de bu konuda ilk defa bu kapsamda çalışma yapılmış olmasıdır. Yaklaşım tarzı olarak oluşturulan algoritmalar ve yazılan programlar Türkiye ötesinde dünyada da ilk defa uygulandığı söylenebilir.

Tez kapsamında yapılan çalışmalarda AES algoritmasının FPGA gerçeklemesinde güç analizi ile algoritmanın en önemli bileşeni olan şifreleme anahtarının eldesinin mümkün olduğunu gördük, gerçeklemelerde farksal güç harcamalarına karşı önlem alınmasının gerekli olduğu elde ettiğimiz sonuçlardan görülmektedir.

Yazılım dili olarak C programlama dili kullanılmıştır. C dili , mühendislik uygulamalarında bit seviyesinde sağladığı operatörleri nedeniyle ve hız açısından diğerlerinden üstünlüğü nedeniyle tercih edilmiştir. Analiz datalarından Matlab ile korelasyon analizi yapılmıştır.

Yazılan programlar ve elde edilen sonuçlar, AES gerçeklemesinin güç analizi saldırılarına karşı korumalı olarak yapılması gerekliliğini göstermiştir. Gerçeklemeler farksal güç analizi saldırısına karşı korumalı olarak tasarlanmalıdır.

KAYNAKLAR

[1] **Yerlikaya, T., Buluş, E., Arda, D.**, 2004. AES Aday Şifreleme Algoritmalarının Yazılım ve Donanım Performans Karşılaştırması ve Uygulamalar, *Elektrik Elektronik Bilgisayar Mühendisliği Sempozyumu (ELECO 2004)*, Bursa-TÜRKİYE, s. 1

[2] **Şahin A., Buluş E., Sakallı T.**, 2005. Modern Blok Şifreleme Algoritmalarının Gücünün İncelenmesi, *II. Mühendislik Bilimleri Genç Araştırmacılar Kongresi,*İ.Ü. İstanbul, 17 – 19 Kasım, s. 4

[3] **J¨org J. Buchholz,** 2001. Matlab Implementation of the Advanced Encryption Standard report , Hochschule Bremen Almanya s. 5 - 28

[4] **Gladman B.**, 2002. A Specification for Rijndael, the AES Algorithm v3.3, 1 Mayıs 2002., s. 2 - 20

[5 **Federal Information Processing Standards Publication.**, 2001. Announcing the Advanced Encryption Standart (AES) report, Kasım 2001, s. 18 - 34

[6] **Loidreau P.**, 2002. Kriptografiye Giriş, Linux Magazine dergisi, Mayıs 2002, http://www.linuxfocus.org/Turkce/May2002/article243.shtml

[7] **Ors S., Gürkaynak F., Oswald E., Preneel B.**,2004. Power-Analysis Attack on an ASIC AES implementation, *International Conference on Information Technology: Coding and Computing (ITCC'04)* Volume 2, s. 546

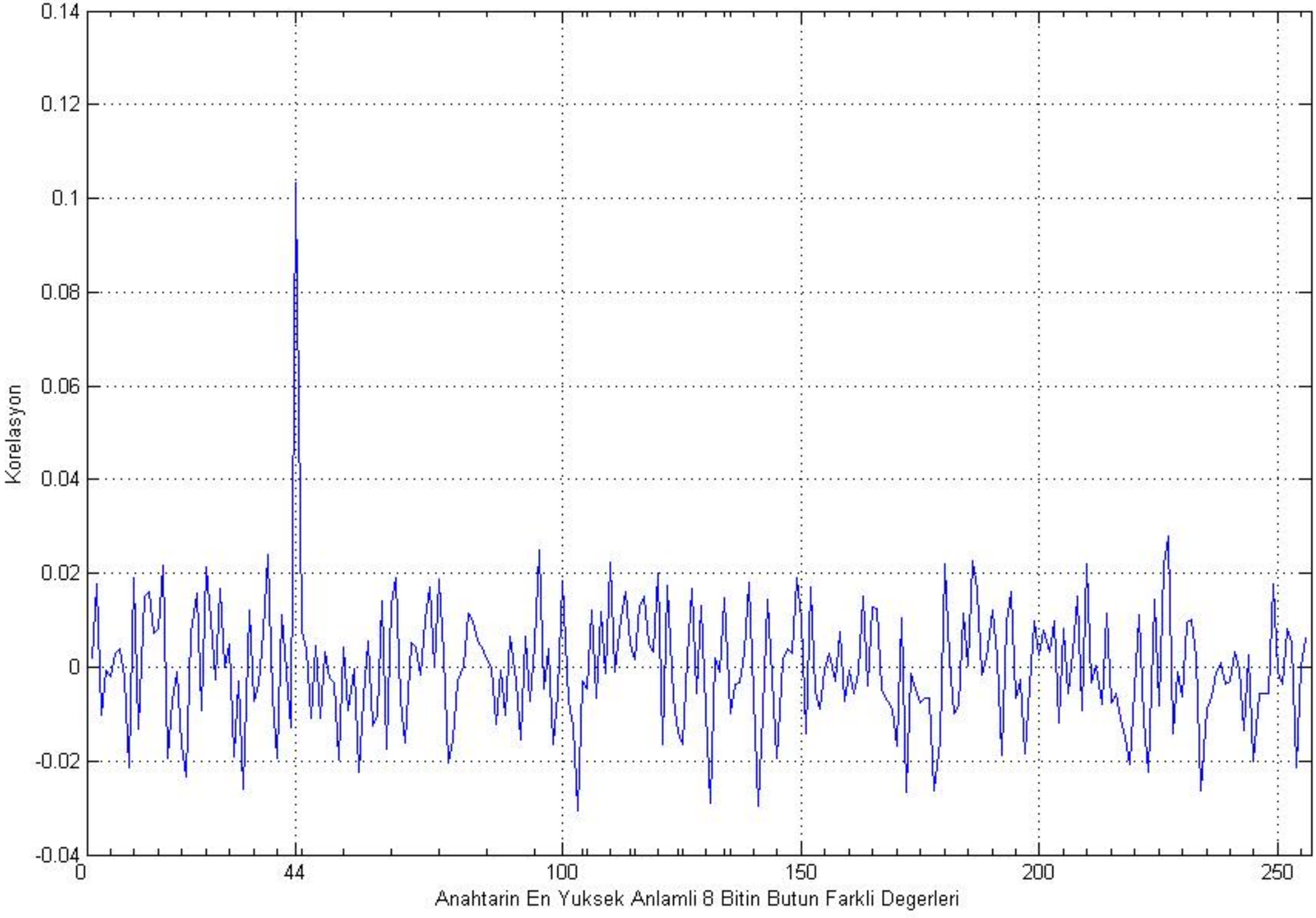
[8] **P. Kocher.**, 1996.Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In N. Koblitz, editor,*Advances in*

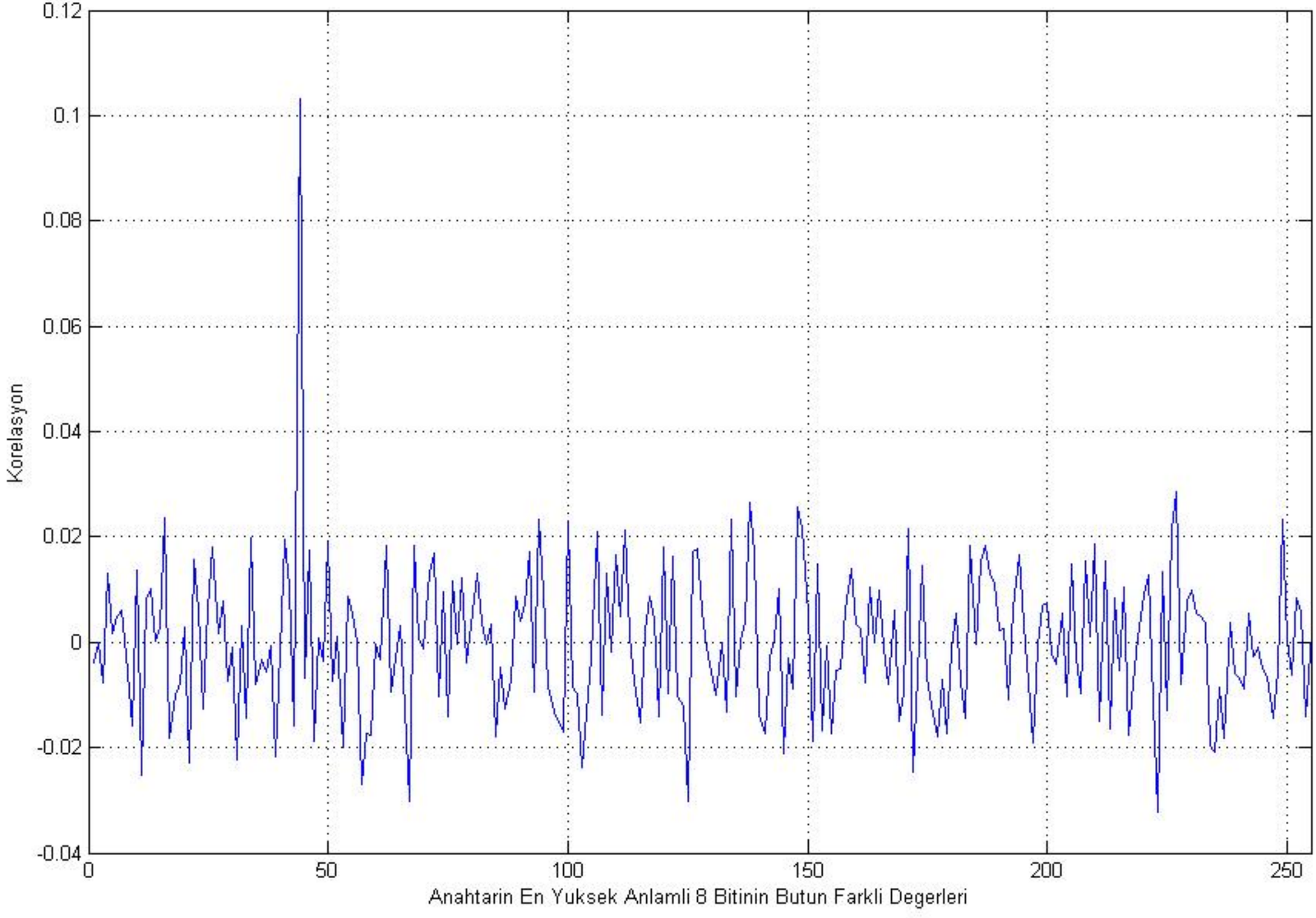
Cryptology: Proceedings of CRYPTO'96,number 1109 in Lecture Notes in Computer Science, s.104–113, Santa Barbara, CA, USA, Springer-Verlag.

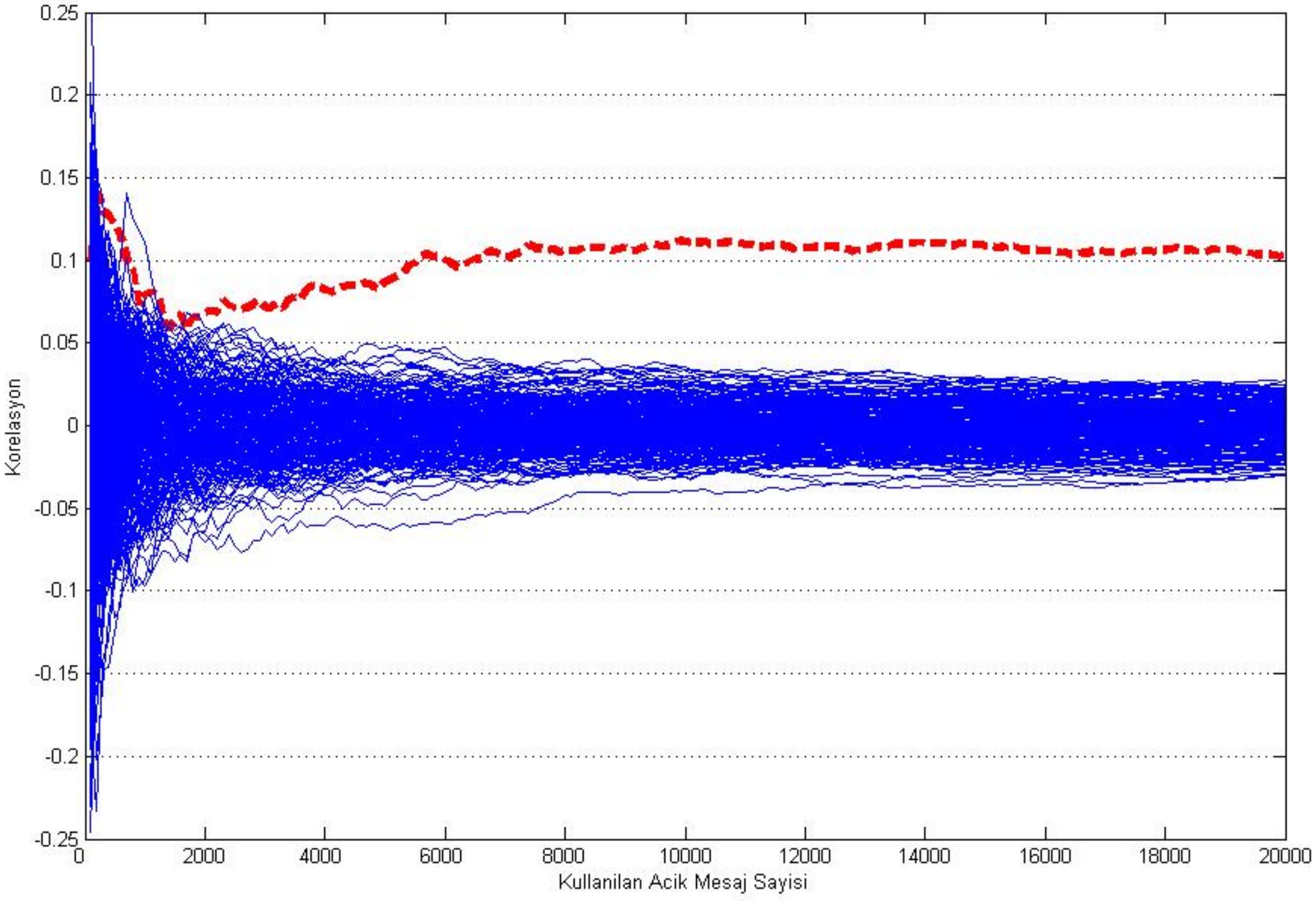
[9] **P. Kocher, J. Jaffe, and B. Jun.**,1999. Differential power analysis.In M. Wiener, editor, *Advances in Cryptology: Proceedingsof CRYPTO'99*, number 1666 in Lecture Notes in ComputerScience, s. 388–397, Santa Barbara, CA, USA, Springer-Verlag.

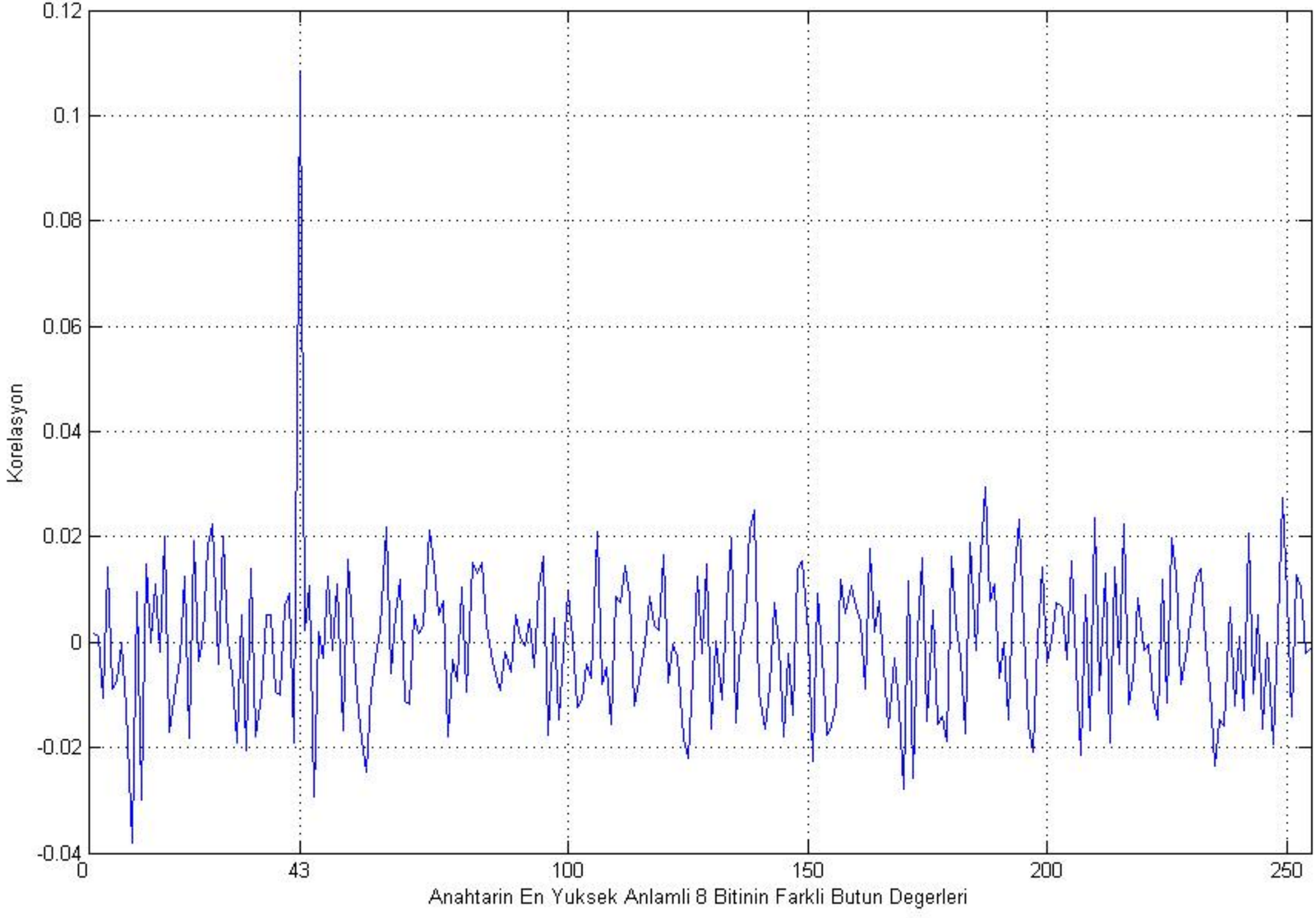
ÖZGEÇMİŞ

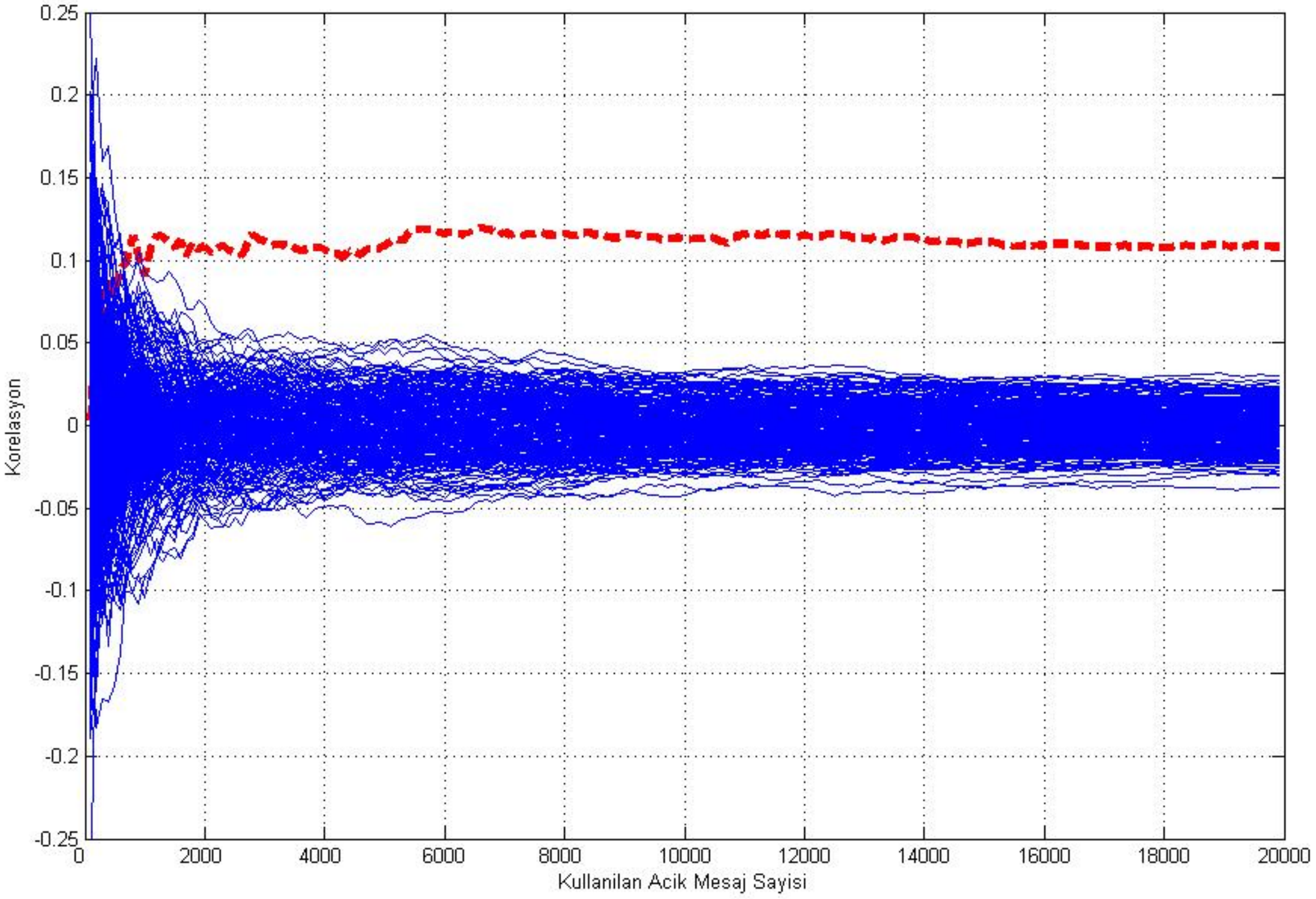
14.06.1973 Suluova/Amasya doğumlu Hakan Kayış, ilk öğrenimini bu şehirde tamamladıktan sonra orta ve lise öğrenimini Samsun Anadolu Lisesi'nde tamamladı. 1991 yılında İstanbul Teknik Üniversitesinde Elekronik ve Haberleşme Mühendisliği eğitimine başladı. İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Bölümünde yüksek lisans eğitimine devam etti. Sırasıyla TEM Elektronik (8 bitlik mikroişlemcilerle sistem tasarımı), AB Develioğlu (değişik tıp cihazları teknik destek ve satış), Pamukbank (yazılım geliştirme), Dışbank (yazılım geliştirme), Turkcell (test mühendisi) firmalarında çalıştı. Meslek yaşamına halen Turkcell'de devam etmektedir.

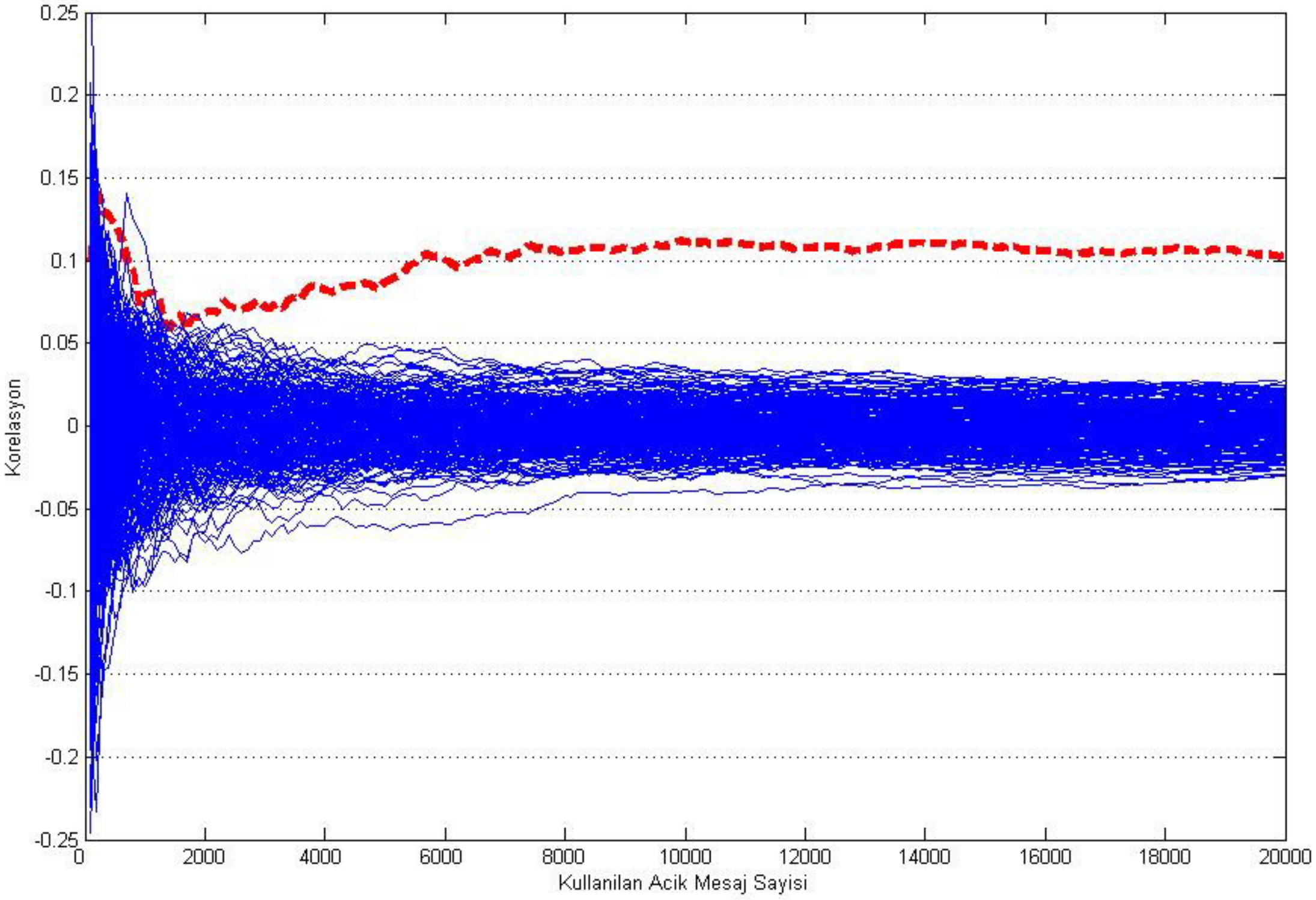

0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
Korelasyon
0
44
100
150
200
250
Anahtarin En Yuksek Anlamli 8 Bitin Butun Farkli Degerleri

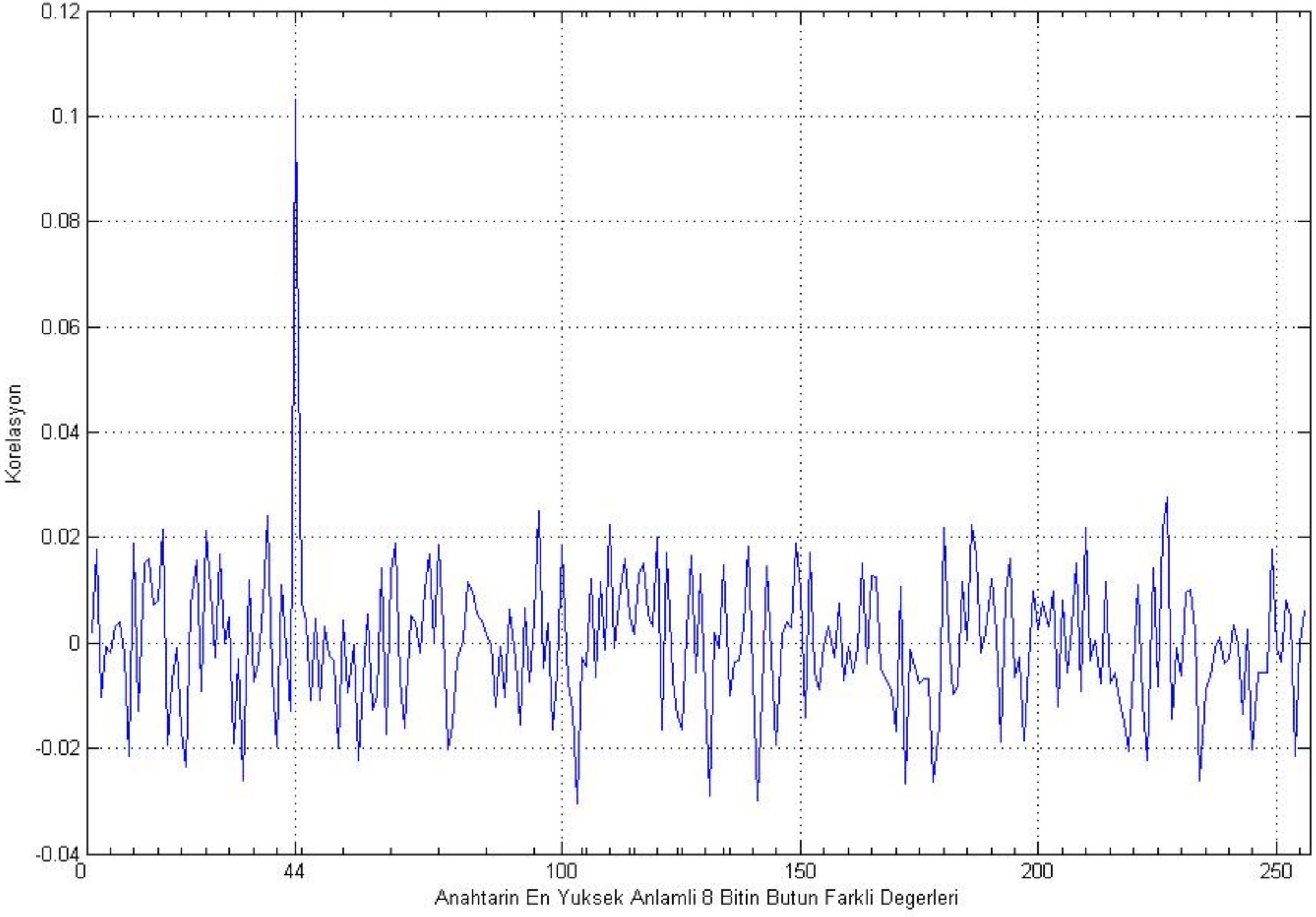

0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
Korelasyon
0
50
100
150
200
250
Anahtarin En Yuksek Anlamli 8 Bitinin Butun Farkli Degerleri

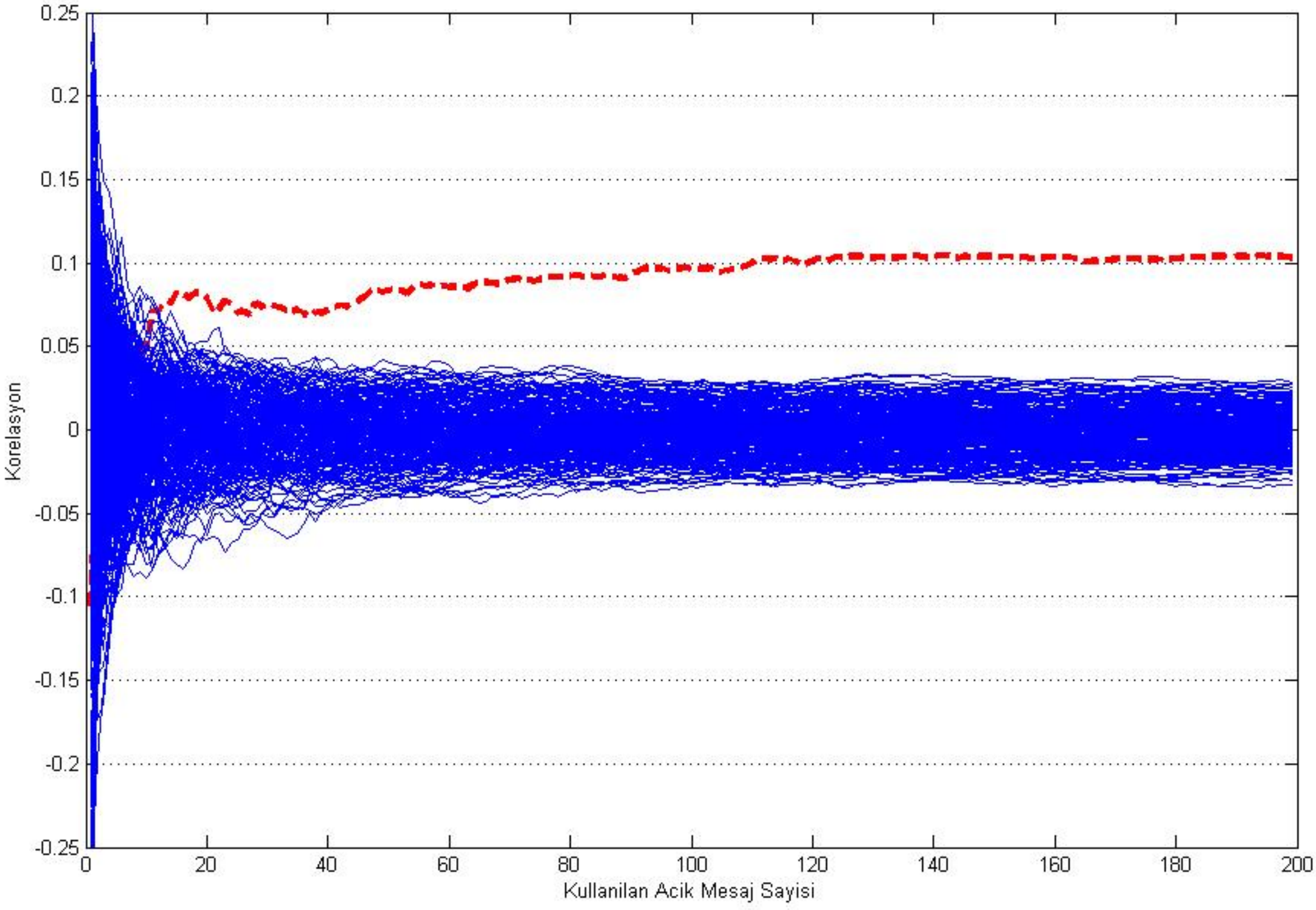

0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
Korelasyon
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
Kullanilan Acik Mesaj Sayisi


0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
Korelasyon
0
43
100
150
200
250
Anahtarin En Yuksek Anlamli 8 Bitinin Farkli Butun Degerleri


0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
Korelasyon
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
Kullanilan Acik Mesaj Sayisi


0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
Korelasyon
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
Kullanilan Acik Mesaj Sayisi


0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
Korelasyon
0
44
100
150
200
250
Anahtarin En Yuksek Anlamli 8 Bitin Butun Farkli Degerleri


0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
Korelasyon
0
20
40
60
80
100
120
140
160
180
200
Kullanilan Acik Mesaj Sayisi