SİMETRİK VE ASİMETRİK ŞİFRELEME ALGORİTMALARINA YAN KANAL SALDIRILARI UYGULANMASI VE ANORMALLİK TESPİTİ

Burcu SÖNMEZ

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Ahmet Bedri ÖZER

TEMMUZ -2018

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SİMETRİK VE ASİMETRİK ŞİFRELEME ALGORİTMALARINA YAN KANAL SALDIRILARI UYGULANMASI VE ANORMALLİK TESPİTİ

YÜKSEK LİSANS TEZİ

Burcu SÖNMEZ

(122129103)

Tezin Enstitüye Verildiği Tarih : 02.07.2018
Tezin Savunulduğu Tarih : 18.07.2018

Tez Danışmanı :	Prof. Dr. Ahmet Bedri ÖZER (Fırat Üniversitesi)
Diğer Jüri Üyeleri :	Prof. Dr. Mehmet KAYA (Fırat Üniversitesi)
	Dr. Öğr. Üyesi Soner KIZILOLUK (Munzur Üniversitesi)

TEMMUZ-2018

ÖNSÖZ

Bu çalışmada değerli vaktini bana ayırarak çalışmamın bitirilmesinde her türlü desteğini esirgemeyen sayın danışman hocam Prof. Dr. Ahmet Bedri ÖZER' e teşekkürlerimi sunmak istiyorum.

Her konuda benim için harcadığı zaman ve çabalardan ötürü değerli meslektaşlarım İbrahim Rıza HALLAÇ'a ve Ali Can ATICI'ya, hayatımın her alanında ilgi, anlayış ve her türlü desteğini esirgemeyen aileme çok teşekkür ediyorum.

Burcu SÖNMEZ

ELAZIĞ - 2018

İÇİNDEKİLER

Sayfa No

ÖZET

Bu tez, simetrik şifreleme algoritmaları ve asimetrik şifreleme algoritmalarına uygulanan yan kanal saldırılarından zaman odaklı önbellek saldırılarını ve güç saldırılarını içermektedir. Zaman odaklı önbellek saldırısı ve güç saldırısı sırasıyla simetrik şifreleme algoritmalarından AES algoritmasına ve asimetrik şifreleme algoritmalarından olan RSA algoritmasına uygulanmıştır. Uygulanan saldırıların performansları değerlendirilerek ve makine öğrenmesi teknikleriyle anormallik tespiti yapılarak saldırılar hakkında bilgi elde edilmiştir. Elde edilen bilgiler sonucunda makine öğrenmesi teknikleriyle saldırıların tespit edilmesi konusunda ve uygulanan yöntemler hakkında bilgi elde edilebileceği ispatlanmıştır.

Anahtar Kelimeler: AES, Önbellek Saldırıları, RSA, Güç Saldırıları, Anormallik Tespiti

SUMMARY

APPLICATION OF SIDE CHANNEL ATTACKS ON SYMMETRIC AND ASYMMETRIC ENCRYPTION ALGORITHMS AND ANOMALY DETECTION

This thesis deals with time-driven cache attacks and power attacks from side channel attacks applied to symmetric cryptographic algorithms and asymmetric cryptographic algorithms. Time-driven cache attacks and power attacks are applied to the AES algorithm, which is a symmetric encryption algorithm, and the RSA algorithm, which is an asymmetric encryption algorithm, respectively. Information about the attacks has been obtained by evaluating the performances of the attacks and by detecting anomaly by machine learning techniques. According to the findings obtained, information about the attacks was obtained. As a result of the information obtained, it has been proved that machine learning techniques can be used to detect attacks and obtain information about applied methods.

Key Words: AES, Cache Attacks, RSA, Anomaly Detection, Power Attacks

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMALAR LİSTESİ

AES : Advanced Encryption Standards
AMD : Advanced Micro Devices
CRT : Chinese Remainder Theorem
CPU : Data Encryption Standards
DPA : Differantial Power Analysis
FIPS : Federal Information Processing Standards
FPGA : Field-Programmable Gate Array
IBM : International Business Machines
L1 : Level 1 Cache
L2 : Level 2 Cache
LLC : Last Level Cache
MA : Micro Architectural Attack
NIST : National Institute of Standards and Technology
OPENSSL : Open Secure Socket Layer
PID : Process Identification Number
RSA : Rivest , Shamir, Adleman
SPA : Simple Power Analysis
XOR : Exclusive Or

1. GİRİŞ

Tarih boyunca insanlar birbirleriyle gizli haberleşme ihtiyacı duymuşlar ve bunun güvenilir yollarını aramışlardır. Geçmişte, kriptoloji bilgiyi yalnızca düşmandan gizlemek amacıyla kullanılmıştır. Günümüzde ise teknolojinin hayatımıza girmesiyle birlikte bilginin güvenliği çok daha önemli hale gelmiştir. Elektronik ticaret, bankacılık, sağlık işlemleri, kimlik doğrulama gibi birçok işlem elektronik ortamda gerçekleştirilmektedir. Hayatımızda oldukça kolaylık sağlayan elektronik ortamlar bilginin güvenliği açısından tehlike oluşturmaktadır. Bu yüzden bilginin güvenli bir şekilde iletilmesi için şifreleme sistemleri kullanılmaktadır. Şifreleme sistemlerinin kullanıldıkları tüm alanlarda güvenlik büyük ölçüde kullanılan kriptografik algoritmalara dayanmaktadır. Yunancada gizli anlamına gelen kripto ve yazılmış bir şey anlamına gelen grafi kelimelerinden oluşan kriptografi gizli yazışma sanatı anlamına gelmektedir. Kriptoloji, matematiğin hem şifre bilimi (kriptografi), hem de şifre analizini (kriptoanaliz) kapsayan dalıdır. Kriptografinin amacı, ileti güvenliğini sağlamaktır. Kriptoanalizin amacı ise var olan şifreleri çözmektir. Bu tezde "Kriptoloji"nin bir alt dalı olan "Kriptoanaliz" yöntemlerinden biri olarak yan kanal saldırıları uygulanmıştır. Yan kanal saldırıları bir sistemde gerçekleşen kriptografik işlemlerin dışarı yaydığı bilgileri kullanarak gerçekleştirilen bir uygulama saldırısıdır. Sistemin dışarı yaydığı bilgiler literatürde yan kanal bilgileri olarak geçmektedir. Gerçekleştirilen saldırı yan kanal bilgilerine bağlı olarak isimlendirilir. Örneğin sistemin güç tüketimi ölçülerek güç saldırıları gerçekleştirilmektedir. Bu saldırıların önemli bir örneği [1]' de verilmektedir. Bir diğer saldırı örneği tezde de kullanılan önbellek saldırısıdır. Kriptografik işlemler gerçekleşirken önbellek davranışlarını yan kanal bilgisi olarak kullanan bu saldırı türünün önemli örnekleri Tsunoo ve arkadaşları [2,3] tarafından sunulmuştur. Daha sonra ise Bernstein'in [4] uyguladığı zaman odaklı önbellek saldırısı popüler olmuştur. Literatürde yer alan diğer önbellek saldırıları [5,6] 'da verilmiştir. Literatürde yer alan yan kanal saldırıları sonucunda birçok şifreleme algoritmasının önlem alınmadığı takdirde yan kanal saldırılarına karşı dirençli olmadığı gözlemlenmiştir. Buna dayanarak saldırılara karşı önlem alınması amacıyla [7,8]'da yan kanal saldırılarının tespit edilmesi için makine öğrenmesi yöntemleri sunulmuştur. Literatürde yer alan bu çalışmalar, tezin ilerlemesine temel kaynaklar olarak ışık tutmuştur.

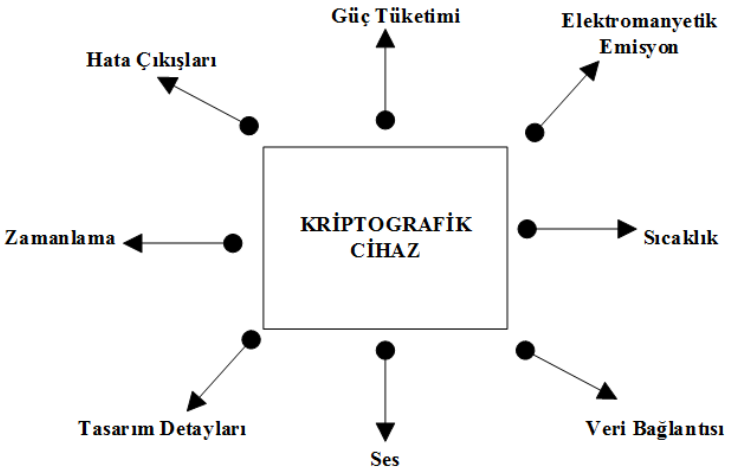
Tezde kullanılan saldırı örneği Bernstein'in zaman odaklı önbellek saldırısıdır. Tezin 2.Bölümü'nde yan kanal saldırıları ele alınmıştır. Bu bölümde genel olarak yan kanal saldırıları açıklanmış ve tezde kullanılan önbellek ve güç analizi saldırılarının teorik alt yapısı bu bölümde detaylandırılmıştır. 3.Bölüm'de saldırı tespit yöntemlerinden biri olan makine öğrenmesi ile saldırı sürecinin saptanması ele alınmıştır. Saldırı sürecinin saptanması amacıyla kullanılan teknikler bu bölümde tanıtılmıştır. Bu tekniklerden biri olan anormallik tespiti ve kümeleme algoritması detaylandırılmıştır. 4.Bölüm'de AES algoritması üzerinde uygulanan zaman odaklı önbellek saldırısı ve anormallik tespiti sonuçları açıklanmıştır. 5.Bölüm'de FPGA üzerinde gerçekleştirilen RSA ile sayısal imza uygulamasının güç analizi ve anormallik tespit yöntemi açıklanmıştır. Sonuç olarak 6.Bölüm'de gerçekleştirilen yan kanal saldırıları ve saldırı tespit yöntemi olarak makine öğrenmesi teknikleri değerlendirilmiştir. AES algoritması OPENSSL ile gerçekleştirilmiştir. Gerçekleştirilen AES şifrelemesine zaman odaklı önbellek saldırısı uygulanmıştır ve saldırı tespit yöntemlerinden anormallik tespiti gerçekleştirilmiştir. RSA algoritması ile FPGA üzerinde sayısal imza uygulaması gerçekleştirilmiştir. FPGA üzerinde sayısal imza uygulaması için iki farklı yöntem olarak Klasik RSA ile Çin Kalan Teoremi kullanılmıştır. Her iki yöntemin güç analizi yapılarak anormallik tespiti gerçekleştirilmiştir.

2. YAN KANAL SALDIRILARI

Kriptografik analiz yöntemleri geleneksel matematiksel saldırılar ve uygulama saldırıları olarak 2 alanda incelenmektedir [9]. Geleneksel matematiksel saldırı yöntemlerinde algoritma ideal matematiksel nesnelere göre modellenmiştir. Bu saldırı türleri genellikle tipiktir ve algoritmaya özel tasarlanmıştır. Bu saldırı yöntemleri literatürde klasik kriptografik analizi olarak da geçmektedir. Klasik kriptografik analizde şifreleme için bir düşmanın kara kutuya erişimi dikkate alınır. Yani düşman için sadece girişler ve çıkışlar önemlidir. Örneğin seçilen bir şifreli metin saldırısında, düşman şifre çözme tahmini için seçtiği şifreli metinleri gönderebilir ve karşılık gelen düz metinleri geri alabilir. Ancak, gerçek hayatta, bir düşman daha güçlü olabilir. Örneğin, düşman ek olarak saldırı altında şifreleme yürütülmesini izleyebilir ve yürütme zamanı ya da güç tüketimi gibi bazı yan kanal bilgileri toplayabilir. Yan kanal analizi arkasındaki fikir bu ekstra bilgilerden bazı gizli verileri anlamaktır. Uygulama saldırıları olarak genellikle yan kanal saldırıları uygulanmaktadır. Bu saldırı yöntemi fiziksel bir uygulama saldırısıdır ve bu saldırı karşısında güvenliği kontrol etmek oldukça zordur.

Klasik kriptografik analizde şifreleme süreci, bir giriş dizisinin özel bir anahtar ile bir çıkış dizisine dönüştüren hesaplamalar zincirinden oluşmaktadır. Klasik kriptografik analizde şifrenin ters yönde kırılması söz konusudur. Yan kanal analizinde ise şifreleme süreci, belli koşullarda belirli işlemler yapan ve izlenebilen karakteristikler sergileyen bir uygulama olarak görülmektedir. Bu analizde belirli koşullarda tekrarlanan karakteristiklerin ölçülmesi ve yorumlanması ile şifre çözülebilmektedir [10].

Kriptografik cihazlar açık veri ve kapalı veri dışında dışarıya bazı istemsiz bilgiler sızdırmaktadır ve bu bilgiler kolaylıkla ölçülmektedir. Örneğin bir işlemin gerçekleşme zamanı, cihazın ne kadar güç harcadığı, ne kadar elektro manyetik yayılım yaptığı, nasıl ve ne şiddette sesler çıkardığı veya ne kadar ısı yaydığı en çok kullanılan bilgilerdir [11].



Şekil 2.1. Yan kanal bilgileri

Cihazdan sızan bu bilgiler, cihaz içindeki gizli bilgilerle bağlantılıysa yan kanal bilgisi olarak adlandırılır. Yan kanal bilgileri ölçülerek şifreleme algoritması, şifre gibi gizli bilgiler elde edilmektedir [12]. Yöntemin uygulanabilmesi için hedefe fiziksel erişimin olması gerekir. Bu yüzden genelde bu yöntemin uygulandığı cihazlar ağ üzerindeki bir bilgisayar yerine basit elektronik cihazlardır.

Yan kanal saldırıları yöntemleri; uygulamaya, ortama ve çözülecek şifreye özel olarak değiştirilip uygulanmaktadır.

Yan kanal saldırıları bozucu, bozucu olmayan ve yarı bozucu saldırılar olarak üç gruba ayrılır. Bozucu saldırılarda tüm devrenin paketi açılarak enformasyon elde edilmeye çalışılır.

Bozucu saldırılar aktif ve pasif olarak ikiye ayrılır. Tüm devrenin açılıp veri hattına ulaşılıp ölçülmesi pasif saldırıdır. Tüm devreye hata ürettirecek şekilde, örneğin bağlantı yollarında açık-devre veya kısa-devre oluşturmak suretiyle, müdahale etmek ise aktif saldırıdır [10]. Bu saldırıya örnek Hata Oluşturma saldırısıdır. İlerleyen bölümlerde bu saldırı incelenecektir.

Bozucu saldırılara karşı tüm devrelerde fiziksel önlemler alınır. Saat frekansı, ışık, UV, sıcaklık, devrenin çektiği akım gibi parametrelerde sapmalar bir saldırı belirtisidir. Bu sapmaları fark eden algılayıcılar kullanılır. Bozucu saldırılar laboratuvar koşulları gerektirdiğinden pahalıdır. Bu nedenle gözleme dayalı ataklar, atak yapanlar tarafından öncelikli olarak tercih edilmektedir [10].

Yarı iletken bir çipin koruma katmanına zarar vermeden ve çip ile fiziksel bağlantı kurmadan, gizli bilgileri elde etme yöntemidir. Genellikle elektromanyetik olarak fiziksel hattı dinleyerek bilgi elde etmeye çalışılır.

Bozucu olmayan saldırıda kriptografik donanıma zarar vermeden sadece dışarıdan gözlem yapılır. İşaret işleme süresinin, elektromanyetik emisyonun, harcanan gücün ölçülmesi bozucu olmayan saldırılardır. Uygulaması daha çok olan saldırılardır. Bu saldırılar pasif saldırı kategorisine de girmektedir.

Literatürde temel olarak 4 saldırı tipi vardır [12]. Bunlar:

- Zamanlama Saldırıları
- Güç analizi Saldırıları
- Elektromanyetik Saldırılar
- Akustik Saldırılar

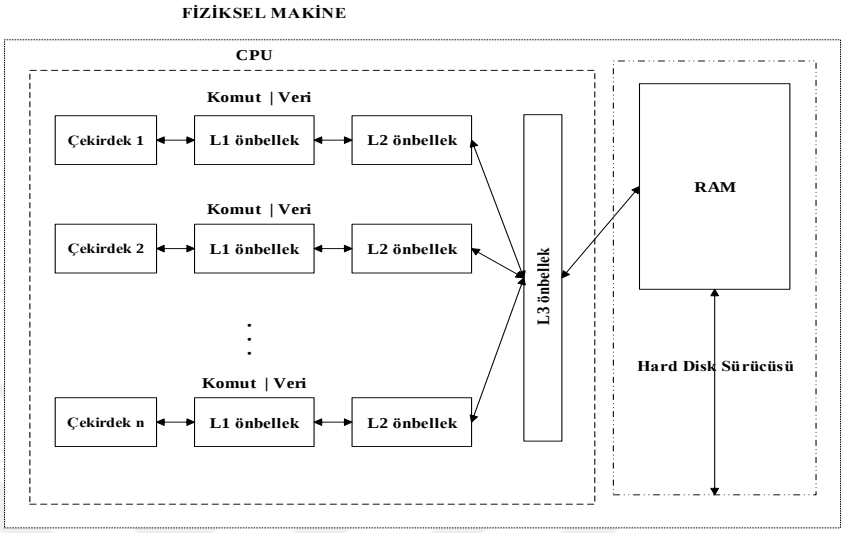
2.1. Önbellek Saldırıları

2.1.1. Önbellek Mimarisi

Önbellek mikroişlemciler bağlamında ana hafıza ve işlemci arasında yer alan hafızanın küçük bir alanıdır. Ana hafızaya erişim işlemci hızıyla kıyaslandığında daha yavaş olduğu için performans genellikle veri yükleme ve depolamaya ihtiyaç duyan işlemler tarafından etkilenmektedir. Önbellek, en sık kullanılan verilerin kopyalarını saklar. İşlemciye daha yakın bir hızda çalışan bir önbellek, en sık kullanılan verileri depolayarak bu sorunun çözülmesine yardımcı olur; böylece bunlara erişme maliyeti azaltılır [13]. Önbellek, bu ekstra seviyeye sahip olmak için ilk ticari bilgisayardaki işlemci ve ana bellek arasındaki bellek hiyerarşisinin seviyesini temsil etmek için seçilen addır [14].

Modern işlemciler önbellek üzerinde Şekil 2.2’ de gösterildiği gibi L1, L2 ve L3 olmak üzere üç tane seviyeye sahip olabilmektedir. Onlardan her biri veri veya talimat olarak özel bir amaç için belirlenebilir. Fakat, donanım mimarisi olarak adlandırıldığında L1 önbelleği genellikle veri önbelleği ve talimat (komut) önbelleğine ayrılmaktadır. L2 önbelleği hem komutu ve hem de veriyi aynı anda tutabilir; L2 önbelleği L1’den daha yavaş fakat daha büyüktür [5]. L3 boyut açısından en büyük önbellektir, ancak bunlar arasında en yavaş olanıdır. L3 seviyesine son seviye önbelleği (LLC:Last Level Cache) de

denmektedir. L3, işlemci çekirdeği arasında paylaşılır ve diğerleri (L1 ve L2) süreçler ve iş parçacıkları arasında paylaştırılır [15].



Şekil 2.2. CPU önbellekleri ve seviyeleri

Bir önbellek boyutu sabit olan bloklara veya çizgilere ayrılır: tipik blok boyutları 32, 64, 128 bayttır. Ancak burada önemli olan nokta L1 ve L2 önbelleklerinin farklı blok boyutlarına sahip olabilmesidir[5].

İşlemci, bellek sistemine bir erişim sorunu olduğunda, başvurulan bellekteki adres ilk önce bir önbellek satırına eşlenir. Önbellekteki, adreslerin eşlendiği önbellek satırlarının bir örneği Tablo 2.1 ‘ de verilmiştir. Tablo 2.1’ deki önbelleğin ilk satırı ‘0b00100000’ adresi, dördüncü satırı ise ‘0b00001101’ adresi içermektedir. Bu satırların dışında kalanlar da önbelleğin boş satırlarını ifade etmektedir.

Tablo 2.1. Satır başına sekiz satır ve dört öğe içeren örnek bir önbellek yapısı

Geçerlilik	Etiket	İçerik			
True	0b001	0x20	0x21	0x22	0x23
False	-------	-------	-------	-------	-------
False	-------	-------	-------	-------	-------
True	0b000	0x00	0x01	0x02	0x03
False	-------	-------	-------	-------	-------
False	-------	-------	-------	-------	-------
False	-------	-------	-------	-------	-------
False	-------	-------	-------	-------	-------

Önbellek ana bellekten daha küçük olduğu için eşleme şeması, birden fazla adresin aynı satıra eşlenebileceği anlamına gelen bir sarmal efekti oluşturmaktadır. Bir önbellek satırı; önbellek satırının geçerli olup olmadığını belirleyen önbellek bayrağı, satırdaki içeriği açıklayan önbellek etiketi ve gerçek bellek içeriğini tutan önbellek verileri olmak üzere üç ana bilgiden oluşur [13].

Önbelleklerin tümünde önbellek satırı, aynı içerik alanında çeşitli farklı adreslerden gelen verileri içerebilir. Yani, birkaç adres aynı satırda eşlenebilir ve bu satırın içinde bulunabilir. Bu, daha büyük veri bloklarının aynı anda ana belleğe aktarılmasını ve daha küçük aktarımlarla uğraşmaktan daha etkili olmasını sağlar. Önbellek etiketi alanı, benzersiz tanımlama yaparak eşleme şemasındaki bu tür karmaşıklıklara izin verir. Böylece önbellek satırının içeriğiyle bir adres her zaman eşleşebilir.

İşlemci, ana bellekteki bir konumu okumak istediğinde, öncelikle verilerin önbellekte olup olmadığını kontrol eder. Veriler zaten önbellekte bulunuyorsa önbellek vuruşu (cache hits) olarak adlandırılır. İşlemci, bir önbellekten önemli ölçüde daha uzun bir bekleme süresi olan ana belleğe erişmek yerine bu verileri hemen kullanır. Aksi halde, yani istenilen veriler önbellekte yoksa bir önbellek hatası (cache miss), veriler hafızadan okunur ve bir kopyası önbellekte saklanır [16]. Tablo 2.2 bir önbellekte başlangıçta olmayan bir veri öğesinin istekten önce ve sonraki basit bir önbelleğini sunmaktadır. İstek öncesi, önbellek X_1, X_2, … X_{n-1} olan son referansları içerir ve işlemci önbellekte olmayan bir X_n kelimesi ister. Bu istek önbellek hatası (cache miss) olarak sonuçlanır ve X_n kelimesi hafızadan önbelleğe getirilir[14].

Tablo 2.2. Önbellekte başlangıçta bulunmayan bir X_n sözcüğüne hemen önce ve hemen sonra önbellek durumu(a,b)

a Xn referansından önce

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_3

b Xn referansından sonra

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_n
X_3

Bu istek, bellekten X_n'i getirmek ve önbelleğe eklemek önbelleği zorlar. Bundan dolayı bir önbellek hatası meydana gelir. Önbellek mimarileri, çalışma zamanı ve güç tüketimi gibi yan kanal bilgileri aracılığıyla şifrelerin önbellek vuruş veya hata (hits/miss) istatistikleri hakkında bilgi sızdırmaktadır. Önbellek davranış analizi anlamına gelen bu karakteristikler önbellek saldırılarının temelini oluşturmuştur. Bir şifreleme uygulaması sabit bir yürütme akışına sahip olsa bile, yani tüm açık metinler ve gizli anahtarlar aynı komut yapısında çalışsa da, yürütme sırasındaki önbellek davranışı programın çalışma süresinde varyasyonlara neden olur. Önbellek saldırıları, bu varyasyonlarda yararlanır ve gizli anahtarların ayrıntılı arama alanını daraltır [6]. Önbellek yan kanal saldırıları, yan kanal analiz saldırıları içerisinde geniş bir kriptografik analiz tekniği grubu olan Mikro Mimari Saldırı (MA) türündedir [17].

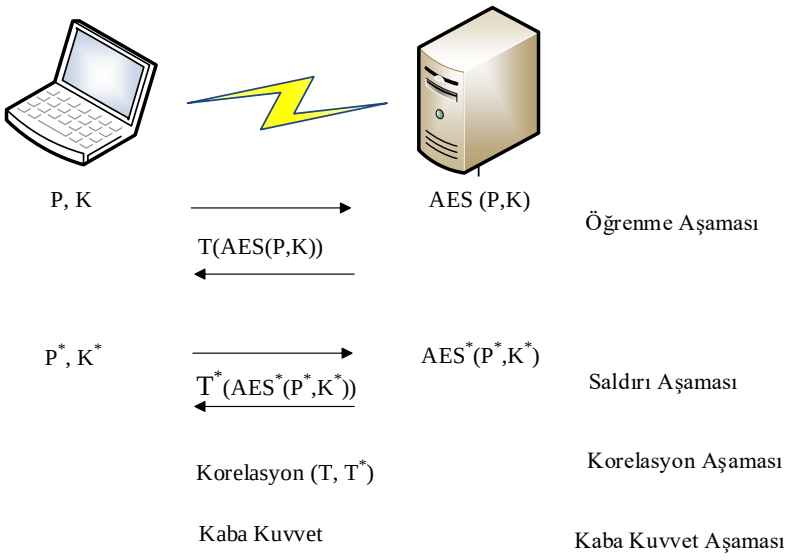
Teorik olarak önbellek saldırıları ilk olarak [13] 'de tanımlanmıştır. Burada zaman odaklı ve iz odaklı önbellek saldırıları olmak üzere iki çeşit önbellek saldırısı tanımlanmıştır. Daha sonra erişim odaklı yeni bir önbellek saldırı türü ortaya çıkmıştır. İz odaklı önbellek saldırılarında düşman, şifrelemelerinin bir örneği için önbellek vuruşlarının ve hatalarının izlerini alır ve bu verileri kullanarak bir kriptosistemin gizli anahtarını kurtarır. İz, bir dizi önbellek vuruşlarının ve hatalarının bir dizisi olarak tanımlanmaktadır. İz odaklı önbellek saldırıları için detaylar [18]'da verilmiştir. Erişim odaklı saldırılar, genel yürütme süresini değerlendirmek yerine önbellek davranışını daha ince ayrıntılarla inceler. Bu saldırıda gizli anahtarın yorumlanması için kriptografik operasyonlar boyunca bir

önbellek satırına erişilip erişilmediği konusundaki bilgi kullanılmaktadır. Erişim odaklı önbellek saldırıların detayları [19]'de verilmiştir. Gerçekleştirilen uygulamada zaman odaklı önbellek saldırısı kullanılmıştır.

2.1.2. Zaman Odaklı Önbellek Saldırıları (Time-Driven Cache Attacks)

Zaman odaklı önbellek saldırıları işlemci üzerinde bir algoritmanın çalışmasındaki zaman farklılıklarını analiz eder. Bu saldırı türünde bilgi sızıntısı olarak önbellek mimarisinin davranışı boyunca zaman farklılıkları kullanılmaktadır. Kriptografik işlemlerin çalışma zamanı önbellek davranışlarından önemli ölçüde etkilenmektedir. Özellikle verinin önbellekte olmaması sebebiyle oluşan önbellek hatası kriptografik işlemlerin çalışma zamanında önemli varyasyonlara neden olmaktadır. Bu varyasyonlar bir şifreleme algoritmasında kullanılan açık metinler ve gizli anahtarlar hakkında bilgi edinmede kullanılabilmektedir. Temeli bu mantığa dayanan zaman odaklı önbellek saldırılarının ilk örneği Tsunoo ve arkadaşları [2,3] tarafından DES (Data Encryption Standards) algoritmasına uygulanmıştır. Bu saldırı türünün önemli bir örneği de Bernstein'in [4] uyguladığı saldırıdır. Bu saldırı AES(Advanced Encryption Standards) 'in OPENSSL uygulaması tarafından kullanılan gizli anahtarı kurtarmak için rastgele oluşturulan mesajların şifreleme zamanlarındaki farklılıkları kullanır. Bernstein'in çalışmasıyla başka bir bilgisayardaki bir ağ sunucusundan AES anahtarının eksiksiz bir şekilde elde edildiği kanıtlanmıştır. Hedeflenen sunucu anahtarı yalnızca Pentium III üzerinde OpenSSL AES uygulamasıyla verileri şifrelemek için kullanılmıştır. Bersntein'in çalışmasında sadece Pentium III işlemciyle sınırlı kalınmamıştır. Bununla birlikte AMD Athlon, Inter Pentium M, IBM PowerPC RS64 IV ve Sun UltraSPARC III işlemciler üzerinde testler gerçekleştirilmiştir [4].

Bu saldırıda bir AES sunucusu ve bir kullanıcısı vardır. AES kullanıcısı, AES sunucusuna rastgele açık metinleri gönderir. AES sunucusu açık metinleri şifreler ve kullanıcıya cevap olarak şifreli metinleri ve şifreleme işlemlerinin zamanını verir. Kullanıcı gizli anahtarı yorumlamak için şifreleme işlemlerinin çalışma zamanını kullanır. Çalışma zamanından gizli şifreyi yorumlamak için korelasyon analizi kullanılır. Saldırı şeması Şekil 3.4 'te gösterilmiştir.



Şekil 2.3. Bernstein saldırı şeması

Şekil 2.3'de gösterildiği gibi saldırı dört aşamadan oluşmaktadır: öğrenme aşaması, saldırı aşaması, korelasyon analizi aşaması ve kaba kuvvet (brute force) saldırı aşaması. Bu aşamalar sonrasında genellikle korelasyon analizi sonucunda gizli anahtara ulaşılmıştır. Ancak korelasyon analizinin yetmediği, gizli anahtarın tamamının elde edilemediği durumlarda kaba kuvvet saldırısı kullanılmaktadır [20].

Öncelikle öğrenme aşamasında saldırgan K anahtarının bilinmesine ihtiyaç duyar. Kullanıcı tarafında P ile gösterilen rastgele açık metinler oluşturulur. Rastgele oluşturulan açık metinler şifrelenmek üzere sunucuya gönderilir. Sunucu tarafında gelen açık metinler bilinen bir anahtar ile (bu aşamada anahtar olarak 00..000 kullanılmıştır) şifrelenir ve aynı zamanda şifreleme süresi ölçülür. Kullanıcı, sunucudan rastgele açık metinlerin şifreli halini ve çalışma süresini alır. Şekil 2.3' de şifreleme işlemi AES(P, K), şifreleme süresi T(AES(P, K)) ile gösterilmektedir. Şifreleme süreci bitinceye kadar elde edilen verilerle bir zaman profili oluşturulur. Öğrenme aşaması için zaman bilgileri açık metin baytının P_i=b (i=0,…,255) olduğu ve bu açık metinlerin şifrelenmiş metinlerinin sayısı n[j][b] 'de, açık metinler için gözlemlenen tüm şifreleme zamanlarının toplam sayısı t_n[j][b]'de tutulmaktadır.

Öğrenme aşaması bittikten sonra saldırı aşaması başlatılır. Saldırı aşamasında da sunucuya kullanıcı tarafından P^* rastgele metinleri gönderilir. Sunucu tarafında gelen rastgele metinler bilinmeyen (rastgele oluşturulan) bir K^* anahtarı ile şifrelenir ve $T^*(AES^*(P^*, K^*))$ çalışma süresi ölçülür. Cevap olarak kullanıcıya şifreli metinler ve

zaman bilgisi verilir. Yine saldırı aşamasında da elde edilen verilerle zaman profili oluşturulur. Saldırı aşamasının zaman bilgileri öğrenme aşamasıyla aynı şekilde t_n^*[j][b] ve n^*[j][b]'de tutulmaktadır. Öğrenme ve saldırı aşamasında elde edilen zaman profillerinin korelasyon aşamasında analizi yapılmaktadır. Bu aşamada öncelikle elde edilen zaman bilgileri ile öğrenme aşaması ve saldırı aşamasının Denklem 3.1 kullanılarak bir zaman imzası oluşturulmaktadır.

$$x[j][b] = \frac{\text{tn}\,[j][b]}{n[j][b]} - \frac{\sum_j \sum_b \text{tn}\ [\text{j}][\text{b}]}{\sum_j \sum_b \text{n}\ [\text{j}][\text{b}]} \tag{3.1}$$

Her iki aşama için zaman imzası oluşturulduktan sonra korelasyon Denklem 3.2' de gösterildiği şekilde hesaplanır.

$$c[j][b] = \sum_{i=0}^{255} \text{x[j][i]} * \text{y[j][i} \oplus \text{b]} \tag{3.2}$$

Korelasyonlar azalan bir sıraya göre sıralanır ve önceden tanımlanmış bir eşik temel alınarak saldırgan her anahtar baytı K_i için potansiyel değerlerin bir listesini elde eder [21]. Gerçekleştirilen korelasyon analizi sonucunda bir anahtar uzayı elde edilmektedir. Bu anahtar uzayının bir örneği Şekil 2.4' te gösterilmiştir [4].

```
 24  0    e3 e2 e5 e4 e6 d3 e0 d6 d2 e7 e1 d1 d5 d4 d7 d0 ...
 16  1    72 77 76 75 71 70 74 73 99 98 9a 9e 9b 9d 9c 9f
  8  2    81 87 83 86 84 85 82 80
  1  3    a9
  4  4    8b 8a 89 88
  8  5    b4 b1 b0 b2 b5 b7 b6 b3
 31  6    66 67 62 65 64 60 63 61 ed e9 ef ec e8 eb ee ea ...
  8  7    a7 a0 a1 a5 a2 a6 a4 a3
  8  8    b9 bb b8 ba be bf bd bc
  8  9    7f 7e 7c 7d 79 78 7a 7b
  8 10    d3 d6 d1 d5 d4 d2 d7 d0
  8 11    3a 3f 3d 3b 39 3c 3e 38
176 12    ef ec eb e9 ea ed ee e8 3f 39 3b 0b 3c 0c 0f 38 ...
 16 13    ec e8 ee ea eb e9 ef ed 94 96 91 95 90 97 92 93
  8 14    3c 3d 3f 38 39 3e 3b 3a
  1 15    35
```

Şekil 2.4. Korelasyon sonuçları

Korelasyon analizi hesaplamalarının sonucunda yüksek korelasyona sahip olan anahtar muhtemelen aday anahtardır. Satırlar sırasıyla bayt, bayt sayısı ve olası değerlerin ayrıntılı listesi için kalan olası değerlerin sayısını verir. Örneğin Şekil 2.4'e göre gizli anahtarın 3. Baytı hexadecimal olarak a9'dur.

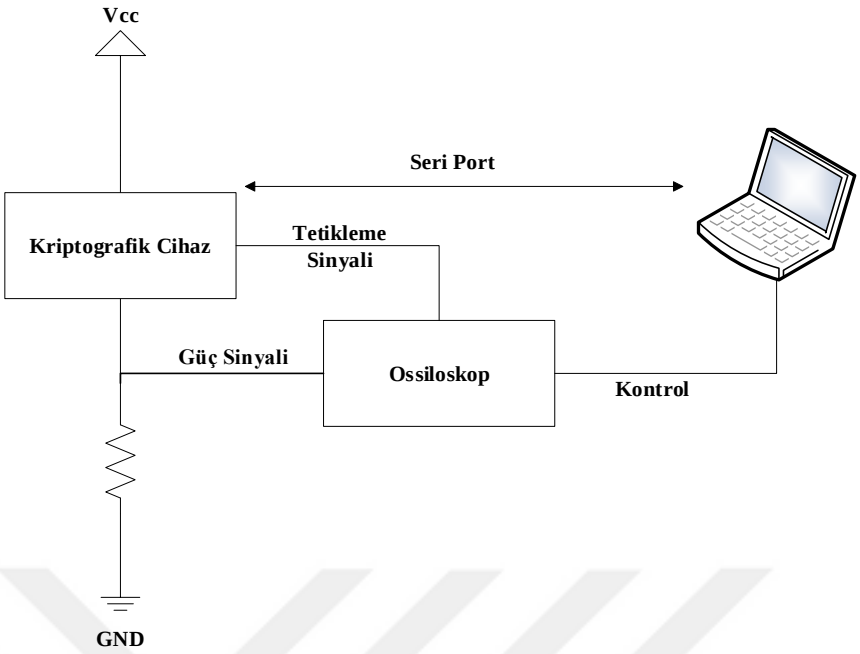
Bernstein'in bahsedilen zaman odaklı önbellek saldırısı literatürde gerçekleştirilen zaman saldırılarında önemli bir referans olmuştur. Örneğin [22] ' de Atıcı ve arkadaşları öğrenme aşaması olmadan da saldırının gerçekleştirileceğini ispatlamışlardır. Ayrıca AES'in son döngüsüne odaklanarak da saldırı gerçekleştirilmiştir. Bu tezde de AES'in son döngüsüne uygulanan Bernstein saldırısı referans alınmıştır. Ayrıca bulut tabanlı birçok çalışmada da Bernstein saldırısı önemli bir kaynak teşkil etmektedir. Algoritmadan bağımsız olarak sadece önbellek yapısı ve zaman varyasyonları kullanıldığı için farklı algoritmalara da uyarlanabilir.

2.2. Güç Analizi Saldırıları

Paul Kocher ve arkadaşları [1] 1999 yılında birçok güç analizi saldırıları geliştirmişlerdir. Bu saldırıya göre bir mikroişlemci genellikle elektromanyetik sızıntı, elektrik tüketimi ve zamanlama davranışı gibi yan kanal bilgileri üretmektedir. Güç analizi, cihazdan sızan güç yan kanalı kullanılarak gerçekleştirilir.

Güç analizi saldırıları, bir şifreleme aygıtının anlık güç tüketimi ile işlediği veriler veya gerçekleştirdiği işlem arasındaki bağımlılığı kullanır [23].

Bilindiği gibi işlemciler transistörlerden oluşmaktadır. Transistörler anahtar gibi davranmaktadır. Eğer anahtar 0 ise akım bloke olur. 1 ise akım oluşur. Bu da transistörün daha fazla enerji harcaması anlamına gelir. Güç analizi ile bu akım farklarından yararlanarak cihazın güç tüketimi izlenir.Güç analizi sayesinde cihazın güç tüketimi ile gizli bilgi ve yapılan işlemler arasında bir ilişki kurularak gizli bilgi elde edilmeye çalışılır. Bunu gerçekleştirmek için öncelikle cihazın güç tüketiminin ölçülmesi gerekir. Bu amaçla, devre ile kaynak arasındaki hat üzerinde küçük değerli bir direnç yerleştirilir ve bu direncin her iki ucundaki gerilim farklılıklarından yararlanarak çekilen akım bilgisi elde edilir [12]. Ölçüm düzeneği oldukça basittir ve maliyeti azdır. Bu nedenle en çok kullanılan yan kanal analiz tekniğidir. Ölçüm düzeneği Şekil 2.5' te gösterilmektedir.



Şekil 2.5. Güç tüketimini izlemek için ölçüm düzeneği

Ölçüm düzeneği, ölçülen güç izlerini analiz eden ve yeni ölçümleri tetikleyen bir bilgisayar ve kriptografik cihaz tarafından tüketilen güçten örnek alan bir ossiloskoptan oluşmaktadır. Ossiloskop ikinci bir kanal tetiklemesi için kullanılabilir [24]. Güç analizi saldırıları iki çeşittir: Basit güç analizi ve diferansiyel güç analizi.

2.2.1. Basit Güç Analizi (SPA: Simple Power Analysıs)

Basit güç analizi saldırılarında, şifreleme cihazı çalışırken doğrudan tek bir güç tüketiminin izleri incelenmektedir. Bununla birlikte saldırgan donanım hakkında ve uygulanan özel algoritma hakkında ayrıntılı bilgi sahibi olmalıdır[24]. Çünkü elde edilen ölçümler gözle incelenerek yorumlanır. Yürütülen işlemlerle güç tüketimi izleri arasında ilişki kurulmaya çalışır. Bu şekilde cihazın işleyişi, cihazda hangi algoritmanın çalıştırıldığı ve hatta kullanılan gizli anahtar bilgisine ulaşılır [12].

SPA ölçümü güçlü donanımlara bağlıdır. Ayrıca gözlemlenen komutun doğru noktası (gerçek zamanı) bilinmelidir. SPA'da genellikle bilgi sızması iki türdür: Hamming ağırlığı sızıntısı ve geçiş sayısı sızıntısı [24].

Önceden şarj edilen bir veri yolu tasarımında, veri yolu üzerine doğrudan yönlendirilen 1'lerin sayısı, kapılardan deşarj edilen akım miktarı ile doğru orantılıdır. Böylece veri yolu üzerindeki Hamming ağırlığı belirlemek mümkündür [24].

Geçiş sayısı bilgi sızıntıları, veri yolu ile yönlendirilen kapıların akım vermesine bağlıdır. Bir kapı girişi sırasında yüksekten düşüğe (1'den 0'a) veya düşükten yükseğe (0'dan 1'e) kısa süreli bir akım gerçekleşecektir. Bu akım geçişi iki kısımdan oluşmaktadır. Büyük bir kısmı başarılı olan kapıların ve yayını bozan dirençlerin şarj/deşarjından kaynaklanır. Diğer kısmı ise V_{dd} ile V_{ss} arasındaki dinamik kısa devre akımından kaynaklanır. Teorik olarak bu yüksekten düşüğe ve düşükten yükseğe bir çıkış durumunun değişikliği ayırt etmek mümkündür [24].

2.2.2. Diferansiyel Güç Analizi (DPA: Differantial Power Analysıs)

Diferansiyel güç analizi, bir güç izi koleksiyonu üzerinde istatistiksel yöntemler kullanan gelişmiş güç analizi tekniğidir. DPA saldırıları akıllı kart teknolojileri için büyük bir tehdit oluşturmaktadır.

Bu saldırı yönteminde gürültüyü filtreleyebilmek için çok sayıda ölçüm yapılır. Hata düzeltme algoritmaları kullanılarak saldırı için gerekli örnek sayısı azaltılır. DPA'nın uygulanabilmesi için gerekli koşullar, algoritma içerisinde bir veya birden fazla ara değerin, az sayıda anahtar biti ve bilinen giriş veya çıkış verisiyle ifade edilmesi ya da en azından ilişkili olmasıdır. Birçok algoritma bu koşulu sağlamaktadır [12].

DPA saldırılarının uygulanması SPA saldırılarına göre daha zordur, ancak DPA saldırıları daha güçlüdür. DPA saldırılarına önlem alınması daha zordur. Ayrıca, donanım analizi ve şifreleme komutları ile ilgili ayrıntılı bilgi gerektirmez. Sinyal işleme ve istatistik konularında ayrıntılı bilgi gerektirir [24].

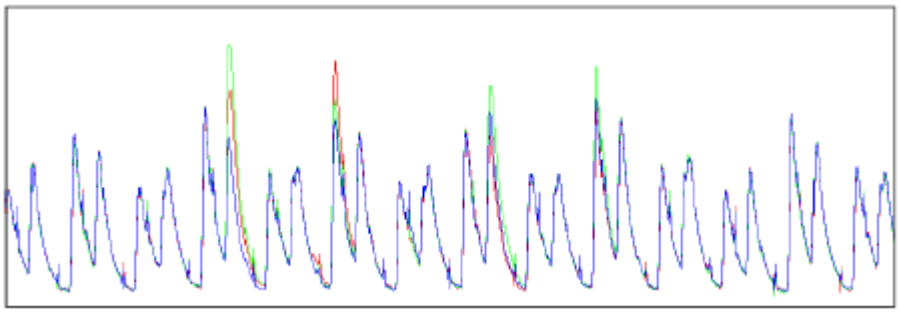
DPA rastgele metinlerle kriptografik cihaz üzerinde güç ölçümleri yaparak şifreyi çözmeye çalışır. Yapılan çalışmalarda rastgele metinler arasından seçim ve tahmin yaparak, seçimlerine göre güç izlerindeki değişim gözlenmiştir ve güç izlerinde en yüksek tepe oluşuyorsa tahminin başarılı olduğu sonucuna varılmıştır.

N düz veya şifreli rastgele metinler olsun. N'e ilişkin güç tüketim grafiği Şekil 2.6'da gösterilmiştir [25].

00 B688EE57BB63E03E

01 125D04D77509F36F

02 C031A0392DC881E6…



Şekil 2.6. N mesajlarına ilişkin güç tüketimi

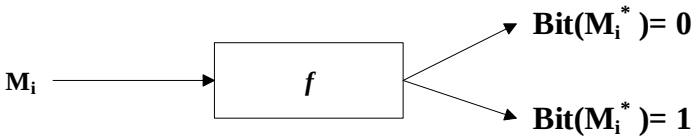
Rastgele bir mesajın bilinen bir deterministik *f* fonksiyonu (transfer fonksiyonu, permütasyon gibi) tarafından işlendiği varsayılır. Şekil 2.7'de gösterildiği gibi bilinen mesaj, *f*'in görüntüsü sayesinde yeniden hesaplanabilmektedir [5].

Şekil 2.7. Mesajın işlenmesi

Mesajın işlenmesinden sonra *M'*deki bitler (*M* bufferında) arasından bir tek bit seçilir. *M*'deki bitlerin varyasyonlarında doğru olan tahmin edilebilir [25].

i	Mesaj	bit		
0	B688EE57BB63E03E	1		
1	185D04D77509F36F	0		
2	C031A0392DC881E6	1	….	for i = 0, N-1

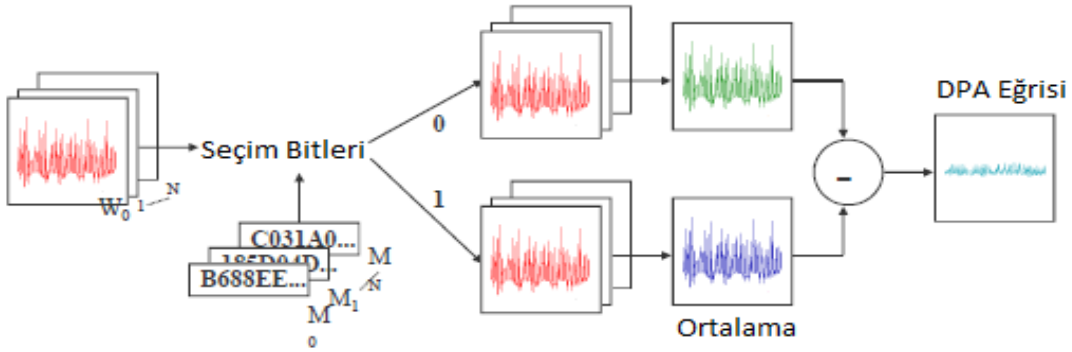
Bit değeri seçimine göre Şekil 2.8'deki gibi iki paket halinde (0 paketi ve 1 paketi olarak) mesajlar ve buna bağlı eğriler ayrılır ve 0 paketine -1 atanır ve 1 paketine +1 atanır [25].



Şekil 2.8. Mesajların ayrılması

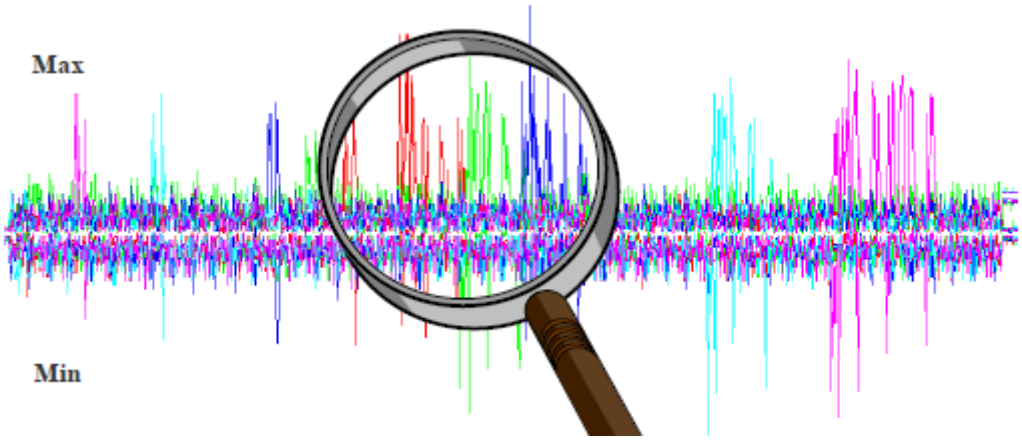
0	B688EE57BB63E03E	1	+1	
1	185D04D77509F36F	0	-1	
2	C031A0392DC881E6	1	+1	… for i = 0, N-1

Daha sonra güç tüketim eğrileri toplanır ve normalize edilir. DPA eğrisinin yapısı Şekil 2.9'te gösterilmiştir [25].



Şekil 2.9. DPA eğrisi oluşumu

Farklı seçim bitleri için DPA eğrisi Şekil 2.10'de gösterilmiştir.



Şekil 2.10. DPA Eğrisi

Seçim bitleri işlendiği zaman tepe yükselmektedir.

3. MAKİNE ÖĞRENMESİ İLE SALDIRI SÜRECİNİN SAPTANMASI

Yan kanal saldırılarının önlenmesi amacıyla son yıllarda geliştirilen yöntemlerden biri makine öğrenmesi ile saldırı sürecinin saptanmasıdır. Makine öğrenmesinin yan kanal saldırılarına uygulanması için saldırı yapılırken şifreleme sisteminin davranışları gözlemlenmektedir. Bu gözlemler önbellek saldırıları için önbellek davranışlarından oluşmaktadır. Makine öğrenmesi için kullanılan özellikler önbellekteki vuruşların sayısı veya önbellek kaçışlarının sayısı, önbelleğin performans verileri olabilmektedir. Bu veriler öğrenmeye girerek bundan sonraki gerçekleşen durumların saldırı süreci olup olmadığı belirlenebilmektedir. Bu kapsamda gerçekleştirilen önemli bir çalışma [7] ve [8]' da verilmiştir. Bu çalışmada önbelleğin L3 seviyesinde gerçekleşen flush-reload [26] saldırısına 3 farklı yöntem kullanılarak saldırı süreci tespit edilmiştir. Marco'nun tezinde [27] kullanılan yöntemlerden ikisi makine öğrenmesi tabanlı diğeri ise korelasyon tabanlıdır. Korelasyon tabanlı yaklaşım donanım performans ölçüm sayaçları kullanılarak toplanan veriler analiz edilerek kurban ve saldırgan arasında bir korelasyon bulmaya dayalıdır. Makine öğrenmesi yöntemlerinden ise anormallik tespit yöntemi ve denetimli öğrenme yöntemlerinden biri olan yapay sinir ağları kullanılmıştır. Bu çalışma referans alınarak anormallik tespiti yöntemi gerçekleştirilen saldırılara uygulanmıştır. Uygulanan saldırılarla ilgili sonuçlar Bölüm 5' te ve Bölüm 6' da verilmiştir. Bu bölümde gerçekleştirilen tespit yöntemleri ve bu yöntemler için kullanılan veriler açıklanmıştır. Zaman odaklı önbellek saldırısında anormallik tespiti için donanım performans sayacı aracılığıyla ölçülen veriler kullanılmıştır. Güç saldırısı için sistemin tükettiği güç enerjisi eğitim verisi olarak anormallik tespitinde kullanılmıştır.

3.1. Donanım Performans Sayaçları

Yüksek performanslı çoğu modern işlemciler işlemcinin performansını izleyen özel çipli donanıma sahiptir. Bu donanım tarafından toplanan veriler, uygulamalar, işletim sistemi ve işlemci hakkında performans bilgisi sağlar. Performans olayları program karakterizasyonu (komut yapısına bağlı olarak yüklemeler, depolamalar, dallanmalar vb.), hafıza erişimi, dallanma tahmini, kaynakların kullanımı ve pipeline yapısı olmak üzere 5 gruba ayrılmaktadır [27]. Önbellek saldırılarında performans olaylarından hafıza erişimi

kullanılmaktadır. Önbellek saldırıları genellikle önbellekte tutulan ve kaçan verilerin oluşturduğu varyasyonlara dayandığı için önbellekteki performans ölçümleri saldırı hakkında bilgi vermektedir.

Donanım performans sayaçları için Linux çekirdeği perf adı verilen özel bir araç kullanmaktadır. Perf, Linux işletim sistemlerinde performans ölçümlerinde donanımsal farklılıkları özetleyen bir komut satırı ara yüzüdür. Donanım olayları, yazılım olayları, donanım önbellek olayları, izleme noktası olayları gibi performansların değerlendirilmesini sağlamaktadır. Linux çekirdeğinin son sürümleri tarafından sunulan perf_events ara yüzüne dayanmaktadır [28].

Linux çekirdeğinde komut satırına perf list yazıldığında perf ara yüzü ile elde edilebilecek donanım performanslarının bir kısmı Şekil 3.1'deki gibi listelenmektedir.

```
perf list

List of pre-defined events (to be used in -e):

 cpu-cycles OR cycles                              [Hardware event]
 instructions                                      [Hardware event]
 cache-references                                  [Hardware event]
 cache-misses                                      [Hardware event]
 branch-instructions OR branches                   [Hardware event]
 branch-misses                                     [Hardware event]
 bus-cycles                                        [Hardware event]

 cpu-clock                                         [Software event]
 task-clock                                        [Software event]
 page-faults OR faults                             [Software event]
 minor-faults                                      [Software event]
 major-faults                                      [Software event]
 context-switches OR cs                            [Software event]
 cpu-migrations OR migrations                      [Software event]
 alignment-faults                                  [Software event]
 emulation-faults                                  [Software event]

 L1-dcache-loads                                   [Hardware cache event]
 L1-dcache-load-misses                             [Hardware cache event]
 L1-dcache-stores                                  [Hardware cache event]
 L1-dcache-store-misses                            [Hardware cache event]
 L1-dcache-prefetches                              [Hardware cache event]
 L1-dcache-prefetch-misses                         [Hardware cache event]
 L1-icache-loads                                   [Hardware cache event]
 L1-icache-load-misses                             [Hardware cache event]
 L1-icache-prefetches                              [Hardware cache event]
 L1-icache-prefetch-misses                         [Hardware cache event]
 LLC-loads                                         [Hardware cache event]
```

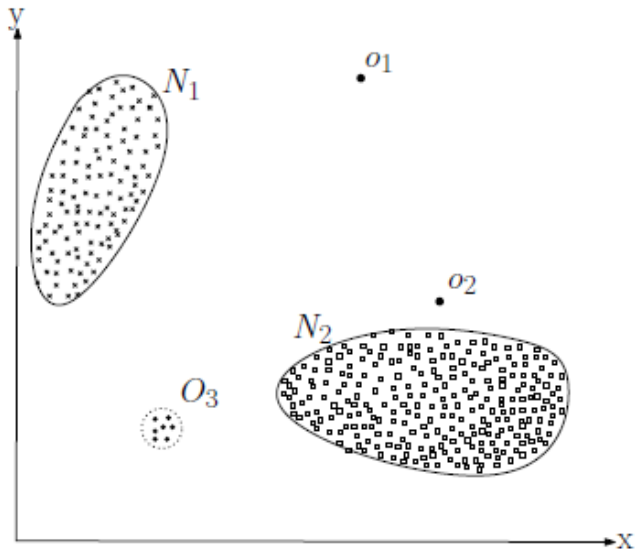
Şekil 3.1. perf-events arayüzü ile elde edilebilecek performans olayları

Perf aracının en önemli komutlarından biri perf-stat komutudur. Bu komut performans sayacı istatistiklerini toplar. Elde edilen istatistikleri kaydetmek için de perf-stat record

komutundan yararlanılmaktadır [29]. Gerçekleştirilen saldırı sürecinde perf aracılığı ile performans verilerinin elde edilmesi Bölüm 4.4'de gösterilmiştir.

3.2. Anormallik Tespiti Yöntemi

Anormallik tespiti, beklenen davranışa uymayan verilerdeki şablonları bulma problemine dayanan makine öğrenmesi tekniklerinden bir yöntemdir. Bahsedilen uygun olmayan şablonlar genel olarak anormaller, aykırı değerler, uyuşmayan gözlemler, istisnalar, sapmalar gibi terimlerle ifade edilmektedir. Anormaller iyi tanımlanmış bir normal davranış kavramına uymayan verilerdeki şablonlardır. Şekil 3.2'de iki boyutlu bir veri setindeki anormallerin örneği gösterilmektedir [30].



Şekil 3.2. Anormallerin bir örneği [30]

Şekil 3.2'de gösterilen x ve y değerlerinin bir araya toplandığı bölgeler N1 ve N2 iki tane normal bölgeyi temsil etmektedir. Bu bölgelerden uzak olan değerler olan O1, O2 ve O3 anormal değerleri temsil etmektedir.

Anormallik tespiti kullanarak, casustan gelen veri örnekleri normal ve diğer işlemlerden gelen veri örnekleri anormaller olarak ele alınabilir.[7] ve [8]'da belirtilen çalışmalara göre saldırı aşamasını tespit etmede makine öğrenmesi tekniklerinin kullanımı daha güvenilir olmaktadır. Makine öğrenmesi yöntemiyle saldırı aşaması gibi görünen ancak saldırı aşamasına ait olmayan veriler olabilmektedir. Bu şekilde elde edilen gürültülü

sonuçların önüne geçilmesi için makine öğrenmesi tekniklerinden yararlanılmaktadır. Anormallik tespiti yönteminde sıklıkla kullanılan makine öğrenmesi yöntemlerinden biri kümelemedir. Kümeleme benzer özellikte olan objelerin gruplanması olarak adlandırılabilir. Yan kanal saldırılarının tespit edilmesi için kümelemenin kullanılması da bu mantığa dayanmaktadır. Aynı kümeye ait olan süreç normal olarak ve kümenin dışında olan süreç anormal olarak etiketlenerek saldırı aşamasının hangi kümeye ait olduğu bulunabilmektedir. Kullanılan saldırıya bağlı olarak anormallik tespiti yönteminde kullanılacak veri kümesi değişiklik göstermektedir. Gerçekleştirilen çalışmada önbellek saldırı için L1 önbelleği hücresinde meydana gelen olaylar dikkate alındığından PID, L1 cache loads, L1 cache misses öznitelikleri kullanılmıştır. Gerçekleştirilen güç analizi saldırısında ise klasik RSA algoritması ve Çin Kalan Teoremi kullanılarak gerçekleştirilen sayısal imzaların güç tüketimi öznitelik olarak kullanılmıştır.

Kümeleme algoritmalarından en yaygın olarak kullanılan K-ortalama kümeleme yöntemi kullanılmıştır. K-ortalama kümeleme algoritması 1975 yılında Hartigan [31] tarafından detaylı olarak tarif edilmiştir.

3.2.1. K-Ortalama Kümeleme Algoritması

K-ortalama algoritması belirli bir veri kümesini belirli sayıdaki kümeler üzerinden gruplara ayıran bir algoritmadır. Bu algoritmanın temelinde yatan fikir gruplara ayrılan her küme için bir tane küme merkezi tanımlamaktır. Daha sonra her bir nokta belirli bir veri kümesine alınır ve o nokta en yakın merkeze ilişkilendirilir. Bekleyen bir nokta yoksa aşama tamamlanır ve gruplama yapılır. Optimum merkez noktalar buluncaya kadar devam edilir. Kümelemede kaç grup oluşacağı verinin içindeki nesnelerin birbirine benzerlik derecesine göre değişiklik gösterir. Farklı kümelerin oluşması için nesnelerin tamamen birbirine benzememesi çok fazla da birbirinden ayrı olmaması gerekir.

Hartigan tarafından detaylı bir şekilde tarif edilen K-ortama kümeleme algoritması N boyutlarında i noktalarının bir matrisini ve N boyutundaki K başlangıç küme merkezlerinin bir matrisini girdi olarak gerektirir. Genel prosedür, noktaları bir kümeden diğerine taşımak suretiyle, yerel olarak optimal küme içinde bir K-bölümü aramaktır [32].

K verisi ilk küme merkezi olarak rastgele seçilmiştir, geri kalan veriler küme merkezine olan mesafesine göre en yüksek benzerliğe sahip kümeye eklenir ve daha sonra her kümenin merkezini yeniden hesaplar. Her küme merkezi değişmedikçe bu adımlar

tekrarlanır. Bu adımların sonunda elde edilen kümenin dışında kalan değerler anormal olarak değerlendirilmektedir.

4. AES'İN SON DÖNGÜSÜNE ZAMAN ODAKLI ÖNBELLEK SALDIRISI UYGULANMASI VE ANORMALLİK TESPİTİ

4.1 AES Algoritmasının Tanımı

Belçikalı kriptograflar J.Daemen ve V.Rijmen [33] tarafından geliştirilen AES genellikle Rijndael simetrik bir blok şifrelemesi olarak tanımlanır. AES algoritması 2001'de NIST tarafından resmi olarak tanıtılmıştır [34].

Rijndael blok ve anahtar boyutu olarak 128, 192 ve 256 bit olmak üzere 3 farklı boyut kullanır. Ancak AES standardı için sadece 128 bit anahtar boyutu gereklidir [35]. AES algoritması giriş ve çıkış değerleri olarak 128 bittir. 128 bitlik giriş ve çıkış değerleri 0'lar ve 1'lerden oluşan bir dizidir. Bu diziye blok adı verilir. Anahtar boyutu ise 128, 192, 256 bit olabilmektedir Simetrik şifrelemede mesajı gönderecek olan kişi açık metni gizli anahtar ile şifreler ve şifreli metni alıcıya gönderir. Alıcı ise kendisine gelen şifreli metni aynı anahtarla deşifreler ve açık metni elde eder. Blok şifrelemede ise aynı anahtarla açık metnin bütün bir bloğu şifrelenir. Açık metin bitlere ayrılmaz bir blok olarak şifreleme algoritması uygulanır.

AES algoritması anahtar boyutuna göre farklı sayıda gerçekleşen döngü aşamalarından oluşmuştur. 128 bitlik anahtar için 10 döngüde, 192 bitlik anahtar için 12 döngüde, 256 bitlik anahtar için 14 döngüde şifreleme gerçekleşmektedir. AES'de döngü işlemi dört farklı transformasyon işleminden oluşmaktadır. Transformasyon işlemlerinden önce açık metin durum matrisi adı verilen 4x4 boyutunda bir dizi olarak ifade edilir. Bu matrisin her bir elemanı açık metnin bir baytına karşılık gelir [33].

Bayt Değiştirme (SubBytes-S-Box) : Bayt değiştirme adımı şifrelemenin doğrusal olmayan bir adımıdır. S-box olarak adlandırılan bir değiştirme tablosu kullanılarak durum matrisinin her bir baytı üzerinde bağımsız olarak yer değiştirme işlemi yapılır.

Satır Kaydırma (ShiftRows) : Durum matrisindeki byte'lar çevrimsel olarak kaydırılır. İlk satır döndürülmez. İkinci satır bir sola kaydırılır, üçüncü satır iki sola kaydırılır ve dördüncü satır üç sola kaydırılır.

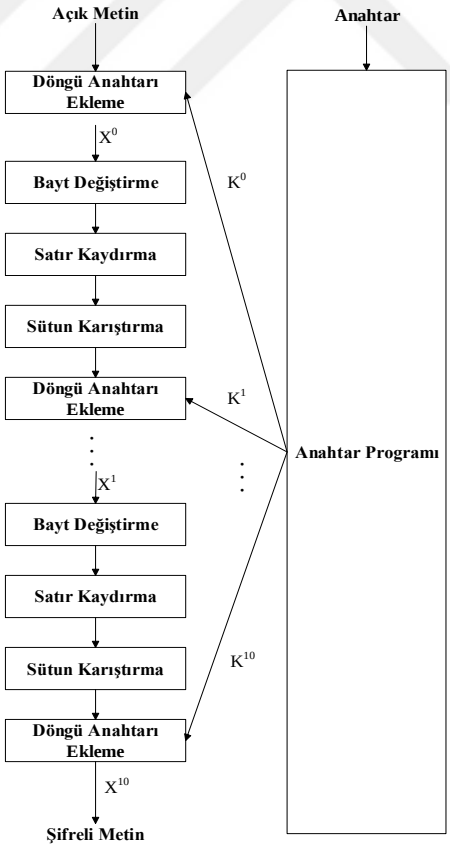
Sütun Karıştırma (MixColoumns) : Durum matrisinin her sütunu 4 byte'lık bloklarla karıştırılır.

Döngü Anahtarı Ekleme (AddRoundKey) : Durum matrisine 4x4 (128 bit) anahtar matrisi eklenmesi işlemi gerçekleştirilir.

AES algoritması için tanımı yapılan bu transformasyon işlemleri sonlu elemanlar olarak yorumlanır [35]. Sonlu elemanlar eklenebilir ve çoğaltılabilir, ancak bu işlemler sayılar için kullanılanlardan farklıdır.

4.2. AES Algoritmasının İç Yapısı

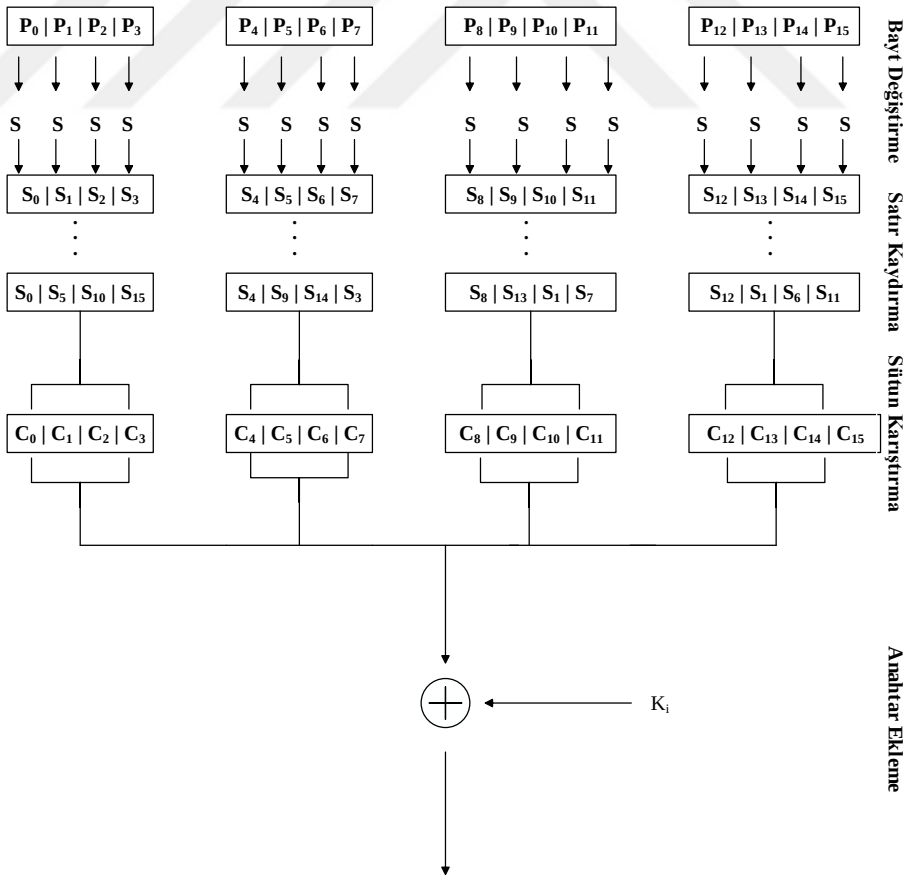
AES algoritması anahtar kaydırma ve döngü transformasyonlarından oluşan bir anahtar yinelemeli blok şifreleme algoritmasıdır. Şekil 4.1'de algoritmanın transformasyon işlemleri ve anahtar kaydırma adımını gösteren genel akış şeması verilmiştir. Uygulanan AES'de 192 bitlik anahtar kullanılmıştır ve toplamda 10 döngü sonunda şifreli metin elde edilmiştir. Şekil 4.1'ün sol kısmında gösterildiği gibi her döngüde 4 aşamalı transformasyon işlemleri gerçekleştirilmektedir.



Şekil 4.1. AES Şifreleme Akış Şeması

Şekil 4.1'in sağ tarafı ardışık döngülerde kullanılacak anahtarı elde etmeyi temsil etmektedir. K^0,..., K^{10} elde dilen döngü anahtarlarını temsil etmektedir. Şekilde ifade edilen X^0 ... X^{10} değerleri ise her bir döngüde elde edilen durum matrisini ifade etmektedir. Her döngüde durum matrisine Bayt Değiştirme, Satır Kaydırma, Sütun Karıştırma ve Döngü Anahtarı Ekleme adı verilen transformasyon işlemleri uygulanmaktadır. Gerçekleştirilen bu işlemler bir sonraki bölümde detaylandırılmıştır.

Şekil 4.2'de tek bir AES döngüsünün diyagramı gösterilmektedir. 16 baytlık giriş P_0 ,. . . , P_{15} bayt cinsinden S-Box'a beslenir. S-Box birden çok bayt değiştirme dönüşümünde ve bir bayt değerinin bire bir değiştirilmesini gerçekleştirmek için anahtar genişletme yordamında kullanılan doğrusal olmayan bir yer değiştirme tablosudur [35]. 16 bayt S_0 , . . . , S_{15} çıktısı satır kaydırma katmanında bayt cinsinden değiştirilir ve sütun karıştırma dönüşümü olan C (x) tarafından karıştırılır. Sonunda, ara sonuç ile 128-bit alt anahtar XOR işlemi uygulanır. Bu işlemler döngü sayısına bağlı olarak her döngüde gerçekleştirilir. Son döngünün çıktısı olarak şifreli metin elde edilir.



Şekil 4.2. AES döngü yapısı

Yazılım uygulamalarında, S-Box genellikle Tablo 4.1'de verilen sabit girdileri olan bir arama tablosudur. S-Box yapısı, bir kısmi haritalandırmadır; diğer bir deyişle, 2^8 = 256 olası girdi elemanlarının her biri bir çıktı öğesine bire bir eşlenmiştir [35].

Tablo 4.1. AES- S-Box: ij baytı için hexadecimal formatta yer değiştirme değerleri

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

Tablo 4.1'de gösterilen S-box arama tablosuna göre, örneğin S-box 'a giriş değeri olarak $P_{i,j}$ = (e5) olduğu farzedilirse, tabloda e5'e karşılık gelen d9 ile yer değiştirme işlemi yapılır.(Tabloda satırlar i baytını sütunlar ise j baytını temsil etmektedir). Kısaca aşağıdaki gibi ifade edilebilir:

$$S((e5)_{hex}) = (d9)_{hex} \quad (4.1)$$

4.3 OpenSSL ile AES Şifreleme

OpenSSL, AES algoritmasının işlevselliklerini sağlayan önemli bir açık kaynaklı yazılım kütüphanesidir [36]. Gerçekleştirilen zaman odaklı önbellek saldırısında bu kütüphane (OpenSSL v0.9.7a Feb 19, 2003) ile uygulanan özel AES'e odaklanılmıştır. AES, 16 baytlık bir anahtar, sabit bir 256 baytlık tablo S = (99, 124, 119, 123, 242 …)

kullanarak 16 baytlık bir girişi karıştırır ve başka bir sabit 256 baytlık tablo S' = (198, 248, 238, 246, 255 …)'dir. Bu iki 256 baytlık tablo, T_0, T_1, T_2, T_3 tarafından tanımlanan dört 1024 baytlık tabloya genişletilir. Bu arama tabloları AES algoritmasında S-box yapısının gerçekleştirildiği tablolardır.

$$T_0[b] = (S'[b], S[b], S[b], S[b] \oplus S'[b]),$$
$$T_1[b] = (S[b] \oplus S'[b], S'[b], S[b], S[b]),$$
$$T_2[b] = (S[b], S[b] \oplus S'[b], S'[b], S[b]),$$
$$T_3[b] = (S[b], S[b], S[b] \oplus S'[b], S'[b]). \quad (4.2)$$

AES, iki adet 16 baytlık yardımcı dizi olan x ve y ile çalışır. İlk dizi, k için, ikinci dizi n⊕k için başlatılır. Öncelikle x, 4 baytlık dizi olarak x_0, x_1, x_2, x_3 şekilde değiştirilir. 4 baytlık e dizisi aşağıdaki gibi hesaplanır.

$$e = (S[x_3[1]] \oplus 1, S[x_3[2]], S[x_3[3]], S[x_3[0]]) \quad (4.3)$$

Daha sonra (x_0, x_1, x_2, x_3) ile (e⊕x_0, e⊕x_0⊕x_1, e⊕x_0⊕x_1⊕x_2, e⊕x_0⊕x_1⊕x_2⊕x_3) yer değiştirir. Sonrasında y, 4 baytlık dizi olarak (y_0, y_1, y_2, y_3) şekilde değiştirilir. (y_0, y_1, y_2, y_3) ile aşağıdaki dizi yer değiştirir.

$$(T_0[y_0[0]] \oplus T_1[y_1[1]] \oplus T_2[y_2[2]] \oplus T_3[y_3[3]] \oplus x_0,$$
$$T_0[y_1[0]] \oplus T_1[y_2[1]] \oplus T_2[y_3[2]] \oplus T_3[y_0[3]] \oplus x_1,$$
$$T_0[y_2[0]] \oplus T_1[y_3[1]] \oplus T_2[y_0[2]] \oplus T_3[y_1[3]] \oplus x_2,$$
$$T_0[y_0[0]] \oplus T_1[y_1[1]] \oplus T_2[y_2[2]] \oplus T_3[y_3[3]] \oplus x_3) \quad (4.4)$$

AES, ⊕1 yerine ⊕2 kullanarak x 'i yeniden değiştirir, sonra y'yi tekrar değiştirir. Daha sonra ⊕4 kullanarak tekrar x'i değiştirir ve tekrar y'yi değiştirir. Bu işlemler 10 döngü boyunca devam eder. X modifikasyonunun sabit değerleri 1, 2, 4, 8, 16, 32, 64, 128, 27, 54'tür. Onuncu döngüde Y'nin modifikasyonu T_0[]⊕T_1[]⊕T_2[]⊕T_3[]' dan ziyade (S[], S[], S[], S[])'i kullanır. Y'nin son değeri AES k (n) çıktısıdır [4].

4.4. AES'in Son Döngüsüne Zaman Odaklı Önbellek Saldırısı Uygulanması ve Anormallik Tespiti

Bölüm 3.2'de bahsedilen Bernstein' in yapmış olduğu zaman odaklı önbellek saldırısında OpenSSL ile gerçekleştirilen AES algoritmasının ilk döngüsüne odaklanılmıştır. Bu saldırı örnek alınarak Atıcı tarafından AES algoritmasının son döngüsünde Bernstein saldırısı uygulanmıştır. Bu tezde de Atıcı' nın uygulamış olduğu son döngüye odaklanan zaman odaklı önbellek saldırısı gerçekleştirilerek bu saldırıya anormallik tespiti yapılmıştır. OPENSSL ile AES algoritmasının son döngüsünde temel olarak AES S-box operasyonunu uygulayan tablo T_4 kullanılır. Bu tabloya erişim sırasında ortaya çıkan önbellek vuruşları ve hataları sayısı, şifrelemenin genel çalışma süresini etkilemektedir. Sadece erişim için kullanılan indeks değerlerine bağlı olan T_4 tablosunun çıktıları, zaman odaklı önbellek saldırı için kullanılan istatistiksel modelleri elde etmek için kullanılmaktadır [37].

Uygulanan saldırı üç aşamadan oluşmaktadır:

1) Öğrenme aşaması (study)
2) Saldırı aşaması (attack)
3) Korelasyon aşaması

Korelasyon aşamasında anahtarın tamamının elde edilmemesi durumunda brute force saldırısına başvurulabilir. Ancak korelasyon aşamasında son döngü tüm anahtar baytlarına ulaşılmıştır.

Yapılan saldırı istemci-sunucu tabanlı çalışmaktadır. Öncelikle öğrenme aşamasında saldırgan k anahtarının bilinmesine ihtiyaç duyar. İstemci tarafında, rastgele açık metinler oluşturulur. Rastgele oluşturulan açık metinler şifrelenmek üzere sunucuya gönderilir. Sunucu tarafında gelen açık metinler bilinen bir anahtar ile (bu aşamada anahtar olarak 00..000 kullanılmıştır) şifrelenir ve aynı zamanda şifreleme süresi ölçülür. İstemci, sunucudan rastgele açık metinlerin şifreli halini ve şifreleme süresini alır. Şifreleme süreci bitinceye kadar korelasyon uygulanmak üzere gerekli veriler txt dosyasına kaydedilir. Korelasyon için öğrenme aşamasındaki verilerin bir örneği aşağıdaki şekilde gösterilmektedir.

```
14  600 159 130395 417.393 16.934 0.057 0.047
14  600 160 130910 417.270 11.992 -0.065 0.033
14  600 161 130222 417.291 13.687 -0.045 0.038
```

Şekil 4.3. Öğrenme aşamasına ait örnek veriler

Öğrenme aşaması bittikten sonra saldırı aşaması başlatılır. Saldırı aşamasında da sunucuya istemci tarafından rastgele metinler gönderilir. Sunucu tarafında gelen rastgele metinler bilinmeyen (rastgele oluşturulan) anahtar ile şifrelenir ve şifreleme süresi ölçülür. Cevap olarak istemciye şifreli metinler ve zaman bilgisi verilir. Yine saldırı aşamasında da elde edilen veriler korelasyon işlemi uygulanmak üzere text dosyasına kaydedilmiştir. Korelasyon için saldırı aşamasındaki verilerin bir örneği aşağıdaki şekilde gösterilmektedir.

```
12  600 134 1051010 409.248 19.053 0.021 0.019
12  600 135 1049132 409.207 18.292 -0.019 0.018
12  600 136 1049796 409.237 18.946 0.010 0.018
```

Şekil 4.4. Saldırı aşamasına ait örnek veriler

Şekil 4.4'deki belirtilen veri örneklerinden ilk satırdaki değerler, n[12] = 134 olan 600 baytlık 1051010 paketlerinin sunucuya gönderildiğini ifade eder. Bu paketler 19.053 döngü sapması ile ortalama 409.248 döngüde işlenmiştir. n[12]'ün tüm seçimleri ile ortalama karşılaştırıldığında n[12]=134 için ortalama 0.021 en yüksek döngüdür. 0.019 sayısı bu farkın tahmini sapmasıdır. Yaklaşık 19.053 döngü sapması ve yaklaşık 1051010 ortalama ile ölçülen zamanların dağılımı normale yakınsa, bu tür zamanlar yaklaşık 0.019 döngü sapmasına sahiptir [4].

Öğrenme ve saldırı aşamasında elde edilen Şekil 4.3 ve Şekil 4.4'deki zaman profillerinin korelasyon analizi yapılır. Gerçekleştirilen korelasyon analizi sonucunda yüksek korelasyona sahip olan anahtar muhtemelen aday anahtardır. Korelasyon sonucu Şekil 4.5'deki gibi bir text dosyasına kaydedilmiştir.

```
 77  0 9d d3 3b 85 91 df f1 67 e8 bf 4d ab e9 cb 29 89 f6
79 d2 51 b8 19 75 e5 ed 2e 95 5a 46 88 3d 8d 13 14 d0 47
  1 13 82
  1 10 52
  1  7 f5
  1  4 92
  1  1 00
  1 14 b5
256 11 cb 85 df d8 13 be 87 9f a6 66 bc 0c f0 30 53 1a 72
88 6a aa c3 92 79 8c 5d 90 0b 8e 65 52 d3 91 18 fd 51 58
a4 32 28 c5 3a 33 21 23 a8 f3 24 1e 89 93 75 e0 44 49 70
95 06 ca e3 60 c0 7d e2 1f 41 8a b9 6c b7 15 b5 0a 57 a7
9b 38 3f 80 47 f4 9d b1 5c 01 ae 3e 22 02 da a2 71 f9 09
0f 10 54 99 de c1 5e f8 73 20 1b e7 50 a0 cd 48 c4 d7 9e
  1  8 35
  1  5 ad
  1  2 76
  1 15 6d
256 12 3d 88 79 b4 ed 84 e9 66 5b 1f 75 8f 9f 3a b9 22 d5
f0 ca 83 31 45 67 15 e0 48 cc 7f 57 33 98 2a c3 96 a7 c1
f7 ce 77 e7 a3 c6 86 5e 4d ac 1e 91 b3 29 24 16 14 3e 42
db 92 70 19 04 71 55 8e 7b 18 f1 26 ad 2b 38 fb 1b a0 f3
7a 0f 21 d7 95 4c 90 1a bb 49 08 4a d1 02 2c f6 76 03 a9
34 bc 68 05 f2 8b b7 9c 54 0e f4 93 41 da d0 be 65 27 09
```

Şekil 4.5. Tahmini anahtar uzayı

Şekil 4.5 muhtemel anahtar uzayını ifade etmektedir. Ve çıkan anahtar değerleri AES ‘in son döngü anahtarıdır. Gerçekleştirilen uygulama Linux tabanlı ubuntu işletim sistemi üzerinde çalıştırılmıştır.

Saldırı L1 cache hücresinde gerçekleştirilmiştir. Varyasyonları yakalamak için tüm adımların aynı cache seviyesinde uygulanması gerekmektedir. Random fonksiyonları dinamik kütüphaneleri her yüklediğinde farklı cache adres alanları vermektedir. Bu da doğru korelasyon elde edilmesinin önüne geçmektedir. Bunu önlemek için uygulamaya başlamadan önce önbellek adresini sabitlemek gerekmektedir. Bu işlem için aşağıdaki komut kullanılmıştır.

echo 0 | sudo tee /proc/sys/kernel/randomize_va_space

Linux terminalde uygulama adımları aşağıdaki gibidir:

1) schedtool –a 0x1 -e ./server 127.0.0.1 < /dev/zero

Bu adımla sunucu 000..000 anahtarı ile başlatılmaktadır.

2) schedtool –a 0x1 -e ./study 127.0.0.1 600 > study.600

Bu adımda rastgele 600 açık metin paketi sunucuya gönderilir ve elde edilen zaman değerleri study.600 dosyasına aktarılır. Tüm paketin zaman profili elde edildikten sonra sunucu durdurulur. Öğrenme aşamasının verileri elde edilmiş olur. Öğrenme aşaması burada tamamlanır.

3) dd if=/dev/urandom of=secretkey bs=16 count=1

Bu adımda saldırı aşamasının zaman profilini elde etmek amacıyla rastgele gizli anahtar oluşturulur.

4) schedtool –a 0x1 -e ./server 127.0.0.1 < secretkey

Oluşturulan gizli anahtarla şifreleme yapması için sunucu bu defa gizli anahtarla başlatılır. Bu aşamanın çıktısı aşağıdaki gibidir:

```
burcu@burcu-Lenovo:~/lastround1$ schedtool -a 0x1 -e ./server 127.0.0.1 < secre
key2&
[1] 27885
burcu@burcu-Lenovo:~/lastround1$ Round keys are:
c3b71a1e75e68b90 c7dfc53e99272206 4970f29c7201537 5a6fe8119db02d2f
4109b287459ebdae 86d677b9dcb99fa8 52d3f9d813da4b5f 8bc11c98e6a6670
9c39d70dceea2ed5 1aefa0b41253b17d df7126284348f125 d7cd37e1cd229755
64376ab7bb464c9f 7ed8ca03a915fde2 a482471cc0b52dab 734f70fd0d97bafe
30b8ee66943aa97a 4e6024653d2f5498 e9ad9731d9157957 9ae2e7ccd482c3a9
7b0076c492ade1f5 356052a1af82b56d 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 0000000a 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
loopa girdi
```

Şekil 4.6. Gizli anahtarla sunucunun başlatılması

5) schedtool –a 0x1 -e ./study 127.0.0.1 600 > attack.600

Bu adımda rastgele 600 açık metin paketi sunucuya gönderilir ve elde edilen zaman değerleri attack.600 dosyasına aktarılır. Tüm paketin zaman profili elde edildikten sonra sunucu durdurulur.

6) (tail -4096 study.600; tail -4096 attack.600) | ./correlate>>attack

Bu adımda toplanan zaman profillerinin korelasyon analizi yapılmıştır. Elde edilen analiz sonucu attack dosyasında Şekil 4.5'deki gibidir. Bu aşamadan sonra AES'in son döngü anahtarı elde edilmiştir.

Saldırı uygulanırken diğer yandan Linux 'un donanım performansını ölçen araçlarından yararlanılarak saldırının aşaması hakkında bilgiler elde edilmiştir. Gerçekleştirilen saldırı L1 önbellek hücresindeki zaman varyasyonlarından yararlandığı için L1 önbelleğindeki performans bilgilerini toplayan aşağıdaki komutlar kullanılmıştır.

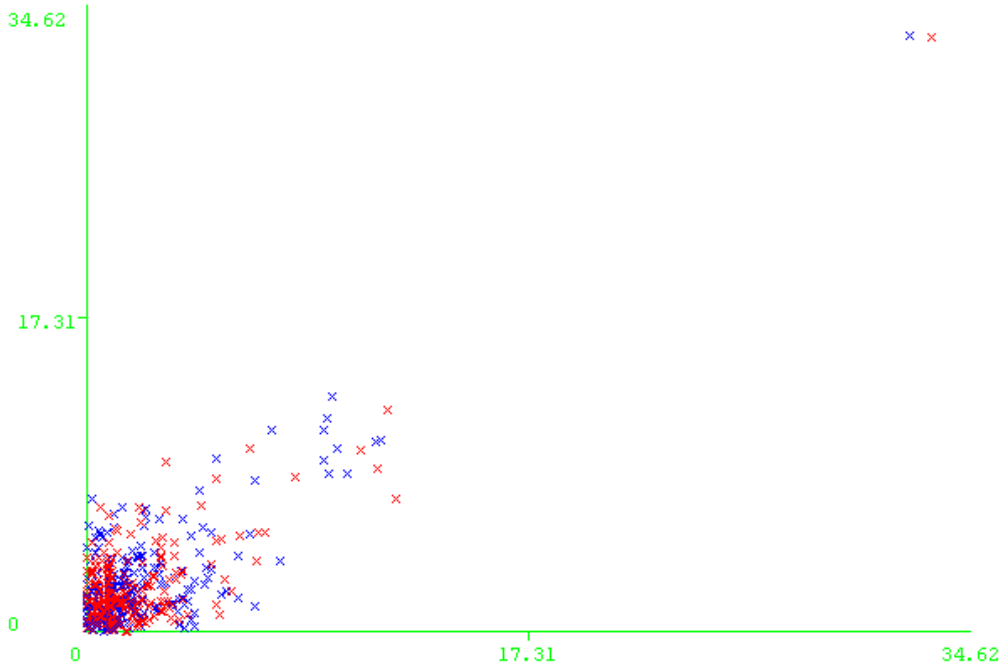
perf record -e L1-dcache-loads ./server

perf report -n > study.csv

perf record -e L1-dcache-loads-misses ./server

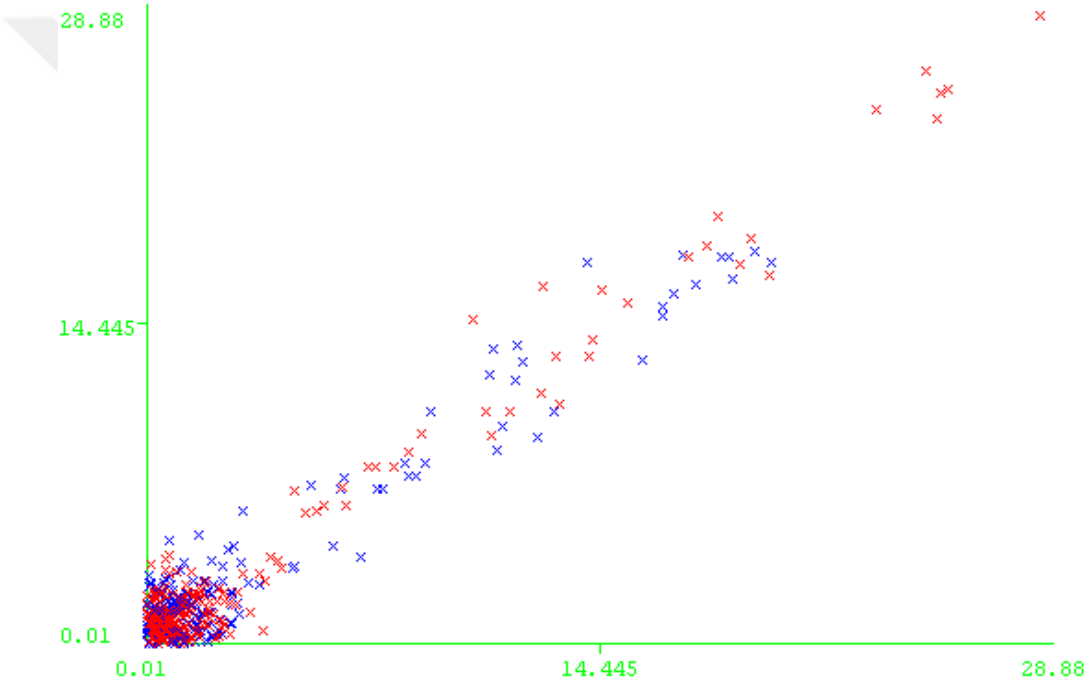
perf report -n > study.csv

Bu komutlar hem öğrenme aşamasında hem de saldırı aşamasında çalıştırılıp sunucunun L1 önbellek hücresinde yüklenen ve kaçan veri oranlarını sayısal değerler olarak kaydetmeyi sağlamıştır. Bu değerler üzerinde k-ortalama kümeleme algoritması uygulanarak verilerin öğrenme aşamasına veya saldırı aşamasına ait olup olmadığı hakkında bilgi elde edilmeye çalışılmıştır. Bu aşamada son zamanlarda saldırı tespit yöntemi olarak kullanılan anormallik tespiti yöntemi devreye girmektedir. Ancak Bernstein'in saldırı yönteminde öğrenme ve saldırı aşamalarında kullanılan cihazların özdeş olması ve öğrenme ile saldırı aşamalarının aynı işlemleri takip etmesi açısından kümeleme algoritması yetersiz kalmıştır. Şekil 4.7' de K-ortalama kümele algoritmasının L1 önbelleği yüklemesine göre sonucu verilmiştir. Saldırı aşaması ile öğrenmesi aşamasında başlangıçta 2^{32} sayıda rastgele açık metin paketlerini önbelleğe yüklendiği için kümeleme algoritması sonucu benzerlik göstermiştir. Şekil 4.7'ye göre verilerin önbellek hücresine yüklenme oranları ile anormallik tespiti yapılamamaktadır. Çünkü öğrenme ve saldırı aşaması önbelleğe veri yükleme bakımından özdeştir. Grafikteki kırmızı değerler saldırı aşamasına ait, mavi değerler ise öğrenme aşamasına ait değerlerdir.



Şekil 4.7. L1 önbellek hücresi yüklemeleri için K-ortalama kümeleme algoritması

Şekil 4.8’de önbelleğin L1 hücresinde verilerin kaçma oranına göre kümele algoritması sonucu verilmiştir. Bu grafikte de kırmızı ile gösterilen değerler saldırı aşamasın ait, mavi ile gösterilen değerler öğrenme aşamasına ait değerlerdir. Gerçekleştirilen saldırı önbellekten kaçan verilerin sebep olduğu zaman varyasyonlarına dayandığı için L1 hücresindeki kaçma oranına göre saldırı aşamasının tespit edilmesi önbellek yüklemesine göre elde edilen sonuçlara göre daha iyidir. Şekil 4.8’de görüldüğü gibi saldırı aşamasına ait veriler kümenin dışında kalmıştır. Bu değerler anormal değerler olarak etiketlenir ve kümeleme algoritması sonucunda saldırı aşamasına ait olan değerler tespit edilebilmektedir.



Şekil 4.8. L1 önbellek kaçan veri oranları için K-ortalama kümeleme algoritması

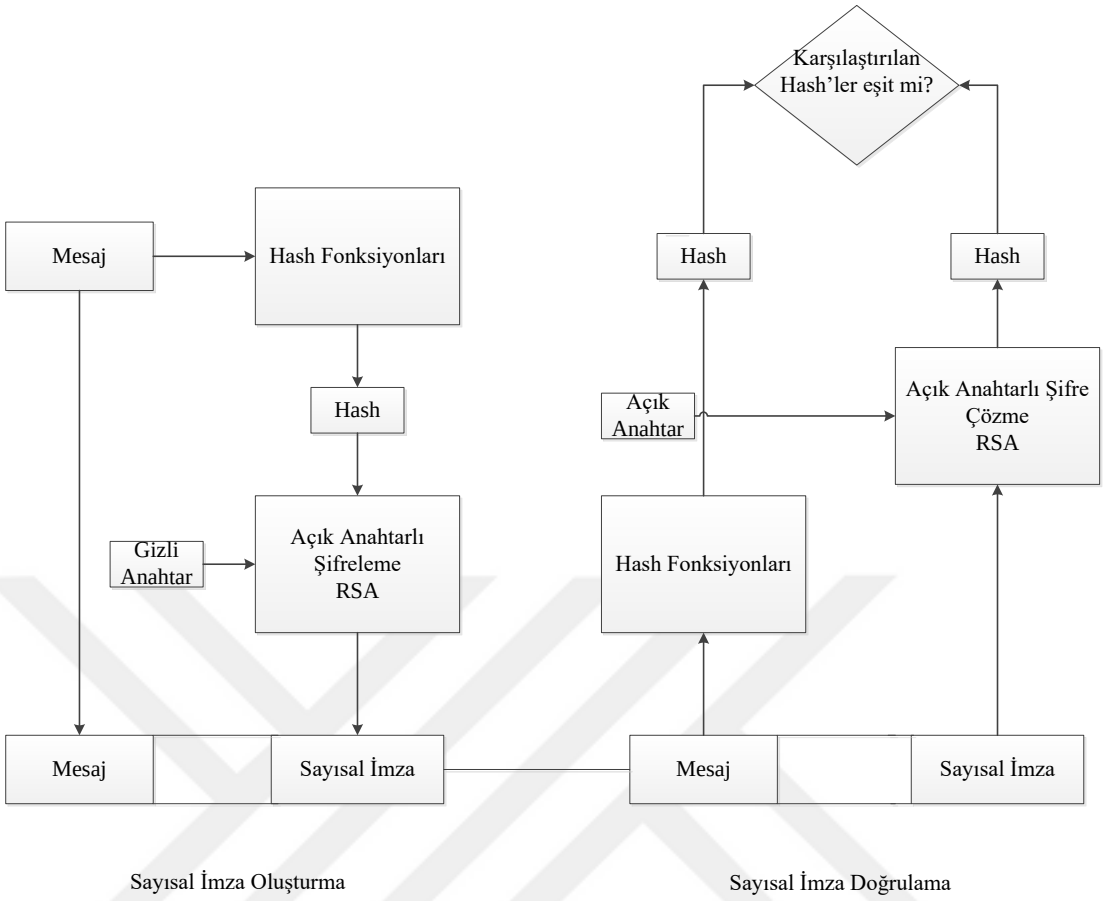
Elde edilen grafikler kümeleme algoritmasının sonucu Bernstein saldırısının tespit edilmesinin kolay olmadığını göstermektedir. Bu yöntemi kullanan saldırgan normal aşamadaki işlemleri tekrarladığı için dışardan gözlemlendiğinde bir saldırı söz konusu olsa dahi, sistemin normal aşamadaki gibi davrandığı gözlemlenebilmektedir.

5. FPGA ÜZERİNDE KLASİK RSA ve CRT ile RSA TABANLI SAYISAL İMZA UYGULAMASININ GÜÇ ANALİZİ VE ANORMALLİK TESPİT

5.1. RSA Algoritması

1976 yılında Whitfield Diffie ve Martin Hellman'in [38]'de belirtilen çalışmasıyla o dönemde kriptografide yeni bir alan olan açık anahtarlı şifreleme sistemi tanıtılmıştır. Bunun bir sonucu olarak kriptologlar açık anahtarlı şifreleme sistemini gerçekleştirebilmek için çeşitli yöntemler araştırmaya başlamışlardır. 1977 yılında Ron Rivest, Adi Shamir, ve Leonard Adleman şu anda da geniş ölçüde kullanılan RSA algoritmasını tasarlamışlardır. Algoritmanın ismi yazarların soy isimlerinin baş harflerinden oluşmuştur. RSA algoritması için çeşitli uygulamalar vardır. Ancak uygulama en sık kullanılanlar, veri şifreleme ve sayısal imzalardır.

RSA [39] ve ElGamal [40] gibi açık anahtarlı bir şifreleme sisteminde, kullanıcı bir belgeyi şifrelemek için alıcının açık anahtarını kullanabilir. Belirli bir alıcı şifreli dökümanı çözebilmek için gizli anahtarını kullanır. Açık anahtarlı şifreleme sistemlerinde açık anahtar (public key) ve gizli anahtar (private key) olmak üzere iki anahtar mevcuttur. Açık anahtar herkes tarafından bilinir ve mesajları şifrelemek için kullanılır. Şifrelenen mesajlar ise sadece gizli anahtarın makul bir sürede kullanılması ile çözülebilmektedir. Öte yandan Şekil 5.1'de gösterildiği gibi, bir kullanıcı kendi gizli anahtarı ile bir belgeyi imzalamayabilir ve herhangi bir alıcı imzalayanın açık anahtarını kullanarak imzayı doğrulayabilir.



Şekil 5.1. Sayısal İmza Şeması

Şekil 5.1'de belirtilen Hash fonksiyonları [41] kısa ve sabit uzunluklu bit dizisidir ve mesajın bir özetini hesaplayan çok yaygın kullanılan bir protokoldür.

Şekil 5.1'de ifade edilen şemada karar aşamasında karşılaştırılan hash ifadesi eşit olduğunda imzalanan belgenin bütünlüğü ve gönderenin kimliği doğrulanmış olur. Böylece, sayısal imza belgeyi gönderenin kimliğini kanıtlamak, iletilen belgenin gizliliğini korumak, alınan belgenin bütünlüğünü doğrulamak için kullanılabilir.

RSA algoritması temelde anahtar oluşturma, şifreleme ve şifre çözme operasyonlarından oluşmaktadır. Buna ek olarak RSA algoritması ile sayısal imza uygulamaları da gerçekleştirilmektedir. Yapılan çalışmada RSA algoritması ile sayısal imza uygulamasına odaklanılmıştır.

Klasik RSA algoritmasında modüler üs alma, soldan sağa doğru kare alma ve çarpma algoritması kullanılmaktadır. Modüler üs alma işlemi sayısal imza oluşturmanın maliyetli bir aşamasıdır. Özellikle güvenlik açısından çok büyük asal sayılar kullanıldığında modüler üs alma ve modüler çarpma işlemleri oldukça maliyetli olmaktadır. Bu yüzden

verimli bir şekilde uygulanması önemlidir. Verimi arttırmak amacıyla birçok yaygın olarak kullanılan hızlı üs alma ve hızlı çarpma yöntemleri vardır. Bu yöntemler literatürde Montgomery Çarpma Yöntemi ve Çin Kalan Teoremi olarak geçmektedir [42,43].

Hızlı üs alma işlemi için kullanılan yöntem Çin Kalan Teoremi'dir. Bu teoremle üs alma işlemi iki ayrı modül şeklinde gerçekleştirilir. Daha sonra imza oluşturma aşamasında ikiye ayrılan modüller birleştirilir[43]. Bu sayede klasik RSA tabanlı sayısal imza uygulamasında tek bir adımda gerçekleştirilen üs alma işlemi yerine, Çin Kalan Teoremi'nde üs alma işlemi iki ayrı modül şeklinde gerçekleştirilmektedir ve performans açısından daha hızlı sonuçlar elde edilmektedir. Çin Kalan Teoremi Bölüm 5.1.4' de yer almaktadır.

RSA algoritmasında gerçekleştirilen tüm şifreleme, şifre çözme, anahtar oluşturma ve sayısal imza operasyonları Z_n tam sayı halkasında [44] gerçekleştirilmektedir ve bu algoritmanın temelinde modüler hesaplamalar yer almaktadır [34]. Ayrıca Euler teoremi [45] ve Euler'in phi fonksiyonu RSA'da önemli rol oynamaktadır.

5.1.1. Anahtar Oluşturma

Anahtar oluşturma operasyonları açık ve gizli anahtarların hesaplamalarından oluşmaktadır. Açık ve gizli anahtarlar hesaplanırken Algoritma-1'deki adımlar izlenir.

Algoritma 1 :

1. Rastgele iki büyük asal sayı seçilir: p ve q
2. Modüler n hesaplanır: n = p*q
3. $\phi(n)$ =(p-1)(q-1) hesaplanır.
4. gcd(e, $\phi(n)$)=1 şartını sağlayan $1<e<\phi(n)$ -1 aralığında bir e açık üssü seçilir.
5. Gizli üs d hesaplanır: $ed=1(\mathrm{mod}\,\phi(n))$

Algoritma 1'de öncelikle açık ve gizli anahtarı oluşturmak üzere kullanılacak olan modülün hesaplanması için iki tane asal sayı seçilmektedir. p ve q ile temsil edilen bu asal sayılar güvenlik amaçlı oldukça büyük asallardan ve rastgele olarak seçilmelidir. Her ne kadar n değeri genel (herkese açık) olsa da n değerinin hesaplanmasının zorluğundan dolayı p ve q değerlerinin herkes tarafından bilinmesi oldukça zordur. Bu aynı zamanda d' nin (Gizli anahtarın) e'den(açık anahtardan) türetilebileceği de gizler [39]. Bu şekilde RSA algoritmasının güvenliği sağlanmaktadır.

Algoritma 1'de 4. Adımda belirtilen gcd(e, ϕ(n))=1 koşulunun sağlandığı durumda e'nin ϕ(n) modülünde bir tersi olduğu kesinleşmiş olur. Böyle d gizli anahtarına daima ulaşılabilir. Verilen algoritmada 4. Ve 5. Adımlarda açık ve gizli anahtarlar hesaplanmaktadır. d ve e anahtarlarının hesabı genişletilmiş öklid algoritması (EEA: Extended Euclidean Algorithm) ile gerçekleştirilmiştir [35].

Bu bölümde RSA algoritmasında gizli ve açık anahtarların nasıl üretildiği anlatılmıştır. Sonraki bölümlerde üretilen anahtarlar kullanılarak şifreleme, şifre çözme ve sayısal imza oluşturma aşamaları anlatılacaktır.

5.1.2. Şifreleme ve Şifre Çözme

RSA şifreleme ve şifre çözme işlemleri Z_n tamsayı halkasında gerçekleşir. Şifrelenecek metin M olarak ifade edilirse, M Z_n={0,1,…,n-1} kümesinin bir elemanı olmalıdır ve bundan dolayı da açık metin M 'in binary (ikili) değeri n'den küçük olmalıdır. Aynı şekilde şifrelenmiş metin de bu kurallara uymaktadır. Şifreleme işlemi aşağıdaki denklemde görüldüğü gibi açık anahtarla gerçekleşmektedir. Açık metnin n modülüne göre açık anahtar üssü alınmaktadır.

$$C = M^e mod n \tag{5.1}$$

Denklem 5.1'de belirtilen C şifrelenmiş metni göstermektedir. Burada şifreleme yaparken girilen metinler sayısal değerlere dönüştürülür daha sonra Z_n halkasında gerekli işlemler yapılır.

Şifre çözme işlemi ise şifreleme işlemine benzerlik göstermektedir. Ancak burada kullanılan üs gizli anahtardır. Denklem 5.2'de gösterildiği şekilde şifre çözme operasyonu gerçekleştirilmektedir.

$$M = C^d mod n \tag{5.2}$$

5.1.3. Sayısal İmzalar

RSA ile bir M mesajının imzalanması için, p ve q iki büyük asal sayı olmak üzere bir N değeri elde edilmektedir ve M mesajı $m \epsilon Z_N$ ' i temsil eden bir hash fonksiyonu ve padding işlemi kullanılarak dönüştürülmektedir. Daha sonra imzalayıcının gizli anahtarı kullanılarak $m^d \ modN$ hesaplanır. Böylece M mesajı için dijital imza elde edilmektedir. Burada d gizli anahtarı temsil etmektedir [46].

M imzalanacak metin olmak üzere, sayısal imza "(5.3)" te gösterildiği gibi elde edilir.

$$S = M^d \ \mathrm{mod} N \tag{5.3}$$

Eğer M imzası için S imzası geçerli ise doğrulayıcı "(5.4)" deki eşitliği kontrol ederek m' i hesaplayabilir [49].

$$m = S^e \ \mathrm{mod} N \tag{5.4}$$

Bu hesaplamaların temeli, kare alma, çarpma ve modüler üs alma algoritmaları tarafından gerçekleştirilebilen modüler üs alma işlemidir. Gerçekleştirilen klasik RSA tabanlı sayısal imza algoritmasının sözde kodu Algoritma 2'de verilmiştir.

Algoritma 2:

1. Birbirine yakın iki asal sayı p ve q seçilir.
2. N=p*q ve ϕ(N) =(p-1)(q-1) hesaplanır.
3. 1<e<ϕ(N) arasında bir e tamsayısı seçilir.
4. ed=1(modϕ(N)) olacak şekilde bir d tam sayısı seçilir.
5. $S = M^d \ \mathrm{mod} N$ sayısal imza hesaplanır.

5.1.4. Çin Kalan Teoremi ile RSA Tabanlı Sayısal İmza

Çin Kalan Teoremi (Chinese Remainder Theorem : CRT) ile RSA uygulaması ,RSA dijital imza algoritmasının yüksek performans gerektiren geniş çaplı kullanılan bir algoritmadır [47].

CRT uygulamasında imzalayıcı öncelikle her bir asal faktörü p ve q' yu imzalama modülünde ayrı ayrı hesaplar. Daha sonra Çin Kalan Teoremi kullanılarak imza $S\ mod N$ hesaplanır. p ve q'nun boyutu N'nin yaklaşık olarak yarısı kadar olduğu için CRT ile üs alma doğrudan üs alma işleminden dört kat daha hızlı gerçekleşir.

$m_1, m_2, ..., m_n$ pozitif asal sayılar ve $a_1, a_2, ..., a_n$ keyfi seçilmiş tam sayılar olsun. Sistem birbirinden farklı $\mathrm{m_1 = m_1 m_2 ... m_n}$ çözüm modülüne sahip olur [48]. (Şöyle ki, $0 \le x \le m$ olmak üzere bir x çözümü vardır ve bu çözüme göre diğer tüm çözümler modm 'in kongürentidir.)

$$\begin{aligned} x &\equiv a_1 \left(\mathrm{mod}\, m_1\right) \\ x &\equiv a_2 \left(\mathrm{mod}\, m_2\right) \\ &\vdots \\ x &\equiv a_n \left(\mathrm{mod}\, m_n\right) \end{aligned} \qquad (5.5)$$

Bu teoremi kanıtlamak için, var olan bir çözüm yöntemine ihtiyaç duyulmuştur ve bu çözüm modm'dir. Eş zamanlı bir çözüm oluşturmak için, öncelikle $k = 1, 2, ...n$ olsun [48].

$$M_k = m / m_k \qquad (5.6)$$

m_i ve $m_k, i \neq k$ olduğunda 1'den daha büyük ortak bölene sahip değillerdir ve bu (5.7)' a karşılık gelir. Sonuç olarak, M_k'nın tersinin m_k olduğunu biliriz [48]. Yani bu (5.8)'a karşılık gelir.

$$\gcd\left(m_k, M_k\right) = 1 \qquad (5.7)$$

$$M_k y_k \equiv 1 \left(\mathrm{mod}\, m_k\right) \qquad (5.8)$$

Eş zamanlı bir çözüm oluşturabilmek için (5.9)'deki toplam oluşturulur.

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + ... + a_n M_n y_n \qquad (5.9)$$

Gerçekleştirilen Çin Kalan Teoremi ile RSA açık anahtarlı şifreleme sistemi kullanılarak sayısal imza uygulaması sözde kodu Algoritma 3'de verilmiştir.

Algoritma 3:

1. Birbirine yakın iki asal sayı p ve q seçilir.
2. N=p*q ve $\phi(N)$ =(p-1)(q-1) hesaplanır.
3. 1<e<$\phi(N)$ arasında bir e tamsayısı seçilir.
4. ed=1(mod$\phi(N)$) olacak şekilde bir d tam sayısı seçilir.
5. $m_{p=}$ m (modp) ve m_q = m (modq) hesaplanır.
6. d_p = d(mod (p-1)) ve d_q=d(mod (q-1)) hesaplanır.
7. $xp = m_p^{dp} \text{ modp}$ ve $xq = m_q^{dq} \text{modq}$ olarak ayrı ayrı modüler üs alma işlemi gerçekleştirilir.
8. Ayrı gerçekleştirilen modüller birleştirilerek sayısal imza hesaplanır.

5.1.5 FPGA Üzerinde Klasik RSA ve CRT ile RSA Tabanlı Sayısal İmza Uygulanması, Güç Analizi ve Anormallik Tespiti

Bu çalışmada FPGA üzerinde Çin Kalan Teoremi ile RSA açık anahtarlı şifreleme sistemi kullanılarak sayısal imza gerçekleştirilmiştir. Gerçekleştirilen sayısal imza uygulaması her iki yöntem için vhdl dilinde ayrı ayrı kodlanmıştır. Klasik RSA ile sayısal imza uygulaması için gerçekleştirilen vhdl kodu aşağıda verilmiştir.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use IEEE.std_logic_arith.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_signed.all;
ENTITY classicRSA IS
generic (p : integer :=7;
 q : integer :=19;
 d : integer :=65;
 e : integer :=5;
 m : integer :=15;
```

```
 n : integer :=133);
PORT(
   clk   :   IN  std_logic;
    s     : OUT integer
          range 0 to 15);
END classicRSA;
ARCHITECTURE beh OF classicRSA IS
begin
process (clk)
BEGIN
s <= (m**d) mod n;
end process;
END beh;
```

Uygulamayı çalıştırmadan önce RSA algoritmasına göre kullanılacak değerler hesaplanıp değişkenlere ataması yapılmıştır. FPGA üzerinde çalıştırılmadan önce gerekli port atamaları gerçekleştirilmiştir. Öncelikle uygulamanın çalıştırılması için bir giriş sinyaline ve çıkışın ossiloskopta gözlemlenmesi için bir çıkış portuna ihtiyaç vardır. Giriş portu bir clock sinyalidir ve uygulama çalıştırıldığında tetikleme gerçekleşir. Deney seti üzerinde yazılan komutlara göre sayısal imza gerçekleşir. Giriş portunun ayarlanması clk : IN std_logic şeklindedir. Burda lojik sinyal olarak 1 geldiğinde kullanılan portun aktif olup çalışması 0 geldiğinde ise portun pasif olması ayarlanır. Çıkış portunun ayarlanması ise s : OUT integer range 0 to 15 şeklindedir. Burada ifade edilen s değeri sayısal imzayı ifade etmektedir. Güç tüketimi için ossiloskopta gözlemleyeceğimiz değer sayısal imza değeri olduğu için çıkış portu olarak bu değer ayarlanmıştır. Range 0 to 15 ifadesi sayısal imza değerinin sayısal olarak sınırını ifade etmektedir. Sayısal imza (s) değeri maksimum 16 bayt olabilmektedir. Aynı atamalar CRT ile RSA tabanlı sayısal imza uygulaması için de gerçekleştirilmiştir. CRT ile ayrı ayrı üs alma işlemi gerçekleştirildiği için klasik RSA' ya göre farklı değişkenler tanımlanmıştır. CRT ile RSA tabanlı sayısal imza uygulamasının vhdl kodu aşağıdaki gibidir.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
```

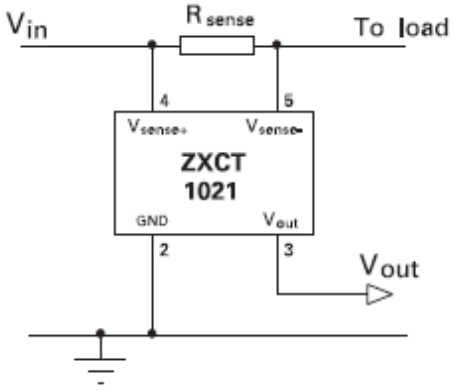
```
use IEEE.std_logic_arith.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_signed.all;
ENTITY crtRSA IS
generic (p : integer :=7;
          q : integer :=19;
          d : integer :=65;
          e : integer :=5;
          m : integer :=15;
          n : integer :=133;
          pt : integer :=19;
          qt : integer :=7);
PORT(
   clk   :   IN  std_logic;
    s     : OUT integer
          range 0 to 15);
END crtRSA;
ARCHITECTURE beh OF crtRSA IS
begin
process (clk)
variable Dp : integer ;
variable Mp : integer ;
variable Mq : integer ;
variable Xp : integer ;
variable Xq : integer ;
variable Dq : integer ;
BEGIN
Mp := m mod p;
Mq := m mod q;
Dp := d mod (p-1);
Dq := d mod (q-1);
Xp := (Mp**Dp) mod p;
Xq:= (Mq**Dq) mod q;
```

```
s <= (q*(qt mod p)*Xp + p*(pt mod q)*Xq) mod n;
end process;
END beh;
```

Bir ölçüm düzeneği kullanılarak gerçekleştirilen sayısal imza uygulamasının güç tüketimi ölçülmüştür. Çin Kalan Teoremi ile klasik RSA tabanlı sayısal imza uygulamalarının güç tüketimleri karşılaştırılmıştır. Gerçekleştirilen sayısal imza uygulaması sonucunda Çin Kalan Teoremi'nin daha az güç tükettiği gözlemlenmiştir.

Gerçekleştirilen CRT ile RSA tabanlı sayısal imza uygulamasında imzalanacak mesaj Mp ve Mq olarak iki ayrı bölüm şeklinde işleme alınmıştır. Modüler üs alma işlemi Çin Kalan Teoremi tanımına uygun olarak Xp ve Xq olmak üzere ayrı ayrı gerçekleştirilmiştir[50]. Ayrı olarak gerçekleştirilen modüler üs alma işlemleri birleştirilerek sayısal imza elde edilmiştir. Çıkış olarak elde edilen s değeri sayısal imzayı ifade etmektedir.

Yapılan uygulamada Altera EP4CE115F29C7 seti kullanılmıştır. Burada harcanan gücün tespit edilebilmesi için ZXCT1021 low offset high-side current monitor entegresi kullanılmıştır. Hassas ölçüm yapabilmek için uygulama devresinde 0.1 ohm'luk direnç kullanılmıştır. Şekil 5.2 uygulama devresini göstermektedir.



Şekil 5.2. ZXCT1021 entegre devresi

Altera EP4CE115F29C7 setinin güç çıkışına 0.1 ohm'luk direnç yerleştirilmiştir. Giriş ve çıkış portları vhdl kodlarındaki gibi ayarlandıktan sonra uygulama için tetikleme sinyali verilmiştir. Uygulama çalışırken ossiloskop yardımıyla direncin her iki ucundaki gerilim farkı ölçülüp kaydedilmiştir. Çin Kalan Teoremi'nin deney setinin üzerinde

gerçekleştirilmesi sonucunda Şekil 5.2'de gösterilen ZXCT1021 entegre devresinin Vout pininden ölçülen güç tüketimi sinyali Şekil 5.3'de gösterilmektedir.

Şekil 5.3. Çin Kalan Teoremi Güç Tüketimi

Klasik RSA ile sayısal imza uygulamasının deney setinin üzerinde gerçekleştirilmesi sonucunda Şekil 5.2'de gösterilen ZXCT1021 entegre devresinin Vout pininden ölçülen güç tüketimi sinyali Şekil 5.4'de gösterilmektedir.

Şekil 5.4. Klasik RSA Güç Tüketimi

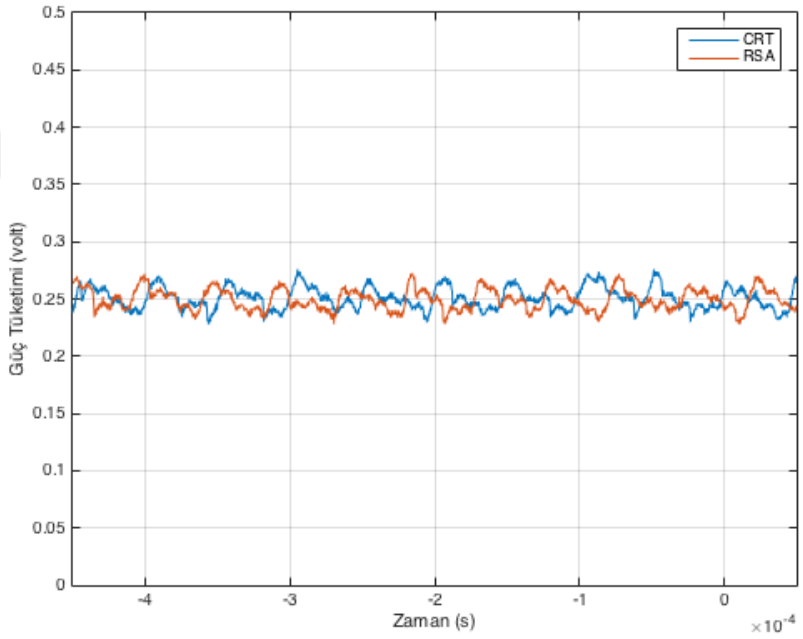
Tablo 5.1'de ölçülen sonuçlara göre Çin Kalan Teoremi kullanılarak gerçekleştirilen sayısal imza uygulamasının klasik RSA tabanlı gerçekleştirilen sayısal imza uygulamasına göre daha az güç tükettiği gözlemlenmiştir.

Tablo 5.1. CRT ile Klasik RSA Tabanlı Sayısal İmza Uygulamalarının Toplam Gerilim Farkları

CRT ile Klasik RSA Tabanlı Sayısal İmza Uygulamasının Ölçülen Gerilim Farkları	
CRT	**RSA**
624,0060 mV	626,7200 mV

Elde edilen gerilim farkları sonucunda Çin Kalan Teoremi'nin modüler üs alma aşamasında sayısal imza uygulamasının işlem yükünü azalttığı gözlemlenmektedir. Özellikle çok büyük asal sayılar kullanıldığında Çin kalan Teoremi'nin klasik RSA tabanlı sayısal imza uygulamasına göre daha fazla avantajı olacaktır.

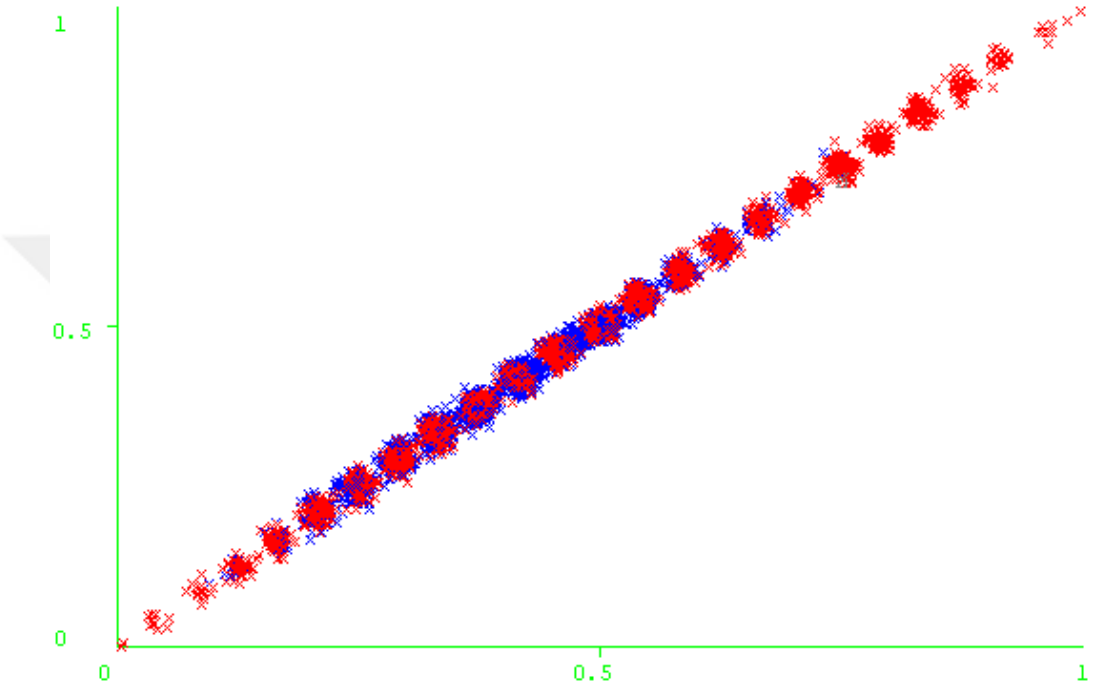
Çin Kalan Teoremi kullanılarak elde edilen sayısal imza ile Çin Kalan Teoremi kullanılmadan RSA algoritmasıyla elde edilen sayısal imza uygulamasının güç tüketimi Şekil 5.5'da gösterilmektedir.



Şekil 5.5. Çin Kalan Teoremi ile RSA Uygulamalarının Güç İzleri

Şekil 5.5'deki grafik incelendiğinde uygulamanın son aşaması olan sayısal imza oluşturma işleminde CRT ile RSA güç tüketimi eğrilerinin birbirinin tersi olduğu gözlemlenmiştir.

Bu grafiklerdeki değerlerden yola çıkılarak bir sayısal imza uygulamasında kullanılan yöntemi ayırt etmek amacıyla sistemin bilgi sızıntısı olarak güç tüketimi kullanılarak anormallik tespit yöntemi uygulanmıştır. Bu yöntemde sistemin güç tüketim değerlerine k-ortalama kümeleme algoritması uygulanmıştır. Kümenin dışında kalan değerler Şekil 5.6'da gösterildiği gibi Çin Kalan Teoremi'ne ait anormal değerler olarak tespit edilmiştir.



Şekil 5.6. Güç tüketimi değerlerine göre kümeleme algoritması sonucu

Şekil 5.6' daki x ve y ekseni güç tüketim değerlerini ifade etmektedir. Mavi işaretli değerler klasik RSA algoritmasına ait, kırmızı işaretli değer ile Çin Kalan Teoremine (CRT) ait değerlerdir. Bu şekilde de görüldüğü gibi 0.75'ten büyük değerler Çin Kalan Teoremine ait anormal olarak tespit edilmiştir. Elde edilen bu sonuca göre bir kriptografik bir cihazın tükettiği güç bilgisinden yararlanılarak makine öğrenmesi teknikleriyle sistemde çalışan algoritmanın yapısı ile ilgili bilgi elde edilebileceği ispatlanmıştır.

6. SONUÇLAR

Bu tezde yan kanal saldırı yöntemleri AES ve RSA algoritmaları üzerinde incelenmiştir. AES algoritması OPENSSL ‘in 0.9.7a versiyonu ile gerçekleştirilmiştir. Gerçekleştirilen AES şifrelemesine literatürde önemli bir kaynak teşkil eden Bernstein’in zaman odaklı önbellek saldırısı uygulanmıştır ve saldırı tespit yöntemlerinden anormallik tespiti gerçekleştirilmiştir. Gerçekleştirilen önbellek saldırısı Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz model bir donanımda ve Linux işletim sisteminde minimum 2^{32} açık metin örneği ile başarıya ulaşmıştır. Bernstein’in önbellek saldırısında donanım sayaçları kullanılarak elde edilen özniteliklerle makine öğrenmesi yöntemlerinden olan anormallik tespitiyle saldırı sürecinin saptanabildiği gözlemlenmiştir. Saldırı esnasında önbellek davranışları analiz edilerek saldırının hangi önbellek davranışlarını kullandığı tespit edilmektedir. Tezin diğer aşaması olan FPGA üzerinde RSA algoritması ile sayısal imza uygulaması gerçekleştirilmesinde güç analizi kullanılmıştır. FPGA üzerinde sayısal imza uygulaması için iki farklı yöntem olarak Klasik RSA ile Çin Kalan Teoremi kullanılmıştır. Her iki yöntemin güç analizi yapılarak anormallik tespiti gerçekleştirilmiştir. Burada makine öğrenmesi için kullanılan öznitelikler FPGA’nın iki farklı yöntem için güç tüketimi olmuştur. Her iki yöntemin güç tüketimi farklarından yararlanılarak anormallik tespiti yapılmıştır.

Bu çalışma sonucunda sistemler için yan kanal önlemleri alınmadığı takdirde bir sistemden sızan yan kanalı bilgisi kullanılarak, sistemin gerçekleştirdiği işlemler, şifrelemeler, anahtarlar gibi gizli bilgilerin elde edilebileceği gözlemlenmiştir. Çalışma bu açıdan yan kanal analizi saldırılarına dikkat çekmektedir. Gelişen teknolojiyle beraber kriptografik cihazların yan kanal saldırıları dikkate alınarak tasarlanması gerekmektedir. Bununla birlikte son yıllarda kriptolojiye destek olarak kriptolojinin yapay zeka ile birlikte kullanımı yaygınlaşmaktadır. Tez de gerçekleştirilen çalışma günümüzde kullanılan saldırı tespit yöntemleri, radar sistemleri gibi güvenlik önlemlerinde yapay zekanın kullanılabileceğini göstermektedir. Gelecekte kriptoloji alanında yapay zeka uygulamalarının gittikçe artacaktır. Bu tezden sonra yapay zeka ile bilgi güvenliği konusunda çalışmalara devam edilecektir.

KAYNAKLAR

[1] **Kocher P., Jaffe J., Jun B.,** 1999."Differential Power Analysis", *In proceedings of CRYPTO 1999*, LNCS 1666, pp. 388-397, Springer-Verlag.

[2] **Yukiyasu T., Teruo S., Tomoyasu S., Maki S., and Hiroshi M.,** Cryptanalysis of DES Implemented on Computers with Cache. In Walter et al. pages 62–76.

[3] **Tsunoo, Y., Tsujihara, E., Shigeri, M., Kubo, H., & Minematsu, K.** 2006. Improving cache attacks by considering cipher structure. *International Journal of Information Security*, *5*(3), 166-176.

[4] **Bernstein D. J.,** 2005. "Cache Timing Attacks on AES," http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[5] **Canteaut A., Lauradoux C., Seznec A.,** 2006. Understanding Cache Attacks, INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE.

[6] **Acıiçmez, O., Schindler, W., Koç, Ç. K.** 2007. Cache based remote timing attack on the AES. In *Cryptographers' Track at the RSA Conference* (pp. 271-286). Springer, Berlin, Heidelberg.

[7] **Chiappetta, M., Savas, E., Yilmaz, C.** 2016. Real time detection of cache-based side-channel attacks using hardware performance counters. *Applied Soft Computing*, *49*, 1162-1174.

[8] **Chiappetta M.** 2016. REAL TIME DETECTION OF CACHE-BASED SIDE-CHANNEL ATTACKS USING HARDWARE PERFORMANCE COUNTERS, *Master's Thesis* Computer Science and Engineering Sabancı University.

[9] **Chaves, R.,** AISS 2012/13. Side Channel Analysis. http://neerci.tecnico.ulisboa.pt/ Departamento de Engenharia Informática, 10 Aralık 2013.

[10] **Konur, E., Özelçi, Y., Arıkan, E., and Ekşi, U.,** 2004. Yan kanal analizlerine dirençli kriptografik tümdevre tasarım yöntemleri geliştirilmesi, SAVTEK, ODTÜ, ANKARA.

[11] **Rizk, M.,** 2002. Side channel attacks. http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf, Erişim: 5 Eylül 2013.

[12] **Ordu, L., Örs Yalçın, S.B.,** 2006. Yan-Kanal Analizi Saldırılarına Genel Bakış, Ulusal Elektronik İmza Sempozyumu, Sheraton Hotel and Convention Center ANKARA, 7-8 Aralık.

[13] **Page D.,** 2002. Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel, Department of Computer Science, University of Bristol.

[14] **Patterson A. D., Hennessy L. J.** Computer Organization And Design, 5th Edition 2014

[15] **Younis, Y., Kifayat, K., Merabti, M.** 2014. Cache side-channel attacks in cloud computing. In *International Conference on Cloud Security Management (ICCSM)* (p. 138).

[16] **Koç, Ç.K.,** 2009. Cryptographic Engineering, City University of Istanbul Tophane, Istanbul Turkey and University of California Santa Barbara, Santa Barbara CA USA.

[17] **Acıiçmez, O., Brumley, B. & Grabher, P.,** 2010. New results on instruction cache attacks. In *CHES'10 Proceedings of the 12th international conference on Cryptographic hardware and embedded systems*. Springer-Verlag Berlin, Heidelberg, pp. 110–124.

[18] **Aciicmez O., Koc Ç.K.,** 2006. "Trace-driven cache attacks on aes (short paper)," in ICICS, ser. Lecture Notes in Computer Science, P. Ning, S. Qing, and N. Li, Eds., vol. 4307. Springer, pp. 112–121.

[19] **Neve, M., Seifert, J. P**. 2006. Advances on access-driven cache attacks on AES. In *International Workshop on Selected Areas in Cryptography* (pp. 147-162). Springer, Berlin, Heidelberg.

[20] **Neve M.,** 2006. "Cache-based vulnerabilities and spam analysis," *PhD Thesis*, Universite catholique de Louvain.

[21] **Spreitzer, R., Gérard, B.** 2014. Towards more practical time-driven cache attacks. In *IFIP International Workshop on Information Security Theory and Practice* (pp. 24-39). Springer, Berlin, Heidelberg.

[22] **Atici, A. C., Yilmaz, C., Savas, E.** 2016. Remote Cache-Timing Attack without Learning Phase. *IACR Cryptology ePrint Archive*, *2016*, 2.

[23] **Le T.H., Canovas C., Clédière J.,** 2008. An overview of side channel analysis attacks, Proceedings of the 2008 ACM symposium on Information, computer and communications security Pages 33-43, Tokyo, March 18 – 20.

[24] **Schramm, K.,** 2002. DES Sidechannel Collision Attacks On Smartcard Implementation, Communication Security Group (COSY), *Thesis*, Departmant of Electrical Engineering and Information Sciences, Ruhr-Universität Bochum.

[25] **Joye, M.,** 2013. Side Channel Attacks on Crptographic Tokens, Crptographic Engineering, Lausanne, July 1-5.

[26] Yarom, Yuval, and Katrina E. Falkner. Flush+ Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack. Proceedings of the 23rd USENIX Security Symposium. USENIX. 2013.

[27] **Sprunt, B.** 2002. The basics of performance-monitoring hardware. *IEEE Micro*, *22*(4), 64-71.

[28] **De Melo, A. C.** 2010. The new linux'perf'tools. In *Slides from Linux Kongress* (Vol. 18).

[29] http://www.brendangregg.com/perf.html, Perf Tool, Erişim: 21 Ağustos 2017.

[30] **Chandola, V., Banerjee, A., Kumar, V.** 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), 15.

[31] **Hartigan J. A.** 1975. WILEY SERIES IN PROBABILITY AND MATHEMATICAL STATISTICS, Clustering Algoirthms, Canada.

[32] **Hartigan, J. A., Wong, M. A.** 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100-108.

[33] **Daemen J., Rijmen V.** 2001. *The design of Rijndael, AES - The Advanced Encryption Standard*. Information Security and Cryptology. Springer.

[34] **PUB, F.** 2001. Advanced Encryption Standard (AES). *FIPS-197*.

[35] **Paar, C., Pelzl, J.** 2009. *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media.

[36] http://www.openssl.org/ OpenSSL, OpenSSL: the Open-source toolkit for SSL / TLS, Erişim: 25 Kasım 2016.

[37] **Diffie, W., Hellman, M.** 1976. New directions in cryptography. *IEEE transactions on Information Theory*, *22*(6), 644-654.

[38] **Rivest, R. L., Shamir, A., Adleman, L.** 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120-126.

[39] **Atici, A. C., Yilmaz, C., Savas, E.** 2013. An approach for isolating the sources of information leakage exploited in cache-based side-channel attacks. In *Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on* (pp. 74-83). IEEE.

[40] **ElGamal, T.** 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, *31*(4), 469-472.

[41] http://csrc.nist.gov/publications/fips/fips180-4/fips180-4.pdf Draft Federal Information Processing Standards Publication 180-4. February 2011.

[42] **Montgomery, P. L.** 1985. Modular multiplication without trial division. *Mathematics of computation*, *44*(170), 519-521.

[43] **Fouque, P. A., Guillermin, N., Leresteux, D., Tibouchi, M., Zapalowicz, J. C.** 2013. Attacking RSA–CRT signatures with faults on Montgomery multiplication. *Journal of Cryptographic Engineering*, *3*(1), 59-72.

[44] **Judson W. T.** Abstract Algebra Theory and Applications, Stephen F. Austin State University, February 14, 2009.

[45] **Koshy T.** Elemantary Number Theory With Applications, 2nd Edition, 2007.

[46] **Walter, D. C., Koç, K, Ç., Paar, C.,** 2003. "Cryptographic Hardware and Embedded Systems-CHES 2003", 5th International Workshop Cologne, pp.254-269.

[47] **Rosen, K. H.,** Sixth Edition, Discrete Mathematics and Its Aplication, Avenue of Americans 8th Edition ,2007

[48] **Quisquater, J. J., Couvreur, C.** 1982. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics letters*, *18*(21), 905-907.

ÖZGEÇMİŞ

1989 yılında Elazığ'da doğan Burcu SÖNMEZ ilköğretim ve lise eğitimini Elazığ'da tamamladı. 2006 yılında Korgeneral Hulusi Sayın Lisesini bitirdikten sonra 2013 yılında Fırat Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümünden mezun oldu. 2013 yılında Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim dalında yüksek lisansa başladı ve halen devam etmektedir. 2014 yılında Ağrı İbrahim Çeçen Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümüne Araştırma Görevlisi olarak atandı. 2016 yılında 39. Madde ile Ağrı İbrahim Çeçen Üniversitesi Bilgi İşlem Daire Başkanlığı Ağ ve Sistem Birimine görevlendirildi. Halen Ağrı İbrahim Çeçen Üniversitesi Bilgisayar Mühendisliği Bölümünde ve Bilgi İşlem Daire Başkanlığı Ağ ve Sistem Biriminde görevine devam etmektedir.