AES Algoritmasına Yapılan Zaman Odaklı Önbellek Saldırılarının Makine Öğrenmesi ile Tespiti

Detecting Time-Based Cache Attacks on AES Algorithm with Machine Learning

Burcu SÖNMEZ SARIKAYA (ORCID: 0000-0002-5385-9949)
Mehmet Ali SARIKAYA (ORCID: 0000-0001-82942007)
Şerif BAHTİYAR (ORCID: 0000-0003-0314-2621)
SPF Lab., Bilgisayar Mühendisliği,
İstanbul Teknik Üniversitesi
{sonmezb18, sarikayameh, bahtiyars}@itu.edu.tr

Öz

Yan kanal saldırıları, hedeflenen sistemdeki şifreleme işlemleri hakkındaki yan kanal bilgisi veya bilgi sızıntısı olarak adlandırılan dolaylı bilgileri kullanır. Bu bilgiler, saldırıları sistem üzerinde oldukça etkili kılar. Bu çalışmada, zaman bilgisini birincil ve önbellek bilgilerini ikincil yan kanal bilgisi olarak kullanan zaman odaklı bir önbellek saldırısı incelenmektedir. Zaman odaklı yan kanal saldırısını gerçekleştirmek için hedef olarak AES algoritması seçilmiştir. Gerçekleştirilen zaman odaklı yan kanal saldırısı ile AES algoritmasının son döngüsündeki gizli anahtarı elde etmektedir. Sistemin zayıf yönlerini belirlemek için ikincil kanallardan bilgi çıkarmak için makine öğrenmesi ve derin öğrenme modelleri kullanılmıştır. Saldırı sırasında oluşturulan zaman profilleri üzerinde ağaç modelleri kullanılarak, saldırının en önemli yan kanal bilgileri değerlendirilmiştir. Karar Ağacı, Rastgele Orman, Gradyan Arttırma ve Ekstrem Gradyan Arttırma algoritmaları veri işleme görevine çok duyarlı olduklarından, bu çalışmada ağaç tabanlı modeller olarak seçilmiştir. Analiz sonuçları, "ortalama döngü" bilgilerinin zaman odaklı önbellek saldırılarında etkili olduğunu göstermektedir. Ayrıca, Ekstrem Gradyan Arttırma algoritması daha iyi sonuçlar vermiştir. Ek olarak, saldırı esnasında elde edilen zaman bilgileri saldırının tespiti için kullanılmıştır. Yapılan deneyler göstermiştir ki zaman odaklı önbellek saldırılarının tespitinde derin öğrenme yöntemleri daha başarılı olmaktadır.

***Anahtar Sözcükler*:** Zaman odaklı önbellek saldırıları, makine öğrenmesi, güvenlik.

Abstract

Side-channel attacks use indirect information about cryptographic operations from the targeted system. This makes the attacks highly effective on the system. In this paper, we investigate the time-based cache attacks that use the time feature and cache information as secondary channel information. We selected AES algorithm to accomplish the time-based side channel attack. The target of the attack is the secret key in the last cycle of AES algorithm. We used machine learning models to extract information from secondary channels to determine vulnerabilities of the targeted system. We applied tree models on time profiles created during the attacks to evaluate the most significant characteristics of the attack. Since Decision Tree, Random Forest, Gradient Boosting, and Extreme Gradient Boosting algorithms are very sensitive to tasks that require high processing, we selected these algorithms. Analysis results show that "cycle on average" information helps to predict the time-driven cache attacks. Moreover, Extreme Gradient

Gönderme ve kabul tarihi: 16.10.2019 - 24.12.2019
Makale türü: Araştırma

Boosting algorithm provides better results than other algorithms. We used the time information extracted during attacks for detection purposes. Experimental evaluations show that deep learning methods provide better detections for time based cash attacks than other machine learning algorithms.

***Keywords*:** Time-driven cache attacks, machine learning, security.

1. Giriş

Yan kanal saldırılarının son bilgi sistemleri için önemli sonuçları vardır. Şifreleme anahtarları gibi bir sistemdeki şifreleme işlemleri sırasında yayılan kritik bilgileri kullanırlar. Yayılan bilgiler literatürde yan kanal bilgisi olarak adlandırılmaktadır. Yan kanal saldırıları bu bilgileri hedeflerine ulaşmak için kullanmaktadır. Bu çalışmada sadece zaman ve önbellek bilgilerini kullanan zaman odaklı yan kanal saldırıları ele alınmaktadır. Bu çalışmada [1]'de olduğu gibi zaman odaklı yan kanal saldırılarında kullanılabilecek açıkları belirlemek amacıyla yan kanal bilgilerinin bir sistemden sızmasını değerlendirmek için ağaç tabanlı makine öğrenme modelleri ve saldırıyı tespit etmek için derin öğrenme yöntemi kullanılmıştır.

Saldırı sırasında oluşan zaman profillerine ağaç tabanlı modeller uygulanmış ve saldırı için önemli öznitelikler değerlendirilmiştir. Veri seçim sürecinin zaman odaklı önbellek saldırısına katkısını doğrulamak için parametrik olmayan birkaç makine öğrenme algoritması kullanılmıştır. Bu yaklaşımlardan biri, birkaç zayıf tahmin modelini genelleştirilebilir bir modelde birleştiren ağaç algoritmalarından oluşmaktadır. XGBoost teknolojisi (Extreme Gradient Boost: Ekstrem Gradyan Artırım) [2] birçok makine öğrenme yöntemleri gibi, son zamanlarda veri bilimcileri arasında popüler hale gelen bir tekniktir [3][4]. Gradyan artırma (GBM), ağaç kümeleri oluştururken kırpmayı en aza indirmek için gradyan azaltma algoritmasını güçlendirme ile birleştirir. XGBoost'ta [2] ağaçların boyutunu ve şeklini kontrol eden, tahminleri daha sağlam ve algoritmayı daha genel yapan başka düzenlileştirme parametreleri vardır. Ağaç, veri özellikleri arasındaki etkileşimi yakalamak için idealdir. Son olarak, çalışmamızda Karar Ağacı, Rastgele Orman, Degrade Artışı ve Aşırı Degrade Yayılım (XGBoost) algoritmaları ağaç tabanlı algoritmalardan seçilmiştir, çünkü farklı işlem görevleri için oldukça hassas algoritmalardır [5][6][7].

Gradyan azaltma tabanlı yükseltme yöntemleri istatistiksel çerçeveler ile ilişki kurmak için kullanılır [7][8][9]. Bu yöntemlere gradyan arttırma makineleri denir. Bu makinelerin öğrenme süreci tahmin modeline daha doğru bir sonuç vermesi için sırayla yeni modeller adapte etmektir. Bu algoritmanın arkasındaki fikir, kayıp fonksiyonunun (loss function) negatif gradyanı ile en yüksek düzeyde ilişkili olduğunu bulmaktır. Detaylar [10]'da verilmiştir.

Yan kanal saldırılarında makine öğrenmesi algoritmaları ile yan kanaldan sızan bilgilerin kullanılması yaygın değildir. Bu çalışma ile birlikte, makine öğrenmesi veya daha akıllı yapay zeka yöntemlerini kullanarak saldırılardaki yan kanal sızıntılarından bilgilerin değerlendirilmesi, seçilmesi ya da özellik çıkarılması gibi önemli olayların göz önüne alınması mümkündür. Yan kanal saldırılarında temiz bilgi edinmek, saldırının başarısında önemli bir rol oynamaktadır. Böylece, makine öğrenme teknikleri gürültü azaltmada ve önemli bilgilerin çıkarılmasında yardımcı olacaktır.

Zaman odaklı önbellek saldırılarında, gizli anahtarı bulmak için zaman profilinde doğru parametreleri bulmak önemlidir. Kritik parametreler belirtilmezse, tüm anahtar verileri saldırı sonucu elde edilemeyebilir ve bunun sonucunda parametreler arasındaki korelasyon analizinde gürültülü sonuçlar elde edilebilir. Bu çalışmadaki motivasyonumuz zaman profilindeki en önemli değişkenleri gürültü bariyeri tarafından engellenmeden bulmaktır. Çalışmanın bir sonucu olarak, zaman profilinde "ortalama döngü" ve "döngü sapması " olmak üzere iki önemli parametre belirlenmiştir.

Bu çalışmanın literatüre iki önemli katkısı vardır. Birincisi, zaman tabanlı önbellek saldırıları ile ilgili kritik bilgileri makine öğrenmesi yöntemleri ile daha yüksek başarım ile elde edilmesidir. Özellikle, yan kanal sızıntı seçiminde daha önce kullanılmamış olan XGBoost ve GBM benzeri makine öğrenme algoritmaları kullanılmıştır. Diğer katkı, makine öğrenmesi ile elde edilen zaman tabanlı önbellek saldırıları ile ilgili bilgilerin derin öğrenme yöntemi kullanılarak bu saldırıların tespitinin gerçekleştirilmesidir.

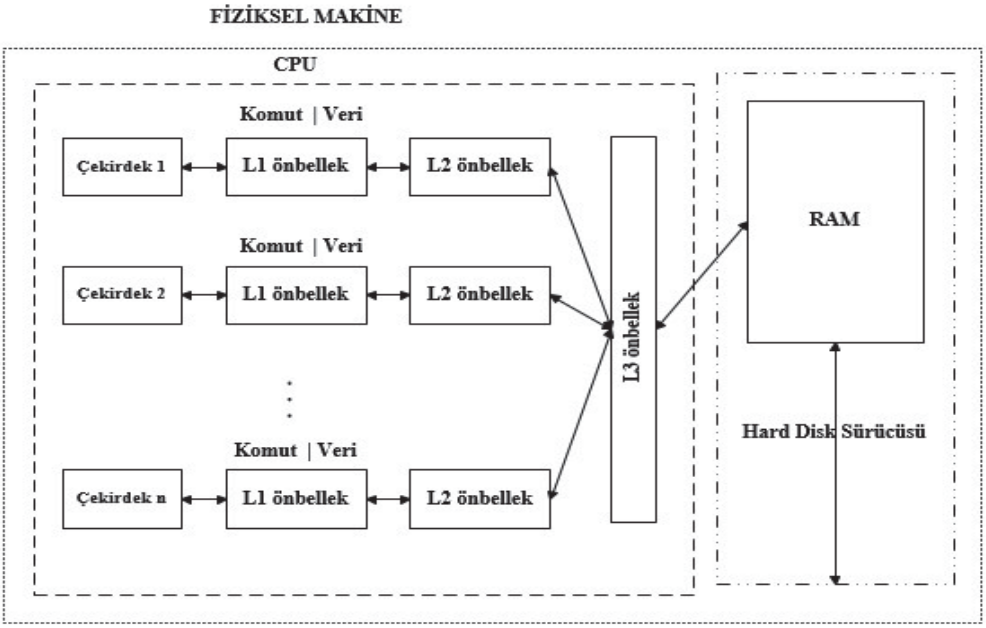
Makalenin geri kalanı şu şekilde düzenlenmiştir. Bölüm 2 zaman odaklı önbellek saldırıları hakkında bilgiler içermektedir. Sonraki bölümde, önbellekteki yan kanal saldırılarına ilişkin ağaç tabanlı yaklaşımlar açıklanmaktadır. 4. Bölümde derin öğrenme yöntemleri ile saldırı tespitinden bahsedilmektedir. Son bölümde sonuçlar yer almaktadır.

2. Zaman Odaklı Önbellek Saldırıları

Önbellek mikroişlemciler bağlamında ana bellek ve işlemci arasında yer alan belleğin küçük bir alanıdır. Ana belleğe 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 [11]. Önbellek, bu ekstra seviyeye sahip olmak için ilk ticari bilgisayardaki işlemci ve ana bellek arasındaki bellek hiyerarşisinin seviyesini temsil etmek için seçilen addır [12].

Modern işlemciler önbellek üzerinde Şekil 1''de gösterildiği gibi L1, L2 ve L3 olmak üzere üç tane seviyeye sahip olabilmektedir.

Onlardan her biri veri veya talimat olarak adlandırılan özel bir amaç için kullanılabilir. 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 [13]. L3 boyut açısından en büyük önbellektir. Ancak önbellekler arasında en yavaş olanıdır. L3 seviyesine son seviye önbelleği (LLC: Last Level Cache) de denmektedir. L3 önbelleği, işlemci çekirdeği arasında paylaşılır ve diğer önbellekler (L1 ve L2) süreçler ve iş parçacıkları arasında paylaştırılır [14].



Şekil-1 : Önbellek Mimarisi

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 [13]. İşlemci, bellek sistemine bir erişim sorunu olduğunda, başvurulan bellekteki adres ilk önce bir önbellek satırına eşlenir. 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 [11]. Önbelleklerin tümünde önbellek satırı, aynı içerik alanında farklı adreslerden gelen verileri içerebilir. Yani, birkaç adres aynı satırda eşlenebilir ve bu satırın içinde bulunabilir. Bu yapı daha büyük veri bloklarının aynı anda ana belleğe aktarılmasını ve daha küçük aktarımlarla uğraşmaktan daha etkili olmasını sağlar. Önbellek etiket 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 önbellekte bulunuyorsa önbellek vuruşu (cache hits) olarak adlandırılır. İşlemci, bir önbellekten daha uzun bir bekleme süresi olan ana belleğe erişmek yerine bu verileri hemen kullanır. İstenilen veriler önbellekte yoksa bir önbellek hatası (cache miss) oluşur. Bu durumda veriler ana bellekten okunur ve bir kopyası önbellekte saklanır [15].

Bellekten veriyi 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 algoritması sabit bir yürütme akışına sahip olabilir. Tüm açık metinler ve gizli anahtarlar aynı komut yapısında çalışabilir. Fakat uygulamanın yürütülmesi sırasındaki önbellek davranışı programın çalışma süresinde varyasyonlar olur. Önbellek saldırıları, bu varyasyonlarda yararlanır ve gizli anahtarların ayrıntılı arama alanını daraltır [16].

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 [11]'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 saldırgan, şifrelemelerinin bir örneği için önbellek vuruşlarının ve hatalarının izlerini alır ve bu verileri kullanarak gizli anahtarı bulmayı hedefler. İz, bir dizi önbellek vuruşlarının ve hatalarının bir serisi olarak tanımlanmaktadır. İz odaklı önbellek saldırıları için detaylar [18]'de verilmiştir.

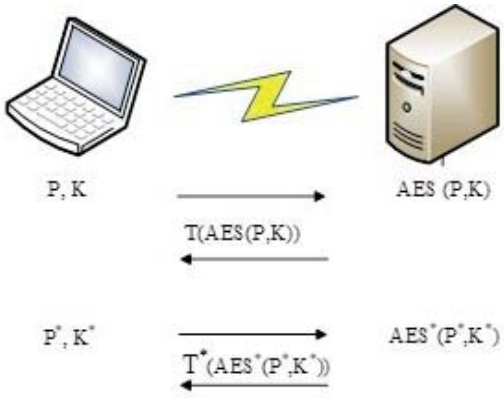
Erişim odaklı saldırılar, genel yürütme süresini değerlendirmek yerine önbellek davranışını daha ince ayrıntılarla inceler. Bu saldırıda gizli anahtarın yorumlanması için kriptografik işlemler boyunca bir önbellek satırına erişilip erişilmediği konusundaki bilgi kullanılmaktadır. Erişim odaklı önbellek saldırıların detayları [19]'da verilmiştir.

Gerçekleştirilen uygulamada zaman odaklı önbellek saldırısı kullanılmıştır. 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 [20][21] tarafından DES (Data Encryption Standards) algoritmasına uygulanmıştır.

Zaman odaklı önbellek saldırı türünün önemli bir örneği de Bernstein'in [22] uyguladığı saldırıdır. Bu saldırı AES (Advanced Encryption Standards)'ın 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 [22].

Atıcı ve arkadaşları öğrenme aşaması olmadan da saldırının gerçekleştirileceğini ispatlamışlardır [23]. Ayrıca AES'in son döngüsüne odaklanarak da saldırı gerçekleştirilmiştir. Bu çalışmada da 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 bu saldırıda uyarlanabilir. 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.



Şekil-2 : Bernstein Saldırı Şeması

Şekil 2'de gösterildiği gibi saldırı dört aşamadan oluşmaktadır. Bunlar öğrenme aşaması, saldırı aşaması, korelasyon analizi aşaması ve kaba kuvvet (brute force) saldırı aşamasıdır. Bu aşamalar sonrasında genellikle korelasyon analizi sonucunda gizli anahtara ulaşılmıştır. Ancak korelasyon analizinin yetmediği, gizli anahtarın tamamının elde edilemediği durumlarda kaba kuvvet saldırısı kullanılmaktadır [24].

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' 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 aşağıdaki denklem yardımıyla saldırı aşamasının bir zaman imzası oluşturulmaktadır.

$$x[j][b] = \frac{tn[j][b]}{n[j][b]} - \frac{\sum_j \sum_b tn[j][b]}{\sum_j \sum_b n[j][b]} \quad (1)$$

Her iki aşama için zaman imzası oluşturulduktan sonra korelasyon aşağıdaki denklemde gösterildiği şekilde hesaplanır.

$$c[j][b] = \sum_{i=0}^{255} x[j][i] * y[j][i \oplus b] \quad (2)$$

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 [25]. Gerçekleştirilen korelasyon analizi sonucunda bir anahtar uzayı elde edilmektedir. Bu anahtar uzayının bir örneği Şekil 3'te de gösterilmiştir.

```
 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-3 : 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 3'e göre gizli anahtarın 14. Baytı onaltılık olarak B5'tir.

3. Ağaç Tabanlı Modeller ile Yan Kanal Seçilimi

Gerçekleştirilen uygulama Linux tabanlı Ubuntu işletim sistemi üzerinde çalıştırılmıştır. AES algoritması OpenSSL'in 0.9.7a versiyonu ile uygulanmıştır. Önbellek saldırısı minimum 2^{32} tane açık metin örnekleri ile Intel (R) Core (TM) i5-4200M CPU @ 2.50GHz özelliklerine sahip makine üzerinde gerçekleştirilmiştir. Saldırı L1 önbellek hücresinde gerçekleştirilmiştir. Varyasyonları yakalamak için tüm adımların aynı önbellek seviyesinde uygulanması gerekmektedir. Rasgele fonksiyonları dinamik kütüphaneleri her yüklediğinde farklı önbellek adres alanları vermektedir. Bu da doğru korelasyon elde edilmesinin önüne geçmektedir. Bunu önlemek için uygulamaya başlamadan önce önbellek adresini sabitlemek gerekmektedir [23]. OpenSSL ile AES algoritmasının son döngüsünde temel olarak AES S-box operasyonunu uygulayan bir başvuru çizelgesio (T) 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 tablosunun çıktıları, zaman odaklı önbellek saldırı için kullanılan istatistiksel modelleri elde etmek için kullanılmaktadır. 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 : Saldırı Aşamasına Ait Örnek Veriler

Öğrenme aşamasında da elde edilen veriler korelasyon işlemi uygulanmak üzere kaydedilmiştir. Şekil 4'te 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 [22]. Öğrenme ve saldırı aşamalarından elde edilen öznitelikler aşağıdaki tabloda tanımlanmıştır.

Çizelge-1: Zaman Odaklı Önbellek Saldırılarında kullanılan Öznitelikler

Öznitelikler	Tanımı
m	T_0 başvuru çizelgesinin indeks değişkenini temsil eder
paket baytları	şifreleme için sunucuya gönderilen paketlerin boyutunu temsil eder
n	T_0 başvuru çizelgesinin indeks değişkenini temsil eder
paketler	şifreleme için sunucuya gönderilen paketlerin sayısını temsil eder
ortalama döngü	paketlerin ortalama döngüsünü ifade eder
döngü sapması	paketlerin döngü sapmasını temsil eder
en yüksek döngü	başvuru çizelgesinde seçilen m'ler için en yüksek döngüyü temsil eder
zaman sapması	paketlerin zaman sapmasını temsil eder

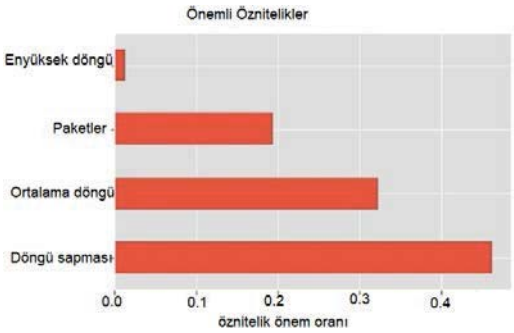
Tabloda belirtilen öznitelikleri içeren veri kümesi oluşturulduktan sonra, saldırı için en iyi sonucu almak üzere ağaç tabanlı modeller ve derin öğrenme modeli kullanılmıştır. Elde edilen veri kümesi, zaman odaklı önbellek saldırıları için yan kanal bilgilerini temsil etmektedir. Özniteliklerin önemi, örnekler arasındaki bir değişkenin, gerçek değeri ile model tahmin değerleri arasındaki farktan (loss) kaynaklanmaktadır [26].

Bu çalışmada toplamda 139264 zaman odaklı yan kanal bilgisi elde edilmiştir. Bu veri seti üzerinde önemli yan kanal bilgileri belirlenmiştir ve derin öğrenme ile saldırı tespiti yapılmıştır.

3.1 Karar Ağaçları

Bir karar ağacındaki özniteliğin önemi şu şekilde hesaplanır. Özniteliğin kullanıldığı tüm düğümlere göz atılır ve ilgili özniteliğin ana düğüme göre varyansı ne kadar azalttığı ölçülür. Tüm anlamlılığın toplamı 100 ile sınırlandırılır. Bu dönüşüm ile herhangi bir önem, modelin genel öneminin bir oranı olarak yorumlanabileceği anlamına gelir [27].

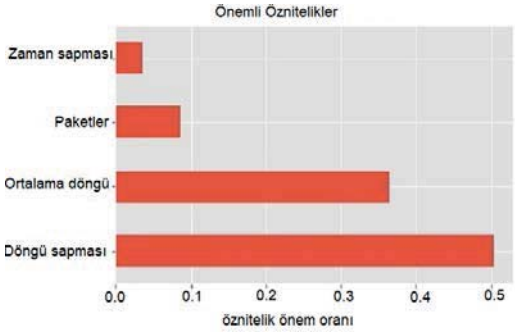
Çalışmamızdaki karar ağacı modeli Şekil 5'te gösterildiği gibi, *"döngü sapması"* özniteliğini 0.997934 AUC (eğri altındaki alan) değerine sahip önemli bir öznitelik olarak seçmiştir. Karar ağacına göre, AES Algoritması üzerinde gerçekleştirilen saldırı için *"döngü sapması"* özniteliği daha başarılı olmuştur.



Şekil-5 : Karar Ağacına Göre Önemli Öznitelikler

3.2 Gradyan Arttırma Modeli (GBM : Gradient Boosting Model)

Genel olarak, arttırma algoritmaları, zayıf tahmin modellerini yinelenen bir şekilde belirli kurallara göre birleştirerek güçlü bir tahmin modeli bulmaya çalışır. Gradyan artırma, ilk yineleme sırasında tahminler üreten bir f_1 işlevi oluşturur. Tahminler ve hedef değer arasındaki farkı hesaplar ve bu farklılıklar için f_2 işlevini oluşturur. İkinci yinelemede, f_1 ve f_2 işlevleri birleştirilir ve yeniden tahminler ile hesaplanan hedefler arasındaki fark birleştirilir. Bu şekilde, f_1 işlevinin başarısını sürekli ekleyerek ve tahminler ile hedefler arasındaki farkı sıfıra indirmeye çalışır. Şekil 6 *"döngü sapması"* özniteliğinin 0.999769 AUC değeri ile saldırı için daha belirleyici bir faktör olduğunu göstermektedir.

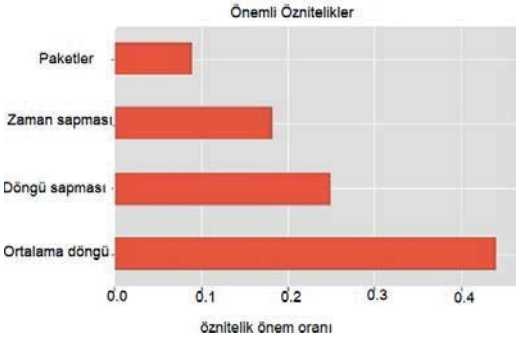


Şekil-6 : Gradyan Arttırma Modeline Göre Önemli Öznitelikler

3.3 Rastgele Ağaçlar

Rasgele orman yöntemlerinin gerektirdiği kullanıcı tanımlı parametrelerin sayısı daha azdır ve bu parametrelerin tanımlanması daha kolaydır. Bu nedenle, öznitelik seçimi için iyi bir ağaç modelidir [28].

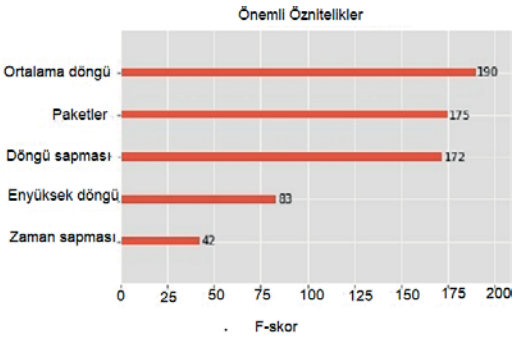
Doğrusal, statik olmayan fakat dinamik modellere benzer şekilde yorumlanabilirliğe olanak tanıyan orman rastgele tahminlerinin yorumlanmasını kolaylaştırmanın çok basit bir yolu vardır. Her bir tahmin modelinin katkılarının toplamı olarak gösterilebilir ve modellerin nasıl belirli bir tahmine yol açtığını gösterir. Şekil 7 *"ortalama döngü"* özniteliğinin 0.999775 AUC değeri ile saldırı için daha belirleyici bir faktör olduğunu göstermektedir.



Şekil-7 : Rastgele Ağaç Modeline göre Önemli Öznitelikler

3.4 Ekstrem Gradyan Arttırma (Extreme Gradient Boosting : XGBoost)

XGBoost'ta ağaçlar farklı sayıda terminal düğümüne sahip olabilir ve daha az ispat ile hesaplanan kalan ağaç ağırlıkları daha da azalır. Newton Boost, gradyan iniş olarak minimal bir düzeye yol sağlayan Newton-Raphson yaklaşım yöntemini kullanır. Ağaçlar arasındaki korelasyonu azaltmak için ilave rastgetirme parametreleri kullanılabilir. Şekil 8 *"ortalama döngü"* özniteliğinin 0.99981 AUC değeri ile saldırı için daha belirleyici bir faktör olduğunu göstermektedir. Bu modelde geçen F-skor metriği, ilgili özniteliğin ayrılan alt ağaçlarda kaç defa geçtiğidir. En çok geçen öznitelik en önemli özniteliği verir.



Şekil-8 : XGBoost Modeline Göre Önemli Öznitelikler

4. Derin Öğrenme ile Zaman Odaklı Yan Kanal Saldırı Tespiti

Derin öğrenme ile saldırı tespiti yapabilmek için öncelikle tespit edilecek saldırı ne tür bir probleme işaret ettiğini değerlenmek gerekir. Bu çalışmada kullanılan zaman odaklı önbellek saldırısı temelde iki aşamalı (öğrenme ve saldırı) olduğu için sınıflandırma problemi olarak ele alınmıştır. Saldırı esnasında elde edilen zaman profilleri, veri kümesi olarak kullanılmıştır. Derin yapay sinir ağı ile öğrenme sonucunda iki tane sınıf ortaya çıkacak şekilde çıktılar oluşturulmuştur. Bu şekilde saldırı aşaması ayrı sınıflandırılarak saldırı tespit edilmiştir. Saldırı aşamaları ile birlikte tespit sisteminin modelleme adımları aşağıda verilmiştir.

Saldırı Tespit Algoritması

Problem Tanımı: Zaman odaklı yan kanal saldırı tespiti ve önemli yan kanal bilgilerinin elde edilmesi

1.Adım: Önbellek adresini sabitleme

2.Adım: Şifreleme için sunucunun '00' anahtar ile başlatılması

3.Adım: Rastgele 600 adet paketin sunucuya gönderilip şifrelenmesi

4.Adım: Şifreleme zaman bilgilerinin ölçülmesi ve kaydedilmesi

5.Adım: Şifreleme için sunucunun rastgele anahtar ile başlatılması

6.Adım: Adım 3 tekrarlama

7.Adım: Adım 4 tekrarlama

8. Adım: Sinir ağ mimarisi oluşturma (Şekil 9)

9.Adım: Her bir epok için eğitimin başlatılması

10.Adım: Eğitim verisindeki hata oranında azalma varsa Adım 11, yoksa Adım 12 işletilir

11.Adım: Doğrulama verisindeki hata oranında azalma varsa Adım 9, yoksa Adım 12 işletilir

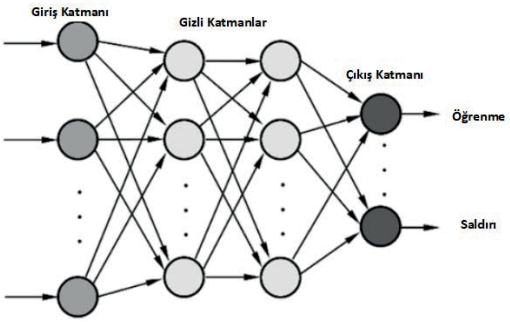
12.Adım: Eğitim verilerinde performans iyi ise adım 13, değilse Adım 8 yeniden iletilir

13.Adım: Test verilerinde performans iyi ise saldırı tespiti başarıya ulaşmıştır, değilse Adım 4' e dönülür.

Saldırı esnasında elde edilen veriler karıştırılıp ayrı ayrı eğitim, doğrulama ve test setlerine ayrıştırılır. Daha sonra model, bir seferde bir epok için eğitim setinde eğitilmiştir.

Her epok sonunda, eğitim seti ve doğrulama seti ile ilgili hatanın azalması sağlanır. Bunlardan biri iyileşmeyi durdurduğunda, modelin test verileri üzerindeki performansı yeterli olursa eğitim sonlandırılır. Performans yeterli olmaz ise mimari yeniden tasarlanır veya toplanan verilerin, yapmak istenilen tahmin için gereken bilgilere sahip olup olmadığını tekrar gözden geçirmesi gerekir.

Eğitimde hatanın iyileştirilmesi durduysa, muhtemelen verilerdeki önemli özellikleri yakalamak için daha iyi bir iş model gerekir. Doğrulama kümesinde hatanın iyileştirilmesi durduysa, muhtemelen aşırı öğrenme problemi (overfitting) ortaya çıkmıştır ve bunun için önlem alması gerekecektir [29]. Bununla birlikte, modelin eğitim verileri üzerindeki performansı yeterli ise modelin bu noktadan önce hiç görmediği test verileri üzerindeki performansı ölçülür. Eğitim sonucu yeterli değil ise veri setine daha fazla veri eklemek gerekir. Çünkü test seti, eğitim setinde iyi temsil edilmemiş örnek tiplerden oluşuyor anlamına gelir. Aksi durumda eğitim tamamlanmış ve problem çözülmüş demektir.



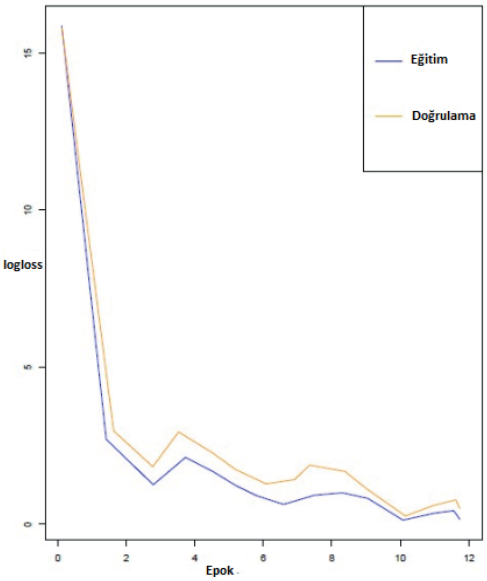
Şekil-9 : Derin öğrenme modeli

Derin Yapay Sinir Ağı, her biri 150 düğümlü iki gizli katmandan oluşur. Son modelde giriş değerlerinden en büyük olanı alan Maxout etkinleştirme fonksiyonu her öğrenme hızında 20 epok ile kullanılmıştır. Şekil 9 saldırı tespiti için önerilen derin öğrenme yapay sinir ağı modelini göstermektedir. Derin öğrenme sinir ağı modelinde her iki veri kümesinde veriler yüzde 70'i eğitim, yüzde 15'i doğrulama, yüzde 15'i test olarak ayrılmıştır. Derin öğrenme sinir ağı modeli

sonucunda öğrenme ve saldırı olmak üzere iki tane sınıf elde edilmektedir.

Gerçekleştirilen derin öğrenme modeli ile 0.9996647 AUC değeri ile saldırı aşamasının tespiti yapılmıştır. Şekil 10'' de görüldüğü gibi gerçek değer ile tahmin değer arasındaki fark (logloss) neredeyse 0'a yaklaşmıştır. Bu da uygulanan modelin saldırı tespiti için doğru eğitildiğini göstermektedir.

Literatürde [30], derin öğrenme diferansiyel güç saldırılarının karmaşıklığını azaltmak için kullanılmıştır. Bu çalışmada kriptografik bir cihazdaki bilgi sızıntısını değerlendirmek amacıyla derin öğrenme kullanılmıştır. Bizim çalışmamızda kullanılan model, zaman odaklı önbellek saldırıları açısından bildiğimiz kadarıyla daha önce çalışılmamış bir konudur. Gelecekteki çalışmalar düşünüldüğünde, önbellek saldırılarının tespiti ve değerlendirilmesi açısından derin öğrenme yöntemlerinin kullanılması hem saldırı hem de savunma tarafları açısından bir alternatif olabilir.



Şekil-10 : Derin öğrenme sonucu hata oranları

5. Sonuçlar

Bu saldırı için iki önemli öznitelik seçilmiştir. Bu özniteliklerden hangisinin daha iyi olduğu Tablo 2'deki AUC (eğri altındaki alan) değerlerine bakılarak anlaşılabilir. Uygulanan modeller arasında, XGBoost ve Rastgele ağaçlar en yüksek AUC değerlerine sahiptir. Bu modeller en önemli özniteliği *"ortalama döngü"* olarak belirlemiştir. GBM ve Karar Ağacı modellerinde ise "döngü sapması" özniteliği daha önemli olarak belirlenmiştir.

Çizelge-1: Zaman Odaklı Önbellek Saldırılarında kullanılan Öznitelikler

Ağaç Tabanlı Modeller	Yan Kanal Seçimi	AUC
Karar Ağaçları	döngü sapması	0.997934
GBM	döngü sapması	0.999769
Rastgele Ağaçlar	ortalama döngü	0.999775
XGBoost	ortalama döngü	0.99981

Bu sonuçlar karşılaştırıldığında, *"ortalama döngü"* özelliğinin zaman odaklı önbellek saldırıları için daha önemli bir yan kanal sızıntısı olduğu görülmektedir. Buna ek olarak derin öğrenme modeli ile saldırının tespiti 0.9996647 oran ile doğru bir şekilde gerçekleştirilmiştir.

Bu çalışmada ortaya konan modelin avantajları, saldırıyı iyileştirmek için önemli öznitelikleri ortaya çıkarması ve saldırı tespiti için eğitim başarısıdır. Bu açıdan çalışma, yan kanal saldırılarına bypass olarak kullanılabilecek makine öğrenmesi ve derin öğrenme yöntemleri ile gelecekteki çalışmalara bir kaynak teşkil edeceğini düşünmekteyiz. Modelin dezavantajı ise uygulanan saldırının özdeş cihazlar gerektirmesinden dolayı temiz veri elde edilmesinin zorlaştırmasıdır. Güvenlik çalışmalarında makine öğrenmesi ve derin öğrenme yöntemlerinin uygulanabilirliği açısından veri kümeleri elde etme oldukça sınırlıdır. Bu çalışmada zaman tabanlı önbellek saldırılarının AES algoritması üzerinde uygulanması ile ilgili kendi veri setimizi deneysel olarak oluşturduk ve derin öğrenme ile yapılan saldırı tespitinde gerçek verileri kullandık. Bu çalışmanın özgün yönleri iki tanedir. İlki zaman tabanlı önbellek saldırıları ile ilgili makine öğrenmesi algoritmaları ile elde edilen özgün ve gerçek veri setidir. Diğer özgün yönü, elde edilen veri setinin kullanıldığı derin öğrenme kullanılarak gerçekleştirilen yüksek doğruluk oranına sahip zaman tabanlı önbellek saldırılarının tespitidir.

Gelecekte, farklı mimariler üzerinde makine öğrenmesi ve derin öğrenme kullanılarak yan dal

saldırılarının tespitindeki başarımlar incelemek için çalışmalar yapılması hedeflenmektedir.

Teşekkür

Bu çalışma MAP-2017-40642 numaralı BAP projesi kapsamında İstanbul Teknik Üniversitesi tarafından desteklenmektedir.

Kaynakça

[1] Daemen, J., & Rijmen, V. (2013). *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media.

[2] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). ACM.

[3] Mangal, A., & Kumar, N. (2016, December). Using big data to enhance the bosch production line performance: A Kaggle challenge. In *2016 IEEE International Conference on Big Data (Big Data)* (pp. 2029-2035). IEEE.

[4] Zhou, Z. H., & Feng, J. (2017). Deep Forest. *arXiv preprint arXiv:1702.08835*.

[5] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

[6] Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*(1), 3-42.

[7] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

[8] Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119-139.

[9] Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, *28*(2), 337-407.

[10] Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, *7*, 21.

[11] Page, D. (2002). Theoretical use of cache memory as a cryptanalytic side-channel. *IACR Cryptology ePrint Archive*, *2002*(169).

[12] Patterson, D. A., & Hennessy, J. L. (2013). *Computer organization and design MIPS edition: the hardware/software interface*. Newnes.

[13] Canteaut, A., Lauradoux, C., & Seznec, A. (2006). Understanding cache attacks.

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

[15] Koc, C. K. (2009). About cryptographic engineering. In *Cryptographic engineering* (pp. 1-4). Springer, Boston, MA.

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

[17] Acıiçmez, O., Brumley, B. B., & Grabher, P. (2010, August). New results on instruction cache attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 110-124). Springer, Berlin, Heidelberg.

[18] Aciçmez, O., & Koc, C. K. (2006). Trace-driven cache attacks on AES.

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

[20] Tsunoo, Y., Saito, T., Suzaki, T., Shigeri, M., & Miyauchi, H. (2003, September). Cryptanalysis of DES implemented on computers with cache. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 62-76). Springer, Berlin, Heidelberg.

[21] 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.

[22] Bernstein, D. J. (2005). Cache-timing attacks on AES.

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

[24] Neve, M. (2006). Cache-based Vulnerabilities and SPAM analysis. *Doctor thesis, UCL*.

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

[26] Palczewska, A., Palczewski, J., Robinson, R. M., & Neagu, D. (2014). Interpreting random forest classification models using a feature contribution

method. In *Integration of reusable systems* (pp. 193-218). Springer, Cham.

[27] Molnar, C. (2018). Interpretable machine learning: A guide for making black box models explainable. *E-book at< https://christophm. github. io/interpretable-ml-book/>, version dated*, *10*.

[28] Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217-222.

[29] Buduma, N., & Locascio, N. (2017). *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.".

[30] Perin, G., Ege, B., & van Woudenberg, J. (2018). Lowering the bar: Deep learning for side channel analysis (White-Paper). In *Proc. BlackHat* (pp. 1-15). (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217-222.