T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AES ALGORİTMASININ BİR GERÇEKLEMESİNE GÜÇ ANALİZİ SALDIRILARI

MUHAMMET ÖZTEMÜR

YÜKSEK LİSANS TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI
HABERLEŞME PROGRAMI

DANIŞMAN
PROF. DR. TÜLAY YILDIRIM

İSTANBUL, 2012

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AES ALGORİTMASININ BİR GERÇEKLEMESİNE GÜÇ ANALİZİ SALDIRILARI

Muhammet ÖZTEMÜR tarafından hazırlanan tez çalışması 21.06.2012 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Prof. Dr. Tülay YILDIRIM

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Prof. Dr. Tülay YILDIRIM

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Sıddıka Berna ÖRS YALÇIN

İstanbul Teknik Üniversitesi _____

Yrd. Doç. Dr. Nihan KAHRAMAN

Yıldız Teknik Üniversitesi _____

ÖNSÖZ

Çalışmalarım boyunca bana yol gösteren, çalışma hayatımda da faydasını göreceğim bilgi güvenliği gibi insan hayatı için önemli olan bir konuda çalışmam yönünde beni teşvik eden hocam Prof. Dr. Tülay YILDIRIM'a teşekkür ederim.

Tez sürecimde, uzman görüşlerini belirterek ve çalışmalarımı değerlendirmek suretiyle bana yol gösteren hocam Doç. Dr. Sıddıka Berna ÖRS YALÇIN'a teşekkür ederim.

Yine tez süreci boyunca; deneyimlerini benimle paylaşarak çalışmalarıma büyük destek olan çalışma arkadaşlarım Yük. Müh. Muhammet ŞAHİNOĞLU ve Yük. Müh. Ebru Akalp KUZU'ya teşekkürü bir borç bilirim.

Ayrıca tüm yüksek lisans sürecim boyunca madden ve manen yanımda olan annem Melahat ÖZTEMÜR'e teşekkürlerimi sunarım.

Son olarak, uzun tez mesailerime gösterdiği anlayış ve çalışmalarıma verdiği destekle, her zaman yanımda hissettiğim eşim Sündüs Büşra ÖZTEMÜR'e teşekkürü bir borç bilirim.

Haziran, 2012

Muhammet ÖZTEMÜR

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

AES	Advanced Encryption Standard
BGA	Basit Güç Analizi Saldırıları
DPA	Differantial Power Analysis
FIPS	Federal Information Processing Standards
FGA	Farksal Güç Analizi Saldırıları
FPGA	Field Programmable Gate Array
IAIK	Angewandte Informationsverarbeitung und Kommunikationstechnologie
NIST	National Institute of Standards and Technology
RMS	Root Mean Square
RSA	Rivest Shami Adleman
JTAG	Joint Test Action Group
VHDL	VHSIC Hardware Description Language
CMOS	Complementary Metal Oxide Semicondctor
GF	Galois Field

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

AES ALGORİTMASININ BİR GERÇEKLEMESİNE GÜÇ ANALİZİ SALDIRILARI

Muhammet ÖZTEMÜR

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Prof. Dr. Tülay YILDIRIM

Kriptografik algoritmalar bilgi güvenliğinin sağlanması için yaygın olarak kullanılırlar. AES (Advanced Encryption Standard)algoritması bu amaçla kullanılan simetrik blok şifreleme algoritmalarından biridir.

AES, Kasım 2001'de elektronik verinin saklanması için kullanılmak üzere Amerikan Ulusal Standartlar ve Teknoloji Enstitüsü (National Institute of Standards and Technology (NIST)) tarafından yayınlanmış bir kriptografik algoritmadır. Yayınlandığı tarihten itibaren kriptografik sistemlerde yaygın olarak kullanılmıştır. AES algoritması bilginin şifrelenmesi ve şifrelenmiş bilginin çözülmesi için kullanılan bir simetrik blok kodlayıcıdır.

AES algoritması, matematiksel olarak güçlü olması nedeniyle klasik kriptanalize karşı dayanıklıdır. Ancak algoritmanın matematiksel yapısı ile ilgilenmeyen başka bir saldırı tekniği mevcuttur. Bu teknik yan kanal analizi saldırılarıdır.

Yan-kanal atakları kriptografi alanında çalışanlar tarafından 1996'da zamanlama analizi konusunda yayınlanan ilk makale ile temel bir tehdit olarak görülmeye başlanmıştır. Bu saldırılarda, kriptografik sisteme fiziksel bir müdahale yapılmaz. Saldırgan sistemin çalışması esnasında ürettiği verileri analiz eder. Eğer bu veriler, sistemin gizliliğini sağlayan parametreler hakkında bilgi veriyorsa bunlara Yan Kanal Bilgisi denir.

Yan Kanal Analizi saldırıları, kullandıkları yan kanal bilgisine göre sınıflandırılırlar. Bunlar; güç analizi saldırıları, zamanlama analizi saldırıları, elektromanyetik analizi saldırıları, akustik analizi saldırılarıdır. Bu saldırılar içerisinde en güçlü olanı, güç analizi

saldırılarıdır. Güç analizi saldırıların bir kısmı kriptografik sistemin tükettikleri güç ile gerçekleştirdiği işlem adımları arasındaki ilişkiyi analiz eder. Bu tür saldırılara Basit Güç Analizi ismi verilir. Diğer bir güç analizi saldırısında ise kriptografik sistemin tükettikleri güç ile işledikleri veri arasındaki ilişki analiz edilir. Bu saldırı tekniğine Farksal Güç Analizi (Differantial Power Analysis (DPA)) ismi verilir. Özellikle DPA tekniği çok yaygın olarak kullanılmaktadır. AES algoritması üzerine de DPA saldırılar gerçeklenmiştir.

AES algoritması uygulamalarını DPA saldırılarına karşı korumak için bazı önlemler önerilmiştir. Bu önlemlerden birisi maskeleme işlemidir. Maskeleme işleminde, algoritma tarafından üretilen ara değerlere rastgele veriler eklenir. Böylece saldırgan, sistemin ürettiği ara değerler ile tüketilen güç arasında bağlantı kuramaz. Bu maskeleme tekniklerinden biri de Oswald tarafından önerilen IAIK maskesidir. IAIK maskesi AES uygulamalarında yaygın olarak kullanılmaktadır. Ancak maskeli kriptografik sistemler içinde farklı saldırı teknikleri önerilmiştir. Bu saldırılar farklı durumlar ve sistemler için uygulanabilirdir. Ayrıca her bir saldırının etkinliği farklı durumlar için değişiklik göstermektedir. Tez çalışması kapsamında bu saldırı teknikleri, FPGA sistem üzerinde gerçeklenmiş Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie (IAIK) maskeli AES algoritması üzerinde pratik olarak gerçeklenecektir. Daha sonra gerçek sonuçlar kullanılarak her bir saldırı tekniğinin etkinliği analiz edilecek ve birbirleri ile karşılaştırılacaktır.

Anahtar Kelimeler: Yan kanal saldırıları, farksal güç analizi, IAIK maskesi.

ABSTRACT

POWER ANALYSIS ATTACKS ON AN IMPLEMENTATION OF AES ALGORITHM

Muhammet ÖZTEMÜR

Department of Electronics and Communications Engineering

MSc. Thesis

Advisor: Assoc. Prof. Dr. Tülay YILDIRIM

Cryptographic Algorithms are widely used in order to ensure information security. AES (Advanced Encryption Standard) algorithm is one of the algorithms of symmetric block encryption that are used for this purpose.

The Advanced Encryption Standard (AES) is a cryptographic algorithm that published by National Institute of Standards and Technology (NIST) to protect electronic data in November 2001. It is widely used in cryptographic systems as from the date it is approved. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher)information.

AES algorithm, since it is mathematically powerful, is resistant to classical cryptoanalysis. However, there exists another attack technique that doesn't care mathematical structure of the algorithm. This technique is side-channel analysis attacks.

Side-channel attacks are considered as a fundamental threat by those who study about cryptography thanks to the first article about timing analysis published in 1996. No physical intervention is carried out to the cryptographic system in those attacks. The attacker analyzes data produced by the system during its runtime. If those data informs about parameters ensuring the confidentiality of the system, this information is called Side-channel Information.

Side-channel analysis attacks are classified with respect to the side channel information that they use. Those are power analysis attacks, timing analysis attacks, electromagnetic analysis attacks and acoustics analysis attacks. The most powerful among them is power analysis attacks. A portion of power analysis attacks analyzes the relationship between the power that the cryptographic system consumes and the process that it carries out. This attack technique is called Simple Power Analysis (DPA).In another power analysis attack; however, the relationship between the power that the cryptographic system consumes and the data it processes is analyzed. This attack technique is called Differential Power Analysis (DPA). Especially DPA technique is pretty widely used. DPA attacks also implemented on AES algorithm.

Some methods are offered to protect AES algorithm application against DPA attacks. One of these methods is masking application. In application of masking, Random data is added to intermediate values produced by the algorithm. That way, the attacker cannot make a connection between the intermediate value that the system produces and the consumed power. One of these masking techniques is IAIK Masking that proposes by Oswald. IAIK Masking is widely used in AES applications. However, different attacking techniques are offered for masked cryptographic systems as well. Those attacks are applicable for different situations and systems. Moreover, efficiency of each attack varies for different situations. Within this thesis, those attack techniques will be practically applied for a IAIK Masked AES Algorithm that implemented on an FPGA system. Later on, efficiency of each attack technique will be analyzed using real outcomes and they will be compared to each other.

Keywords: Side channel attacks, differantial power analysis, IAIK mask.

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Güvenli haberleşme tarih boyunca insanoğlunun ihtiyaç duyduğu önemli bir gereksinimdir. Bu gereksinimin karşılanması için güvenli sistemlerin kurulması gerekmektedir. Kriptografik algoritmalar bu güvenli sistemlerin en önemli parçasını oluşturmaktadır. Bu nedenle kriptografik algoritmaların güvenli bir şekilde geliştirilmesi ve uygulanması sistemlerin güvenliğinin sağlanması açısından kritik önem taşımaktadır.

İlkel kriptoloji uygulamalarında çoğu zaman kriptografik algoritmanın gizliliği önem taşımakta ve dolayısıyla sistemlerin güvenliğine yönelik saldırılarda kriptografik algoritmayı ele geçirmek önem taşımaktaydı. Ancak Kerchhoff tarafından temeli atılan modern kriptoloji uygulamalarında algoritmanın gizliliği değil şifrelemede kullanılan anahtarın gizliliği temel alınmaktadır [1]. Bu nedenle bir kriptografik sisteme yapılan saldırıda amaç gizliliği sağlayan anahtar bilgisini ele geçirmektir.

Kriptografik sistemlere yönelik saldırılar iki şekilde yapılabilmektedir. Bunlar matematiksel olan saldırılar ve gerçeklemeye özgü saldırılardır [2], [3]. Matematiksel saldırılarda sistem algoritma düzeyinde değerlendirilmekte ve sistemdeki matematiksel yapıdan kaynaklı zayıflıklar araştırılmaktadır. Gerçeklemeye özgü saldırılarda sistemin doğrudan ürettiği sonuçları kullanmak yerine genellikle istemsiz ürettiği çıkışlar kullanılmaktadır. Bu tür saldırılar aktif ve pasif olarak iki grupta toplanmaktadır.

Aktif saldırılarda, sistemin yapısına doğrudan müdahale edilerek ölçümler yapılmakta [4] (ölçme saldırıları) veya sistem çalışması anında müdahale edilerek oluşturulan hatalardan [5]

(hata oluşturma saldırıları) faydalanılmaktadır. Bu saldırı türünün en çok uygulanan örneği lazer istasyonları yardımı ile yapılmaktadır. Özellikli hatalar yaptırılarak bütün saldırılarda yapıldığı gibi gizli anahtar elde edilmeye çalışılmaktadır.

Pasif saldırılar, yan kanal saldırıları olarak da adlandırılmakta ve algoritmayı gerçekleyen sistemin ürettiği bazı istemsiz çıkışların (zamanlama, güç tüketimi, elektromanyetik radyasyon, ses gibi) gizli bilgiyle ilişkili olmasından faydalanılmaktadır. Bu istemsiz çıkışların içerdiği bilgiler çeşitli ölçümlerle elde edilip, bunlardan faydalanılarak gizli bilgiye ulaşılmaya çalışılmaktadır. Bu saldırılar, ilk olarak Kocher tarafından 1996'da başarılı şekilde uygulamıştır[2]. Yan kanal saldırıları, kullandıkları yan kanal bilgisine (gizli bilgiyle ilişkilendirilebilen istemsiz çıkışlar) göre, dört gruba ayrılmıştır; zamanlama analizi saldırıları, güç analizi saldırıları, elektromanyetik analizi saldırıları ve akustik analizi saldırıları. Bu saldırılar içerisinden en etkin olarak kullanılan güç analizi saldırılarıdır.

Yan kanal analizi saldırıları; gerçeklenmesi için çok özel sistemlere ihtiyaç duymaması ve uygulamaya bağlı olarak matematiksel güçlülüğü garanti olarak görülen kriptografik algoritmalara karşı açık bir tehdit oluşturması bakımından önemli bir bilimsel çalışma alanı oluşturmuştur.

Kasım 2001'de, Gelişmiş kodlama standardı (Advanced Encryption Standard (AES)) Amerikan Ulusal Standartlar ve Teknoloji Enstitüsü (National Institute of Standards and Technology (NIST)) tarafından federal bilgi işleme standardı (Federal Information Processing Standards (FIPS)) olarak yayınlanmıştır [6], [7]. Daha önce kullanılan Veri Kodlama Standardı (Data Encryption Standart (DES)) [8] algoritmasının yerini almıştır.

Bu tarihten itibaren AES algoritması kriptografik sistemlerde yaygın olarak kullanılmaya başlanmıştır. Bu nedenle AES algoritmasının uygulamaları üzerine de yan kanal analizi saldırıları gerçeklenmiş ve başarılı sonuçlar alınmıştır [9]. Bunun üzerine [10] çalışmasında olduğu gibi AES algoritmasını yan kanal analizi tehdidine karşı korumak için yöntemler önerilmiştir.

Ancak ilerleyen süreçte bu yöntemlere karşı da farklı saldırı türleri ortaya çıkmıştır[11], [12], [13]. Anlatılan süreçten de görüldüğü gibi çalışmalar; farklı saldırı tekniklerinin önerilmesi,

bunlara karşın önlemlerin bulunması ve önlemlere karşı yeni saldırı tekniklerinin önerilmesi şeklinde devam etmektedir.

1.2 Tezin Amacı

AES algoritması; güvenlik, performans, gerçekleme kolaylığı gibi özellikleri nedeniyle yaygın olarak kullanılmakta ve buna bağlı olarak ta birçok saldırı tehdidinin hedefi olmaktadır. Bunun üzerine AES algoritması uygulamalarını yan kanal analizi tehdidine karşı koruyacak güvenlik önlemlerinin geliştirilmesi üzerine çalışmalar yapılmıştır. Bu kapsamda algoritmanın işlediği verilerin saldırgan tarafından bilinmeyen rastsal sayılar ile karıştırılarak gizlenmesi esasına dayanan farklı maskeleme yöntemleri önerilmiştir. Bu maskeleme yöntemleri içerisinde; güvenlik, performans ve uygulanabilirlik açısından IAIK maskesi öne çıkmıştır [10].

Ancak AES algoritmasının maskelenmesinde yaygın olarak kullanılan IAIK maskesinin sağladığı korumanın aşılması içinde farklı saldırı teknikleri önerilmiştir. Tez çalışması kapsamında; FPGA üzerinde gerçeklenmiş ve IAIK maskesi ile güvenliği sağlanmış AES algoritması üzerine literatürde önerilen teknikler ile saldırılar gerçeklenecek ve bu saldırıların etkinliği analiz edilecektir.

1.3 Orijinal Katkı

AES algoritmasının IAIK maskesi ile korunarak kullanılması uygulamasına, güvenlik amaçlı sistemlerde yaygın olan kullanılması nedeniyle, birçok saldırı yöntemi tanımlanmış ve gerçeklenmiştir. Ancak; konu ile ilgili önerilen farklı yöntemlerin gerçeklenebilmesi farklı koşullara bağlıdır. Ayrıca bu yöntemlerin etkinlik seviyeleri farklı olup, bu etkinlik düzeyi uygulamanın türüne göre değişmektedir.

Literatürde gerçekleştirilen saldırılar incelendiğinde, elde edilen başarıların ölçüm sayısı dikkate alınarak diğer saldırı yöntemleri ile karşılaştırılması ile değerlendirildiği görülmektedir. Veya aynı saldırı yöntemi, farklı sistemler için uygulanarak başarı karşılaştırması yapılmaktadır. Ancak bu karşılaştırmalarda, temel kriter olarak çoğunlukla saldırının başarıya ulaşması için kullanılan ölçüm sayısının dikkate alındığı diğer etmenlerden satır arası ayrıntılar olarak bahsedildiği görülmektedir. Bunlara örnek olarak [13] ve[14] çalışmaları verilebilir.

Ayrıca her bir çalışmada, karşılaştırma işlemine tabi tutulan diğer saldırıların sınırlı olduğu bu çalışmada olduğu kadar kapsamlı olmadığı görülmektedir. Sonuç olarak, literatür araştırması yapıldığında tam olarak bu çalışmaya benzeyen bir çalışmaya rastlanmamıştır.

Tez çalışması kapsamında; FPGA üzerinde gerçeklenmiş ve IAIK maskesi ile güvenliği sağlanmış AES algoritması üzerine literatürde önerilen teknikleri belirlenecek ve bu yöntemlerin her biri sistem üzerinde uygulanacaktır. Elde edilen sonuçlara göre her bir saldırının başarı düzeyi belirlenerek birbirleri ile karşılaştırılacaktır. Bu karşılaştırma esnasında saldırganın başarıya ulaşabilmesi için ihtiyaç duyacağı gereksinimler dikkate alınacaktır. Çalışma sonunda *IAIK Maskeli AES Algoritması'*na yapılacak bir güvenlik analizi için yönlendirici bir rehber tablo sunulacaktır.

BÖLÜM 2

ÖN BİLGİLER

Bu bölümde, çalışma boyunca kullanılacak olan matematiksel kavramlar tanıtılacaktır. Bunun yanı sıra kullanılan terminoloji açıklanacak ve kısaca kriptografik algoritmalar konusuna değinilecektir.

2.1 Temel Matematiksel Kavramlar

Aşağıda AES algoritmasında kullanılan *Galois Alanları*'nı anlamak için bazı temel matematiksel tanımlar verilecektir.

2.1.1 Abelian Grubu

Bir G kümesi ve bu kümenin elemanları üzerinde tanımlanmış olan bir "+" işleminden oluşur [15]. Bir grubun *Abelian Grubu* olması "+" işlemi için aşağıdaki özellikleri sağlaması gerekir [16].

1. Kapalılık: $\forall$ a,b $\in$ G : (a + b) $\in$ G
2. Birleşme: $\forall$ a,b,c $\in$ G : (a + b) + c = a + (b + c)
3. Değişme: $\forall$ a,b $\in$ G : a + b = b + a
4. Etkisiz Eleman: $\exists$ 0 $\in$ G, $\forall$ a $\in$ G : a + 0 = a
5. Ters Eleman: $\forall$ a $\in$ G, $\exists$ b $\in$ G : a + b = 0

Örneğin; Tamsayılar kümesi ve 'toplama' işlemi, < Z, + >, bir *Abelian Grubu* oluşturmaktadır. Benzer şekilde 0'dan n -1' e kadar olan tamsayıların oluşturduğu küme ve 'modulo n toplama' işlemi, < Z_n, + >, de bir *Abelian Grubu* oluşturmaktadır.

2.1.2 Halka

R boş kümeden farklı bir küme olmak üzere bir <R, Δ, □ > kümesi ve bu kümenin elemanları üzerinde tanımlanmış olan iki adet işlemden oluşur.

R kümesi üzerinde tanımlanmış olan "Δ" ve "□" işlemlerinin aşağıdaki koşulları sağlıyorsa bir '*Halka*' oluşturur.

Öncelikle < R, Δ> bir *Abelian Grubu* oluşturmalıdır. Ayrıca "□" işlemi *R* üzerinde kapalılık ve birleşme özelliği sağlamalı ve "□" işleminin, "Δ" işlemi üzerinde dağılma özelliği olmalıdır.

Eğer "Δ"işlemi *değişme özelliğine* sahipse, < R, Δ, □ >halkası *'Değişmeli Halka'* olarak adlandırılır. "Δ"işleminin birim elemanı 0, "□" işleminin birim elemanı ise 1'dir [20].

Örneğin; tamsayılar kümesi, 'toplama' ve 'çarpma' işlemleri ile birlikte, < Z, +, * >, bir *Halka* oluşturmaktadır.

2.1.3 Alan

Bir F kümesi, üzerinde tanımlanmış "Δ" ve "□"işlemleriyle birlikte aşağıdaki koşulları sağlıyorsa bir *'Alan'* oluşturur.

Öncelikle (*F*, Δ) bir *Abelian Grubu* olmalıdır, ancak sadece 0 elemanı için *ters eleman* olmayabilir. Ayrıca < F, Δ, □ > bir *Halka* olmalıdır. Yani "□" işleminin "Δ"işlemi üzerinde *dağılma özelliği* olmalıdır [17].

Örneğin; Gerçel Sayılar Kümesi 'toplama' ve 'çarpma' işlemleri ile birlikte bir *Alan* oluşturur.

2.1.4 Sonlu Alan

Sonlu Alan, adından da anlaşılacağı üzere, sonlu sayıda elemana sahip bir '*Alan*'dır. Aynı sayıda elemana sahip tüm Sonlu Alan'lar eş yapılıdır. Bu demektir ki, tamamıyla aynı matematiksel yapıyı gösterirler, sadece elemanlarının gösterilişinde farklılıklar vardır. Kriptografik algoritmalarda yaygın olarak kullanılan bir sonlu alan türü *Galois Alanları'*dır. Bu nedenle bir sonraki bölüm 2.1.5'te *Galois Alanları* hakkında ayrıntılı bilgi verilecektir.

2.1.5 Galois Alanları

P sayısı asal olmak üzere {0,1,2...,p-1} tamsayı elemanlarından oluşan ve üzerinde modülo *p* toplama "Δ" modülo *p* çarpma "□"işlemleri tanımlı alana '*Galois Alanı'* denir ve "*GF(p)*" olarak gösterilir. *P* sayısı '*Galois Alanının Karakteristiği'* olarak adlandırılır [16].

Bir başka deyişle, *p* bir asal sayı olmak üzere, *p* sayıda elemana sahip bir sonlu alan '*Galois Alanı'* olarak adlandırılır ve *GF*(*p*) ile gösterilir.

GF(p^n) ise *GF*(*p*)'nin *Genişletilmiş Sonlu Alan'*ını ifade etmektedir.Eleman sayısı ise *GF*(*p*) sonlu alanının elaman sayısının *n* katıdır. *GF*(p^n)'de yapılan "Δ" ve "□"işlemlerinin alan içerisinde kalmasını sağlamak için *n.* dereceden polinoma ihtiyacı vardır Kullanılan n. dereceden polinomun çarpanlarına ayrılamaması gerekmektedir. Bu polinomun"*İndirgenemez Polinom*" olması gerekmektedir [16].

2.1.5.1 GF(2^n) Galois alanı

Karakteristiği iki olan *Galois Alanları*'dır. 2^n tane eleman içermektedir. *GF*(2^n) *Galois Alanı*, içerisinde tanımlanan matematiksel işlemlerin gerçekleştirilmesini çok kolaylaştırmaktadır. Çünkü bu alanın elemanlarını yan yana yazılmış bitler {0,1} oluşturmaktadır. Bu nedenle yazılım ve donanım uygulamalarında, kriptografik algoritmaların gerçeklenmesinde yaygın olarak kullanılmaktadır.

Karakteristiği iki olan *Galois Alanları*'nda farklı gösterim şekilleri olmakla beraber en yaygın olanı polinomsal gösterimdir. *GF*(2^n)için polinomsal baz,{ $x^{n-1}, x^{n-2}, \ldots\ldots, x^2, x, 1$ } kümesinden oluşmaktadır. *GF*(2^n)'nin bir elemanının polinomsal gösterimi, polinomsal baz vektörünün her bir elamanının *GF*(2)'ye ait bir elemanla çarpılması ile elde edilir. Örneğin, *GF*(2^7)'nin bir elemanı olan {1001011} polinomsal olarak şu şekilde gösterilir.

$a(x) = x^6 + x^3 + x + 1$

2.1.5.2 GF(2^n) Üzerinde Matematiksel İşlemler

Toplama İşlemi

GF(2^m) sonlu alanı üzerinde toplama ve çıkarma işlemi, bu alan üzerinde tanımlı elemanların polinomsal gösterilimlerinin toplanması ve daha sonra polinom katsayılarının modülo iki olarak düzenlenmesi ile gerçekleştirilir. Örnek 2.1'de *GF*(2^8) için toplama işlemi uygulaması gösterilmiştir.

Örnek 2.1:

$a, b, c \in$ GF(2^8) olmak üzere,

$a = (11001111) = \{x^7+x^6+x^3+x^2+x+1\}$,

$b = (10001000) = \{x^7+x^3\}$,

$c = a+b$

$= x^7+x^6+x^3+x^2+x+1+x^7+x^3$

$= 2x^7+x^6+2x^3+x^2+x+1$ bulunur.

katsayıların modulo iki'deki değerleri alındığında,

$c = x^6+x^2+x+1 = (01000111)$ elde edilir.

Yapılan işlemler incelendiğinde toplama işleminin bit bit xor işleminden ibaret olduğu görülebilir. Bu özellik donanım gerçeklemelerinde büyük kolaylık sağlamaktadır.

Çarpma İşlemi

İki *GF*(2^n) polinomsal elemanının çarpımı, iki polinomun aritmetik çarpımının alınmasıyla elde edilir. Ancak, bu iki polinomun çarpımı, *Sonlu Alan*'ın derecesinden daha yüksek dereceli bir polinom elde edilmesine neden olabilir. Bu nedenle çarpım sonucunda oluşan polinomu tekrar *Sonlu Alan* içerisindeki bir polinoma karşı düşürmek gerekir. Bu sebeple sonuç polinomunun n. dereceden bir polinom ile modülünün alınması gerekir. Modül alma işlemi için kullanılan bu polinom, '*İndirgeme Polinomu'* olarak adlandırılır.

İndirgeme Polinomu, $GF(2^n)$içerisinde tanımlı ve iki farklı polinomun çarpımına ayrılamayan bir polinomdur.

Örnek 2.2:

a, b, c $\in GF(2^8)$ ve $P(x) = x^8+x^4+x^3+x+1$ $GF(2^8)$ için *İndirgeme Polinomu* olmak üzere,

a = (01010111) = $\{x^6+x^4+x^2+x+1\}$,

b = (10000011) = $\{x^7+x+1\}$,

$c = a * b = \{x^6+x^4+x^2+x+1\} * \{x^7+x+1\}$

$= x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1$elde edilir.

Görüldüğü gibi elde edilen sonuç $GF(2^8)$ sonlu alanının dışında bir değerdir. Bu nedenle sonucun $GF(2^8)$ için tanımlı olan indirgenemez polinoma göre modülü alınır.

$c = x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \pmod{P(x) = x^8+x^4+x^3+x+1}$

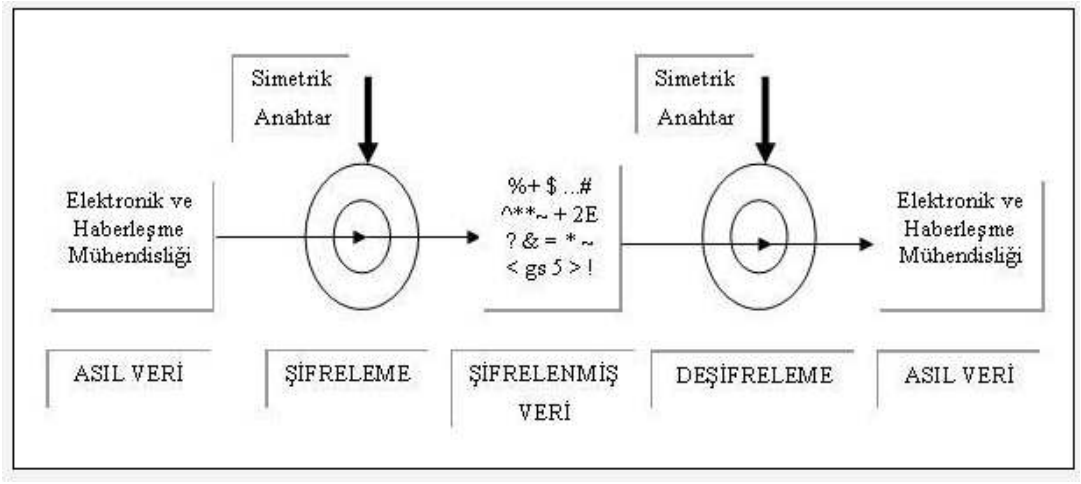
$= x^7+x^6+1 \in GF(2^8)$ elde edilir.

2.2 Kriptografik Algoritmalar ve Sınıflandırılmaları

Şifre bilimi anlamına gelen kriptografi, çeşitli iletilerin, bilgilerin herkese açık ortamlarda iletilirken istenmeyen kişiler tarafından öğrenilmesini veya değiştirilmesini önlemek amaçlı kullanılır. Kriptografi, bilgiyi güvenli bir şekilde sadece istenilen kişiye ulaştırmak amacıyla uzun süredir kullanılmaktadır. Tarihin bilinen ilk şifreleme yöntemi yer değiştirme ve harf değiştirme yöntemidir. Bu yöntemlerden ilki bir yazıdaki harflerin yerlerini değiştirerek, ikincisi ise harfleri başka harflerle değiştirerek gerçekleştirilir [18].

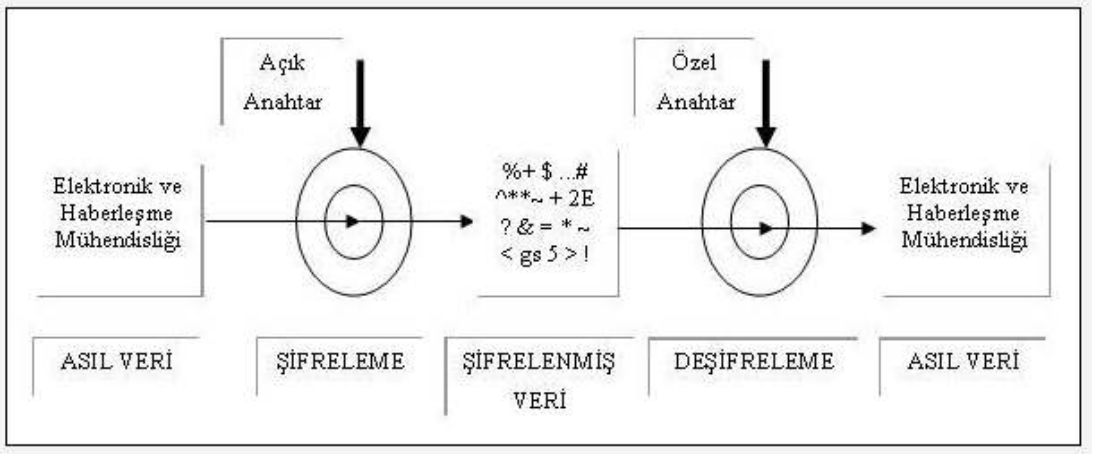
Kriptografik algoritmalar en temelde kullandıkları anahtar sistemine göre '*simetrik algoritma'* ve 'a*simetrik algoritma'* olmak üzere iki kısma ayrılır.

Veriyi şifreleme ve çözmede aynı anahtar kullanılıyorsa bu tür algoritmalara '*Simetrik Algoritma*' denir. AES algoritması ve DES algoritması bu algoritmaların en bilinen örnekleridir.



Şekil 2.1 Simetrik şifreleme

Veriyi şifreleme ve çözmede farklı anahtar kullanılıyor ise buna da 'A*simetrik Algoritma* ismi verilir. Asimetrik algoritmalar için en bilinen örnek Rivest-Shamir-Adleman (RSA) ve Eliptik Eğri algoritmalarıdır.

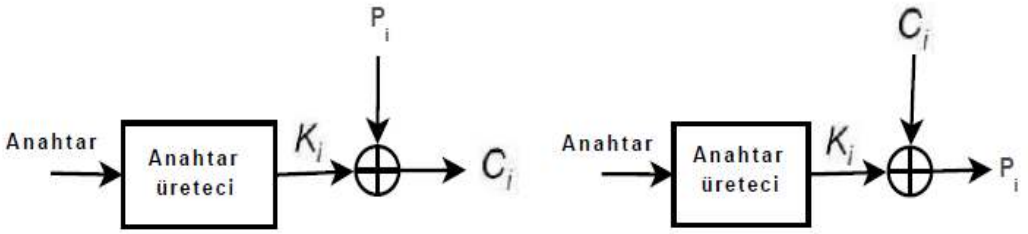


Şekil 2.2 Asimetrik şifreleme

Kriptografik algoritmalar veri işleme yöntemlerine göre ise *'dizi şifreleme algoritmaları'* ve *'blok şifreleme algoritmalar'* algoritmaları olmak üzere iki kısma ayrılırlar. *Dizi Şifreleme Algoritmaları*, küçük veri bloklarını alıp işlerken, *Blok Şifreleme Algoritmaları* çok daha büyük verileri işleyerek çalışırlar.

2.2.1 Dizi Şifreleme Algoritmaları

Dizi Şifreleme Algoritmaları, zamanla değişen anahtar yardımıyla düz metinin her bir bitini sırayla şifrelerler. Dizi şifreleyiciler yüksek hızlı iletişim için en iyi alternatiflerden birisidir. Yüksek hatalı iletişim ortamlarında hata riskini azalttığından tercih sebebidirler. Dizi şifreleme sistemleri genel olarak Şekil 2.3'te verilen yapıya sahiptirler.



Şekil 2.3 Dizi şifreleme sistemleri

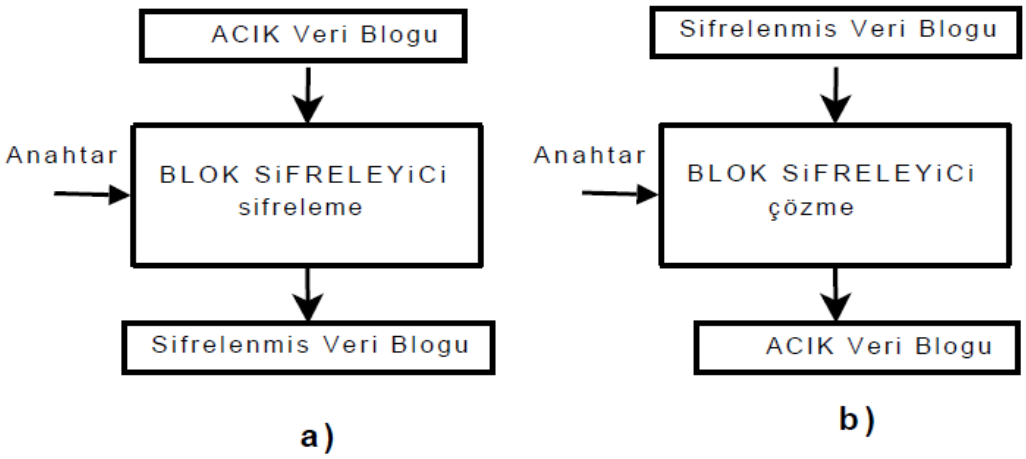
Şifrelenecek olan düz metnin bit dizisi şeklindeki ifadesi olan P dizisinin bir biti ile üretilen anahtar dizisinin bir biti xor işlemine tabi tutularak şifrelenmiş metine ait olan C biti elde edilir. Bu işlem düz metine ait bütün bitler için uygulanarak istenilen şifrelenmiş metin elde edilir. Şifrelenmiş metinden düz metin elde edilmek istendiği takdirde ise xor işleminin özelliğinden faydalanarak; aynı şekilde şifrelenmiş metinin bir biti, biti şifrelemek için kullanılan aynı anahtar biti ile xor işlemine tabi tutulur. Bu işlem şifrelenmiş metine ait bütün bitler için tekrarlandığı takdirde düz metin elde edilmiş olur.

2.2.2 Blok Şifreleme Algoritmaları

Blok Şifreleme Algoritmaları, sabit uzunluktaki bloklar halinde aldıkları açık veriyi, yine aynı uzunluktaki bloklar halinde şifrelenmiş veriye çeviren simetrik anahtarlı şifreleme algoritmalarıdır. Bu dönüştürme işlemi kullanıcı tarafından belirlenen gizli bir anahtar kullanılarak yapılır. Şifre çözme işlemi de, yine sabit uzunluktaki bloklar halinde alınan kapalı verinin bu kez ters dönüşümden geçirilerek, bloklar halinde açık veriye dönüştürülmesiyle gerçekleştirilir.

Blok şifreleyiciler günümüzde şifreleme işlemlerinde yaygın olarak kullanılmaktadır. En çok bilinenlere örnek olarak; 64 bitlik blokları kullanan DES ve 128 bitlik blokları kullanan AES gösterilebilir.

Yukarıda da belirtildiği gibi, blok şifreleyiciler sabit uzunluktaki blokları yine sabit uzunluktaki başka bloklara dönüştürürler. Şifrelenen verinin tekrar çözülebilmesi ya da bunun tersinin olabilmesi için gerek koşul, bu dönüştürme işleminin birebir olmasıdır. Bu açıdan bakıldığında, matematiksel anlamda, blok şifreleyiciler giriş uzaylarındaki değerleri yine giriş uzaylarındaki başka değerlere atayan 'sıra-değiştirme' (substitution) fonksiyonlarıdır. Bu atama işlemi de kullanılan gizli anahtarın kontrolünde yapılmaktadır [19].



Şekil 2.4 Blok şifreleme ve çözme işlemi

BÖLÜM 3

YAN KANAL ANALİZİ SALDIRILARI

Güvenli haberleşme tarihten günümüze kadar insanların yaşamında çok önemli bir yer tutmuştur. Bunun başarılabilmesi için en çok kullanılan yöntem ise bilginin yetkisiz kişilerce anlaşılamayacak formatlara çevrilmesidir. Bu çevirme işlemini gerçekleştiren uygulamalara kripto algoritmaları olarak isim verilmektedir.

Güvenli haberleşme ihtiyacının artarak devam ettiği günümüzde de bu ihtiyacın karşılanması için çalışmalar devam etmektedir. Bu çalışmalar kapsamında güvenliği sağlayacak algoritmalar geliştirilmekte ve bu algoritmalara dayalı güvenli kriptografik sistemler oluşturulmaktadır.

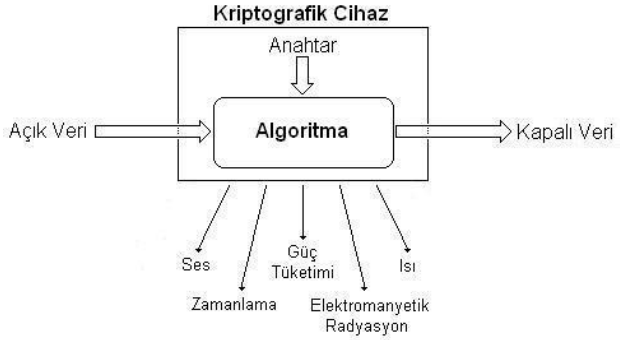
Kriptografik sistemlerin kritik seviyede önem taşıması, bu sistemlere yönelik saldırı çalışmalarını da beraberinde getirmiştir. Bu çalışmalara genel olarak kriptanaliz ismi verilir.

Kriptanaliz, kriptografik sistem analiz edilerek gizli bilgiye (genellikle kullanılan anahtara) ulaşılmaya çalışılır.

Klasik kriptanaliz, kriptografik algoritmayı, girişindeki N_i-bitlik veriyi çıkışı olan N_O-bitlik veriye çeviren matematiksel bir yapı olarak ele alır. Algoritma matematiksel olarak modellenmeye, açık ve/veya kapalı veri kullanılarak çeşitli yöntemlerle gizli bilgiye ulaşılmaya çalışılır. Örneğin blok şifreleyicilere yapılan saldırılarda, istatistiksel saldırılar olan diferansiyel [20] ve doğrusal [21] kriptanaliz kullanılır.

Günümüzde tasarlanan ve genelde standart hale getirilen algoritmalara karşı bu şekilde saldırı yapılma işlemi zorlaşmıştır.

Kriptografik sistemlere yönelik gerçekleşen diğer bir saldırı türü ise yan kanal analizi saldırılarıdır. Yan kanal analizinde kriptografik sistemin çalışması esnasında istemsiz olarak dışarıya yaydığı bazı bilgilerden yararlanılarak sistemin gizliliğini sağlayan parametrelerin ele geçirilmesi amaçlanır. Kriptografik sistem tarafından dışarıya verilen yan kanal bilgileri Şekil 3.1'de gösterilmektedir.



Şekil 3.1 Kriptografik sistem tarafından dışarıya verilen yan kanal bilgileri

Yan kanal analizi saldırıları, aktif ve pasif olarak iki gruba ayrılmaktadır. Aktif saldırılar ya da diğer adıyla kurcalama saldırıları [22], kriptografik cihazın içindeki devrelere ulaşılmasını gerektirir. Bu nedenle uygulanmaları daha zordur ve oldukça gelişmiş ve pahalı düzeneğe ihtiyaç duyulur.

Pasif saldırılarda ise cihazın çalışmasına müdahale edilmez. Cihazın normal çalışması sırasında ürettiği yan-kanal bilgileri kullanılır. Bu saldırılar çok daha basit ölçüm düzenekleriyle yapılabilmektedir.

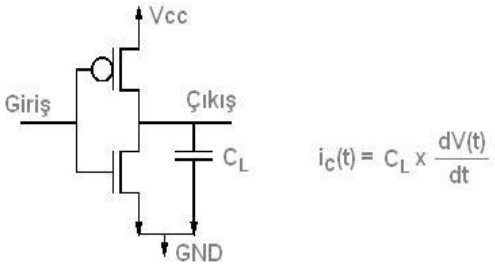
Pasif saldırılar kullandıkları yan-kanal bilgisine göre dört gruba ayrılır: Zamanlama Analizi Saldırıları, Elektromanyetik Analiz Saldırıları, Akustik Analiz Saldırıları ve Güç Analizi Saldırıları. Güç analizi saldırıları ise basit ve farksal güç analizi saldırıları olmak üzere iki kısma ayrılmaktadır.

Tez çalışmasının kapsamında bu saldırılardan güç analizi saldırıları üzerinde durulacaktır.

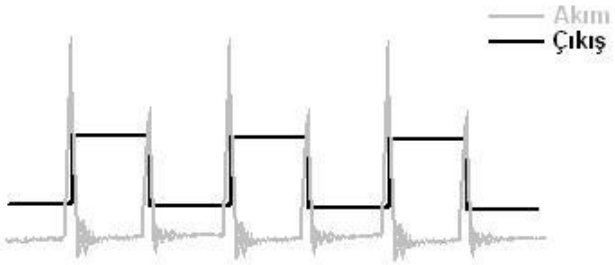
3.1 Güç Analizi Saldırıları

Günümüzde sayısal çalışan elektronik tüm devrelerin gerçeklenmesinde tamamlayıcı metal oksitli yarı iletken transistörler (CMOS) sıklıkla kullanılmaktadır [23]. CMOS transistörler

çalışmaları sırasında farklı durumlar için farklı seviyede güç tüketimi gerçekleştirirler. Bu çalışma karakteristiğinin anlaşılması için Şekil 3.2'de görülen CMOS evirici yapısı örnek verilebilir. Bu yapıda, CMOS transistorün konum değiştirmediği durumlarda tükettiği güç minimum seviyededir. Konum değiştirme anlarındaki güç tüketimi artmaktadır. Konum değiştirme anlarındaki güç tüketimine dinamik güç tüketimi ismi verilir. Dinamik güç tüketimi $0 \rightarrow 1$ geçişlerinde, $1 \rightarrow 0$ geçişlerine oranla daha yüksektir. Şekil 3.3'te CMOS evirici yapısının farklı durumlar için çektiği akım (dolayısıyla güç) değerleri görülmektedir.



Şekil 3.2 CMOS evirici yapısı



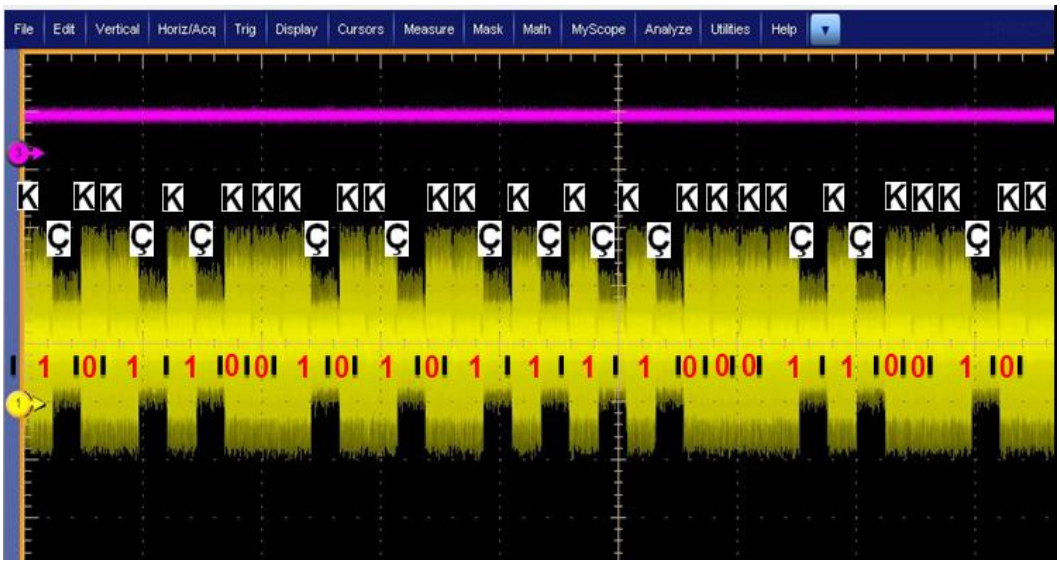
Şekil 3.3 CMOS eviricinin farklı geçişler için güç tüketimi

CMOS transistörlerinin bu karakteristiği kriptografik sistem içerisinde işlenen gizli bilgilerle ilintili olması yan kanal analizi saldırılarının gerçeklenmesine olanak sağlamaktadır. Aşağıda kriptografik sistemin çektiği güç üzerinde analiz yapılarak gerçekleşen 2 farklı yan kanal analizi saldırısı anlatılmıştır.

3.1.1 Basit Güç Analizi Saldırıları

Basit Güç Analizi Saldırıları (BGA), kriptografik sistemin farklı işlem adımlarını gerçekleştirirken farklı seviyede güç tüketimi gerçekleştirmesi açıklığına dayanılarak gerçekleştirilen saldırıdır. Örneğin farklı mikroişlemci komutları işlenirken ya da toplama ve

çarpma gibi farklı işlemler gerçekleştirilirken tümleşik devreler farklı miktarda güç tüketirler. Bir koddaki dallanmalar bu tür farklılıkların genel olarak oluştuğu bölümlerdir. Güç tüketimi ölçümleri incelenirken bu faklılıklar kolayca gözlemlenebilir. Diğer bir yöntemde ise kripto algoritmasının çalışması esnasında kullanılan anahtar değerine bağlı olarak farklı işlemler gerçeklenmekte ve bu esnada gözlemlenen güç tüketimi bilgisi ile anahtar değeri ele geçirilebilmektedir.



Şekil 3.4 RSA algoritmasında anahtara göre farklı güç tüketiminin oluşması

Basit güç analizinin önüne geçilebilmesi için tasarımlarda koşullu dallanmalardan sakınılmalı veya saldırganı yanıltmak amaçlı çok sayıda sahte dallanma gerçeklenmelidir.

Diğer bir önlem yöntemi ise güç tüketiminin az olduğu noktalarda, sistemin tükettiği gücü dengelemek için sürekli etkisiz işlemler (NOP: No Operation) yapılmasıdır. Bu yöntemle sistemin güç tüketimi sabitlenerek saldırganın analiz yapmasının önüne geçilir [19].

3.1.2 Farksal Güç Analizi Saldırıları

Farksal Güç Analizi Saldırıları (FGA) ile BGA saldırılarının aksine kriptografik sistemin yaptığı iş ile değil, işlediği veri ile ilişki kurularak yan-kanal saldırısı gerçeklenir. Bu işlemin gerçeklenmesi için çok sayıda verinin işlenmesi anında güç ölçümü alınır. Saldırının gerçeklenebilmesi esnasında kripto algoritmasının çalışması esnasında ürettiği ara değerle anahtar değerinin ilişkisi kontrol edilir.

FGA saldırılarının uygulanması, BGA saldırılarına göre daha zordur. Ancak FGA saldırıları çok daha güçlü saldırılardır. Ayrıca BGA saldırısının gerçeklenmesi için incelenen sistem hakkında çok fazla bilgi sahibi olunması gerekmez. Konunun daha iyi anlaşılabilmesi için FGA saldırısının gerçeklenme süreci bir örnek üzerinde anlatılacaktır.

3.1.3 AES Algoritmasının Gerçeklemesine FGA Saldırısı

AES algoritması NIST tarafından standart olarak belirlendiği 2001 yılından beri kriptografik sistemlerde yaygın olarak kullanılmaktadır. Bu yaygın kullanımdan dolayı üzerine çok sayıda FGA saldırısı gerçeklenmiştir [11], [12], [13]. Ayrıca tez çalışması kapsamında üzerinde çalışılan sistemde de AES algoritması yer almaktadır. Bu nedenle DPA saldırısının gerçeklenme adımlarını AES algoritması üzerinden örneklemek uygun olacaktır. DPA Saldırısının gerçekleşmesinde aşağıda sıralanan adımlar takip edilir.

- Algoritmanın çalışması esnasında DPA saldırısının gerçekleştirileceği algoritma ara değeri belirlenir. Bu ara değer genellikle bir bellek elemanının çıkışı olarak seçilir. Bu adımın içerisinde anahtar verisinin kullanıldığı ilk veya son tur ara değerlerinden biri olması saldırının pratikte gerçeklenebilmesi için gereklidir.
- AES algoritmasının giriş verisi formatında *n* adet örnek içeren bir açık veri seti oluşturur. Açık veri setindeki örnek sayısı (*n*) , saldırıyı gerçekleştirecek kişinin saldırının başarısı için öngördüğü miktar adedidir. Pratik uygulamalarda bu değer 10.000 ile 1.000.000 arası olabilmektedir.
- Oluşturulan açık veri setindeki her bir veri kriptografik sisteme girdi olarak verilerek şifreleme/çözme işlemi gerçeklenir.
- Osiloskop kullanılarak her bir şifreleme esnasında belirlenen çıkış değerinin değiştiği saat darbesini hedef alan ve *s* adet örnekten oluşan, güç ölçümleri yapılır. Böylelikle, *n* X *s* boyutunda bir *Güç Ölçüm Matrisi* (M_1) oluşur.

M_1 (*n* X *s*)

$$\begin{bmatrix} t_{00} & t_{01} & t_{02} & . & . & . & . & . & . & t_{0s} \\ t_{10} & t_{11} & t_{12} & . & . & . & . & . & . & t_{1s} \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ t_{n0} & t_{n1} & t_{n2} & . & . & . & . & . & . & t_{nk} \end{bmatrix}$$

Şekil 3.5 Güç ölçüm matrisi

Güç ölçüm Matrisi üzerinde çok sayıda noktanın bulunduğu ve bilgisayar hafızasında çok fazla yer kaplaması nedeniyle işlenmesi zordur. Bu matrisin boyunu azaltmak için bazı teknikler uygulanır. Bu tekniklerin en başında geleni her bir açık verinin işlenmesinde kaydedilen gücün karekök ortalama (root mean square (RMS))değerini almaktır. Karekök ortalama, değişen miktarların büyüklüğünün ölçülmesinde kullanılan istatistik bir ölçüttür. Güç ölçümleri üzerinde RMS alma işlemi gerçeklenerek sadece ölçüm üzerinde değişikliklerin oluştuğu noktalara yoğunlaşılmış olur. Yapılan işlemin güç ölçümü verisinin karakteristiğini bozmaması için saat darbelerinin gerçekleştiği noktalarda örnekler alınması gerekmektedir. RMS işlemi sonunda ölçüm üzerinde yer alan nokta sayısı, örnek alınan toplam nokta sayısını gösteren *k* değerine düşmüş olacaktır. Analiz sonunda elde edilen *n* X *k* boyundaki M_2 matrisine "*Güç Analiz Matrisi*" ismi verilir.

M_2 (*n* X *k*)

$$\begin{bmatrix} t_{00} & t_{01} & . & . & t_{0k} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ t_{n0} & t_{n1} & . & . & t_{nk} \end{bmatrix}$$

Şekil 3.6 Güç analiz matrisi

Bilgisayar üzerinde AES algoritması en azından saldırının gerçekleneceği ara değer noktasına kadar modellenir. Modelleme işleminde daha önce oluşturulan açık veri setindeki tüm P açık veri değerleri olası tüm anahtar değerleri ile algoritmaya sokulur. Örneğin 8 bit anahtar değerinin tahmini için 2^8=256 adet anahtar değeri modelleme işlemine sokulacaktır. Modelleme sonucunda 256 farklı anahtar verisinin *n* adet açık veri ile işleme sokulduğu bir M_3 (*n* X 256) matrisi oluşturulur. M_3 matrisinin her bir sütunu 256 farklı anahtar değerinden birinin kullanılması sonucu elde edilecek sonuç değerini (ör: AB,C9,85) göstermektedir. Olası tüm anahtar değerlerinin denendiği M_3 matrisinde sütunlardan birinin doğru anahtar değeri ile üretilmiş verileri içerdiği kesin olmaktadır.

M_3 (*n* X *256*)

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} & \cdots & c_{0256} \\ c_{10} & c_{11} & c_{12} & \cdots & c_{1256} \\ \vdots & \vdots & \vdots & & \vdots \\ c_{n0} & c_{n1} & c_{n2} & \cdots & c_{n256} \end{bmatrix}$$

Şekil 3.7 Ara değer sonuç matrisi

- M_3 matrisinin elde edilmesinden sonra 2 farklı yöntem kullanılarak sürece devam edilir.
    - M_3 matrisinde yer alan her bir veri üzerinde yer alan "1" sayısı sayılarak matristeki o elemanın yerine koyulur. Buna "hammingweight" yöntemi denir.
    - M_3 matrisindeki her bir veri için bir önceki durumuna göre üzerinde gerçekleşen $0 \rightarrow 1$ sayısı sayılarak matriste o elemanın yerine koyulur. Bu yönteme "hammingdistance " yöntemi denir.

Her iki yöntemin sonunda elde edilen M_4 matrisine "*Tahmin Matrisi*" adı verilir. *Tahmin Matrisi*'nin her bir elemanı yukarıda belirtilen iki yöntemden birine bağlı olarak 0 ile 8 arasında bir değer olmaktadır.

M_4 (*nX256*)

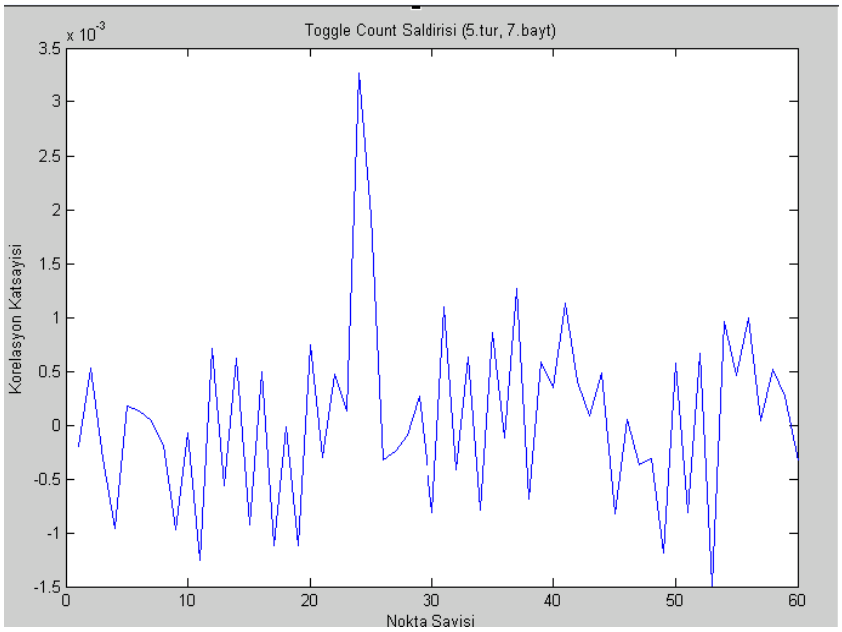
$$\begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} & \cdots & h_{0,256} \\ h_{1,0} & h_{1,1} & h_{1,2} & \cdots & h_{1,256} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ h_{n,0} & h_{n,1} & h_{n,2} & \cdots & h_{n,256} \end{bmatrix}$$

Şekil 3.8 Tahmin matrisi

- *Tahmin Matrisi*'nin elde edilmesinden sonra *Korelasyon* veya *Kocher Yöntemleri*'nden biri kullanılarak analize devam edilir.

<u>Korelasyon Yöntemi</u>

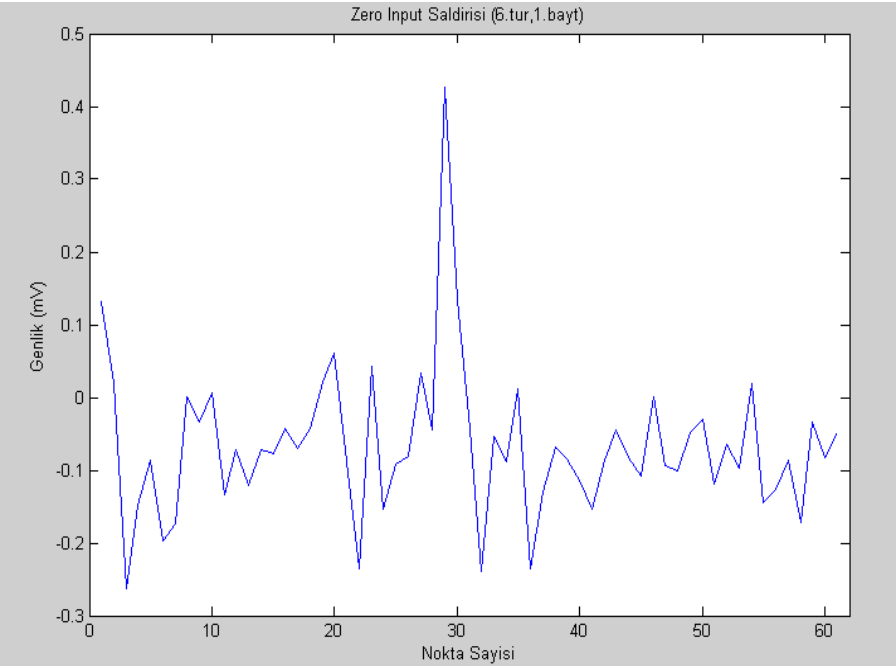
M_4 *Tahmin Matrisi'*nin her bir sütunu M_2 *Güç Analiz Matrisi* ile korelasyon işlemine tabi tutulur. M_4 matrisinin doğru anahtar tahminine karşı düşen sütununun, M_2 sütun matrisi ile korelasyonu yüksek çıkarken, diğer sütunlarının M_2 sütun matrisi ile korelasyonu oldukça düşük çıkacaktır. Şekil 3.9'da tez çalışması kapsamında yapılan başarılı bir korelasyon analizi sonucu görülmektedir.



Şekil 3.9 Örnek korelasyon analizi sonucu

Kocher Yöntemi

Tahmin Matrisi'nin her bir sütununa göre M_2*Güç Analiz Matrisi'*nin her bir elemanı çok güç tüketen ve az güç tüketen olmak üzere iki kısma ayrılır. Genelde *Tahmin Matrisi*'nde 4'ten büyük olan satırlara karşılık gelenler çok güç tüketen, 4'ten küçük olan satırlara karşılık gelenler az güç tüketen olarak kabul edilir. Öncelikle çok güç tüketenler kendi aralarında az güç tüketenler de kendi aralarında toplanır. Daha sonra çok güç tüketenlerden az güç tüketenler çıkarılır. *Tahmin Matrisi*'ndeki her bir sütuna karşılık gelen değerler için bu işlem tekrar edilir. Doğru anahtar değeri ile üretilen sütun ile yapılan tahminde, çok güç tüketenlerle az güç tüketenler düzgün sınıflandırıldığı için gözle görülür bir işaret farkı oluşur. Diğer sütunlara göre yapılan işlem rastgele olduğundan bir sonuç elde edilemez. Bu sayede doğru anahtar değeri tahmin edilebilir. Bu yöntem ilk olarak Kocher [3] tarafından önerildiği için tez kapsamında *Kocher Yöntemi* olarak isimlendirilecektir. Şekil 3.10'te tez çalışması kapsamında yapılan başarılı bir *Kocher Analizi* sonucu görülmektedir.



Şekil 3.10 Örnek Kocher analizi sonucu

3.1.4 Farksal Güç Analizi Saldırılarına Karşı Alınabilecek Tedbirler

Farksal Güç Analizi ilk olarak Paul Kocher tarafından 1998 yılında gündeme getirilmiş ve kriptografi dünyasında ses getirmiştir. Bu tarihten itibaren FGA saldırılarına karşı önlem geliştirme çalışmaları başlamıştır. Bu önlemler temelde donanımsal ve yazılımsal olmak üzere iki kısma ayrılabilir.

Donanımsal önlemler; güç ölçümü alma işlemini, gürültüyü artırarak ya da sızan yan-kanal bilgisini azaltarak zorlaştırmaya çalışır. Bu yöntemlere örnek olarak devre içerisine bir rastgele sayı üreteci yerleştirilerek [3], ortamdaki gürültü seviyesi artırılması çalışması verilebilir.

Ayrıca devre üzerine güç işaretlerini filtreleyen bir devre yerleştirilerek yan kanal bilgisinin genliğini düşürme yöntemi olabilir [24], [25]. Ancak her iki yöntemde de devre üzerine ek yapılar eklenmekte ve bu durum günümüz şartlarında etkin kullanım için tüm devrelerin boyutlarının küçültülmesi ve güç ihtiyacının azaltılması prensibine ters düşmektedir. FGA saldırılarına karşı en temel önlem ise güç tüketimi işlediği veriye göre değişmeyen kapıların kullanılmasıdır [26]. Ancak günümüz teknolojisinde bu yapılar aktif olarak kullanılamadığı için bu önlem teorik düzeyde kalmaktadır.

Yazılımsal önlemlerde ise, işlenen verinin işlem zamanlarının rastgele hale getirilmesi yöntemi uygulanabilir [27], [28]. Bu işlemde algoritma işlem blokları arasına gereksiz beklemeler konulur. Böylece saldırganın modellediği sisteme bağlı olarak güç ölçümlerini uygun şekilde üst üste getirmesinin önüne geçilmiş olur. Bu durum uygulanabilir bir yöntem olmakla beraber algoritma işlemlerinin gereksiz düzeyde uzaması dezavantajına sebebiyet verir.

Yazılımsal karşı önlemlerin en çok uygulanan örneği ise maskeleme yöntemidir [10], [29]. Daha önce de belirtildiği gibi saldırganın FGA saldırısını gerçekleyebilmesi için gerekli ön şart algoritma içerisinde sadece anahtar ve bilinen giriş/çıkış verisine bağlı ara değerlerin bulunmasıdır. Maskeleme yönteminde algoritma tarafından üretilen anahtar ve açık veriye bağlı çıkış verilerine 3. bir değer yani maske eklenir. Saldırgan FGA saldırısı gerçekleştirmek için kendi modelini bildiği açık verilere ve tahmin ettiği anahtar değerine bağlı olarak oluşturur. Ancak maske önlemi alınmış sistemde üretilen ara değere bir de saldırganın

bilemediği maske değeri eklendiğinden saldırganın sonuç elde etmesi teorik olarak imkânsız hale gelir.

BÖLÜM 4

AES ALGORİTMASI

DES algoritması 1977 yılında NIST tarafından veri şifreleme standardı olarak yayınlanmış ve o tarihten itibaren 20 yılı aşkın bir süre yaygın olarak kullanılmıştı. Ancak gelişen teknoloji ve buna bağlı olarak üretilen hızlı işlem kapasiteli bilgisayarlar DES algoritmasının güvenliği için tehdit oluşturmaya başlamıştı. Çünkü DES algoritması tarafından kullanılan 56 bit uzunluklu anahtar değerinin yeni teknoloji ile üretilen sistemler ile kırılacağı öngörülmekteydi. Bu gelişmeler üzerine Ocak 1997'de ABD Ulusal Standartlar ve Teknolojiler Enstitüsü (NIST), yeni bir şifreleme standardının geliştirilmesi için bir çalışma başlattı. Geliştirilecek yeni şifreleme standardının mevcut standart olan Veri Kodlama Standardı (Data Encryption Standart (DES))'nın yerini alması düşünülüyordu. Bu nedenle yeni seçilecek algoritmada, DES algorittması için problem oluşturan 64 bit anahtar boyu yerine, 128 bitlik blok boyunu ve 128, 192 veya 256 bitlik anahtar uzaylarını destekleyen bir simetrik blok şifreleyici olması talep ediliyordu. Yeni şifreleme standardını belirlemek amacıyla bir yarışma düzenlendi. Dört yıl boyunca süren değerlendirme ve eleme süreci sonrasında, 2000 yılında sonuç açıklandı. NIST, Joan Daemen ve Vincent Rijmen tarafından tasarlanan, Rijndael algoritmasının hiçbir değişiklik talep etmeden Gelişmiş Kodlama Standardı (Advanced Encryption Standard (AES)) olarak kullanılacağını ilan etti. Rijndael algoritmasında blok ve anahtar uzunlukları 128 bitten 256 bite kadar 32'lik araklıklarla birbirinden bağımsız olarak değişebildiği halde, AES yukarıda belirtilen sabitlenmiş blok ve anahtar uzunluklarıyla kullanılmaktadır.

AES Algoritması genel olarak *Tur İşlemleri* ve tur işlemleri içerisinde gerçekleştirilen *Tur Dönüşüm İşlemleri* adımlarından oluşur [7].

Bu bölümde AES algoritmasının temel birimleri ve işleyişi incelenecektir.

4.1 AES Algoritması Tur İşlemleri

Bir algoritma içerisinde tekrar tekrar yürütülen işlemlerin oluşturduğu yapıya *'Tur'* adı verilir. AES algoritması standart olarak 128 bit veri blokları işlerken anahtar boyu farklı uzunlukta olabilmektedir. Algoritma içerisinde gerçekleşecek tur sayısı da anahtar uzunluğuna bağlı olarak değişmektedir.

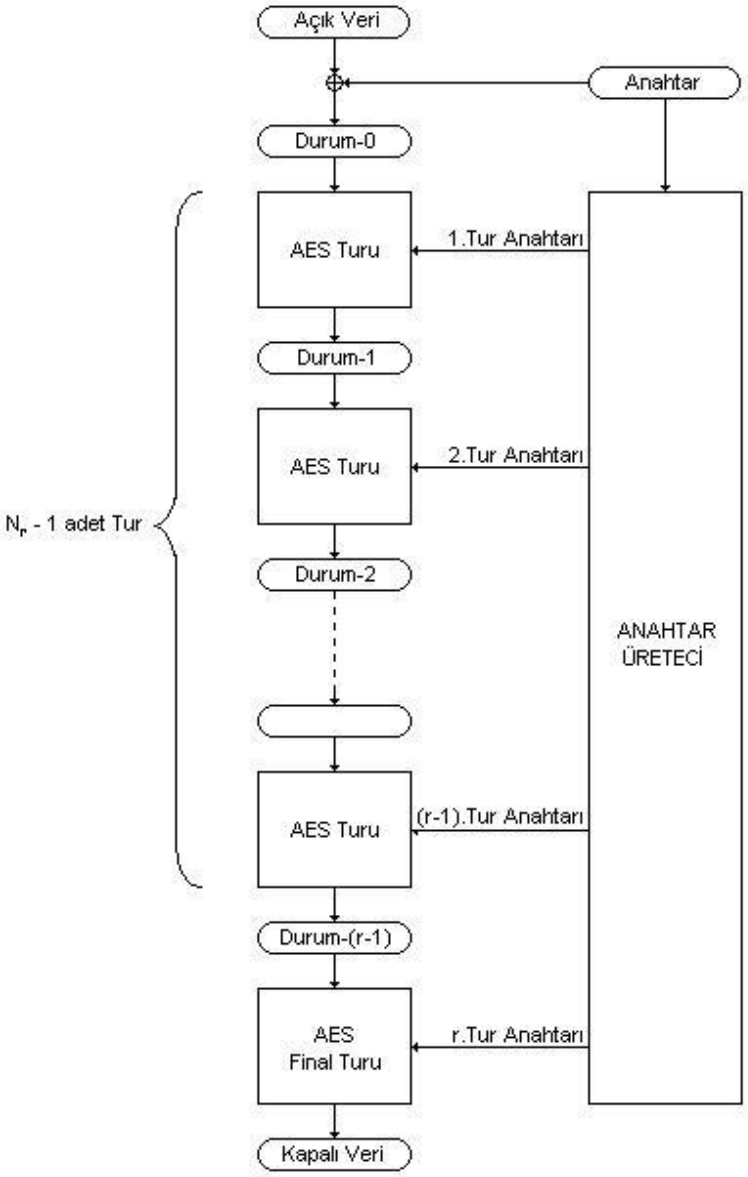
Çizelge 4.1'de farklı anahtar uzunlukları için AES algoritması tur sayıları verilmektedir.

Çizelge 4.1 AES algoritmasın anahtar uzunluğuna göre tur sayıları

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

AES algoritması genel olarak iki bloktan oluşur; ilk blok tur dönüşüm, ikinci blok ise anahtar üretim bloğudur. Bu iki blok birbirine paralel olarak çalışmakta, anahtar üretim bloğu her bir tur dönüşüm bloğu için anahtar üretimini gerçeklemektedir.

Algoritma başlangıcında öncelikle 16 bayt uzunluğundaki giriş verisi Şekil 4.2'de gösterildiği gibi *Durum Tanımlama* işlemi yapılarak *Durum Matrisi* elde edilir. Daha sonra *Durum Matrisi* içerisindeki tüm veriyi başlangıç anahtarı ile XOR işlemine tabi tutarak *Başlangıç Anahtarı Ekleme* işlemi gerçeklenir. Bu aşama ile birlikte şifreleme/çözme işlemine başlanmış olur. Bu aşamadan sonra anahtar boyuna bağlı olarak belirlenen tur sayısının bir eksiği kadar tur işlemleri gerçekleşir. Bu esnada her bir tur işleminden çıkışta elde edilen değer bir sonraki turun giriş verisini oluşturmaktadır. Ayrıca her bir turda anahtar üretim bloğu tarafından üretilen tur anahtarı kullanılmaktadır. Algoritmanın son aşamasını ise final turu oluşturmaktadır. Final turunun diğer turlardan tek farkı Bölüm 4.3'te anlatılan *Sütunları Karıştırma* işleminin bu turda gerçeklenmemesidir. Final turu sonunda elde edilen veri algoritmanın sonuç verisidir. Bu sonuç şifreleme işlemi için şifreli veri çözme işlemi için ise girişteki şifreli veriye karşılık düşen açık veridir.Şekil 4.1'deAES algoritmasının blok diyagramı görülmektedir. "N_r" tur sayısını "r" tur değerini göstermektedir.



Şekil 4.1 AES algoritması blok diyagramı

4.2 AES Algoritması Tur Dönüşüm İşlemleri

AES algoritması önceden de belirtildiği gibi tekrarlı bir yapıdan oluşur. Tur dönüşüm işlemi anahtar uzunluğuna bağlı olarak çok defa tekrarlanır. Tur dönüşüm işlemleri içerisinde; *Bayt Değiştirme, Satırları Kaydırma, Sütunları Karıştırma* ve *Tur Anahtarını Ekleme* işlemleri gerçekleştirilir. Bu işlemlerin her birine '*Adım*' olarak isim verilir. Final turu haricindeki tüm turlarda bu dört adım sırası ile tekrar edilir. Final turunda ise *Sütunları Karıştırma* işlemi gerçekleştirilmez.

AES algoritması ile kodlama işleminin yapılabilmesi için öncelikle 16 baytlık kodlanacak verinin *Durum Tanımlama* işlemi yapılır. *Durum Tanımlama*, 16 bayt uzunluğundaki veriyi 4X4'lük matris haline getirme işlemidir. Bu işlem Şekil 4.2'de gösterildiği gibi yapılmaktadır.

D_0	D_4	D_8	D_{12}
D_1	D_5	D_9	D_{13}
D_2	D_6	D_{10}	D_{14}
D_3	D_7	D_{11}	D_{15}

Şekil 4.2 Giriş verisi için durum tanımlama işlemi

Tablodaki gibi 4X4'lük matris haline getirdikten sonra ise diğer tur dönüşüm işlemleri yapılacaktır.

4.2.1 Bayt Değiştirme

Bayt Değiştirme işlemi, *Durum Tanımlama* ve *Başlangıç Anahtarı Ekleme* işlemlerinden sonra tur işlemlerinin ilk adımını oluşturur.

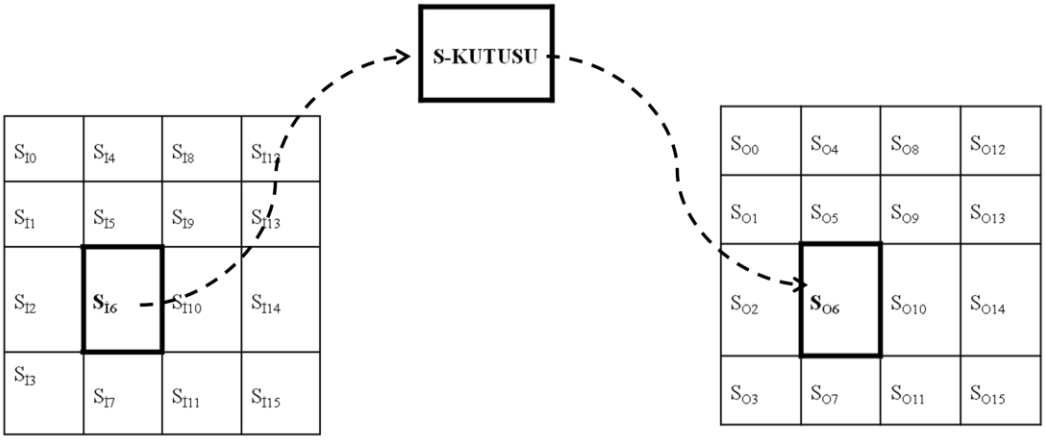
Bayt Değiştirme dönüşümü, girişindeki durumun her bir bayt'ını, Şekil 4.3'te görülen ve *S-Kutusu (S-Box) Çıkış Tablosu* adı verilen bir değiştirme tablosu kullanarak başka bir bayta dönüştürür.

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Şekil 4.3 S-Kutusu çıkış tablosu

Bu işlem aşağıda belirtilen adımlar ile gerçeklenir.

- *Bayt Değiştirme* işlemi gerçekleştirilecek bayt hexadecimal olarak ifade edilir.
- Elde edilen sonucun en önemli 4 bit kısmını gösteren değer alınır. Bu değer *S-Kutusu* tablosunda bakılacak olan satır değerini gösterir.
- Elde edilen sonucun en önemsiz 4 bit kısmını gösteren değer alınır. Bu değer *S-Kutusu* tablosunda bakılacak olan sütun değerini gösterir.
- *S-Kutusu* üzerinde, bakılan satır ve sütun değerlerinin kesiştiği hücre içerisindeki bulunan değer *Bayt Değiştirme* işleminin sonucu olarak alınır ve bayt yer değiştirme işlemine giren sayının yerine konur.



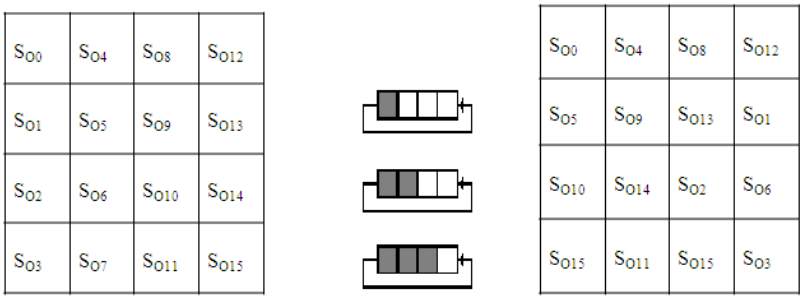
Şekil 4.4 Bayt değiştirme işlemi

Bayt Değiştirme işlemi, AES algoritması içerisinde doğrusal olmayan tek işlemdir. *Bayt Değiştirme* işleminin tersinin alınması mümkündür.

4.2.2 Satırları Kaydırma

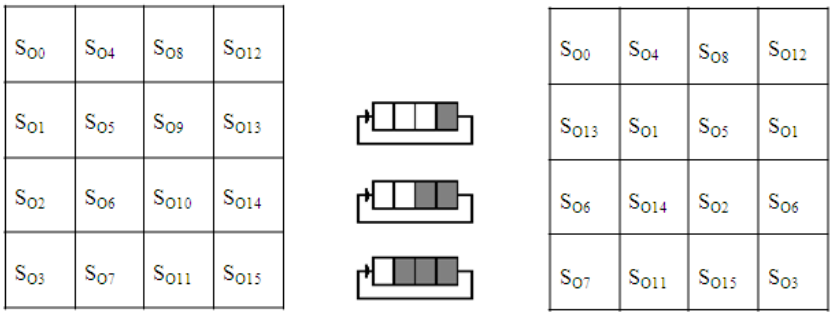
Satırları Kaydırma işlemi *Bayt Değiştirme'*den sonraki tur işlemlerinin 2. adımını oluşturmaktadır. *Bayt Değiştirme* işlemi sonucunda elde edilen 16 bayt uzunluğundaki çıkış verisi *Satırları Kaydırma* işleminin giriş verisini oluşturur. Bunun için öncelikle 16 byte uzunluğundaki veri Şekil 4.2'de durum matrisi tanımlamasında olduğu gibi 4x4'lük bir matris haline getirilir.

Şifreleme için, ilk satır aynı bırakılır. İkinci satır sağdan sola doğru bir pozisyon, değiştirecek şekilde, döngüsel olarak, kaydırılır. Döngüsel kaydırma nedeniyle, 1. sütuna gelen eleman kaydırıldığında 4.sütuna geçer. Üçüncü satır benzer şekilde iki pozisyon, dördüncü satır da üç pozisyon döngüsel olarak kaydırılır. Satırları kaydırma dönüşümünün durum baytlarının yerlerini nasıl değiştirdiği Şekil 4.5'de gösterilmiştir.



Şekil 4.5 Satırları kaydırma işlemi

Tersi işleminde ise Şekil 4.6'da gösterilmiş sekliyle işlem yapılır. 1. satır için kaydırma yapılmaz. 2. satır 1 adım soldan sağa, 3. satır 2 adım soldan sağa ve son olarak da 4. satır 3 adım soldan sağa kaydırılır. Tersi işlemi bahsedildiği gibi kodlamanın çözülmesi sırasında kullanılmaktadır.



Şekil 4.6 Ters satırları kaydırma işlemi

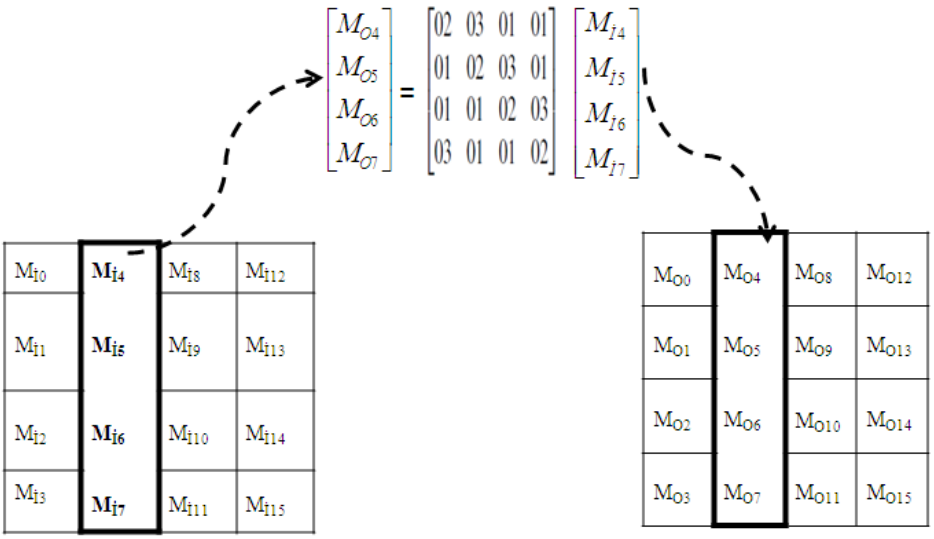
4.3 Sütunları Karıştırma

Tur işlemlerinin 3. adımı *Sütunları Karıştırma* işlemidir. *Sütunları Karıştırma* işlemi Durum matrisindeki her bir sütun üzerinde bağımsız olarak gerçeklenir. Bu işlem gerçeklenirken her bir satır $GF(2^8)$'de bir polinom olarak düşünülür. Her bir satır üzerinde bir $A(x)$ polinomu ile modülo(x^4 +1)'de çarpma işlemi gerçekleştirilir. Çarpma için kullanılan polinom aşağıdaki gibidir.

$$A(x)=\{03\}x^3+\{01\}x^2+\{01\}x^1+\{02\} \quad (4.1)$$

Sütunları Karıştırma işleminden önceki sütunların oluşturduğu polinoma $M_i(x)$, sütunları karıştırma işleminden sonraki sütunların oluşturduğu polinoma $M_O(x)$ denirse, $M_O(x)$polinomu aşağıdaki şekildeki gibi oluşturulmaktadır.

$$M_{O(x)}=A(x)\ M_{i(x)} \qquad (4.2)$$



Şekil 4.7 Sütunları karıştırma işlemi

Çözme işlemindeyse, *Sütunları Karıştırma* dönüşümünün tersi kullanılmaktadır. Ters dönüşümde de benzer işlemler yürütülmektedir. Aynı şekilde, giriş durumunun sütunlarından oluşturulan polinomlar sabit bir polinomla çarpılmaktadır. Yalnız ters alma işlemindeki sabit polinom $GF(2^8)$'de kodlama işleminde kullanılan sabit polinomun çarpmaya göre tersi olarak hesaplanır.

$$A'(x)=\{0b\}x^3+\{0d\}x^2+\{09\}x^1+\{0e\} \qquad (4.3)$$

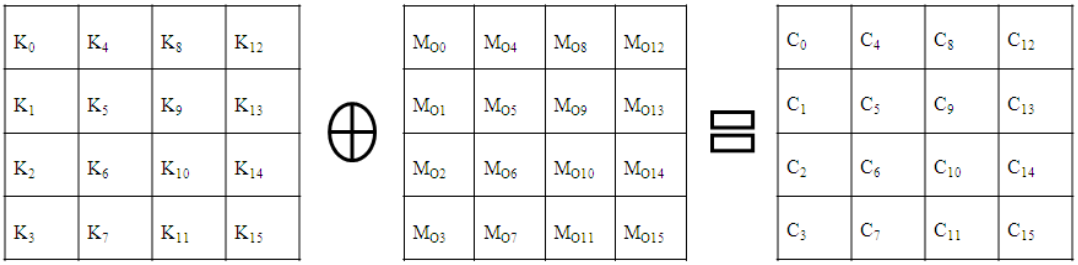
4.3.1 Anahtar Ekleme İşlemi

Turun son adımı olan *Tur Anahtarını Ekleme* işlemi $GF(2^8)$ üzerinde doğrusal bir işlem olup şifreleme ve çözme işlemleri için bu dönüşüm tamamen aynıdır. Yani bu dönüşümün tersi kendisine eşittir.

Tur Anahtarını Ekleme işleminin gerçeklenmesi için öncelikle *Tur Anahtarı Matrisi*'nin aşağıda belirtildiği şekilde oluşturulması gerekir. *Tur Anahtarı Matrisi*'nin oluşturulmasından sonra

Sütunları Karıştırma İşlemi sonunda elde edilen durum matrisi ile anahtar matrisi XOR işlemine tabi tutulur.

Şekil 4.8'de *Anahtar Ekleme* gösterilmektedir. Bu işlemde $M_{O(x)}$ sütun karıştırma sonucu elde edilen veriyi, $K_{(x)}$*Tur Anahtarı Matrisi*'ni, $C_{(x)}$ ise tur sonucu elde edilen veriyi göstermektedir.



Şekil 4.8 Tur anahtarı ekleme işlemi

<u>Anahtar Üretme Fonksiyonu:</u>

AES algoritmasının tur işlemleri başlamadan önce şifreleme/çözme anahtarının ana hali ile giriş verisi toplanır. Daha sonra tur işlemleri aşamasına geçilir. Her bir tur sonunda tekrar anahtar ekleme işlemi gerçeklenir. Tur sonlarında eklenen anahtar değeri başlangıç anahtarından *Anahtar Üretme Fonksiyonu* kullanılarak üretilir. Aşağıda Anahtar Üretme Fonksiyonu adımları sıralanmaktadır [30].

N_K Anahtarın kelime sayısı olmak üzere,

- İlk N_K sütun ana anahtar yazılarak Şekil 4.9'daki gibi oluşturulur.

K_0	K_4	K_8	K_{12}
K_1	K_5	K_9	K_{13}
K_2	K_6	K_{10}	K_{14}
K_3	K_7	K_{11}	K_{15}

Şekil 4.9 Ana anahtar ilk N_K Sütunu

- Sonraki sütunlar, kendisinden bir önceki sütun ile N_K önceki sütunun karşılıklı baytlarının xor işlemi yapılmasıyla oluşturulur. Yalnız oluşturulmakta olan sütun numarası N_K'nın tam katı olan bir sütun numarası ise bu sütun oluşturulur kendisinden bir önceki sütun bir dönüşüm işleminden geçirildikten sonra kendisinden N_K önceki sütun ile karşılıklı bayt bayt xor işlemine sokulur. Bu dönüşüm ise 3adımda yapılmaktadır [7].
    - Dönüşümün ilk adımında sütunda bir adım kaydırma işlemi yapılır. Üzerinde işlem yapılacak sütun aşağıdan yukarıya doğru olmak üzere bir adım döngüsel olarak kaydırılır. Örnek olarak Şekil 4.9'daki 3.sütunu kaydıralım.

      $\{K_8,K_9,K_{10},K_{11}\}\rightarrow\{K_9,K_{10},K_{11},K_8\}$
    - Dönüşümün ikinci adımında ise *Bayt Değiştirme* işlemi uygulanır. Bu işlem için Bölüm 4.2.1'de anlatıldığı gibi *S-kutusu* kullanılır.
    - Dönüşümün son adımında ise elde edilen sütun tur sabiti işlemi ile xor işlemine sokulur. Bu tur sabiti bütün turlar için değişen bir sabittir.

Çizelge 4.2 Anahtar üreteci tur sabitleri

Tur Sayısı	Tur Sabiti	Tur Sayısı	Tur Sabiti
1	01 00 00 00	6	20 00 00 00
2	02 00 00 00	7	40 00 00 00
3	04 00 00 00	8	80 00 00 00
4	08 00 00 00	9	1B 00 00 00
5	10 00 00 00	10	36 00 00 00

Yukarıda verilen adımlar izlenerek N_R tur sayısını göstermek üzere, her bir tur için $4 \times (N_R + 1)$ boyutunda *Tur Anahtarı Matrisi* oluşturulur. Bu matrisin yan yana gelen 4 sütunu 128 bitlik tur anahtarlarını oluşturmaktadır[30].

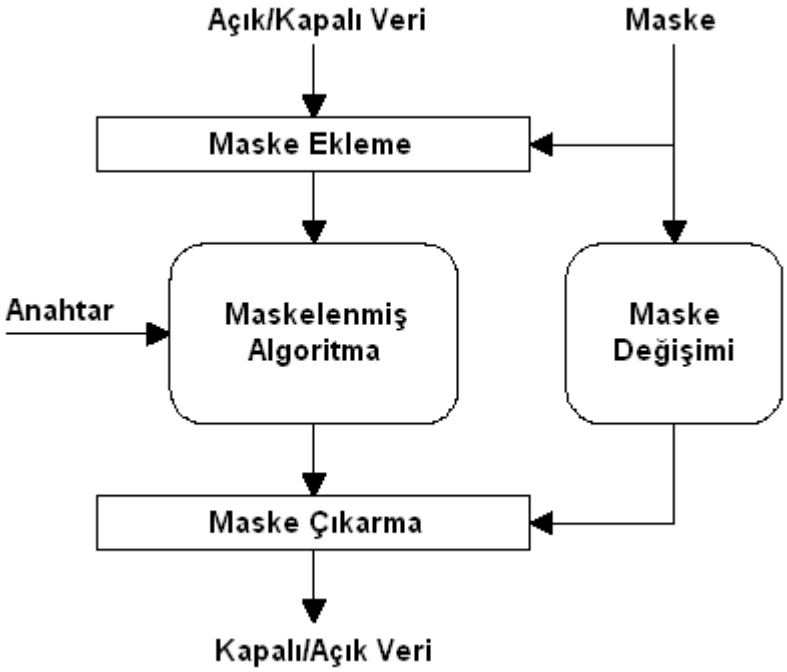
4.4 AES Algoritmasının Maskelenmesi

Yan kanal analizi saldırılarının, özellikle de *Farksal Güç Analizi* saldırılarının kriptografik sistemlere yönelik ciddi bir tehdit oluşturması sonucunda, bu saldırıları önlemek için kullanılacak karşı önlem yöntemleri üzerinde durulmuştur. Bu bağlamda Bölüm 3.1.4'te anlatılan farklı önlemler önerilmiştir. Bu önlemler içerisinde maskeleme yöntemi en fazla tercih edilen güvenlik yöntemi olarak öne çıkmıştır.

Daha önce belirtildiği gibi saldırgan, FGA saldırısını gerçekleştirmek için saldırmak istediği kriptografik sistemin bir modelini oluşturur. Bu modele dayanarak, çeşitli giriş değerleri için cihazın hedef alınan bir kısmının tüketeceği gücü tahmin etmeye çalışır. Hedef alınan algoritma parçasının çıkışının az sayıda anahtar biti ve bilinen giriş/çıkış verisine bağlı olması gerekir. Böylece, tahmini anahtar değerleri için, giriş/çıkış verisi de kullanılarak algoritma parçasının çıkışı hesaplanabilir. İşte maskeleme yöntemi tam bu noktada devreye girmektedir. Çünkü maskeleme yönteminde algoritmanın ürettiği ve saldırganın modelini çıkardığı ara değere saldırganın bilmediği "maske" isminde bir değer daha eklenir. Bu durumda artık saldırı gerçeklenen noktada saldırganın ürettiği model değerlerinden tamamen farklı başka bir değer yer alacaktır. Saldırgan, ürettiği model ile gerçek güç tüketim değerinin arasında bir korelasyon elde edemeyecektir. Maske değerini de hesaba katarak bir model oluşturma istendiğinde ise pratikte gerçeklenemeyecek hesap sayıları ortaya çıkacaktır.

Maskeleme yönteminde rastgele olarak üretilen maske değeri anahtar ve açık veri değeri ile beraber algoritma girişine eklenir. Algoritma boyunca üretilen tüm ara değerlere ekli olan maske değeri saldırganın gerekli güç modelini oluşturmasına karşı bir önlem olarak kullanılır. Ancak algoritma sonunda, girişe verilen açık veri ve anahtar verisine karşılık düşen doğru şifreli veri değerinin elde edilmesi için maske değerinin ayrılması gerekmektedir. Bunun sağlanması için algoritma işlemleri devam ederken bir taraftan da maske bloğu işlenmeye devam edilir. Algoritma sonunda üretilen maskeli çıkış verisi ile girişten çıkışa kadar

algoritma evrelerinden geçmiş maske verisi birlikte xor işlemine tabi tutularak maskesiz şifreli veri elde edilir.

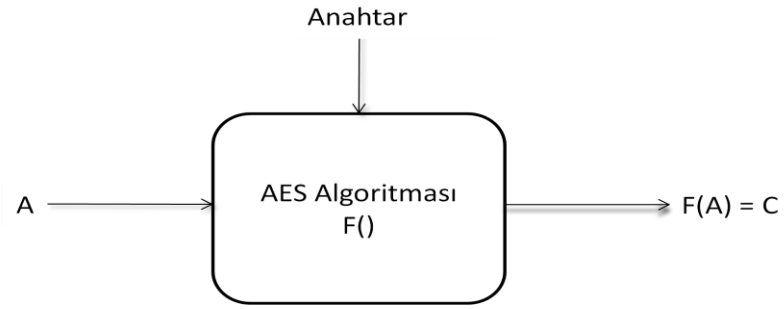


Şekil 4.10 Maskeleme işlemi akış diyagramı

Yukarıda belirtilen adımların doğru bir şekilde işlemesi algoritmanın doğrusal bir şekilde çalışması şartına bağlıdır.

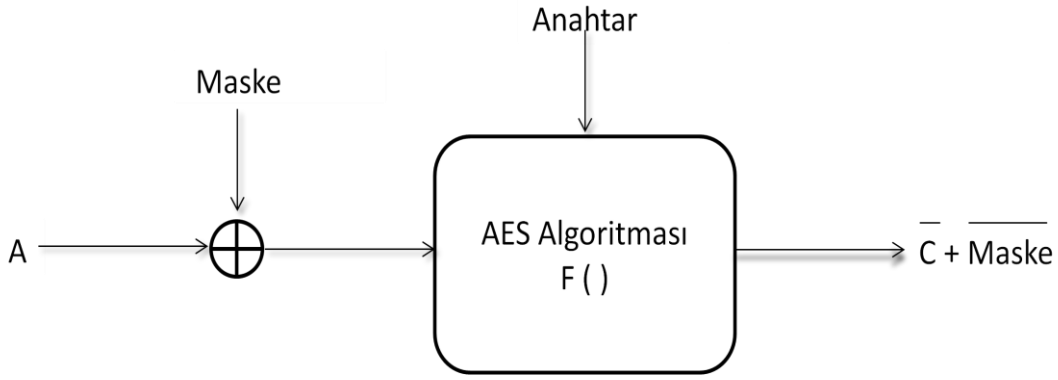
$f(x)= 2x$ iken $f(x+m) = 2x + 2m$ eşitliğini sağlayan sistemlere *Doğrusal Sistemler* adı verilir. $f(x)= x^2$ iken $f(x+m) = x^2 + 2xm+ m^2$ seklinde olan sistemlere ise *Doğrusal Olmayan Sistemler* adı verilir [17].

AES algoritmasında gerçekleştirilen *Bayt Değiştirme* işlemi içerisinde yer alan $G(2^8)$'de ters alma işlemi doğrusal bir işlem değildir. Bu nedenle, giriş verisine toplamsal bir değer eklenmesi, çıkıştaki verinin olması gerekenden farklı bir değere sahip olmasına neden olur. Şekil 4.11'de maskesiz olarak çalışan algoritmanın normal çalışması ve ürettiği sonuç görülmektedir.



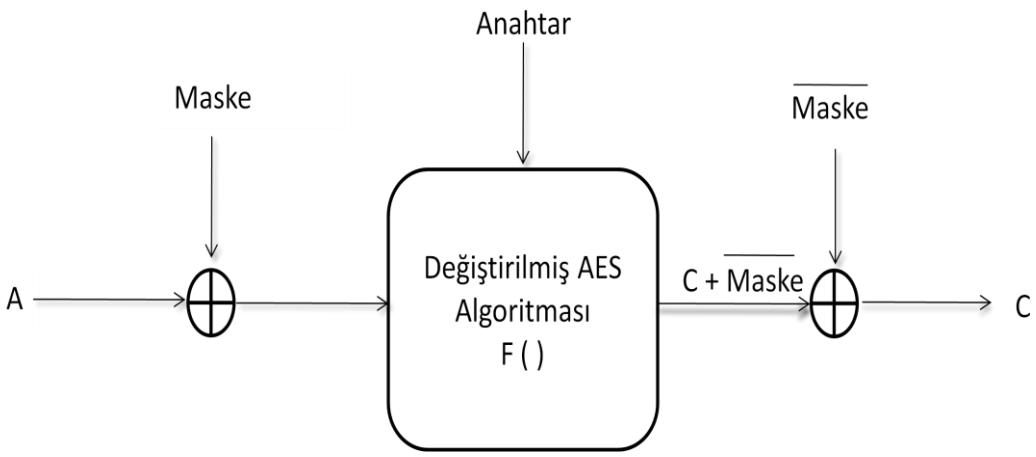
Şekil 4.11 Maskesiz AES algoritması sonuç üretimi

Şekil 4.12'de ise aynı algoritmaya maske eklenmesi durumunda ortaya çıkacak çalışma yapısı ve üretilecek sonuçlar görülmektedir. Görüldüğü gibi algoritmanın bu şekilde kullanılması durumunda üretilen çıkış değeri doğru çıkış değeri olmayacaktır.



Şekil 4.12 AES algoritmasına maskeli giriş uygulanması

Bunun önüne geçilmesi için AES algoritmasının maskeleme verisine karşı doğrusal davranacak şekilde değiştirilmesi gerekir. Değiştirilmesi gereken adım ise *Bayt Değiştirme* adımı ve bu adım içerisinde de $G(2^8)$'de ters alma işlemidir. Değiştirme işleminden sonra oluşacak sistem Şekil 4.13'te verilmiştir.



Şekil 4.13 Değiştirilmiş AES algoritmasına maskeli giriş uygulanması

4.5 Maskeleme Yöntemi

AES S-Kutularının, dolayısıyla ters alma işleminin maskelenmesine yönelik ilk çalışma Akkar tarafından yapılmış, daha sonra bu konuda pek çok farklı yöntem önerilmiştir [7]. İlk akla gelen maskeleme yöntemi *Tablo Değiştirme* yöntemidir.

Bayt Değiştirme işlemi bölüm 4.2.1'de belirtildiği şekilde 8-bit verilerin, Şekil 4.3'te verilen *S-kutusu* tablosuna göre dönüştürülmesi işlemi ile elde edilir. Veri değerine karşılık düşecek maske değeri ise rastgele seçilmiş bir değerdir. Bu değer analiz edilen bir bayt için 2^8=256 farklı değer alabilir.

Tablo Değiştirme yönteminde, giriş verisine eklenecek 256 farklı maske değeri için ayrı tablo oluşturulur. Giriş verisine eklenen maske değerine göre bu tablolardan biri seçilerek *Bayt Değiştirme* işlemi gerçekleştirilir.

Tablo Değiştirme yöntemi, oluşturulacak tabloların tüm devrelerde kaplayacağı yer nedeniyle tercih edilmeyen bir yöntemdir [19].

Tablo değiştirme yöntemi dışında başka birçok maskeleme yöntemi önerilmiştir. Bu yöntemlerden [29] ve [31]'in, *Sıfır Değeri Saldırıları'*na karşı açık olduğu görülmüştür [32]. [31] ile önerilen yöntemin ayrıca 1.derece FGA saldırısına karşı açıklığı bulunduğu tespit edilmiştir [33].

[32] ile önerilen yöntem karmaşık olması nedeniyle pratik olarak uygulamaya müsait değildir.

[34] ve [35] ile önerilen yöntemler teorik olarak güçlü ve uygulanabilir görülmektedir.

Bu yöntemlere ek olarak önerilen diğer bir maskeleme yöntemi Oswald ve arkadaşları tarafından önerilen IAIK yöntemidir [11]. Bu yapı, [34] ile benzer özellik göstermesine karşın uygulamada çok daha yaygın olarak kullanılan bir yöntemdir.

Tez çalışması kapsamında üzerinde çalışılan uygulamada, yan kanal analizi saldırısına karşı alınan güvenlik önlemi de IAIK maskesi kullanılmıştır. Bu nedenle Bölüm 4.5.1'de bu konu ayrıntılı olarak anlatılacaktır.

4.5.1 IAIK Yöntemi ile AES Algoritmasının Maskelenmesi

Kriptografik algoritmaların yan kanal analizine karşı maskeleme yöntemi ile korunmasında ilk akla gelen yöntem *Tablo Değiştirme* yöntemidir. Ancak bu yöntem, büyük işlemsel yük getiren tabloların her AES şifreleme/çözme işleminin yürütülmesinde tekrar hesaplanmasını ya da tüm olası maske değerleri için önceden hesaplanmış tabloların ROM'da saklanarak büyük alan tüketimi oluşması durumlarını gerektirmektedir. *Tablo Değiştirme* yönteminin dezavantajlarını ortadan kaldırmak için önerilen yöntemlerin güvenilirlik ve uygulanabilirlikleri ise yukarıda belirtilmiştir.

Başta *Sıfır Değeri Saldırıları* olmak üzere, FGA saldırılarına karşı teorik bir zayıflığı olmayan ve daha düşük karmaşıklığa sahip olması nedeniyle gerçeklemeyi mümkün kılan bir yöntem Oswald tarafından önerilmiştir [11]. Bu yöntemde, sıfır değeri saldırılarına engel olabilmek, için tüm ara değerler toplamsal maskeyle maskelenmiştir.

Ancak daha önce de belirtildiği gibi *Bayt Değiştirme* adımı içerisinde yer alan $G(2^8)$'de ters alma işlemi doğrusal bir işlem değildir. Bu durum maskesiz veriye karşılık üretilecek çıkışın geri dönüşü olmayacak şekilde kaybedilmesine neden olmaktadır. Bunun önüne geçilmesi için IAIK maskesinde ters alma işlemini gerçekleştiren fonksiyon toplamsal maskeyi koruyarak doğru çıkışı üretecek şekilde değiştirilmiştir. Fonksiyon üzerinde yapılan değişiklik *Bayt Değiştirme* işleminin tablo ile değil de kombinezonsal yöntemlerle gerçekleşmesi ihtiyacını doğurmuştur.

IAIK maskeleme yönteminin anlaşılabilmesi için $G(2^8)$ ters alma işleminin anlaşılması gerekmektedir.

GF(2^8)'de Maskeli Ters Alma İşlemi

GF(2^8)'de maskeli ters alma işlemi için öncelikle *GF*(2^8)'deki bir eleman, katsayıları *GF*(2^4)'ün elemanları olan ikinci dereceden bir polinomla temsil edilir.

$$a \in GF(2^8)$$
$$a = a_h \mathrm{x} + \mathrm{a}_1 \qquad a_h, a_l \in GF(2^4) \tag{4.4}$$

Toplamsal maske kullanımı, alt alana geçişte bir zorluk yaratmaz. Çünkü geçiş işlemi (polinom katsayılarının oluşturulması) doğrusal bir işlemdir;

$$a, m \in GF(2^8)$$
$$(a+m) = (a_h + m_h)\mathrm{x} + (a_l + m_l) \qquad a_h, a_l, m_h, m_l \in GF(2^4) \tag{4.5}$$

Bu polinomun katsayıları kullanılarak, GF(2^8)'de ters alma işlemi, GF(2^4)'te ters alma işlemi kullanılarak gerçekleştirilebilir;

$$a^{-1} = b = b_h x + b_l \qquad b_h, b_l \in GF(2^4)$$

$$b_h = a_h \times \overline{d} \tag{4.6}$$

$$b_l = (a_h + a_l) \times \overline{d} \tag{4.7}$$

$$d = (a_h + a_l) \times a_l + \lambda \times a_h^{\,2} \qquad \lambda, d \in GF(2^4) \tag{4.8}$$

$$\overline{d} = d^{-1} \tag{4.9}$$

Ters alma işlemi, maskeli giriş değerleri için, yukarıdaki denklemler (4.6, 4.7, 4.8, 4.9) aşağıda önerildiği gibi değiştirilebilirse, toplamsal maskeyi koruyarak gerçekleştirilebilir;

$$(a+m)^{-1} = (b+m) = (b_h + m_h)x + (b_l + m_l) \qquad b_h, b_l \in GF(2^4)$$

$$(b_h + m_h) = (a_h \times \overline{d}) + m_h = f_{b_h}((a_h + m_h), (a_l + m_l), (d + m_d), m_h, m_l, m_d) \tag{4.10}$$

$$(b_l + m_l) = (a_h + a_l) \times \overline{d} + m_l = f_{b_l}((a_h + m_h), (a_l + m_l), (d + m_d), m_h, m_l, m_d) \tag{4.11}$$

$$\begin{aligned}(d + m_d) &= (a_h + a_l) \times a_l + \lambda \times a_h^{\,2} + m_d \\ &= f_d((a_h + m_h), (a_l + m_l), (d + m_d), m_h, m_l, m_d)\end{aligned} \tag{4.12}$$

$$(\overline{d} + \overline{m_d}) = d^{-1} + m_d = f_{\overline{d}}((a_h + m_h), (a_l + m_l), (d + m_d), m_h, m_l, m_d) \tag{4.13}$$

Burada dikkat edilmesi gereken nokta, yukarıdaki ifadeleri (4.10, 4.11, 4.12, 4.13) elde edebilmek için, elimizde sadece maskeli değerler ($(a_h + m_h), (a_l + m_l)$) ve maske değerleri (m_h, m_l) bulunmasıdır. f fonksiyonları, bu terimleri kullanarak, yukarıdaki ifadelerin sağ tarafını elde etmemizi sağlayacak şekilde oluşturulmalıdır [22].

_f_ Fonksiyonlarının Oluşturulması

(4.3, 4.4, 4.5, 4.6) denklemlerinin sağ taraflarında, (a_h, a_l, d) değerleri yerine maskeli değerler ($(a_h + m_h), (a_l + m_l), (d + m_d)$) yerleştirerek, (4.7, 4.8, 4.9, 4.10) denklemleri elde edilmeye çalışılır. Bu sırada ortaya çıkacak olan fazladan terimler ise, yine ($(a_h + m_h), (a_l + m_l)$) ve maske değerleri (m_h, m_l) kullanılarak üretilen düzeltme terimleriyle yok edilir. Sırasıyla f fonksiyonları şu şekilde elde edilebilir;

(6.3) denkleminde a_h yerine $(a_h + m_h)$, $\overline{d}$ yerine de $(\overline{d} + \overline{m_d})$ koyulursa;

$$(a_h + m_h) \times (\overline{d} + \overline{m_d}) = \underbrace{a_h \times \overline{d}}_{\substack{\text{istenilen} \\ \text{terimler}}} + \underbrace{m_h \times \overline{d} + a_h \times \overline{m_d} + m_h \times \overline{m_d}}_{\substack{\text{istenilmeyen} \\ \text{terimler}}} \quad (4.14)$$

elde edilir. İstenmeyen terimlerin yok edilmesi ve m_h teriminin eklenmesiyle istenilen fonksiyon elde edilebilir. $\overline{m_d} = m_l$ seçilirse, f_{b_h} fonksiyonu şu şekilde elde edilir;

$$f_{b_h} = (a_h + m_h) \times (\overline{d} + \overline{m_d}) + \underbrace{(\overline{d} + m_l) \times m_h + (a_h + m_h) \times m_l + m_h \times m_l}_{\substack{\text{düzeltme} \\ \text{terimleri}}} + \underbrace{m_h}_{\substack{\text{eklenen} \\ \text{terim}}} \quad (4.15)$$

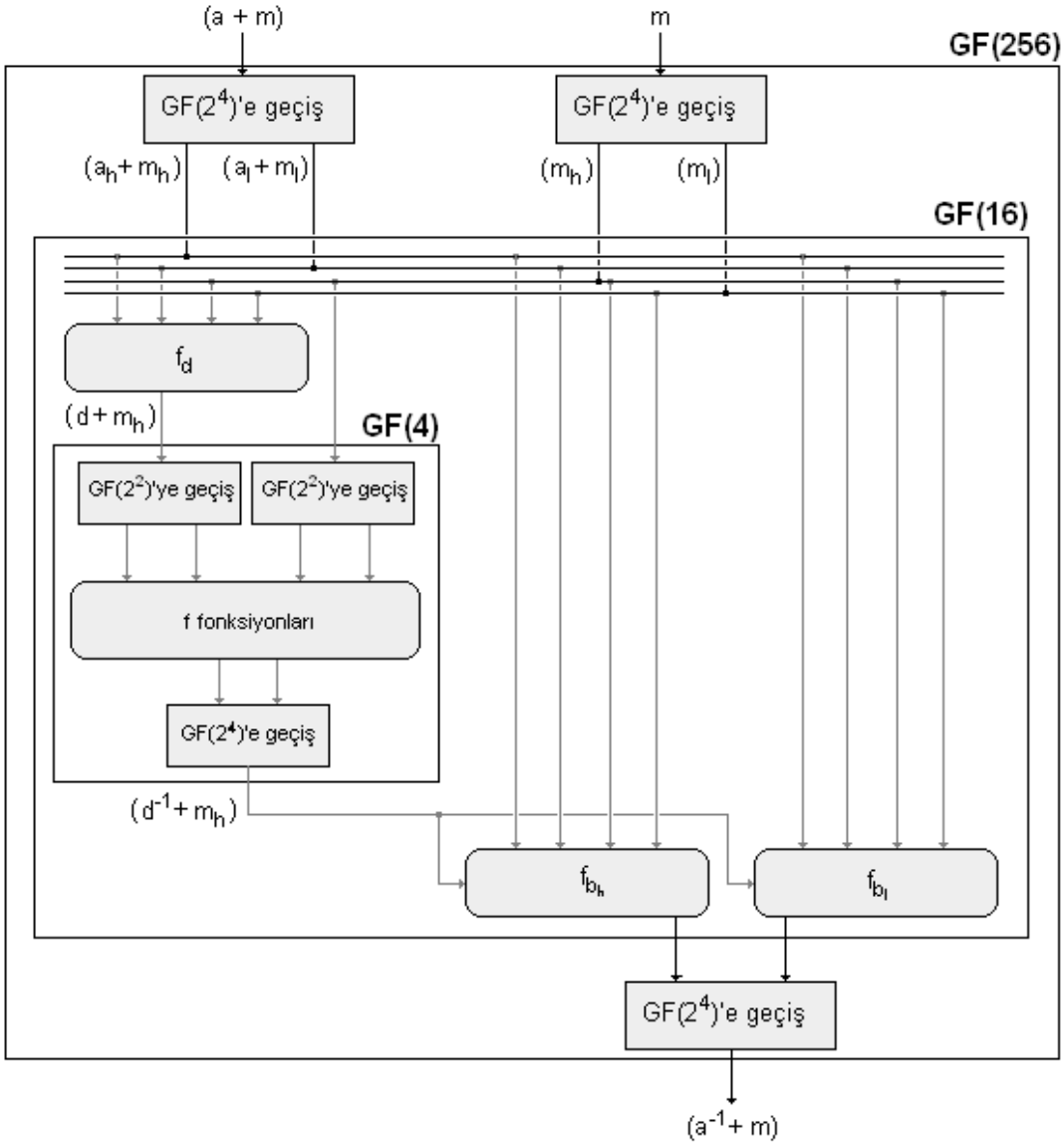
(4.15) denkleminin sonucu ($a_h \times \overline{d} + m_h$) kullanılarak, yine düzeltme ve ekleme terimlerinin yardımıyla, $\overline{m_d} = m_h$ seçilirse, f_{b_l} fonksiyonu şu şekilde elde edilir;

$$\begin{aligned} f_{b_l} = {} & \underbrace{(a_h \times \overline{d} + m_h)}_{b_h \text{ ifadesi}} + \underbrace{m_l}_{\substack{\text{eklenen} \\ \text{terim}}} \\ & + \underbrace{(a_l + m_l) \times (\overline{d} + m_h) + (\overline{d} + m_h) \times m_l + (a_l + m_l) \times m_h + m_h + m_h \times m_l}_{\substack{\text{düzeltme} \\ \text{terimleri}}} \end{aligned} \quad (4.16)$$

(4.7) denkleminde; a_h yerine $(a_h + m_h)$, a_l yerine $(a_l + m_l)$, $\overline{d}$ yerine de $(\overline{d} + \overline{m_d})$ koyulup, yine düzeltme ve ekleme terimlerinin yardımıyla, f_d fonksiyonu şu şekilde elde edilir;

$$f_d = (a_h + m_h)^2 \times \lambda + (a_l + m_l) \times (a_h + m_h) + (a_l + m_l)^2 + (a_h + m_h) \times m_l + (a_l + m_l) \times m_h + m_h^2 \times \lambda + m_l^2 + m_h \times m_l + m_h \quad (4.17)$$

Geriye bir tek 4.14 denkleminin gerçekleştirilmesi kalmaktadır. Bu işlem GF(2^4)'te ters alma işleminden ibarettir. $(d + m_d)$'in tersinin maskeyi koruyarak alınması ($(d^{-1} + m_d)$) gerekmektedir. Bu amaçla *GF*(2^4)'ten *GF*(2^2)'ye inilerek, (4.14, 4.15, 4.16, 4.17) denklemleri *GF*(2^2) için yeniden oluşturulur. GF(2^2)'deki f fonksiyonları benzer şekilde gerçekleştirilebilir. *GF*(2^2)'de $(i + j)$'nin tersinin bulunması kare alma işlemine denk düşer. Bu işlem de doğrusal bir yapıdadır ($(a + m)^2 = a^2 + m^2$). Böylelikle ters alma işleminde maske korunmuş olur. Tekrar *GF*(2^4) alanına çıkılarak $(d^{-1} + m_d)$ değeri elde edilir. Bu değer (4.15, 4.16, 4.17) denklemlerinde kullanılarak, maskeyi koruyacak şekilde *GF*(2^8)'de ters alma işlemi gerçekleştirilmiş olur. IAIK yöntemiyle ters alma işlemini gerçekleştiren devrenin blok şeması Şekil 4.14'de gösterilmiştir.



Şekil 4.14 IAIK yöntemiyle ters alma işleminin maskelenmesi

Maskeleme işlemi için Şekil 4.14'deki gibi yeniden oluşturulan ters alma bloğu AES algoritmasında kullanılmak üzere *Bayt Değiştirme* işlemindeki standart ters alma bloğunun yerine yerleştirilir. Bu sayede *Bayt Değiştirme* işlemindeki ters alma bloğunun non-lineer olmasından kaynaklanan maskenin kontrol edilememesi durumundan kaçınılmış olacaktır. Şekil 4.14'e dikkat edilirse maske ters alma bloğuna nasıl girmişse hiçbir değişikliğe uğramadan o şekilde çıkmaktadır. Bu durum en son adımda şifreli veriden maskenin etkisinin kaldırılabilmesi için önemlidir.

BÖLÜM 5

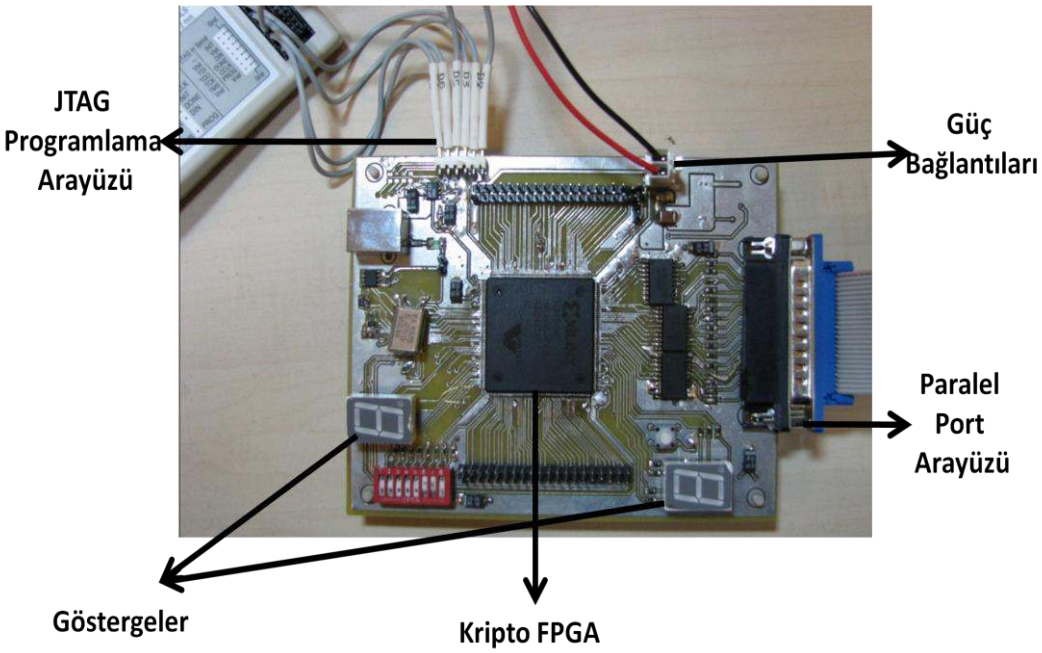
FPGA ÜZERİNDE IAIK MASKELİ AES ALGORİTMASI ÇALIŞMA DÜZENEĞİ

Bu bölümde tez çalışması kapsamında üzerinde çalışılan sistemden bahsedilecektir. Bu kapsamda; sistemi oluşturan donanımsal ve yazılımsal bileşenler, ölçüm alma düzeneğinin oluşturulması için gerekli olan ek modüller, ölçüm alma düzeneği ve ölçüm alma işlemi anlatılacaktır.

5.1 Sistem Donanımı

Kriptografik sistemin üzerinde çalıştığı elektronik devre modülü Doktora Tezi Çalışması kapsamında Yüksek Mühendis Abid Üveys Danış tarafından tasarlanan elektronik karttır. Analiz yapılan AES kripto algoritması, kart üzerinde yer alan Virtex E ailesinin XCV1000E modeli FPGA üzerinde koşmaktadır. FPGA elemanına kod yükleme işlemi kart üzerinde özel pinlerle tanımlanmış JTAG ara yüzü aracılığı ile gerçeklenmektedir. Sistem üzerinde güç ihtiyacının karşılanması için harici güç kaynağı bağlantı noktaları ve bilgisayar ile haberleşmeyi sağlamak için Paralel Port ara yüzü vardır. Ayrıca sisteme şifrelemek için gönderilen açık veriyi ve şifrelenen kapalı verinin 1 baytını gösteren 2 adet 7 parçalı gösterge kart üzerinde yer almaktadır.

Şekil 5.1'de sistem donanımının görünümü verilmiştir.



Şekil 5.1 Tez çalışması kapsamında üzerinde çalışılan sistem donanımı

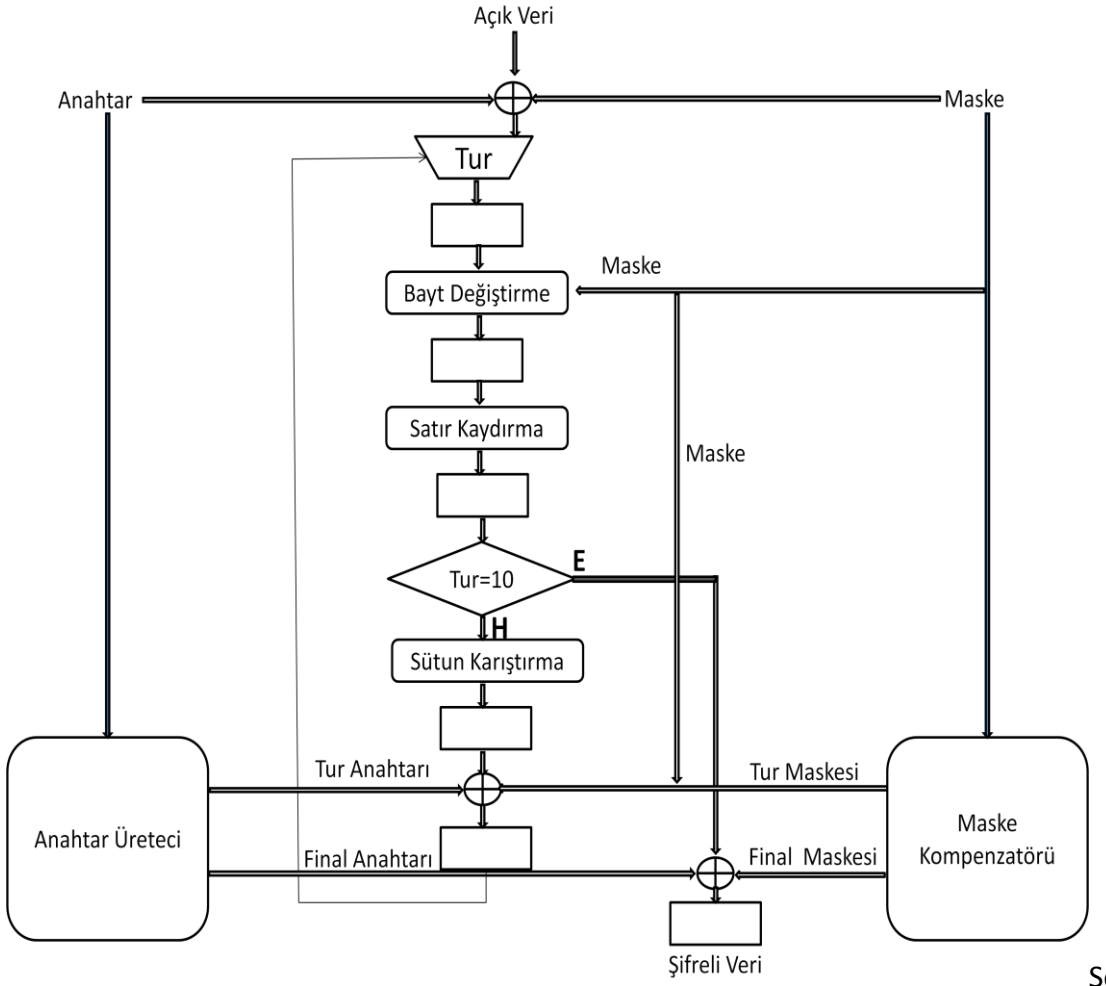
5.2 Sistem Yazılımı

Çalışılan sistemin yazılımını oluşturan "*IAIK Maskeli AES Algoritması*" uygulamasına ait VHDL kodu, Yüksek Lisans Tezi çalışması kapsamında Yüksek Müh. Levent ORDU tarafından geliştirilmiştir [19]. Çalışmada, AES Algoritmasına yapılacak yan kanal analizi saldırılarına karşı güvenlik sağlamak amacıyla, teorik olarak önerilen IAIK maskesinin pratik bir uygulaması gerçeklenmiştir. Bölüm 0'da yapılan saldırı uygulamalarının anlaşılabilmesi için üzerinde çalışılan uygulamaya ilişkin temel tasarım özelliklerinin bilinmesi gerekmektedir. Her bir saldırıya ilişkin ayrıntılı tasarım özellikleri ise söz konusu saldırı başlığı altında anlatılacaktır. *IAIK Maskeli AES Algoritması* aşağıdaki özelliklere sahiptir.

- Standart AES uygulaması 128-bit uzunluğundaki bloklar halinde veri işlemektedir.
- Kullanılan anahtar boyu 128-bit uzunluğundadır. Anahtar boyuna bağlı olarak, algoritma 10 tur işleminden meydana gelmektedir.
- Bölüm 4.2'de verildiği gibi her bir turda 4 adet işlem gerçekleşmektedir. Üzerinde çalışılan uygulamada, herhangi bir turun çıkışının bir sonraki tur için giriş oluşturması adımı eklenmiştir. Dolayısıyla algoritmanın bir turu 5 adımda yaptığı düşünülebilir. Ayrıca son turda gerçeklenmemesi gereken *Sütunları Karıştırma* işlemi de turların

standarda uyması açısından gerçeklenmiş ancak sonucu kullanılmamıştır. Sonuç olarak her bir AES işlemi 50 adımda gerçeklenmiştir.

- Algoritmanın işleyişine paralel olarak maske verisi de işleme tabi tutuluştur. Her bir tur sonunda işlenmiş maske verisi algoritma tur sonucu ile işleme sokularak doğru çıkış verisinin elde edilmesi sağlanmıştır.
- IAIK maskeleme yönteminin gereği olarak *Bayt Değiştirme* işlemi içerisinde yer alan $G(2^8)$'de ters alma işlemi kombinezonsal olarak gerçeklenmiştir.



Şekil 5.2

Üzerinde çalışılan IAIK maskeli AES algoritması akış diyagramı

5.3 Ölçüm Alma Düzeneği

Üzerinde çalışılan sistemde DPA saldırısının gerçeklenmesi için çok sayıda güç ölçümü alınması gerekmektedir. Ölçüm alma düzeneğinin kurulması için aşağıda listelenen donanım ve yazılım modüllerine ihtiyaç vardır.

Donanım Modülleri

Güç Kaynağı: Kripto Sistemi için ihtiyaç duyulan 5V DC gücü sağlamaktadır.

JTAG Programlama Modülü: Kripto algoritması konfigürasyon dosyasının FPGA üzerine yüklenmesi için kullanılacaktır.

Ölçüm Probu: Kripto sistemi güç hattındaki güç tüketimini ölçerek osiloskopa taşır.

Osiloskop: Ölçüm probu tarafından alınan ölçümleri göstermekte kullanılır.

Bilgisayar: Ölçüm alma sürecinde; açık verilerin kripto sistemine gönderilmesi ve şifreli verilerin alınması, osiloskop tarafından gösterilen güç ölçümlerinin depolanması işlemlerini yerine getirecektir.

Paralel Port Kablosu: Bilgisayar ile kripto sistemi arasında haberleşmeyi sağlamak için kullanılır.

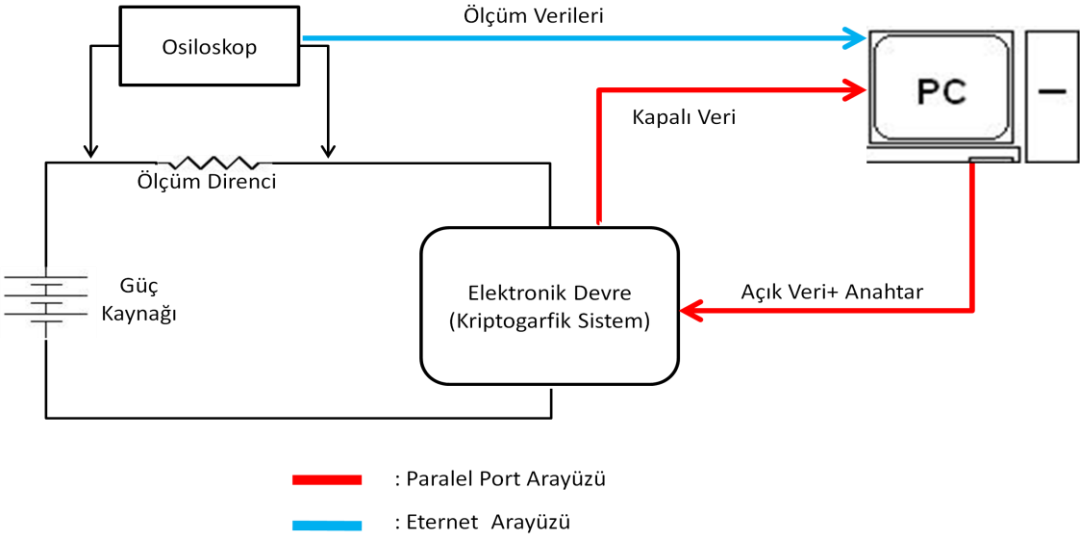
Eternet Kablosu: Bilgisayar ile osiloskop arasında haberleşmeyi sağlamak için kullanılır.

Yazılım Modülleri

Ölçüm Alma Düzeneğinin çalışabilmesi için C++ geliştirme ortamında *Ölçüm Alma Programı* yazılım modülü geliştirilmiştir. Program temel olarak ölçüm alma sürecini yönetmekle beraber aşağıda listelenen işlevleri yerine getirmektedir.

- Bilgisayar ile kripto sistemi/osiloskop arası haberleşmeyi kurar ve yönetir.
- Güç ölçümü için kullanılacak açık veri ve anahtar verisini kripto sistemine gönderir.
- Şifreleme işlemi sonunda kripto sistemi tarafından üretilen şifrelenmiş verileri depolar.
- Osiloskop tarafından alınan güç ölçümü verilerinin bilgisayar tarafından alınarak depolanmasını sağlar.

Yukarıda listelenen donanım ve yazılım modüllerinin birleşiminden oluşan ölçüm alma düzeneği Şekil 5.3'te verilmiştir.

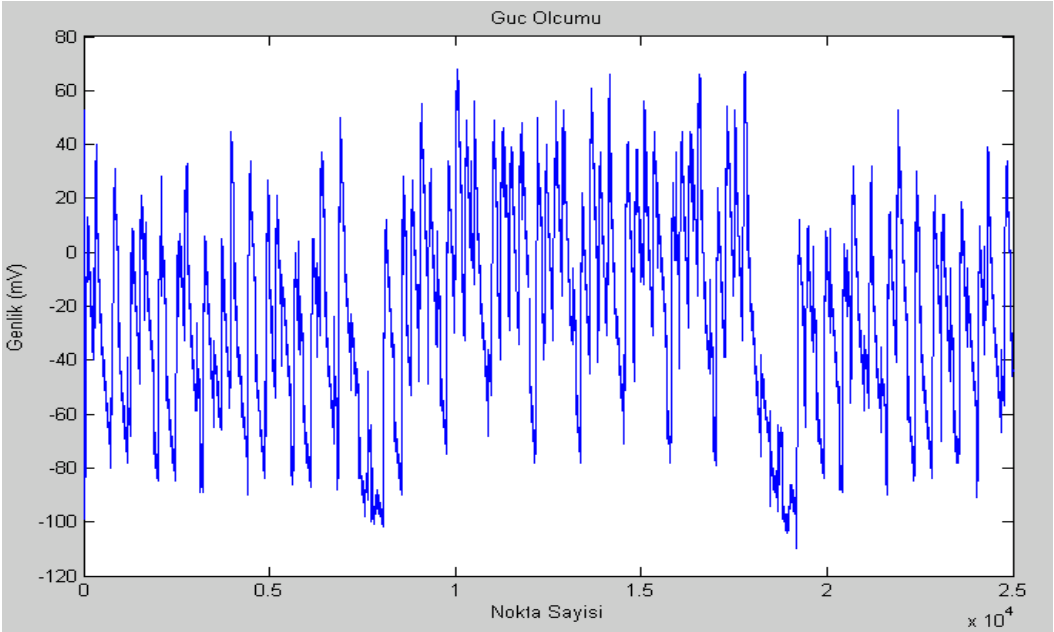


Şekil 5.3 Ölçüm alma düzeneği

5.4 Ölçüm Alma İşlemi

Bölüm 5.3'te anlatılan düzeneğin kurulmasından sonra ölçüm alma işlemine geçilmiştir. Analiz çalışmaları için en fazla 800 bin ölçüm ile çalışılması öngörülmüştür. Bu nedenle 800 bin adet 128 bit rastgele veri içeren bir veri paketi hazırlanmıştır.

Bu veriler için, belirlenen bir anahtar ile şifreleme işlemleri yaptırılarak 500 Msample/sn ile 800 bin ölçüm alınmıştır. Her bir veri için yapılan şifreleme işlemi esnasında alınan güç verisi 25.000 nokta ile kaydedilmiştir. Şekil 5.4'te örnek bir güç ölçümü verilmiştir.



Şekil 5.4 Örnek bir güç ölçümü

Ölçüm alma işlemi sonunda elde edilen 800 bin ölçüm verisi analiz işleminde kullanılacaktır. Bu kadar çok miktarda ölçümün analiz edilebilmesi için her bir ölçüm verisinin boyutunun küçük olması avantaj sağlayacaktır. Bu nedenle her bir ölçüm için RMS değeri hesaplanmıştır. Bu işlemin gerçeklenmesi için her bir darbesinde örneklemede kullanılan nokta sayısının tespit edilmesi gerekmektedir. Bunun için aşağıda yer alan işlemler sırası ile yapılmıştır.

- Üzerinde çalışılan uygulamanın çalışma frekansı 2 MHz'dir. Bu durumda her bir saat darbesi 500 ns olmaktadır.
- Ölçüm alma işlemlerinde kullanılan örnekleme oranı 500 Msample/sn' dir.

1 sn' de → $500x10^6$ örnek alınırsa

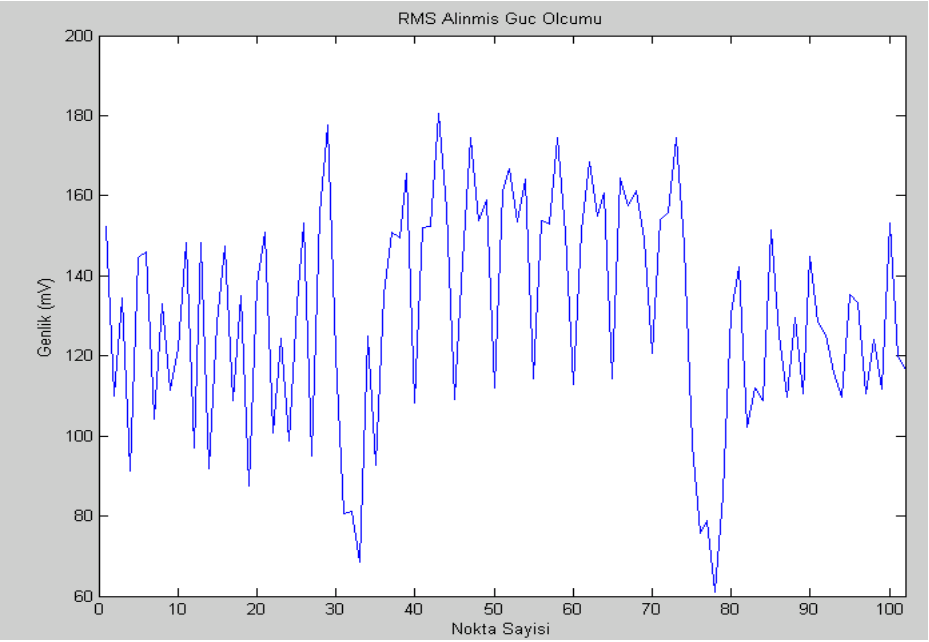
500 ns'de → k örnek alınır

orantısı kurulur.

$$k = \frac{500\ x10^{6}\ x500\ x10^{-9}}{1} = 250$$ değeri bulunur.

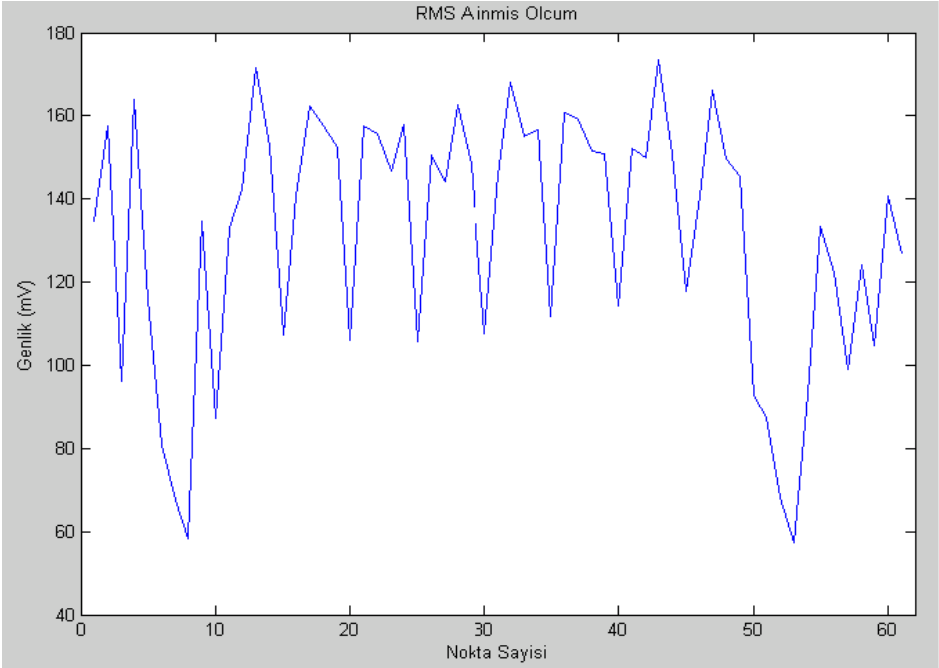
Elde edilen sonuç her bir saat adımının 250 nokta ile örneklendiğini göstermiştir.

- Matematiksel sonuç elde edilmesine karşın Şekil 5.4'te örneği görülen güç ölçümü göz ile analiz edilmiştir. Bunun için gözle saat darbelerinin gerçekleştiği noktalar tespit edilmiştir. Bu şekilde tespit edilen 10 adet saat darbesi arasındaki nokta sayısı sayılmıştır. Sayılan noktalar 10 sayısına (sayılan saat darbesi) bölünmüştür. İşlem sonunda 245 değeri elde edilmiştir.
- Göz ile yapılan analiz dikkate alınarak 245 noktada bir örnek alarak ölçüm verilerinin RMS değeri hesaplanmıştır. İşlem sonunda Şekil 5.5'te örneği görülen ve 102 nokta içeren ölçüm verileri elde edilmiştir.



Şekil 5.5 RMS alınmış bir güç ölçümü

Daha sonra ölçüm verisinin her bir analizde kullanılacak kısmı tespit edilmiştir. Tespit sonunda analizde kullanılmayacağına karar verilen kısımlar ölçümden atılmıştır. İşlem sonunda Şekil 5.6'da bir örneği görülen güç ölçüm verileri analiz işlemi için hazır hale gelmiştir.



Şekil 5.6 Ön işlem süreçlerinden geçmiş bir güç ölçümü örneği

BÖLÜM 6

FARKSAL GÜÇ ANALİZİ SALDIRILARININ GERÇEKLENMESİ

Bu bölümde tez çalışması kapsamında *IAIK Maskeli AES Algoritması* üzerine yapılan saldırılar ve elde edilen sonuçlar anlatılacaktır. Daha öncede belirtildiği gibi, tez çalışmasının amacı, *IAIK Maskeli AES Algoritması* üzerine yapılan saldırıların etkinliğini analiz etmektir. Bu nedenle; söz konusu saldırı yöntemlerinin tespit edilmesi, tespit edilen yöntemlerin uygulanması ve sonuçlarının elde edilmesi tez çalışmasının en önemli adımlarından birini oluşturmaktadır.

Saldırı çalışmasına geçilmeden önce, öncelikle üzerinde çalışılan *IAIK Maskeli AES Algoritması* için gerçeklenebilecek saldırıların tespit edilebilmesi için literatür tarama çalışması yapılmıştır. Yapılan araştırma sonunda farklı durum ve uygulamalar için geçerli saldırıların 5 başlık altında toplanabileceği görülmüştür.

- 1. Derece FGA Saldırısı
- 2. Derece FGA Saldırısı
- Geçiş Sayısı (Toggle Count) Saldırısı
- Sıfır Giriş (Zero Input) FGA Saldırısı
- Sıfır Ofset (Zero Ofset) FGA Saldırısı

Yukarıda listelenen her bir saldırı için; saldırının tanımı, önceki çalışmalar, saldırının gerçeklenmesi ve elde edilen sonuçlar ilgili saldırı başlıkları altında anlatılacaktır.

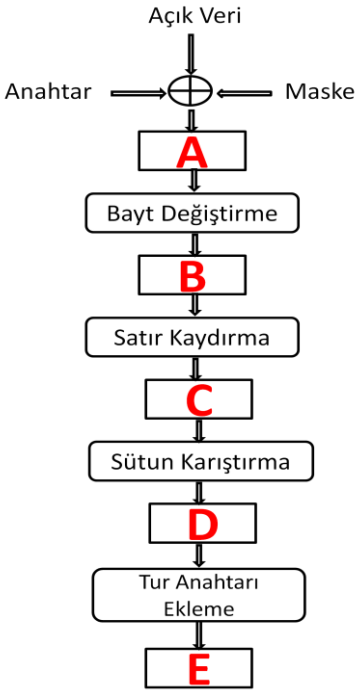
6.1 1.Derece Farksal Güç Analizi Saldırısı

6.1.1 Saldırının Tanımı

Kocher ve arkadaşları tarafından önerilen yöntemdir [3]. Saldırının gerçeklenme adımları Bölüm 3.1.3'te ayrıntılı olarak anlatılmıştır. 1. Derece Farksal Güç Analizi Saldırısı tek başına bir saldırı yöntemi olduğu gibi, aynı zamanda diğer saldırı teknikleri için de bir saldırı adımı olarak kullanılmaktadır.

6.1.2 Saldırının Gerçeklenmesi

1. Saldırının gerçeklenmesinden önce üzerinde çalışılan sistemin analiz edilmesi ve saldırının gerçeklenebileceği noktaların tespit edilmesi gerekmektedir.
   Analiz işleminde, algoritma tarafından üretilen ara değerler, bunların yazıldığı yazmaçlar tespit edilmiştir.



Şekil 6.1 Analiz edilen maskeli AES uygulaması tur adımları ve yazmaçlar

Şekil 6.1'de görülen algoritma akış diyagramına göre her bir tur adımı sonucu A, B, C, D ve E yazmaçlarına aşağıda verilen değerlerin yazıldığı tespit edilmiştir.

Başlangıç Anahtarı Ekleme Sonrası Yazmaca Yazılan Veri

$A = IK_O = P \oplus K \oplus M$[1]

Bayt Yer Değiştirme Sonrası Yazmaca Yazılan Veri

$B = S_O = S(P \oplus K \oplus M)$[2]

Satırları Kaydırma Sonrası Yazmaca Yazılan Veri

$C = SR_O = SR(S(P \oplus K \oplus M))$[3]

Sütunları Karıştırma Sonrası Yazmaca Yazılan Veri

$D = MC_O = MC(SR(S(P \oplus K \oplus M)))$[4]

Tur Sonucu Yazmaca Yazılan Veri

$E = TS_O = MC(SR(S(P \oplus K \oplus M))) \oplus K_R \oplus M \oplus M_R = MC_P \oplus K_R \oplus M$[5]

2. Bölüm 3.1.3'te ayrıntılı anlatıldığı gibi *Güç Analiz Matrisi* ve A, B, C, D ve E noktaları için *Tahmin Matrisi* oluşturularak *Kocher Analizi* gerçeklenmiştir.

6.1.3 Saldırının Sonucu

Yukarıda örnekleri verilen yazmaçlara dikkat edildiğinde hepsinin üzerine yazılan verinin içerisinde de "maske" verisinin olduğu görülmektedir.

Oysa 1.Derece FGA gerçeklemek için model oluşturulması aşamasında üretilen ara değerler, giriş ve anahtar verilerini dikkate alarak üretilmektedir. "maske" önlemi alınmış olan uygulamamızda ise yazmaçlara yazılan değerler "maske" verisinden etkilenmiş değerler olacaktır.

---

[1] **P:** Açık Giriş Verisi, **K:** Başlangıç Anahtarı, **M:** Başlangıç Maskesi, **IK_O:** Başlangıç Anahtarı Ekleme İşlemi Sonrası Oluşan Veri

[2] **S:** Bayt Değiştirme İşlemi, **S_O:** Bayt Değiştirme İşlemi Sonrası Oluşan Veri

[3] **SR:** Satırları Kaydırma İşlemi, **SR_O:** Satırları Kaydırma İşlemi Sonrası Oluşan Veri,

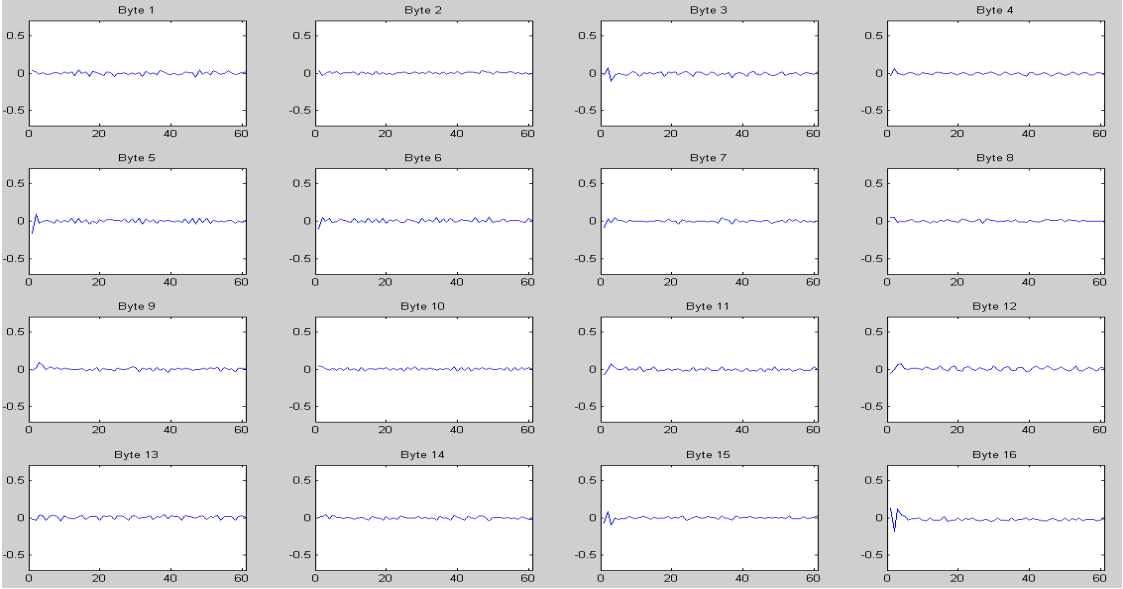
[4] **MC:** Sütunları Karıştırma İşlemi, **MC_O:** Sütunları Karıştırma İşlemi Sonrası Oluşan Veri,

[5] **TS_O:** Tur Anahtarı Ekleme Sonrası Oluşan Veri, **MC_P:** Sütunları Karıştırma İşlemi Sonunda Oluşan Verinin Maske Etkisi Çıkarılmış Hali **M_R:** Tur Sonu Maskesi, **K_R:** Tur Anahtarı

Dolayısıyla eğer maskeleme önlemi düzgün bir şekilde uygulanmış ise 1.Derece FGA Saldırısı sonunda bir sonuç elde edilmemiş olunması gerekmektedir.

Saldırıların hiçbirinde başarılı bir sonuç elde edilememiştir. Bu durum [19] çalışması kapsamında yapılan ve üzerinde çalışılan uygulamanın başarılı olduğunu göstermektedir.

Şekil 6.2'de *Başlangıç Anahtarı Ekleme Sonrası* için elde edilen sonuçlar görülmektedir.



Şekil 6.2 Başlangıç anahtar ekleme için yapılan 1.derece FGA saldırısı sonucu

6.2 2.Derece Farksal Güç Analizi Saldırıları

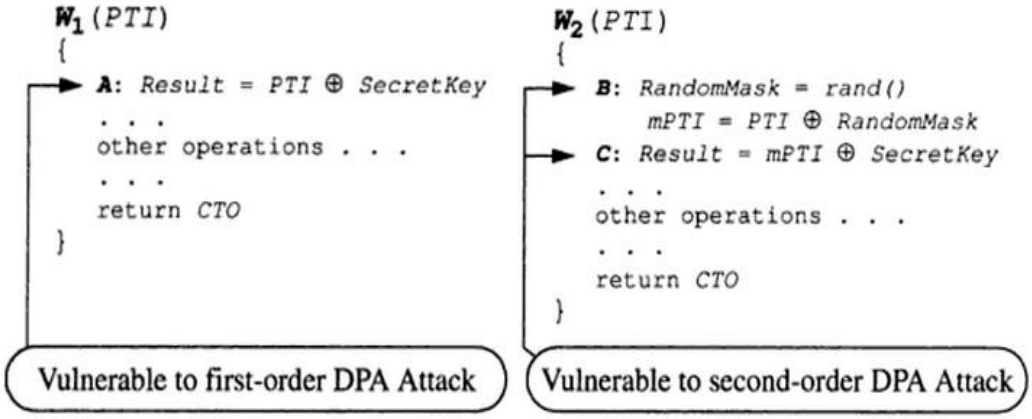
6.2.1 Saldırının Tanımı

Güç Analizi Saldırıları'nın ortaya çıkmasından sonra öncelikle 1.Derece FGA Saldırıları üzerinde yoğunlaşılmıştır. 2.Derece FGA'dan ilk defa Kocher tarafından söz edilmiştir [10]. Ancak bu saldırı tekniğine gereksinimin belirlenmesi, teorik olarak tanımlanması ve pratik olarak gerçeklenmesi Messerges tarafından yapılmıştır [36]. Maske önleminin 1. Derece Farksal Saldırısı'na karşı bir önlem olarak önerilmesinden sonra 2. Derece FGA Saldırısı bu güvenlik önlemini aşmaya yönelik olarak kullanılmıştır [11].

Ek olarak 2. Derece FGA kapsamında "maskeli veri" ve "maske" değerlerinin aynı anda işlendiği özel durumlar için saldırılar geliştirilmiştir [12], [37]. Bu çalışmalar kapsamında önerilen saldırılar tez çalışması kapsamında da gerçeklenmiştir.

2.Derece FGA Saldırısı'nı gerçekleştirmeden önce bu saldırının neden önemli olduğu konusunda bilgi vermek gerekir.

Yukarıda da belirtildiği gibi 2.Derece FGA Saldırısı'nın gerekliliği ilk olarak [36] çalışması ile öngörülmüştür.



Şekil 6.3 Farklı uygulamalar için uygulanabilecek 1. ve 2. derece FGA saldırıları

Şekil 6.3'te sözde kodlarla ifade edilen iki farklı algoritma uygulama parçası görülmektedir.

Sol tarafta yer alan sistemde *A* noktasında, algoritma adımı sonucu ve yazmaca yazılan veri sadece açık veri (PTI) ve anahtar değerlerini içermektedir. Eğer bu çıkış noktasına 1.Derece FGA Saldırısı gerçeklenirse başarılı sonuç alınması beklenmektedir.

Sağ tarafta yer alan sistemdeki *C* noktasında ise, gösterilen algoritma adımı sonucu yazmaca yazılan veri, "açık veri (PTI)" ve "anahtar" dışında bir de saldırganın bilemediği "maske (random mask)" değerine bağlıdır.

Bu durumda 1.Derece FGA Saldırısı gerçekleyen saldırganın "açık veri" ve "anahtar" değerine bağlı olarak oluşturduğu model işe yaramayacak ve saldırı başarısız olacaktır.

2.Derece FGA ile sağ tarafta görülen yapıdaki bir sisteme karşı nasıl başarılı bir saldırı gerçekleneceği aşağıda anlatılacaktır.

6.2.2 Saldırının Gerçeklenmesi

Bu bölümde adımlar halinde 2.Derece FGA Saldırısı'nın gerçeklenmesi anlatılacaktır. Aynı zamanda tez çalışması kapsamında yapılan saldırı çalışması adımları da anlatım içine eklenecektir.

1. Algoritma uygulaması içerisinde güvenliği sağlayan "maske" verisinin kullanıldığı iki farklı noktanın tespit edilmesi gerekmektedir. Bu kapsamda, üzerinde çalışılan uygulama VHDL kodu üzerinden analiz edilmiştir.

   Şekil 6.1'de verilen A, B, C, D ve E noktaları her bir algoritma adımı sonrası gerçekleşen yazma işlemlerini göstermektedir.

   "A" olarak verilen İlk Anahtar Ekleme adımı sonrasında yazılan veri:

   $A = IK_O = P \oplus K \oplus M$[1]

   "E" noktası olarak verilen 1. Tur Sonunda yazılan veri:

   $E = TS_1 = MC\,(SR\,(S\,(P \oplus K \oplus M\,))) \oplus K_1 \oplus M \oplus M_R = MC_P \oplus K_1 \oplus M$[2]

   Bu veri içerisinde *Sütunları Karıştırma* işlemi sonuna kadar taşınmış olan maske verisi M_R değeri, MC (SR(S(P ⊕ K ⊕ M))) içerideki maske etkisini götürmektedir. Sonuç olarak,

   A noktası için: $IK_O = P \oplus K \oplus M$

   E noktası için: $TS_1 = MC_P \oplus K_1 \oplus M$

   değerleri elde edilir. Görüldüğü gibi "M" maske değeri ortak olarak kullanılmaktadır.

2. Analiz işleminde kullanılacak *n* adet ölçüm için *k* adet ölçüm verilerini içeren M_1 *Güç Analiz Matrisi* Bölüm 3.1.3'te belirtildiği şekilde oluşturulmuştur.

   Bu matriste yer alan her bir güç ölçümü için, ölçüm üzerinde yer alan her bir nokta ölçüm üzerindeki tüm diğer noktalardan çıkarılmıştır. Bu sayede saldırıya karşı

---

[1] **P:** Açık Giriş Verisi, **K:** Başlangıç Anahtarı, **M:** Başlangıç Maskesi, **IK_O**: Başlangıç Anahtarı Ekleme İşlemi Sonrası Oluşan Veri

[2] **TS_1:** 1.Tur Anahtarı Ekleme Sonrası Oluşan Veri, **MC_P** Sütunları Karıştırma İşlemi Sonunda Oluşan Verinin Maske Etkisi Çıkarılmış Hali **M_R:** Tur Sonu Maskesi, **K_1:** 1.Tur Anahtarı

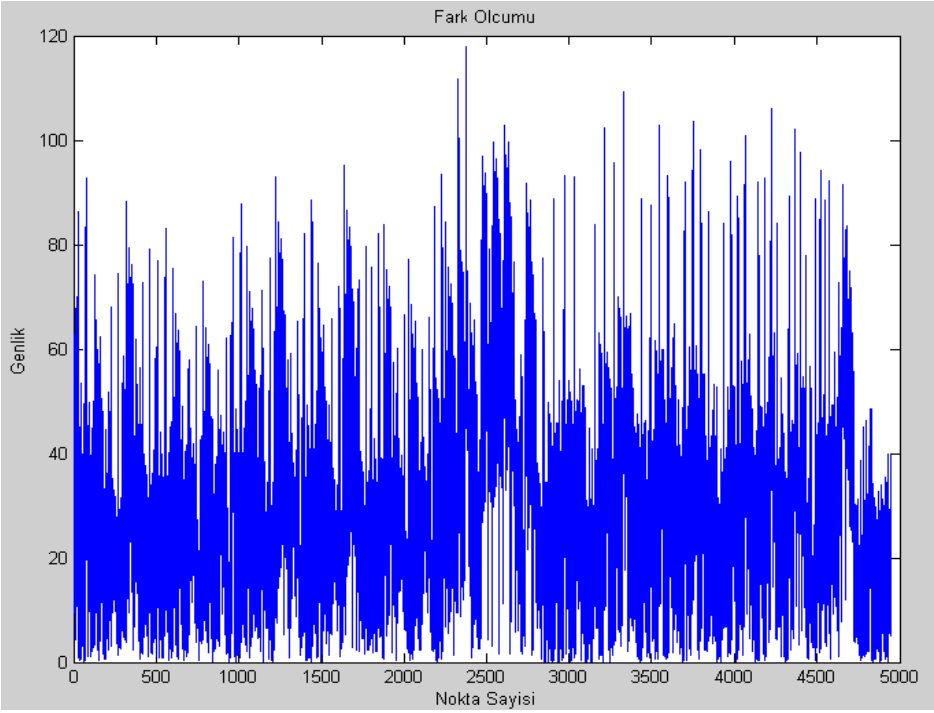
güvenliği sağlayan ve algoritmanın farklı noktalarında kullanılan “Maske” verisinin kullanıldığı iki noktayı birbirinden çıkararak bu noktalarda maske verisinin toplam güç tüketimine etkisinin sıfırlanmasının sağlanması hedeflenmektedir.

Örneğin Şekil 5.5’te analiz işlemlerinde kullanılmak üzere RMS hesaplanmış güç ölçümü görülmektedir. Ölçüm üzerinde 102 nokta vardır.

Bu ölçüm üzerinde 2-102 numaralı noktalar arası tüm ölçüm değerleri 1 noktasından çıkarılıştır. Daha sonra 3-102 numara arası tüm noktalar 2 noktasından çıkarılarak bir önceki işlem sonunda elde edilen sonucun ardına eklenmiştir. Bu şekilde tüm noktalar birbirinden çıkarılmıştır.

Sonuç olarak *k*=102nokta içeren bir ölçüm için $\frac{k(k-1)}{2}$ = *p*=4950uzunluğunda yeni bir ölçüm elde edilmiştir. Daha sonra bu ölçümde yer alan noktaların her birinin mutlak değeri alınmıştır.

Bu işleme *Fark Alma İşlemi* denir. Şekil 6.4’te *Fark Alma İşlemi* sonunda elde edilen *p* uzunluğunda Fark *Ölçümü* görülmektedir.



Şekil 6.4 Fark ölçümü

Analiz işleminde kullanılacak olan *n* adet ölçüm verisi için aynı işlem uygulanmıştır.

Sonuç olarak, (*n* X *p*) boyutunda M_2*Fark Güç Analiz Matrisi* elde edilmiştir.

3. Bu noktanın anlaşılabilmesi için öncelikle 2.Derece FGA saldırılarının temelini oluşturan bir varsayımın anlatılması gerekmektedir.

   Varsayım:

   a ve b $\in$ $(0,1)^n$ olmak üzere, C(a) ve C(b): a ve b değeri nedeniyle gerçekleşen güç tüketimini göstersin. C (x)güç tüketiminin, yaklaşık olarak x değerinin Hammingweight değerine eşit olduğu farz edilsin. Yani C(x) $\approx$ HW (x)kabul edilsin.

   Bu takdirde;

   $HW(a \oplus b) = |H(a) - HW(b)|$ (6.1)

   olur.

   Buna bağlı olarak da $|C(a) - C(b)|$ değeri de (6.1) ile bağlantılı kabul edilebilir.

   Varsayım doğrultusunda Adım-1'de tespit edilen değerleri tekrar yazalım. Şekil 6.1'e göre

   A noktası için: $IK = P \oplus K \oplus M$ ifadesi *Varsayım*'da *a* noktasına

   E noktası için: $TSO_1 = MC_P \oplus K_1 \oplus M$ ifadesi *Varsayım*'da *b* noktasına karşılık gelmektedir.

   Eğer IK ve TS_{O1} verileri xor işlemine tabi tutulursa *Varsayım*'a göre iki değerin farkı alınmış olur. Olası tüm K ve K_1 değerleri için bu işlem gerçeklendiğinde elde edilen sonuç *Tahmin Matrisi* verisidir.

   $$\textit{Tahmin Matrisi} = IK \oplus TS_{O1} = \{P \oplus K \oplus M\} \oplus \{MC_P \oplus K_1 \oplus M\}$$
   $$= P \oplus K \oplus MC_P \oplus K_1$$

   olarak elde edilir.

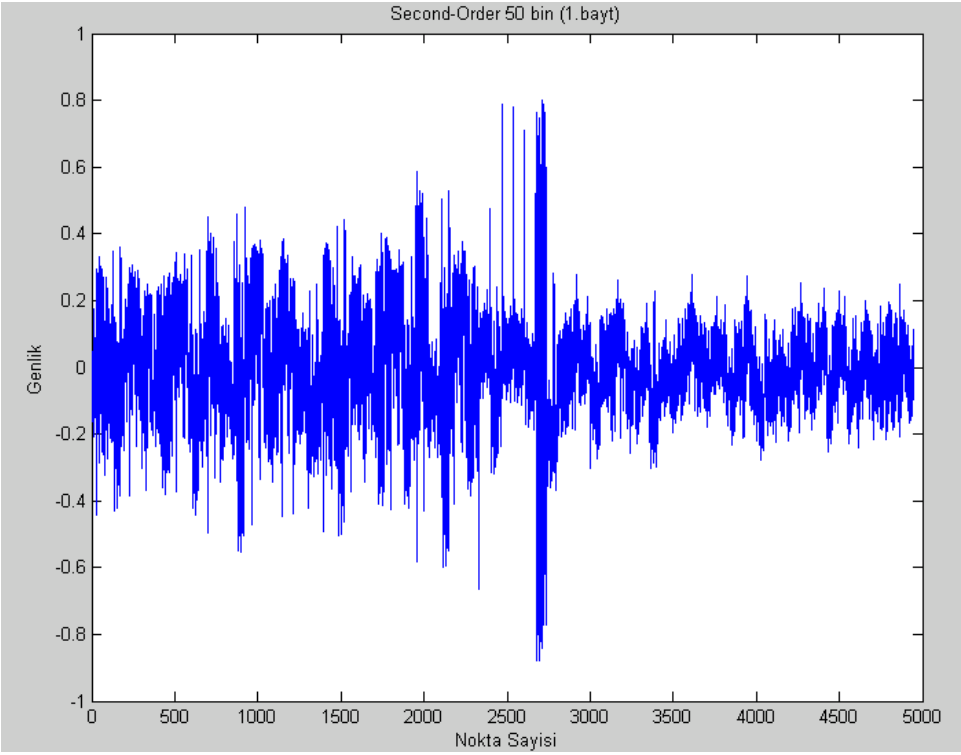
4. 2.Adım'da elde edilen M_2*Fark Güç Analiz Matrisi* ile 3. Adım'da elde edilen *Tahmin Matrisi* kullanılarak Bölüm 3.1.3'te anlatıldığı şekilde *Kocher Analizi* gerçeklenmiştir.

6.2.3 Saldırının Sonucu

Saldırı sonunda 1. derece FGA Saldırısı'nın aksine teorik olarak başarı elde edilmesi beklenmektedir. Analiz sonucunda da saldırı başarı ile gerçeklenmiştir.

İlk Açıklık

Analiz sonunda ilk açıklık 50 bin ölçümde ortaya çıkmıştır. Şekil 6.5'te 1.baytta ortaya çıkan ilk açıklık görülmektedir.



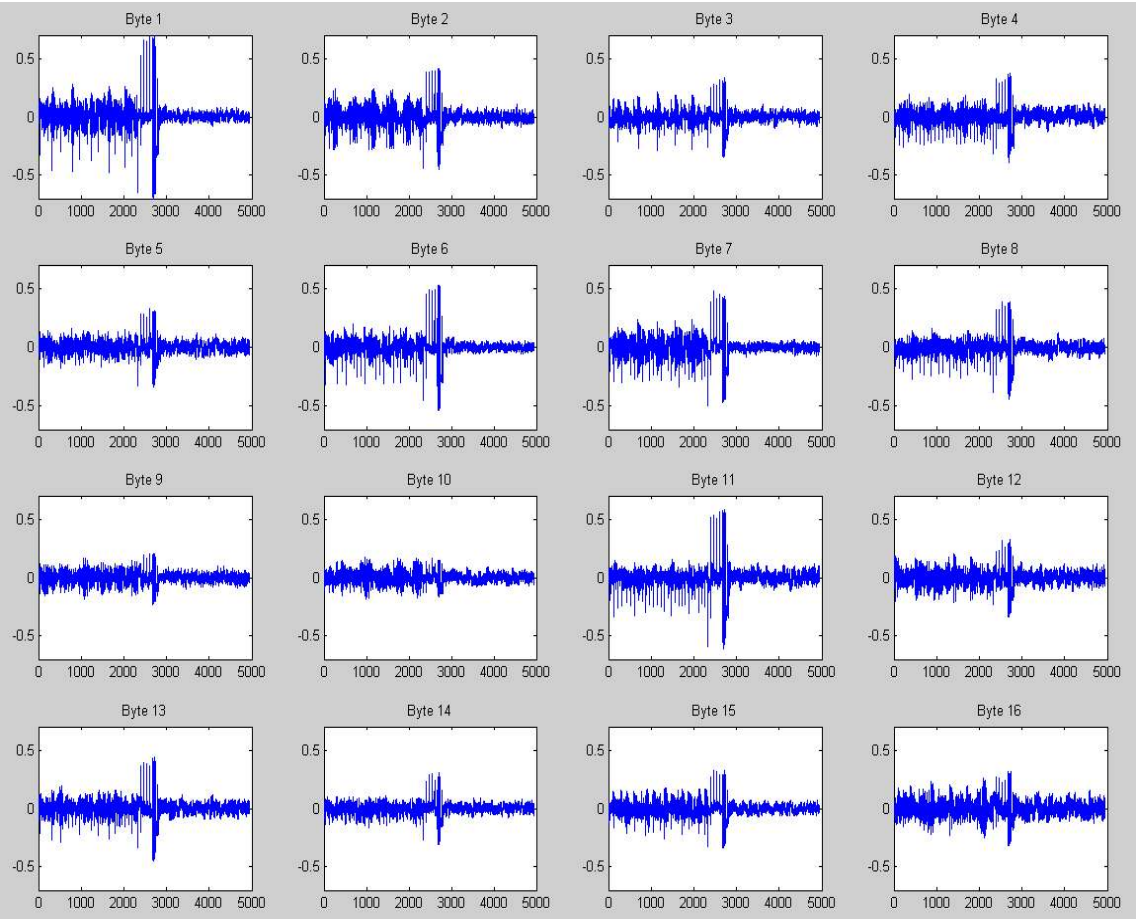
Şekil 6.5 2. derece FGA saldırısı ilk açıklık

50 bin ölçümde açıklığın oluştuğu tek bayt Şekil 6.5'te görülen bayt değildir. 5 bayt için daha açıklık belirtileri ortaya çıkmıştır.

Anahtarın Elde Edilmesi

Analiz sonunda 100 bin ölçümde 16 anahtar baytından 9 tanesi için açıklık ortaya çıkmaktadır. 300 bin ölçüm ile analiz yapıldığında ise 15 bayt için açıklık oluşmakta. **Hata! Başvuru kaynağı bulunamadı.**'da 300 bin ölçüm için analiz sonuçları görülmektedir. Ancak analiz sürecinde, *Tahmin Matrisi* oluşturulurken, ana anahtar ve buna karşılık ana anahtar üzerinden üretilen 1.tur anahtarı "matlab" ortamında üretilerek doğrudan yerine yazılmıştır. Bu sayede saldırı sonunda yukarıda belirtilen kapsamda başarı elde edilebilmiştir. Öte yandan, ana anahtar değerinin bilinmediği bir durumda bu saldırının gerçeklenebilmesi için 2^{40} adet anahtar değerinin tahmin edilmesi gerektiği belirtilmelidir. Bu değerlerden 2^{8} tanesi

ana anahtar tahmini içindir. Geriye kalan 2^{32} tanesi 1. tur anahtarı içindir. 1.tur anahtarı için ihtiyaç duyulan tahmin sayısının artmasının nedeni, sütunları karıştırma işleminin ana anahtar etkisini dağıtmasıdır. Bu durumun pratik olarak saldırının gerçekleşme ihtimalini zorlaştırdığı görülmektedir.



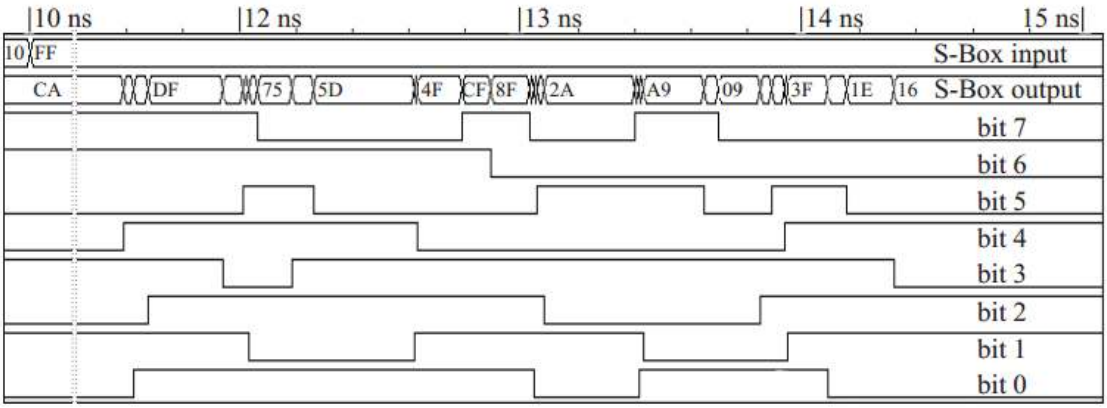
Şekil 6.6 300 bin ölçüm için elde edilen 2. Derece FGA saldırısı sonucu

6.3 Geçiş Sayısı Saldırısı

6.3.1 Saldırının Tanımı

Bölüm 4.4'te, AES Algoritmasına karşı gerçeklenen FGA Saldırıları'na karşı bir önlem olarak önerilen ve yaygın olarak kullanılan IAIK maskesinin gerçeklenebilmesi için, *Bayt Değiştirme* işlemi içerisinde yer alan G (2^8)'de ters alma işleminin kombinezonsal olarak gerçekleşmesi gerektiği anlatılmıştı. Bu durum kombinezonsal devrelerin çalışma yapısından kaynaklanan başka bir açıklığın doğmasına neden olmuştur.

Kombinezonsal devreler içerisinde yer alan kapılarda oluşan gecikmeler nedeniyle girişlerine uygulanan değerleri çıkışlarına doğrudan yansıtamazlar. Giriş verisine karşılık gelen çıkış değeri oturana kadar devre çıkışında ve ara noktalarda anlık değişimler olur. Bu anlık geçişlere "glitch" veya "toggle" ismi verilmiştir [38].



Şekil 6.7 Bayt değiştirme bloğu çıkışının girişe göre değişimi

Şekil 6.7'de, *Bayt Değiştirme* kombinezonsal devre bloğu girişinde oluşan bir değişime karşılık devre çıkışında oluşan geçişler görülmektedir. Giriş verisinin 0x10 değerinden 0xFF değerine değişmesi durumunda devre çıkışı, asıl alması gereken 0x16 değerine oturmadan önce birçok anlık geçiş yaşamaktadır.

Ayrıca, devre çıkışlarında ve ara noktalarda oluşan bu geçişlerin sayısı, devre girişlerine uygulanan veri değerine göre değişmektedir. Bu değişim sayesinde yapısı bilinen kombinezonsal devreler modellenebilmekte ve işledikleri gizli bilgiler analiz edilebilmektedir.

Kombinezonsal devrelerde oluşan gecikmelerin yan kanal bilgisi olarak kullanılabileceği ilk olarak [39] çalışmasında bahsedilmiştir. *SPICE* simülasyonu ile bu durumun gerçekliği teyit edilmiştir. Ayrıca [38] çalışmasında da kombinezonsal devrelerde oluşan glitchlerin maskeli uygulamalar üzerinde etkili olacağı analiz edilmiştir.

Saldırının ilk kez pratik olarak gerçeklenmesi ise, özel tasarlanmış bir ASIC uygulaması üzerinde başarılı olarak uygulanmıştır [13].

[14] çalışmasında oluşan glitchlerin kaynaklanma noktası üzerinde analiz yapılmıştır.

6.3.2 Saldırının Gerçeklenmesi

Tez çalışması kapsamında üzerinde çalışılan *IAIK Maskeli AES Algoritması* VHDL kodu, analiz edilmiştir. Analiz sonunda Geçiş Sayısı Saldırısı'nın gerçeklenebileceği değerlendirilmiştir. Saldırının gerçekleşmesinde aşağıda sıralanan adımlar takip edilmiştir.

1. Uygulamada VHDL kodu içerinde *Bayt Değiştirme* bloğunun girdi ve çıktıları analiz edilmiştir.

Şekil 6.8 Bayt değiştirme bloğu giriş ve çıkışı

2. Kombinezonsal devre, 256 giriş ve her bir giriş için 256 farklı maske değeri olmak üzere toplam 256 X 256=65536 durum için simüle edilmiştir.

   Simülasyon işlemi, *MODELSIM* programı ile devre içerisinde oluşan gecikmeleri ve buna bağlı anlık değişimleri tespit edebilmesi için *POST-MAP* olarak gerçeklenmiştir. İdeal durumlar için tanımlı olan ve yaygın olan *Behavioral Simülasyon* işlemi, saldırıda kullanılan parametreleri dikkate almadığından kullanıma uygun değildir.

   Simülasyonu gerçekleştirmek için *XILINX ISE* ortamında; olası tüm giriş değerlerini bir dosyadan okuyan, her bir giriş anında çıkışta oluşan tüm anlık çıkışları tespit ederek başka bir dosyaya kaydeden bir testbench aracı geliştirilmiştir.

   Simülasyon işlemi sonunda 256X256 boyutunda M_1 matrisi elde edilmiştir. Matrisin elemanları, her bir satır başında tanımlanan olası tüm "maskeli veri" girişlerine ve her bir sütun başında tanımlanan olası tüm "maske" girişlerine karşılık düşen geçiş sayılarıdır.

M_1 (256 X 256)

$$\begin{array}{c} \\ d_0 \\ d_1 \\ . \\ . \\ d_{254} \\ d_{255} \end{array} \begin{array}{c} \begin{array}{cccccc} m_0 & . & . & m_1 & . & . & . & . & m_{255} \end{array} \\ \begin{bmatrix} Td_0 m_0 & & Td_0 m_1 & . & . & . & Td_0 m_{255} \\ Td_1 m_0 & & . & . & . & . & Td_1 m_{255} \\ . & & . & . & . & . & . \\ . & & . & . & . & . & . \\ . & & . & . & . & . & . \\ Td_{255} m_0 & & . & . & . & . & Td_{255} m_{255} \end{bmatrix} \end{array}$$

Şekil 6.9 Olası tüm "maskeli veri-maske" girişleri için geçiş sayısı matrisi

3. Her bir "maskeli veri" değeri ile beraber giren 256 "maske" değeri için oluşan geçişlerin ortalaması alınır. İşlem sonunda 256X1 boyutunda M_2 matrisi elde edilmiştir.

M_2(*256* X1)

$$\begin{array}{c} \\ d_0 \\ d_1 \\ . \\ . \\ d_{254} \\ d_{255} \end{array} \begin{bmatrix} Td_0 m_{ort} \\ Td_1 m_{ort} \\ . \\ . \\ . \\ Td_{255} m_{ort} \end{bmatrix}$$

Şekil 6.10 Olası tüm maskeli veri girişleri için ortalama geçiş sayısı matrisi

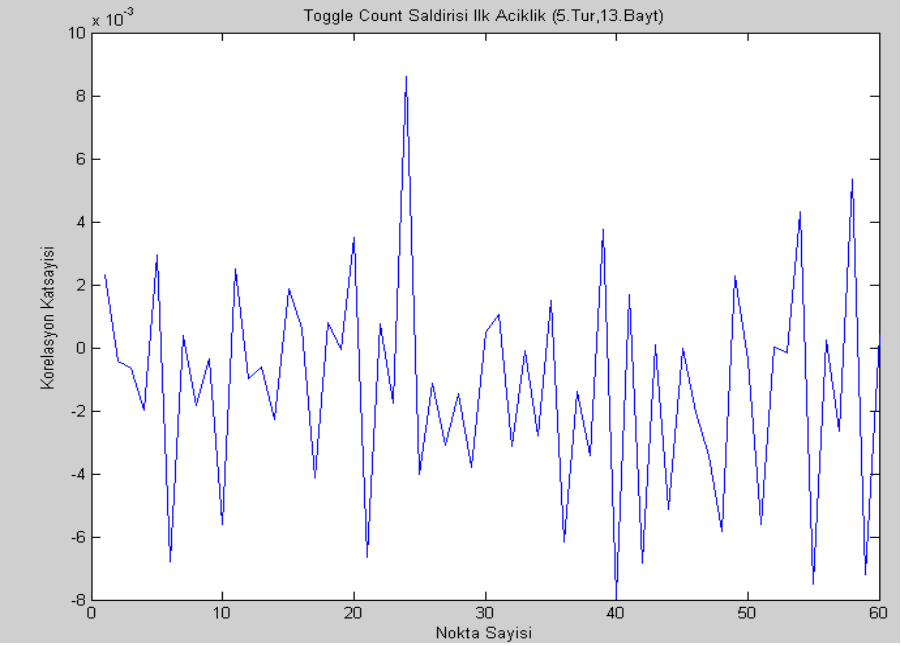
4. Geçiş Sayısı Saldırısı'nın gerçekleneceği *Bayt Değiştirme* işlemi girişi için Bölüm 3.1.3' te anlatıldığı şekilde *Tahmin Matrisi* oluşturulmuştur.
5. Tahmin matrisinde yer alan her bir elamanın yerine M_2 matrisinde o değere karşılık simülasyon sonucunda elde edilen geçiş sayısı konulmuştur. Elde edilen *n* X 256boyutundaki M_4 matrisine, Geçiş Sayısı Saldırısı için özel olmasından dolayı *Geçiş Sayısı Tahmin Matrisi* ismi verilmiştir.

   Geçiş Sayısı Tahmin Matrisi ile ölçüm alma işleminde elde edilen *Güç Analiz Matrisi* arasındaki korelasyona bakılmıştır.

6.3.3 Saldırının Sonucu

İlk Açıklık

Analiz sonunda ilk açıklık 50 bin ölçümde ortaya çıkmıştır. Şekil 6.11'de 1.baytta ortaya çıkan ilk açıklık görülmektedir.



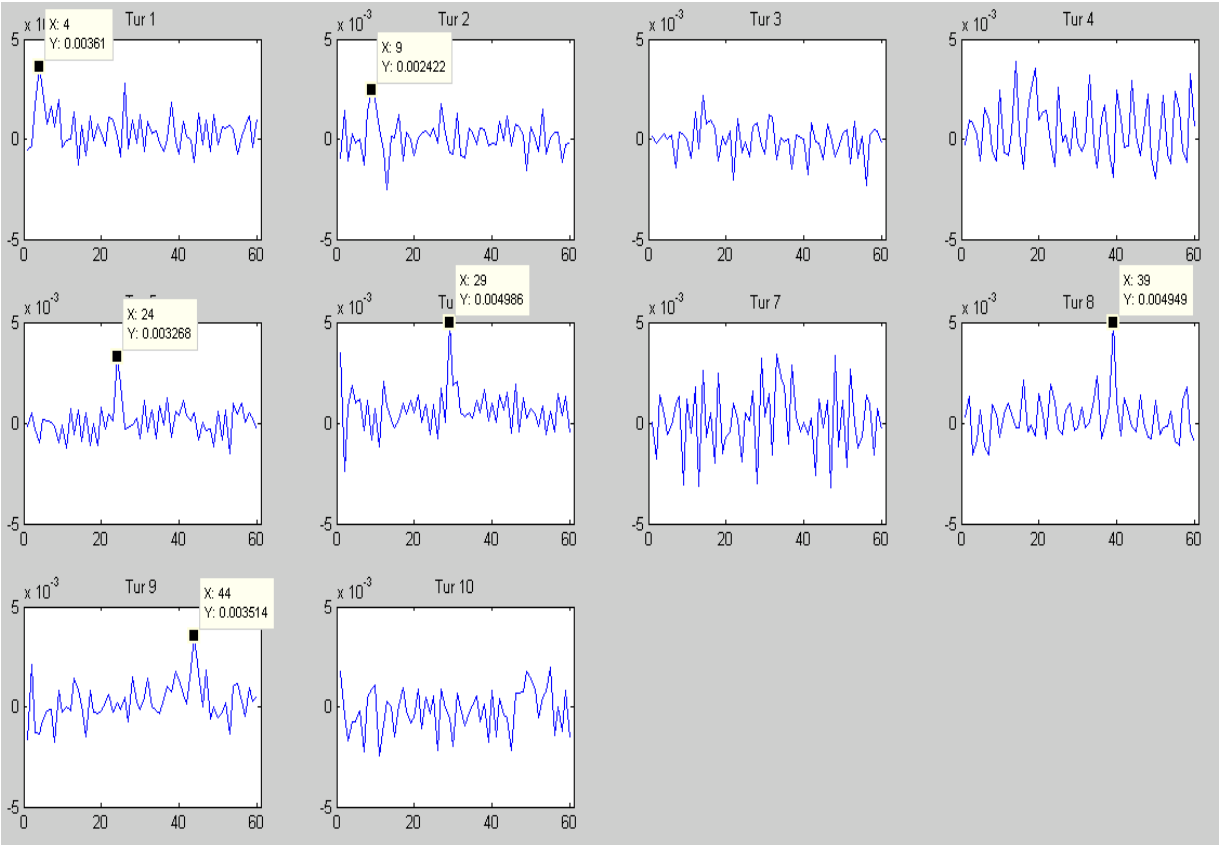
Şekil 6.11 Geçiş sayısı saldırısı ilk açıklık

Anahtarın Elde Edilmesi

800 bin ölçüm kullanılarak analiz yapıldığında ölçümün başarısının arttığı görülmektedir. Şekil 6.12'de 7.bayt için tüm turlara beraber bakıldığında elde edilen analiz sonucu görülmektedir. Ancak anahtarın elde edilmesi için sadece ilk ve son turlarda alınan sonuçlar kullanılabilecektir. Bu nedenle ilk ve son turlarda alınan sonuçlara yoğunlaşılmıştır. Çizelge 6.1'de bu turlarda açıklık bulunan baytlar "**+**" ile gösterilmiştir.

Çizelge 6.1 Geçiş sayısı saldırısı ile ilk ve son turda açıklık bulunan baytlar

Bayt No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
İlk Tur	+					+	+	+			+				+	
Son Tur	+	+						+				+	+		+	

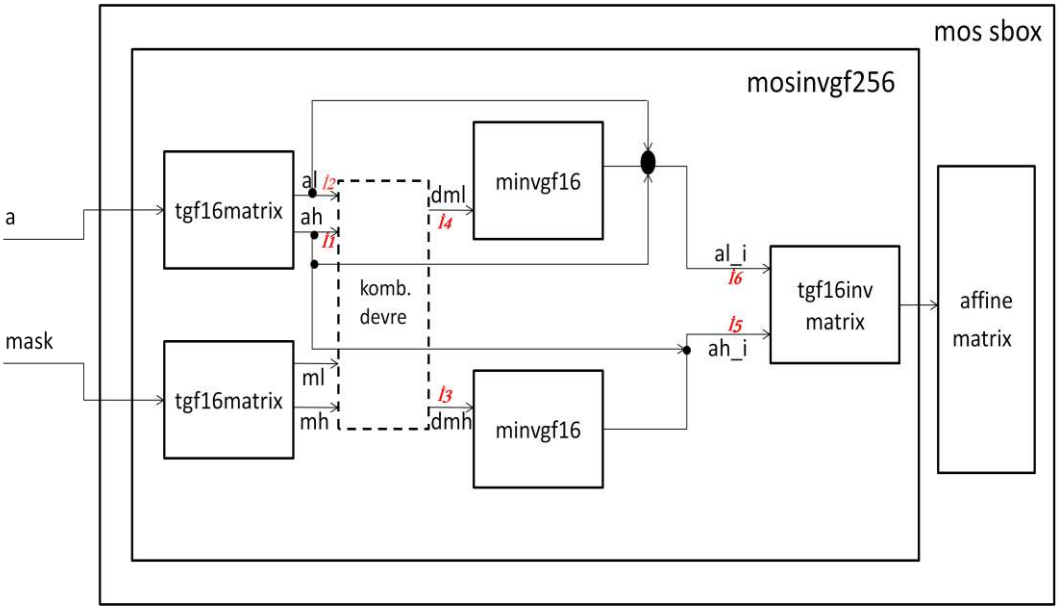


Şekil 6.12 Bayt değiştirme işlemine yapılan geçiş sayısı saldırısı sonucu

6.3.4 Geçiş Sayısı Saldırısı Ek Analiz Çalışmaları

Tez çalışması kapsamında üzerinde çalışılan *IAIK Maskeli AES Algoritması* için; *Bayt Değiştirme* girişlerine verilen farklı "maskeli veri" ve "maske" değerlerine karşılık sistem çıkışında oluşan geçişler sayılmış ve bu değer kullanılarak sisteme başarılı bir saldırı gerçeklenmiştir. Yapılan çalışma Bölüm 0'den itibaren anlatılmıştır. Bu bölümde *Geçiş Sayısı Saldırısı* kapsamında *Bayt Değiştirme* içerisinde yer alan sinyaller üzerinden yapılan ek analiz çalışmaları anlatılmıştır [13].

Bu kapsamda öncelikle tez çalışması kapsamında üzerinde çalışılan *IAIK Maskeli AES Algoritması* VHDL kodu analiz edilmiştir. Analiz sonunda üzerinde çalışılan uygulanan *Bayt Değiştirme* içeriğinin Şekil 6.13'te görüldüğü gibi olduğu tespit edilmiştir. Çizilen yapıda analiz yapılabilecek sinyaller gösterilmiştir. Sinyal ve blok isimleri gerçek tasarımda kullanıldığı gibi şekle yansıtılmıştır.



Şekil 6.13 IAIK maskeli AES algoritması bayt değiştirme bloğu içeriği

1. Kombinezonsal devre içerisinde yer alan ve içerisinde "maskeli veri" etkisi bulunan ve dinlenebilen (blok işlemler arasında gerçeklenmeyen) sinyaller dinlenmeye çalışılmıştır. Bu sinyaller Şekil 6.13'te $İ_1$..... $İ_6$olarak gösterilmiştir. *Bayt Değiştirme* girişlerine uygulanan olası tüm "maskeli veri" ve "maske" değerlerine karşılık sinyallerde oluşan geçiş durumları aşağıda verilmiştir.

- $\dot{I}_1$ ve $\dot{I}_2$: Sinyalin hemen oturduğu ve geçiş olmadığı görülmüştür. Bunun yapılan işlemin çok kısa bir işlem olmasından kaynaklandığı düşünülmektedir.
- $\dot{I}_3$ ve $\dot{I}_4$: Sinyal üzerinde az sayıda geçiş tespit edilmiştir. Bunun nedeninin girişten sinyale kadar yer alan işlemlerin sayısının bir önceki sinyale göre çok, çıkışa göre ise az olmasından kaynaklandığı düşünülmektedir.
- $\dot{I}_5$ ve $\dot{I}_6$ **:** Sinyal üzerinde geçişler tespit edilmiştir. Bu geçişlerin sayısı $\dot{I}_3$ ve $\dot{I}_4$ sinyalleri üzerinde oluşan geçişlerden daha fazla olmasına karşın çıkışa göre halen daha azdır. Geçiş sayısının, sinyale gelen verilerin işlenme süreçleri ile bağlantılı olduğu düşünülmektedir.

Geçiş sayılarının tespit edilmesinden sonra tüm sinyal geçişleri toplanmıştır. Bu veri kullanılarak Bölüm 6.3.2'te anlatıldığı şekilde *Geçiş Sayısı Saldırısı* gerçeklenmiştir. Saldırı sonunda başarı elde edilmiştir. Analiz sonunda ilk açıklığın bulunması ve anahtar değerinin bulunması kapsamında elde edilen başarının yukarıda anlatılan saldırı (doğrudan çıkıştaki geçişlerin sayılması) başarısı ile ortalama olarak aynı düzeyde olduğu görülmüştür.

2. *Bayt Değiştirme* içerisinde yer alan sinyallerdeki geçiş sayıları kullanılarak ayrı ayrı her bir nokta için saldırı gerçeklenmiştir. Bu kapsamda aşağıda verilen sonuçlar elde edilmiştir.
   - $\dot{I}_1$: Sinyal hemen oturduğu ve geçiş olmadığı için sonuç elde edilememiştir.
   - $\dot{I}_2$: Sinyal hemen oturduğu ve geçiş olmadığı için sonuç elde edilememiştir.
   - $\dot{I}_3$: Sinyal üzerinde geçişler tespit edilmiştir. Ancak bu geçiş değerleri kullanılarak 800 bin ölçümde başarılı bir saldırı gerçeklenememiştir.
   - $\dot{I}_4$: Sinyal üzerinde geçişler tespit edilmiştir. Bu değerler kullanılarak başarılı sonuçlar alınmıştır. İlk açıklık 300 bin ölçümde bulunmuştur. Saldırı başarılı olmakla beraber çıkış noktası dinlenerek yapılan saldırıdan daha kötü sonuç alınmıştır.

     $\dot{I}_3$ ve $\dot{I}_4$ sinyalleri üzerinde oluşan geçişlerin sayısı yaklaşık olarak aynıdır. İ3 sinyali dinlenerek yapılan saldırı başarılı olmadığı halde bu saldırının

başarılı olmasının nedeninin bu sinyalde veriye eklenen düşük değerli maske verisinin maskeli veri üzerinde daha etkili olmasından (maske etkisini daha fazla sıyırmasından) kaynaklandığı düşünülmüştür.

- $\dot{I}_5$: Sinyal üzerinde geçişler tespit edilmiştir. Bu değerler kullanılarak başarılı sonuçlar alınmıştır. İlk açıklık 600 bin ölçümde bulunmuştur. Saldırı başarılı olmakla beraber çıkış noktası dinlenerek yapılan saldırıdan daha kötü sonuç alınmıştır.
- $\dot{I}_6$: Sinyal üzerinde geçişler tespit edilmiştir. Bu değerler kullanılarak başarılı sonuçlar alınmıştır. İlk açıklık 50 bin ölçümde bulunmuştur. Saldırının bazı noktalarda çıkış noktası dinlenerek yapılan saldırıdan daha başarılı olduğu bazı noktalarda ise daha başarısız olduğu görülmektedir.

$\dot{I}_6$ sinyali üzerinde oluşan geçişlerin sayısının kullanılması ile yapılan saldırı $\dot{I}_5$ ile yapılan saldırıdan daha başarılıdır. Bunun nedeninin Şekil 6.13'te görüldüğü gibi; $\dot{I}_5$ sinyaline sonradan sadece maskeli verinin bir bileşeninin (ah) eklenmesi, $\dot{I}_6$ sinyaline ise her iki bileşeninin (al, ah) eklenmesi olduğu düşünülmektedir.

Sonuç

Geçiş Sayısı Saldırısı için yapılan ek analiz çalışması sonucunda başarılı saldırılar gerçeklenmiştir. Ancak bu başarılar yapılan temel saldırının başarısını aşamamıştır. "*IAIK Maskeli AES Algoritmasının FPGA Üzerinde Gerçeklemesine Yapılan Güç Analizlerinin Etkinlik Değerlendirmesi*" tez çalışması kapsamında *Geçiş Sayısı Saldırısı'*nın etkinliği ve diğer saldırılara göre başarısı üzerine yoğunlaşılmaktadır. Bu kapsamda saldırının başarısına etki etmeyen analizlere yoğunlaşılmamıştır. Ek analiz çalışması kapsamında elde edilen sonuçlara getirilen yorumlar temel öngörülere dayanmaktadır. Bu sonuçların kesin olarak doğru yorumlanması için yapılan tez çalışmasından bağımsız olarak ayrıca kapsamlı bir çalışma yapılmalıdır.

6.4 Sıfır Giriş FGA Saldırısı

6.4.1 Saldırının Tanımı

Geçiş Sayısı Saldırısı ve bununla ilgili yapılan çalışmalarda *Bayt Değiştirme* işlemi esnasında kombinezonsal devrelerde meydana gelen geçiş sayıları analiz edilmiştir. Bu analizde kombinezonsal devrelerde en az geçiş sayısının devre girişine "0x00" değeri verildiğinde oluştuğu görülmüştür. Bu ön bilgilerden yola çıkılarak Bayt *Değiştirme* işlemi girişindeki yazmaçlara "0x00" değeri yazıldığında tüketilen gücün diğer tüm durumlarda tüketilen güçten daha az olduğu sonucu çıkarılmıştır. Bu sonuç temel alınarak maskeli AES uygulamaları üzerine saldırı gerçeklenmiştir [14] .

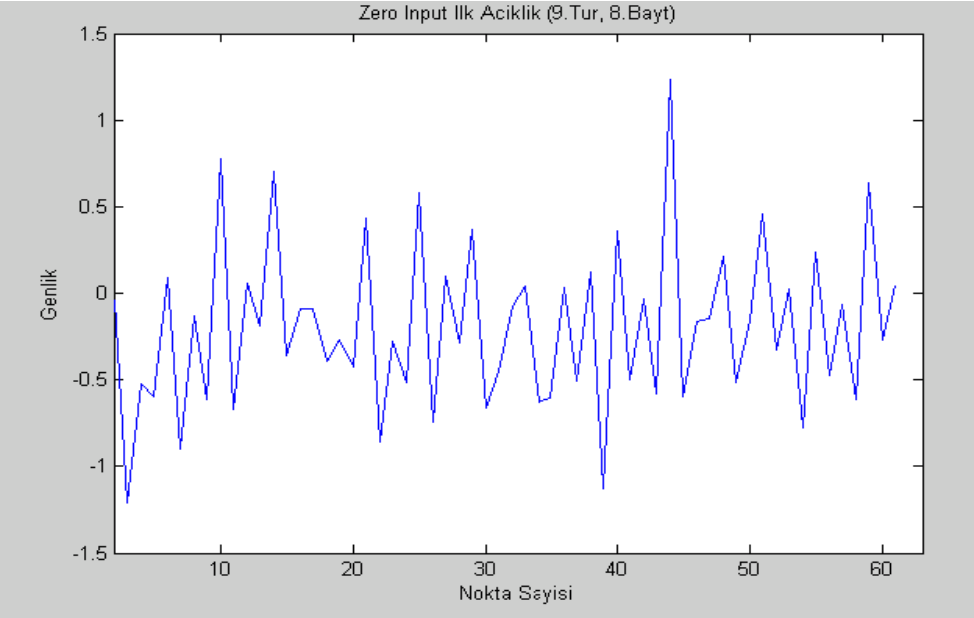
6.4.2 Saldırının Gerçeklenmesi

1. Ölçüm alma işlemi gerçeklenerek Bölüm 3.1.3'de belirtildiği şekilde M_1 Güç *Analiz Matrisi* ve *Tahmin Matrisi* elde edilmiştir.
2. Bu iki matris kullanılarak Bölüm 3.1.3'de belirtildiği şekilde *Kocher Analizi* gerçeklenmiştir. Ancak *Tahmin Matrisi*'ne bağlı olarak ölçümlerin ayrılması aşamasında; içeriği tamamen "0" olan verilere karşılık düşen ölçümler "az güç tüketen" sınıfına, diğer tüm ölçümler "çok güç tüketen" sınıfına ayrılmıştır.

6.4.3 Saldırının Sonucu

<u>İlk Açıklık</u>

Analiz sonunda ilk açıklık 50 bin ölçümde ortaya çıkmıştır. Şekil 6.14'te 1.baytta ortaya çıkan ilk açıklık görülmektedir.



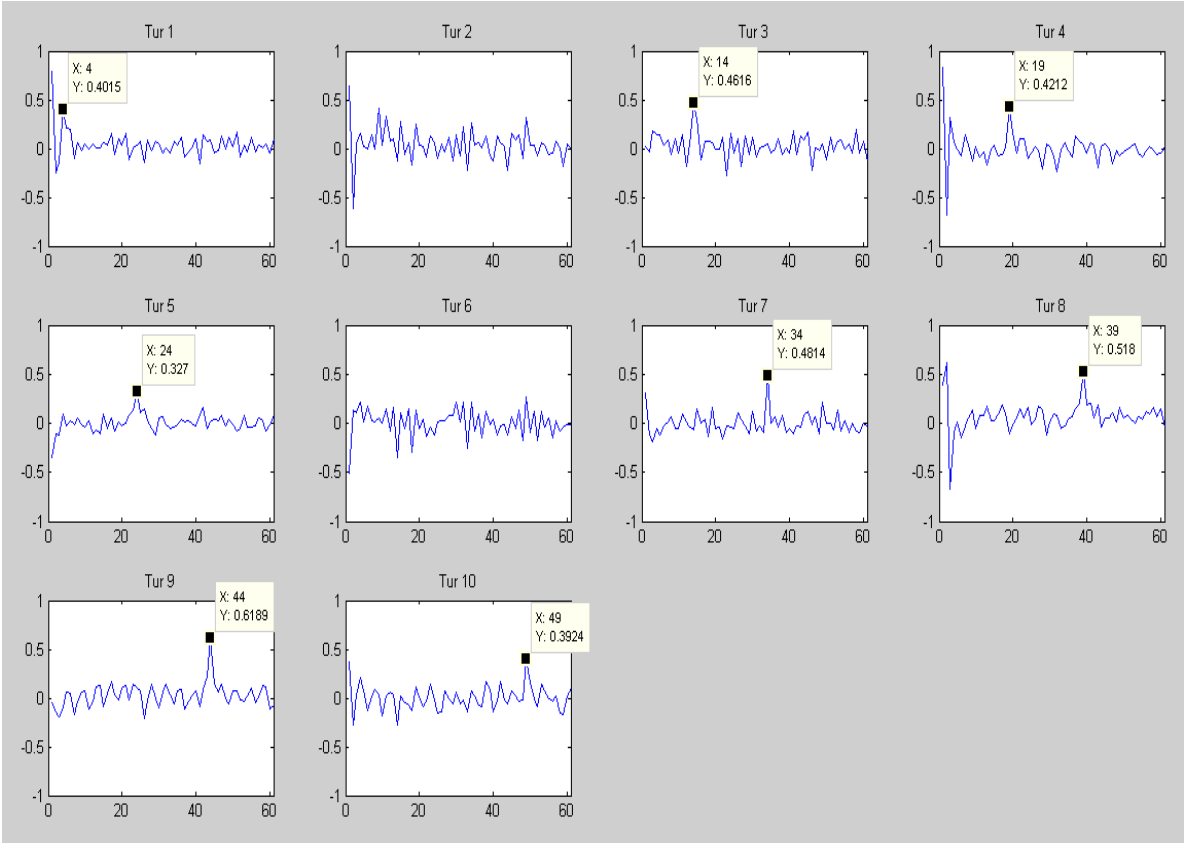
Şekil 6.14 Sıfır giriş saldırısı ilk açıklık

800 bin ölçüm kullanılarak analiz yapıldığında ölçümün başarısının arttığı görülmektedir. Şekil 6.15'de 7.bayt için tüm turlar beraber bakıldığında elde edilen analiz sonucu görülmektedir. Ancak anahtarın elde edilmesi için sadece ilk ve son turlarda alınan sonuçlar kullanılabilecektir.

Bu nedenle ilk ve son turlarda alınan sonuçlara yoğunlaşılmıştır. Çizelge 6.2'de bu turlarda açıklık bulunan baytlar "**+**" ile gösterilmiştir.

Çizelge 6.2 Sıfır giriş saldırısı ilk ve son tur açıklık bulunan baytlar

Bayt No	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
İlk Tur	**+**	**+**			**+**	**+**				**+**	**+**	**+**				**+**
Son Tur		**+**					**+**	**+**						**+**		



Şekil 6.15 Bayt değiştirme işlemine yapılan sıfır giriş saldırısı sonucu

6.5 Sıfır Ofset FGA Saldırısı

Sıfır Ofset FGA Saldırısı, "maskeli veri" ve "maske" verisinin aynı saat darbesinde işlendiği özel durum için önerilmiş bir 2.Derece FGA saldırısı olarak tanımlanabilir. Saldırının temelinde, güç ölçümü üzerinde yer alan bazı noktaların karesini almak suretiyle değerini yükselterek açıklık oluşturma fikri bulunmaktadır. Bu şekilde bir saldırı gerçekleşebileceğinden ilk olarak Chari S. tarafından söz edilmiştir [27]. Ancak saldırının teorik olarak tanımlanması Waddle J. ve Wagner D. tarafından yapılmıştır [12].

Sıralı çalışma yapısından dolayı, önerilen özel durumun mikrodenetleyici uygulamaları üzerinde gerçekleşmesi mümkün değildir. Ancak paralel işlemlerin gerçeklenebildiği FPGA ve ASIC uygulamalarında nadir de olsa söz konusu özel durumun oluşması ihtimali mevcuttur.

6.5.1 Saldırının Tanımı

n : ölçüm sayısı ve *i* : ölçüm numarası olmak üzere $0 < i < n$ şeklinde bir ölçüm seti ele alınsın.

t : zaman ve m : maximum zaman değeri olmak üzere $0 < m < t$ arasında değişen ölçüm verileri ele alınsın.

b : Algoritma tarafından üretilen ara değer kabul edilsin.

$S_i^b(t)$: *i* numaralı ölçümü de, *b* ara değeri üretildiği *t* anında işarete katkı yapan gürültü parametresi olarak kabul edilsin.

d : Bir katsayı olmak üzere;

dR_i^b : "Maske" değerinin toplam güç tüketimine katkısı olarak kabul edilsin.

$d(-1)^b R_i^b$: "Maskeli veri" değerinin toplam güç tüketimine katkısı olarak kabul edilsin.

Bu girdiler doğrultusunda, doğru anahtar verisi için ölçümler 6.1' deki gibi gruplanır.

$$T_i^b(t) = \begin{cases} S_i^b(t) + dR_i^b & t = c_0 \\ S_i^b(t) + d(-1)^b R_i^b & t = c_1 \\ S_i^b(t) & diger \end{cases} \quad (6.2)$$

6.2 denkleminde $t = c_0$ anı "maske" nin, $t = c_1$ anı ise "maskeli veri" nin işlendiği anı göstermektedir.

Eğer sistemde "maske" ve "maskeli veri" aynı anda işlenirse 6.3 denklemi ortaya çıkar.

$$T_i^b(t) = \begin{cases} S_i^b(t) + dR_i^b + d(-1)^b R_i^b & t = c_0 \\ S_i^b(t) & diger \end{cases} \quad (6.3)$$

Bu durum [15] çalışmasında atağın gerçekleşebilmesi için oluşması gereken özel durumdur.

6.3 denklemi analiz edilirse

$b = 1$ için,

$$T_i^1(t) = S_i^1(c_0) \quad (6.4)$$

$b = 0$ için,

$$T_i^0(c_0) = S_i^0(c_0) + 2dR_i^0 \quad (6.5)$$

elde edilir.

6.4 denkleminde görüldüğü üzere $b = 1$ durumu için sonuç elde edilecek bileşen sadece gürültü içermektedir.

6.5' te ise doğru anahtar tahmininde kullanılacak verilerin katkısı görülmektedir. Ancak katkı yapması beklenen bileşen bir *Bimodal Bileşen'*dir ve bu bileşen "0" ortalama değerine sahiptir.

Sıfır Ofset FGA Saldırısı için, bu bileşenin etkisini ortaya çıkarmak amacıyla ölçüm gruplarının karesi alınır.

Bu durumda $b = 0$ durumu için elde edilen ortalama değer saldırının gerçekleşmesine olanak sağlayan farkın ortaya çıkmasını sağlar. Aşağıda kare alma işlemi sonunda b'nin alacağı değere göre oluşacak beklenen değerler gösterilmiştir.

$b = 0$ için,

$$\mathrm{E}\left[\sum_{i=0}^{n-1}\left[T_i^0(c_0)\right]^2\right] = \sum_{i=0}^{n-1}\mathrm{E}\left[S_i^0(c_0))^2 + 4dS_i^0(c_0)R_i^0 + 4d^2(R_i^0)^2\right]$$

$$= \sum_{i=0}^{n-1} \left(\mathrm{E}\left[S_i^0(c_0))^2 \right] + \mathrm{E}\left[4dS_i^0(c_0) R_i^0 \right] + \mathrm{E}\left[4d^2 (R_i^0)^2 \right] \right)$$

$$= \sum_{i=0}^{n-1} (1 + 0 + 4d^2)$$

$$= 4nd^2 + n \qquad (6.6)$$

b= 1 için,

$$\mathrm{E}\left[\sum_{i=0}^{n-1} \left[T_i^1(c_0) \right]^2 \right] = \sum_{i=1}^{n-1} \mathrm{E}\left[(S_i^1(c_0))^2 \right]$$

$$= \sum_{i=0}^{n-1} 1$$

$$= n \qquad (6.7)$$

Eşitlik 6.6 ve 6.7 için oluşan durumların birbirinden çıkarılması ile atağın teorik olarak başarılı olacağı görülmektedir [12].

6.5.2 Saldırının Gerçeklenmesi

Sıfır Ofset Saldırısı ile ilgili temel bilgilerin verilmesi sürecinde de bahsedildiği gibi, bu saldırı çok özel bir durum için geçerlidir.

Tez çalışması kapsamında üzerinde çalışılan *IAIK Maskeli AES Algoritması* VHDL kodu analiz edilmiştir.

Analiz sonucunda başlangıç anahtarı ekleme sonrasında hem "maske" verisinin hem de "maskeli veri" değerinin aynı anda işlendiği görülmüştür. Bunun üzerine Sıfır Ofset FGA Saldırısı gerçeklenmiştir.

Sıfır Ofset FGA Saldırısı yukarıda anlatılan özel durum için, kare alma ön işleminden geçirilmiş normal bir 1.Derece FGA Saldırısı olduğu gözlemlenmiştir. Saldırının gerçeklenmesi için aşağıdaki adımlar takip edilmiştir.

1. Ölçüm alma işlemi sonucunda elde edilen ölçümlerden oluşan *Güç Analiz Matrisi*'nde yer alan her bir ölçüm için kare alma işlemi uygulanarak M_1*Güç Ölçüm Matrisi* elde edilmiştir.
2. Saldırının gerçekleneceği *Başlangıç Anahtarı Ekleme Sonrası* için Bölüm 3.1.3'te belirtildiği üzere *Tahmin Matrisi* elde edilmiştir.
3. Bayt ve bit bazında Bölüm 3.1.3'te anlatılan *Kocher Analizi* uygulanmıştır.

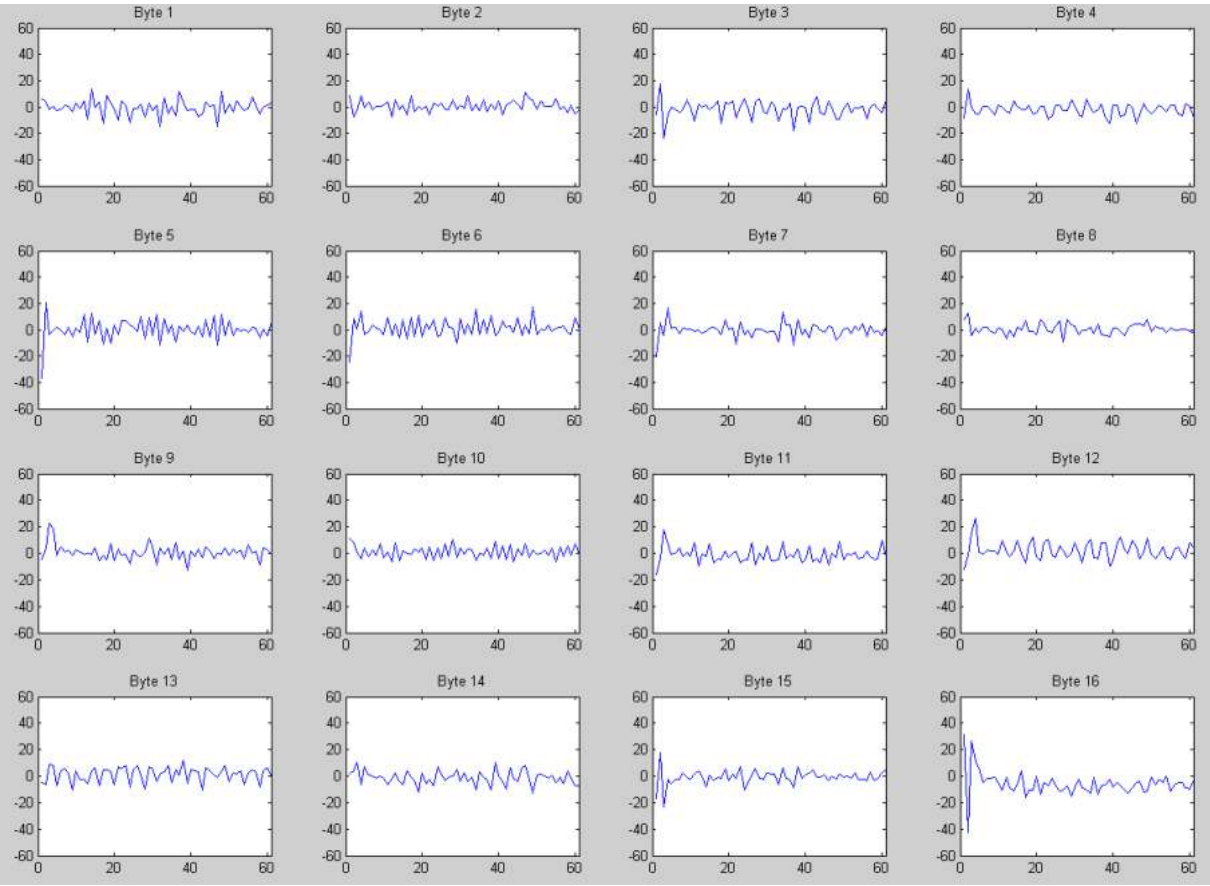
6.5.3 Saldırının Sonucu

Teorik olarak saldırının başarıya ulaşması beklenmiştir. Ancak bayt ve bit bazında yapılan saldırılardan bir sonuç elde edilememiştir.

Bu kapsamda saldırının tanımlandığı [12] çalışması tekrar incelenmiştir. Bu çalışmada pratik olarak bir saldırı gerçeklenmediği görülmektedir.

Ayrıca yine Sıfır Ofset FGA saldırısının [14] çalışmasında gerçeklendiği görülmüştür. Bu çalışmada ASIC uygulaması üzerinde 1 milyon ölçümde başarı elde edilemediği görülmüştür.

FPGA uygulamaları üzerinde FGA saldırıların gerçeklenmesi kriptografik işlemlerin yanı sıra üzerinde gerçekleşen fazladan gürültü etkileri nedeniyle daha zordur [40]. Bu kapsamda bizim çalışmamızda başarılı sonuç alınamamasının beklenebilecek bir sonuç olduğu değerlendirilmiştir. Şekil 6.16'te tüm baytlar için elde edilen başarısız Sıfır Ofset FGA sonuçları görülmektedir.



Şekil 6.16 Tüm baytlar için elde edilen sıfır ofset FGA sonuçları

BÖLÜM 7

IAIK MASKELİ AES ALGORİTMASI ÜZERİNE YAPILAN SALDIRILARIN ETKİNLİK ANALİZİ

Bölüm 6'da, tez çalışması kapsamında üzerinde çalışılan *IAIK Maskeli AES Algoritması* üzerine gerçeklenen farklı saldırılar anlatılmıştır. Bu bölümde, elde edilen sonuçlar kullanılarak saldırıların birbirlerine göre etkinliği analiz edilecektir. Ancak karşılaştırma yapılabilmesi için bazı kriterlerin belirlenmesi gerekmektedir. Ayrıca saldırı sonucunda elde edilen "başarı" değerinin farklı açılardan değerlendirilmesi gerekmektedir.

7.1 Karşılaştırma Kriterleri

Tez çalışması kapsamında üzerinde çalışılan *IAIK Maskeli AES Algoritması* örneğinde olduğu gibi, aynı kriptografik sisteme farklı yöntemler ile saldırarak başarı elde edilebilir. Ancak, bu yöntemlerin her birisi farklı özellikler içerebilir. Örneğin bir saldırı yöntemi, çok az sayıda güç ölçümü ile başarı elde edebilir. Ancak bu yöntem çok özel durumlar için tanımlanmış veya kriptografik sistem hakkında çok ayrıntılı bilgi gerektiriyor olabilir. Aşağıda karşılaştırma işleminde kullanılacak olan kriterler listelenmiştir.

7.1.1 Tasarım Hakkında Bilgi Sahibi Olma

Bazı saldırıların gerçekleştirilmesi için saldırganın tasarım hakkında bilgi sahibi olması gerekmeyebilir. Ancak bazı saldırıların gerçeklenebilmesi için tasarım bilgisine ihtiyaç duyulur. Bir saldırının gerçekleştirilmesi için tasarım bilgisine ihtiyaç duyulması saldırı için zorlaştırıcı bir etkendir. Dolayısıyla etkinlik analizi kapsamında, o saldırının etkinliğini düşüren bir parametre olarak dikkate alınacaktır.

7.1.2 Kaynak Kod Bilgisine Sahip Olma

Bazı saldırılarda tasarım bilgisine sahip olmak bile saldırı için yeterli olmamakta, doğrudan kaynak kod bilgisine ihtiyaç duyulmaktadır. Bir saldırının gerçekleştirilmesi için kaynak kod bilgisine ihtiyaç duyulması saldırı için zorlaştırıcı bir etkendir. Dolayısıyla etkinlik analizi kapsamında, o saldırının etkinliğini düşüren bir parametre olarak dikkate alınacaktır.

7.1.3 Ek Uzmanlık Bilgisine İhtiyaç Duyma

FGA saldırılarında en çok kullanılan analiz yazılımları, matlab, java vb. yazılım programlarıdır. Dolayısıyla bir saldırganın tüm FGA saldırılarında başarılı olabilmesi için bu analiz programlarından birini kullanabilmesi gerekmektedir. Ancak bazı saldırıların gerçekleşmesi için ek olarak başka yazılım geliştirme ortamlarını da kullanabilmek gerekebilir. Bu şekilde ek bir bilgiye ihtiyaç duyulması saldırı süresini uzatacağından, o saldırının etkinliğini düşüren bir parametre olarak dikkate alınacaktır.

7.1.4 Ek Donanım Sistemlerine İhtiyaç Duyulması

Her bir FGA saldırısının gerçekleştirilmesi için bilgisayar ve osiloskop başta olmak üzere Bölüm 5.3'te tam olarak düzeneği verilen donanım gereksinimlerine ihtiyaç duyulmaktadır. Ancak bazı saldırıların gerçekleştirilmesi için standart bir bilgisayar yeterli olmamaktadır. Özellikle analiz sürecinde yüksek anlık hafıza (RAM) ve işlemci gücü değerlerine sahip kuvvetli bilgisayar sistemlerine ihtiyaç duyulmaktadır. Bu şekilde bir donanıma ihtiyaç duyulması, saldırının maliyetini artırdığı için o saldırının etkinliğini düşüren bir parametre olarak dikkate alınacaktır.

7.1.5 Saldırının Teorik Olarak Dayanaksız Olması

Üzerinde çalışılan kriptografik sistemler için bazı saldırılar, teorik olarak gerçeklenebilir görünmekle beraber saldırı yapılan sistem üzerinde başarıya erişemeyebilir. Ancak bazı saldırıların, en başından teorik olarak da başarılı olamayacağı öngörülebilir. Saldırının teorik olarak dayanaksız olması, etkinlik analizinde saldırının etkisini "sıfır" seviyesine düşüren bir parametre olarak dikkate alınacaktır.

7.1.6 Başarı İçin İhtiyaç Duyulan Ölçüm Sayısı

FGA saldırılarında, bir saldırının başarı durumunu göstermek için en çok kullanılan ölçüttür. Tez çalışması kapsamında yapılan saldırıların başarısı değerlendirilirken de bu ölçüt kullanılacaktır. Ancak "başarı" kavramı ile ne kastedildiği Bölüm 7.2'de ayrıca değerlendirilecektir.

7.2 "Başarı" Kavramının İrdelenmesi

FGA yöntemi kullanılarak kriptografik sistemlere yönelik saldırıların yapıldığı makaleler ve analiz laboratuarları tarafından yapılan çalışmalar incelendiğinde, saldırı sonunda gizli anahtar değerinin 1 bit değerinin bile bulunması durumunda o saldırının "başarılı" olarak kabul edildiği görülmektedir. Bu bakış açısı bir bakıma doğrudur. Çünkü bir saldırı ile anahtar verisinden çok küçük de olsa bir değerin bulunması, ölçüm sayısının yeterince artırılması durumunda, anahtarın diğer kısımlarının da bulunacağı yönünde bir göstergedir.

Ancak diğer bir açıdan bakıldığı zaman bazı durumlarda anahtar parçaları elde edilse de bunun tam olarak bir "başarı" olmayacağı görülmektedir.

İlk olarak tek başına anahtar verisinin küçük bir kısmını bulmak, asıl amaç olan anahtarı elde etme sonucu için yetersizdir. Örneğin 128-bit uzunluğundaki AES anahtarının 1-bit değerinin bulunduğu düşünülsün. Bu durumda halen bulunması gereken 127-bit bulunmaktadır. Bu uzunlukta bir anahtar değerinin günümüz teknolojisi kullanılarak deneme yanılma yöntemi ile bulunması mümkün değildir.

Yine benzer şekilde AES algoritmasının analizinde, doğru anahtar modeli ile ilk ve son tur dışında kalan turlarda bulunan açıklıklar anahtar değerinin elde edilmesine katkı sağlamayacaktır. Çünkü Bölüm 3.1.3'te ayrıntıları anlatılan Tahmin Matrisi'nin oluşturulması sürecinde pratik olmayan sayıda anahtar değeri denenmek zorunda kalınacaktır. Örneğin; Geçiş Sayısı Saldırısı kapsamında, AES algoritmasının 5. turu içerisinde yer alan bir bayt için 2^{40} tane anahtar değerinin denendiği bir modelin oluşturulması gerekmektedir. Bu nedenle anahtar verisinin elde edilmesi için ölçüm sayısının artırılarak analiz işlemlerine devam edilmesi gerekir. Bu işlemin devamında bazı saldırılar için nispeten az sayıda ölçüm kullanmak yeterli olurken bazı saldırılarda

bu sonucu almak için diğerinin katları düzeyinde ölçüm kullanmak gerekebilir. Her iki saldırıda da ilk anahtar parçasının bulunduğu noktanın aynı olması durumunda, başarı için sadece anahtar değerinin bir parçasının bulunduğu nokta dikkate alınırsa iki saldırıda eşit olarak "başarılı" görülebilir. Ancak asıl hedef olan anahtar değerinin elde edilmesi için ihtiyaç duyulan ölçüm sayılarına dikkat edildiğinde iki saldırının eşit başarıda olmadığı görülmektedir.

Sonuç olarak tez çalışması kapsamında, saldırıların başarısı değerlendirilirken iki farklı "başarı" durumu belirlenecektir. Bunlar: “ilk açıklık başarısı” ve “anahtarı bulma başarısı” durumlarıdır.

7.2.1 İlk Açıklık Başarısı

Anahtar parçasının miktarına veya bulunduğu algoritma turuna bakılmaksızın, ilk değerin bulunduğu ölçüm sayısıdır.

7.2.2 Anahtarı Bulma Başarısı

Analiz işlemleri ile bir kriptografik sistemin anahtarının elde edilmesi için, bilinmeyen anahtar uzunluğunun deneme yanılma yöntemi (brute force) ile bulunacak sayıya indirilmesi yeterlidir. Bu konuda ölçü olarak 56 bit anahtar uzunluğunun yetersiz olması nedeniyle güvensiz kabul edilen DES algoritması alınmıştır [41]. Yani bulunamayan anahtar miktarının 56 bit sayısına düştüğü anda kullanılan ölçüm sayısı saldırının "anahtarı bulma başarısı" olarak kabul edilmiştir.

7.3 Saldırıların Karşılaştırılması ve Etkinliğinin Analiz Edilmesi

Bu bölümde, Bölüm 6'da gerçeklenen saldırılar ve elde edilen sonuçlar karşılaştırma kriterleri açısından değerlendirilmiş ve saldırıların birbirlerine göre etkinliği analiz edilmiştir. Değerlendirme sonuçları Çizelge 7.1’de yer almaktadır.

Çizelge 7.1 Gerçeklenen saldırıların karşılaştırılması

	Tasarım Hakkında Bilgi Sahibi Olma	Kaynak Kod Bilgisine Sahip Olma	Ek Uzmanlık Bilgisine İhtiyaç Duyma	Ek Donanım Sistemlerine İhtiyaç Duyulması	Saldırının Teorik Olarak Dayanaklı Olması	Başarı İçin İhtiyaç Duyduğu Ölçüm Sayısı	
						İlk Açıklık Başarısı	Anahtar Bulma Başarısı
1. Derece FGA Saldırısı	Gerekli Değil	Gerekli Değil	Gerekli Değil	Gerekli Değil	Dayanaksız	Yok	Yok
2. Derece FGA Saldırısı	Gerekli	Gerekli Değil	Gerekli Değil	Gerekli (Yüksek RAM kapasitesi)	Dayanaklı	50 bin	300 bin -14 bayt
Sıfır Giriş FGA Saldırısı	Gerekli Değil	Gerekli Değil	Gerekli Değil	Gerekli Değil	Dayanaklı	50 bin	800 bin-11 bayt
Geçiş Sayısı Saldırısı	Gerekli	Gerekli	Gerekli (VHDL ortamında yazılım geliştirebilme)	Gerekli Değil	Dayanaklı	50 bin	800 bin-9 bayt
Sıfır Ofset FGA Saldırısı	Gerekli	Gerekli Değil	Gerekli Değil	Gerekli Değil	Dayanaklı	Yok	Yok

Çizelge 7.1 incelendiğinde saldırıların etkinliği üzerine şu yorumlar yapılabilir.

1. Bir saldırının etkinliğindeki en temel nokta saldırının en az sayıda ölçüm elde ettiği başarıdır. 2. Derece FGA Saldırısı'nın " ilk açıklık başarısı" kriterinde diğer ölçümlerden geri kalmadığı, "anahtar bulma başarısı" kriterinde ise diğer saldırılara göre çok daha ileride olduğu görülmektedir. Öte yandan Sıfır Giriş FGA Saldırısı'nın " anahtar bulma başarısı ", 2.Derece FGA Saldırısı'na göre daha geride olmakla beraber, 2. Derece FGA saldırısının ihtiyaç duyduğu ek gereksinimlere ihtiyaç duymadan başarı yakalayabilmektedir.

   İki saldırı tekniği diğerlerine göre daha etkili olmakla beraber, birbirlerine göre farklı durumlar için etki üstünlüğü sağlamaktadır.

   Ancak her durumda, 2. Derece FGA Saldırısı ve Sıfır Giriş FGA Saldırısı'nın *IAIK Maskeli AES Algoritmasının FPGA Üzerinde Gerçeklemesine Yapılan Güç Analizleri* içerisinde en etkili iki saldırı yöntemi olduğu değerlendirilmiştir.
2. Geçiş Sayısı Saldırısı'nın elde ettiği başarının Sıfır Giriş FGA Saldırısı ile yaklaşık aynı düzeyde olduğu görülmektedir. Ancak Geçiş Sayısı Saldırısı'nın gerçeklenebilmesi için saldırganın başta kaynak kod olmak üzere ek gereksinimlere ihtiyaç duyması saldırının etkinliğini azaltacak bir faktördür. Bu

nedenle, *IAIK Maskeli AES Algoritmasının FPGA Üzerinde Gerçeklemesine Yapılan Güç Analizleri* içerisinde, Geçiş Sayısı Saldırısı'nın başarı elde eden diğer yöntemlerin gerisinde kaldığı ve 3. en etkili saldırı yöntemi olduğu değerlendirilmiştir.

3. Teorik olarak dayanaklı olan bir saldırının, bir sistem üzerine uygulandığında başarılı olamamış bile olsa, daha fazla ölçüm sayısı için başarı gösterme ihtimali mevcuttur. Sıfır Ofset FGA saldırısı bu kapsamda incelendiğinde, teorik olarak dayanaklı olsa da üzerinde çalışılan uygulamada başarı elde edememiştir. Sonuç olarak Sıfır Ofset FGA saldırısının, *IAIK Maskeli AES Algoritmasının FPGA Üzerinde Gerçeklemesine Yapılan Güç Analizleri* içerisinde 4. en etkili FGA saldırısı olduğu değerlendirilmiştir.
4. Bir saldırının başarılı olması teorik olarak mümkün değilse o saldırının hiçbir şekilde etkinliği yoktur. Bu nedenle 1. Derece FGA Saldırısı, *IAIK Maskeli AES Algoritmasının FPGA Üzerinde Gerçeklemesine Yapılan Güç Analizleri* içerisinde en etkisiz saldırıdır.

BÖLÜM 8

SONUÇ ve ÖNERİLER

Gelişen teknoloji ve buna bağlı olarak oluşan yaşam şeklimizde, bilgi güvenliğinin sağlanması çok önemli bir durum arz etmektedir. Bilgi güvenliğini sağlayan sistemlerin arkasında ise kriptografik uygulamalar ve bu uygulamaların çalışmasını sağlayan kriptografik algoritmalar yer almaktadır. AES algoritması bu alanda en yaygın olarak kullanılan algoritmalardan biridir. Matematiksel olarak güçlü olması nedeniyle AES algoritması klasik kriptanalize dayanıklı görülmektedir. Ancak uygulamaya bağlı olarak gerçeklenebilen yan kanal analizi saldırıları AES algoritması uygulamaları için ciddi bir tehdit oluşturmaktadır. Bu tehdide karşı olarak AES algoritması için; algoritma tarafından üretilen ara değerlere rastgele sayıların eklendiği farklı maskeleme teknikleri önerilmiştir. Bu tekniklerin içerisinde en yaygın kullanılanlarından biri IAIK maskesidir. Ancak kriptografik uygulamalarda geliştiriciler tarafından önerilen her önleme karşı saldırganlar tarafından farklı saldırı teknikleri geliştirilmektedir. Bu bağlamda IAIK Maskeli sistemler için de uygulanabilecek saldırı teknikleri önerilmiştir. Ancak bu saldırı tekniklerinin uygulanabilirlikleri ve etkinlikleri farklı durumlar için değişkenlik göstermektedir.

Tez çalışması kapsamında IAIK Maskesi ile korunmuş bir AES sisteminin pratik olarak analiz edilmesi ve üzerinde gerçeklenebilecek saldırıların etkinliğinin analiz edilmesi hedeflenmiştir. Bu kapsamda öncelikle genel kriptografik sistemler ve IAIK Maskeli AES Algoritması çalışma yapısının anlaşılması için teorik araştırmalar yapılmıştır. Bu kapsamda elde edilen bilgiler Bölüm 2 ve Bölüm 4'te anlatılmıştır. Bu teorik araştırma kapsamında Bölüm 3'te anlatılan Yan Kanal Analizi konusunda kapsanarak ileride yapılacak analiz çalışmaları için teorik altyapı hazırlanmıştır. Daha sonra *IAIK Maskeli*

AES Algoritması uygulamasının pratik olarak gerçeklendiği bir düzenek aranmıştır. Başka bir tez çalışması kapsamında gerçeklenmiş bir düzenek temin edilerek bu düzenek üzerinde analiz uygulamaları gerçeklenecek şekilde çalışır hale getirilmiştir. Üzerinde çalışılan düzeneğin donanım ve yazılım özellikleri Bölüm 5'te anlatılmıştır. Aynı bölüm kapsamında düzenek üzerinden analiz işlemlerinde kullanılacak verilerin toplanması da anlatılmıştır.

Verilerin toplanmasından sonra IAIK Maskeli AES Algoritması üzerine gerçekleşebilecek saldırılar üzerine yoğunlaşılmıştır. Bu kapsamda her bir saldırı için teorik araştırma ve pratik gerçeklemeler yapılmıştır. Saldırılar ve elde edilen sonuçlar Bölüm 6'da ayrıntılı olarak anlatılmıştır. Bölüm 7'de ise, her bir saldırının gerçeklenebilme durumları ve etkinliği analiz edilmiştir.

Yapılan çalışma ile yeni bir saldırı tekniği önerilmemekle beraber, pratik bir sistem üzerinde pratik saldırılar gerçekleyerek literatürde daha önce bizim gerçekleştirdiğimiz kapsamda, geniş olarak benzeri yapılmamış olan bir etkinlik analizi çalışması yapılmıştır. Yapılan çalışmanın benzeri bir sistemi tasarlayacak veya analiz edecek olan kişiler için bir kılavuz olması ümit edilmektedir.

Bilgi güvenliği kapsamında, “güvenli kriptografik uygulamalar ve bunlara yönelik saldırılar” konusunun önemini koruyacağı görülmektedir. Bu bağlamda, bu tez çalışması kapsamında gerçekleştirilen saldırılar ve önerilmesi muhtemel yeni saldırı tekniklerine karşı maskelemeye ek olarak alınabilecek yeni güvenlik önlemleri üzerinde çalışılması önerilmektedir. Bu tür çalışmalar hali hazırda aktif olarak devam etmektedir. Ancak daha fazla çalışmanın gerçeklenmesi, bilgi güvenliği ihtiyacının çözümü için temel noktayı oluşturan güvenli kriptografik sistemlerin tasarlanması için önemlidir.

KAYNAKLAR

[1] Kerckhoffs A., (1883). "La cryptographie militaire", Journal des sciences militaires, IX: 5–83,

[2] Kocher P., (1996). "Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems", Advances in Cryptography: CRYPTO'96, 1109: 104-113.

[3] Kocher P., Jaffe J. ve Jun, B., (1999). "Differential power analysis", Advances in Cryptography: CRYPTO'99, 1666: 388-397.

[4] Kommerling O. ve Kuhn, M.G., (1999). "Design principles for tamper resistant smartcard processors", Proceedings of the USENIX Workshop on Smartcard Technology.

[5] Joye, M. ve Lenstra, A.K., (1999). "Chineese remaindering based cryptosystem in the presence of faults", Journal of Cryptology, 4: 12-241-245.

[6] Daemen, J. ve Rijmen, V., (2002). The Design of Rijhndael AES-The Advanced Encryption Standart, Springer.

[7] FIPS 197, (2001). Advanced Encryption Standard. National Institute of Standarts and Technology (NIST).

[8] FIPS 46-3, (1999). Data Encryption Standard. National Institute of Standarts and Technology (NIST).

[9] Standaert, F.X. ve Örs, S.B., Preneel, B., (2004). "Power analysis attack on an FPGA implementation of AES", CHES-2004, 3156: 30-44.

[10] Oswald E., Mangard S., Premstaller N. ve Rijmen V., (2004). "A Side Channel Analysis Resistant Description of the AES S-Box", FSE 2005, 3557.

[11] Oswald E., Mangard S., Herbst C. ve Tillich S., (2006). "Practical Second Order DPA Atacks for Masked Smart Card İmplementation of Block Ciphers". CT-RSA 2006, 3860: 192-207

[12] Waddle J., Wagner D., (2004). "Towards Efficent Second Order Power Analysis", 6th International Workshop, Cambridge, USA, 3116: 1-15.

[13] Mangard, S., Pramstaller N. ve Oswald, E., (2005). "Successfully attacking masked AES implementations", CHES-2005, 3659: 157-171.

[14] Mangard S. ve Shramm K., (2006). "Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementation" , CHES 2006, 4249: 76-90.

[15] Fuchs, László, (1999). Infinite abelian groups, Pure and Applied Mathematics, New York-London: Academic Press. 36: 290.

[16] Axler, S., (1997) Linear Algebra Done Right, 2e, Springer.. Abstract algebra theory.

[17] Allenby R.B.J.T. Rings, Fields and Groups. (1991). Butterworth-Heinemann.

[18] Doğan, A.Y., (2008). AES Algoritmasının FPGA Üzerinde Düşük Güçlü Tasarımı, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Türkiye.

[19] Ordu, L., (2006). AES Algoritmasının FPGA Üzerinde Gerçeklenmesi ve Yan Kanal Analizi Saldırılarına Karsı Güçlendirilmesi, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, Türkiye.

[20] Shamir, A. ve Billiam, E., (1993). "Differential Cryptanalysis of DES-like syaytems", Journal of Cryptography, 4: 3-72.

[21] Matsui, M., (1994). "The first experimental cryptanalysis of the data encryption standard", Proceedings of CRYPTO'94, 1-11.

[22] Anderson, R. ve Kuhn, M., (1996). "Tamper resistance – a cautionary note", Proceedings of the 2nd USENIX Workshop on Electronic Commerce, 1-11.

[23] Kang S. M. ve Leblebici Y., (2002). CMOS Digital Integrated Circuits: Analysis and Design. Mc Graw Hill.

[24] Coron, J.S. ve Goubin, L., (2000). "On boolean and arithmetic masking against differential power analysis", Proceedings of the 2nd International Workshop on CHESS, number 1965: 231-237.

[25] Shamir, A., (2000). "Protecting smart cards from passive power analysis with detached power supplies", CHES-2000, 1965: 71-77.

[26] Tiri, K. ve Verbauwhede I., (2003). "Securing encryption algorithms against DPA at logic level: Next generation smart card technology", CHES-2003, 2779: 125-136.

[27] Chari S., Jutla C.S., Rao J.R. ve Rohatgi P., (1999). "Towards sound approaches to counteract power-analysis attacks", Advances in CRYPTO'99, 1666: 398-412.

[28] Coron J.S., (1999). "Resistance against differential power analysis for eliptic curve systems", CHES-1999, 1717: 292-302.

[29] Akkar M.L. ve Giraud C., (2001). "An Implementation of DES and AES, Secure Against Some Attacks", CHES-2001, 2162: 309-318.

[30] Şahinoğlu, M. (2008). AES Algoritmasının FPGA Üzerinde Gerçeklemesine Elektromanyetik Alan Saldırıları, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, Türkiye.

[31] Trichina E., De Seta D. ve Germani L., (2003). "Simplified Adaptive Multiplicative Masking for AES" , CHES 2002, 2435: 187-197.

[32] Goliç J. D. ve Tymen C., (2003). "Multiplicative Masking and Power Analysis of AES", CHES 2002 2535: 198-212.

[33] Akkar M.L., Bevan R. ve Goubin L., (2004). "Two Power Analysis Attack Against One Mask-Method", 11th International Workshop, 3017: 332-347.

[34] Blömmer J., Guarajardo J. ve Krummel V., (2005). "Provably Secure Masking of AES". 11 th International Workshop, SAC 2005, 3357: 69-73.

[35] Marioka S. ve Akishita T., (2004). "A DPA Resistant Compact AES S-Box Circuit Using Additive Mask". Computer Security Composium, 679-684.

[36] Messerges T. S., (2000). "Using Second Order Power Analysis to Attack DPA Resistant Software". 1965: 238-251.

[37] Peeters E., Standaers F. X., Donckers N. ve Quisquater J., (2005). Improved Higher Order Side-Channel Attack with FPGA Experiments. CHES 2005.

[38] Suzuki D., Saeki M. ve Ichikawa T., (2005). "DPA Leakage Models for CMOS Logic Circuit", CHES 2005, 3559: 366-382.

[39] Mangard S., Popp T. ve Gammel M. (2005). "Side Channel Leakage of Masked CMOS Gates", CT-RSA 2005, 3376: 351–365

[40] Wollinger W. ve Paar C., (2004). "Security Aspect of FPGAs in Cryptographic Application", Chair for Communication Security (COSY), Ruhr-UniversitÄat Bochum, Germany.

[41] Kumar S., Paar C., Pelzl J. ve Schimmler M. , (2006). "Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker" CHES 2006.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Muhammet ÖZTEMÜR

Doğum Tarihi ve Yeri : 01.08.1985 Çarşamba/SAMSUN

Yabancı Dili : İngilizce

E-posta : muhammetoztemur@gmail.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Elektronik ve Haberleşme Mühendisliği	Yıldız Teknik Üniversitesi	2007
Lise	Fen Bilimleri	Bafra Lisesi	2002

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2009	TÜBİTAK BİLGEM UEKAE	Araştırmacı/Bilgi Güvenliği
2008	Nortel Netaş	Donanım Tasarımcısı