İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK BAŞARIMLI YAZILIM TABANLI IPSEC GÜVENLİK GEÇİDİ TASARIMI

YÜKSEK LİSANS TEZİ

Müh. Ural ERDEMİR
504031530

Tezin Enstitüye Verildiği Tarih : 8 Mayıs 2006
Tezin Savunulduğu Tarih : 19 Haziran 2006

Tez Danışmanı : **Yrd.Doç.Dr. Feza BUZLUCA (İ.T.Ü)**

Diğer Jüri Üyeleri: **Prof.Dr. Coşkun SÖNMEZ (Y.T.Ü)**

Doç.Dr. Sema OKTUĞ (İ.T.Ü)

Haziran 2006

ÖNSÖZ

Tez çalışmalarım süresince göstermiş olduğu anlayış ve yardımlardan dolayı tez danışmanım sayın Yrd. Doç. Dr. Feza Buzluca' ya, katkılarından dolayı çalışma arkadaşım Umut Tekin' e, hayatımın her anında olduğu gibi tez çalışmalarım boyunca da benden yardımlarını esirgemeyen aileme sevgi ve saygılarımı sunarım.

Haziran 2006 Ural ERDEMİR

İÇİNDEKİLER

Sayfa No

KISALTMALAR

AES	: Advanced Encryption Standard
AH	: Authentication Header
ARP	: Address Resolution Protocol
CBC	: Cipher Block Chaining
CIDR	: Classless Inter-Domain Routing
CRC	: Cyclic Redundancy Check
CSMA / CD	: Carrier Sense Multiple Access with Collision Detection
DUT	: Device Under Test
ESP	: Encapsulating Security Payload
HMAC	: Keyed-Hashing for Message Authentication Code
Gbps	: Gigabits per second
GoF	: Gang of Four, Gamma E., Helm R., Johnson R., Vlissides J.
GRASP	: General Responsibility Assignment Software Patterns
ICMP	: Internet Control Message Protocol
ICV	: Integrity Check Value
IETF	: Internet Engineering Task Force
IP	: Internet Protocol
IPSec	: IP Security, Security Architecture for the Internet Protocol
IPv6/IPng	: IP version 6/ IP Next Generation
ISAKMP	: Internet Security Association and Key Management Protocol
IV	: Initialization Vector
MAC	: Message Authentication Code
Mbps	: Megabits per second
MD5	: Message Digest 5
NAT	: Network Address Translation
NIST	: National Institute of Standards and Technology
PPS/PBS	: Packet per second / paket bölü saniye
RED	: Random Early Detection
QoS	: Quality of Service
RFC	: Request For Comment
RSA	: Rivest-Shamir-Adleman Güvenlik Firması
SA	: Security Association
SAD	: Security Association Database
SHA1	: Secure Hash Algorithm
SPD	: Security Policy Database
SPI	: Security Parameter Index
TCP/IP	: Transmission Control Protocol / Internet Protocol
TCAM	: Ternary Content Addressable Memory
TFC	: Traffic Flow Confidentiality
TTL	: Time to Live
UP	: Unified Process
VPN	: Virtual Private Network, Sanal Özel Ağ

TABLOLAR

Sayfa No

ŞEKİLLER

Sayfa No

YÜKSEK BAŞARIMLI YAZILIM TABANLI IPSEC GÜVENLİK GEÇİDİ

ÖZET

Veri haberleşmesinde güvenlik konusu günümüzde üzerine en çok araştırma yapılan konulardan biri haline gelmiştir. Özellikle kaynak ve zaman paylaşımının yapıldığı bilgisayar ağları gibi sistemlerde, verilerin korunması ve saldırıların önlenmesi için bir çok yöntem tasarlanmıştır. Tasarlanan yöntemler temelde verinin iletimi esnasında korumasını esas almaktadır. Bugün bilgisayar ağlarında kullanılan temel iletişim protokolü IP protokolüdür ve IPSec, ağ katmanında IP protokolü için tasarlanmış yaygın kullanımlı standart güvenlik mimarisidir.

IP (IPv4) ile birlikte isteğe bağlı olarak genellikle kurumsal bazda şıkça kullanılan IPSec' in yeni nesil IP (IPv6 ya da IPng) protokolüyle birlikte kullanımı zorunlu hale getirilmiştir. Artan güvenlik gereksinimleri ve IPv6'nın yaygınlaşmasıyla beraber IPSec' in kullanımı dünya genelinde daha da artacağı düşünülmektedir. IPSec' in bu şekilde artan kullanımı sonucunda IPSec yapan güvenlik geçitlerinin başarımları konusunda bugün için düşünülmeyen yeni problemler ve araştırma konuları ortaya çıkacaktır. IPSec güvenlik politikası veritabanlarının, kullanıcı sayısı ile doğru orantılı olarak büyümesi, güvenlik politikası veritabanlarında kararlar ile paketlerin eşleşmesini zorlaştırarak, IPSec başarımında ciddi bir dar boğaz oluşturacaktır.

Bu konuda ortaya çıkan problemler için çeşitli çözümler düşünülse bile günümüz ağ cihazlarının statik ve değişime açık olmayan tasarımları bu çözümlerin gerçekleştirilmesini zorlaştırmakta hatta çoğu zaman imkansız hale getirmektedir. Ağ cihazlarının mimarilerinin donanım tabanlı olması bunun olmasının en büyük sebebidir. Ancak gelişen teknoloji sayesinde bugün işlemci hızları çok yüksek mertebelere ulaşmış ve bu gibi sistemlerin yazılım tabanlı olarak da yüksek başarımlı bir biçimde yapılabilmesine imkan vermiştir.

Bu tez çalışması kapsamında öncelikle yüksek başarımlı ve yazılım tabanlı IPSec' yapan bir güvenlik geçidi tasarlanıp gerçeklenmiştir. Tasarım nesneye dayalı bir biçimde, modülerlik ve esneklik ön planda tutularak yapılmıştır. Gerçeklenen modüler ve esnek yapı sayesinde yeni fikirlerin kısa sürede gerçeklenip sonuçlarının alınabileceği örnekler ile gösterilmiştir. IPSec' protokolünde güvenlik politikası veritabanının boyutlarının büyümesinin cihaz başarımına etkileri test sonuçları gösterilmiş. Bu problem için çeşitli çözümler önerilmiş ve test sonuçlarıyla bu çözüm önerilerin sonuçları ortaya konmuştur. Bu çalışma sonucunda karar tablosunun boyutundan çok az etkilenen yüksek başarımlı ve yazılım tabanlı modüler bir güvenlik geçidi gerçeklenmiştir. Normal bir PC donanımı üzerinde yapılan testler sonucunda 16.000 kural sayısında bile yaklaşık 700.000 pps gibi çok yüksek bir başarım elde edilmiştir.

HIGH PERFORMANCE SOFTWARE BASED IPSEC SECURITY GATEWAY

SUMMARY

Nowadays, security has become one of the most interesting research issues in data communication, especially in computer networks where resource and time sharing is very common. There exist many standard systems designed for protection of data and preventing attacks. They are based on protection of data at the time of transmission.

Today IP is the most popular protocol of the computer networks. Because security is not inherently built into the protocols of IP, securing IP traffic has largely been a chore for the higher layers. IPSec provides security services at the network layer. It is commonly used to refer to the secure IP packets of the AH and ESP protocols, because these provide the major security services. IPSec design is a framework for multiple services, algorithms and granularities The security services that IPSec provides can include access control, authentication (through data origin authentication and connectionless integrity), packet anti-replay protection, confidentiality through encryption, and limited traffic flow confidentiality. The new IPSec standards (IPSec version 3) defined by IETF, in December 2005.

After its proven success in IPv4 and by making IPSec a mandatory part of IPv6, it is expected that IPSec will be much more widely used. On the other hand, the bandwidth requirement and number of nodes in network is growing exponentially, these will bring up new problems and research issues. With increasing number of rules in the security policy databases, packet classification will be one of the significant performance bottlenecks for IPSec security gateways. Another important topic for security devices is the reliability of the cryptographic algorithms and protocols they use. A reliable algorithm for today can be broken and become unreliable in the near future. In these cases, there will be a need for adapting new algorithms and protocols in a short critic time interval.

Even there exist some suggestions for such problems; because of their closed, static, and inflexible designs applying them in today's network devices is very difficult and mostly impossible especially for hardware based network processors.

In this thesis, we designed a high performance, its object oriented software based IPSec security gateway. The most significant features of our design are its modularity, extensibility and flexibility. It is shown that, new ideas can be integrated easily with these features. Also in this study, a new scheme for IPSec policy search presented and its performance measured

Designed software runs in the Linux kernel and for a size of 16,000-rules security policy database, its maximum IPSec forwarding rate is 700,000 64-byte packets per second on a conventional PC hardware.

1 GİRİŞ

İçinde bulunduğumuz bilgi çağının temel gerekleri bilgiye ulaşım, bilginin paylaşımı ve etkin kullanımıdır. Dünyayı kuşatan telefon ağlarının kurulması, radyo ve televizyonun icadı, bilgisayar endüstrisinin doğuşu ve benzersiz büyümesi, iletişim uydularının fırlatılması bu çağın en önemli gelişmeleri olarak sıralanabilir. Teknolojinin gelişmesi ile artan, bilgiye hızlı ve kolay ulaşma ihtiyacı, Internet'i geçtiğimiz yüzyılın en önemli buluşlarından biri yapmıştır. Internet en genel anlamda birbirine bağlı bilgisayarların oluşturduğu ağların ağı olarak tanımlanabilir.

60'lı yılların sonunda, askeri bir araştırma projesi olarak ortaya çıkan Internet, o yıllardaki klasik devre anahtarlamalı telefon ağlarından farklı paket anahtarlamalı çalışacak şekilde tasarlanmıştı ve asıl amacı birbirine bağlı ağlar üzerinden, bu ağlardan bir kısmı çalışmasa bile bağımsız olarak haberleşmeyi sürdürebilmekti.

Internet'i kısa zamanda bu denli popüler yapan, milyonlarca insana fikirlerini, bilgilerini, çalışmalarını paylaşacak bir ortam sağlaması. ve çıkış felsefesinde yatan açık mimari, baskın bir kontrol merkezinin olmaması , herkese kolayca erişim imkanı sunma düşünceleridir[1]. 90'lı yıllarda büyük ticari şirketlerin ve akademik çevrenin Internet altyapısını toplu iletişim aracı olarak kullanmaya başlaması ve WWW' in gelişmesi IP'nin yaygınlaşmasını hızlandırmıştır.

Internet ağlarını, bir arada tutan temel yapı taşı Internet'in ağ katmanı protokolü IP[1]'dir. IP 20 yıl öncesinin ihtiyaçlarına göre tasarlanmış bir protokoldür. O yıllarda bilgisayar ağları üniversite araştırmacıları arasında elektronik posta yollama, dosya paylaşımı, haber gruplarına erişme, yazıcı paylaşma gibi uygulamalarda kullanılıyordu ve bu tür uygulamalarda öncelikli konu iletişim güvenliği değildi. Ancak 90'lı yıllardan sonra Internet'in milyonlarca kişinin bankacılık işlemlerini yaptığı, elektronik ticaret vb. parasal uygulamalarda, şirketlerin ve kurumların özel verilerini taşıyan kritik uygulamalarda kullanılmaya başlamasıyla iletişim güvenliği önemli bir boşluk olarak ortaya çıkmıştır.

[1] Tez kitapçığında IP, Internet Protokol versiyon 4'ün kısaltması olarak kullanılmaktadır. Internet Protokol versiyon 6, IPv6 olarak ayrıca vurgulanmaktadır.

Internet'in özel sektörde yaygınlaşması ile beraber çıkan diğer bir konu da iç ağ ve dış ağ kavramlarıdır. İç ağ belli bir kurumun Internet ağ teknolojilerini (TCP/IP) kullanarak oluşturduğu kurum içi özel iletişim ağıdır. Dış ağ, kurumların iç ağlarını oluştururken, coğrafi olarak birbirinden uzakta bulunan şube veya ofislerini, Internet gibi herkese açık bir iletişim ortamı üzerinden bağlamalarıyla oluşan özel ağdır. Böyle bir ağda kuruma ait özel bilgiler herkese açık bir ortamdan geçirileceği için iletişim güvenliğinin sağlanması şarttır. Bu amaçla oluşturulan Sanal Özel Ağlar (VPN) kişilere ya da kurumlara ait özel bilgilerin Internet gibi herkese açık ağlar üzerinden güvenli bir şekilde iletilmesini sağlar. Sanal özel ağlar ile sadece güvenilir ve yetkilendirilmiş sistemlerin hassas ve önemli verilere ulaşması sağlanabilir. Geleneksel ortamda kurumlar, kiralık hatlar ya da farklı ortamlar ile Internet'e açık olmayan iletişim hatlarını kullanarak güvenli haberleşmeyi sağlayabilirler. Ancak bu yöntem hem ekonomik hem de esnek olmamaktadır.

IPSec, IETF tarafından tanımlanmış Internet Protokolü güvenlik standardıdır. IPSec mimarisi ile kendi içinde güvenli ağlar, güvensiz ağlar üzerinden haberleştirilerek sanal özel ağlar oluşturulabilir. Bu tip VPN uygulamasında temelde biri kullanıcı-güvenlik geçidi ve diğeri güvenlik geçidi-güvenlik geçidi olarak adlandırılan iki tür bağlantı türü yapılabilir. Bu yapıda güvenilmeyen ağdan geçen paketler IPSec güvenlik geçidi tarafından şifrelenmesine ve diğer uçtaki güvenlik geçidinde karşı ucun kimlik kontrolü yapılıp, paketin şifresi çözülmesi dayanır. Böylece iki uç arasında dışarıya mantıksal olarak kapalı bir tünel bağlantısı oluşturulur.

Bilgisayar ağlarındaki düğüm sayısı ve band genişliği ihtiyacı üstel bir şekilde artarken, güvenlik tehditleri de artmakta ve iletim güvenliği firmalar için kaçınılmaz olmaktadır. Güvenli iletişim elbette, iletişim alt yapısı üzerinde bazı ek maliyetler de getirmektedir. Bu maliyetler başta güvenlik yönetimi işlemleri ve ağ başarımında azalmadır.

Son zamanlarda IPSec' in sanal özel ağ uygulamalarında yaygınlaşması ile IPSec güvenlik geçidi tasarımı önemli bir konu haline gelmiştir. Kullanıcılar IPSec güvenlik geçidi ürünlerinden, kolay yönetilebilirlik, ucuzluk, yüksek başarım beklerken, üretici firmalar rekabet edebilmek için bu isteklere ek olarak, hızlı ürün çıkarabilme, modüler yapılar ile tekrar kullanılabilirlik, güncelleme hata ve bakım maliyetinin azlığını istemektedir. Donanım tabanlı sistemler yüksek başarım vermelerinin yanında, yazılım tabanlı sistemlere kıyasla maliyetleri oldukça

yüksektir. Günümüzde genel amaçlı işlemcilerin başarımı giderek artmakta ve zamanla daha yüksek başarımlı işlemciler ucuza alınabilmektedir.

Güvenlik gibi kritik uygulamalarda güncelleme oldukça önemli bir konudur. Dün güvenli olan bir algoritma ya da sistem bugün güvenliliğini yitirmiş olabilir, örneğin MD5[2] ve SHA1[3] algoritmalarında zayıflıklar bulunması[4,5] bu algoritmaları kullanan güvenlik geçitlerinin, dolayısıyla haberleşme bağlantılarının güvenliğini azaltmıştır. Kritik tasarım hataları da güvenlik sistemi üzerinde güncellemeler gerektirebilir. Yazılım tabanlı sistemlerde bu tip sorunlar basit bir yazılım güncellemesi ile çözülebilecekken donanım tabanlı sistemlerde durum her zaman bu kadar kolay olmamaktadır. Hata durumlarında güncellemenin bu denli yüksek olması donanım tabanlı sistemlerin test süreçlerinin de daha uzun sürmesini gerektirmekte ve toplam maliyeti arttırmaktadır.

Bu tez çalışmasında yeni standartlarla tanımlı IPSec (versiyon 3)[6] mimarisi incelenmiş ve yazılım tabanlı yüksek başarımlı bir IPSec güvenlik geçidi tasarlanmaya çalışılmıştır. Yazılım tabanlı sistemlerin en büyük zayıf noktası olan başarım dar boğazları incelenmiş ve başarımı arttıracak çalışmalarla sistemin limitleri belirlenmeye çalışılmıştır.

Tez kitapçığında öncelikle IPSec güvenlik mimarisi hakkında kısaca bilgi verilmiştir. Üçüncü bölümde tasarlanan sistemin mimarisinde büyük etkisi olan Click yönlendirici yazılım mimarisi tanıtılmıştır. Dördüncü bölümde esnek, modüler, hata bağışıklığı yüksek hedef sistemin nesneye dayalı modellenmesi ve tasarımı anlatılmıştır. Beşinci bölümde yazılım tabanlı IPSec güvenlik geçidi başarımında ciddi bir dar boğaz oluşturan Güvenlik Politika Veritabanlarında paket eşleşmesi için çok boyutlu paket sınıflandırma problemi ve önerilen çözümler anlatılmıştır. İzleyen bölümde tasarlanan sistem üzerinde yapılan önemli iyileştirmelerden bahsedilmiştir. Son olarak yapılan iyileştirmelerin tasarlanan sistemin başarımına etkileri ölçülmüş ve sonuçlar tartışılmıştır.

2 IPSEC

IPSec, IP ağlarının güvenliği için IETF tarafından tanımlanmış Internet Protokolü güvenlik standardıdır. Kendi içerisinde güvenli yerel alan ağlarının, güvensiz ağlar (örneğin Internet) üzerinden güvenli haberleşmesini sağlar [6].

IPSec' in avantajları şu şekilde sıralanabilir:

- Tüm üst katmanlar(3.katman ve yukarısı) için güvenlik sağlar.
- Uygulamalarda değişiklik gerektirmez.
- Ağdaki tüm trafik korunur (ICMP v.b.).
- Yönlendiricilerle birlikte kullanıma uygun.
- Ayrı bir cihaz olarak gerçekleştirilmeye uygun.(düşük maliyet, Esneklik, kolay yönetilebilirlik)

2.1 Güvenlik Mimarisi

IPSec güvenliği IP(IPv4 ve IPv6) ağ katmanı için sağlar.

IPSec ile sağlanabilen güvenlik hizmetleri şunlardır:

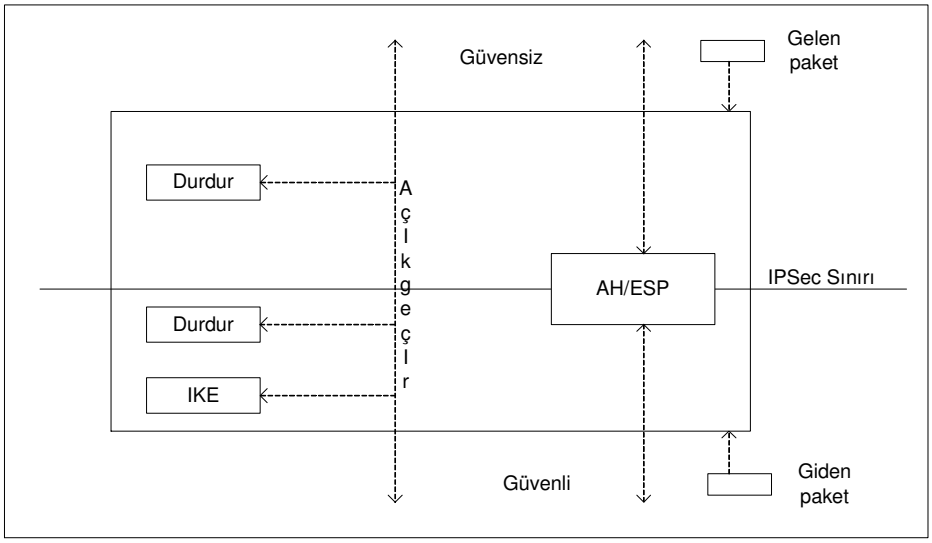
- Veri gizliliği (Confidentiality): Gizlililik, verinin sadece izin verilen kişilerin izin verilen yollarla erişimini garanti etmektir. ve şifreleme ile sağlanır
- Veri kaynağının asıllanması (Data origin authentication): Veri göndericisinin kimliğinin doğrulanmasıdır.
- Veri bütünlüğü (Integrity): Verinin iletim esnasın değişmediğinin garanti edilmesidir. Şifreleme, mesajların anlaşılamamasını sağlamaktadır, ancak iletilen mesajın değiştirilmesine karşın bir etkinliği yoktur.
- Yinelenmiş paketlerin reddedilmesi (Anti-replay protection): aynı iletim verisinin, ağdaki herhangi bir kişi tarafından tekrar gönderildiğinin fark edilip verinin atılmasıdır.

- Kısmi Trafik akış gizliliği (Limited traffic flow confidentiality): İç IP adreslerinin ve akan trafik miktarının gizlenmesidir.
- Güvenlik derecesine göre öğelerin ayrımı, trafik filtreleme

IPSec güvenli ve güvensiz arayüzler arasında bir sınır oluşturur. (Şekil 2.1) Bu sınır üzerinden geçen tüm trafik akışı, IPSec konfigürasyonundan sorumlu kullanıcı tarafından belirlenen erişim kontrollerinden geçer.

Bu kontroller sonucu paket üzerinde 3 temel işlev yapılabilir.

- Paket açık bir şekilde sınırdan geçirilebilir.(Bypass)
- Paket sınırdan geçirilmez ve atılabilir. (Discard)
- Pakete ESP ya da AH ile güvenlik servisleri uygulanabilir.(IPSec)



Şekil 2.1: Üst seviye IPSec modeli

Bu bağlamda güvenli arayüz kırmızı arayüz, güvensiz arayüz siyah arayüz olarak ta adlandırılmaktadır. Güvenli arayüz dış fiziksel bir arayüz olabileceği gibi, dahili bir arayüz de olabilir. (Örneğin işletim sistemi tarafından tanımlanmış bir soket katmanı arayüzü). Bu yazı boyunca giren trafik (inbound) terimi güvensiz taraftan gelen ve güvensiz tarafa yönlendirilen paketler için, çıkan trafik (outbound) terimi, güvenli taraftan gelen ve güvensiz tarafa yönlendirilen paketleri ifade etmektedir. IPSec gerçeklemeleri sınırın her bir tarafı için birden fazla fiziksel ve mantıksal arayüze sahip olabilirler.

IPSec tasarımı, çoklu güvenlik servisleri, algoritmalar ve taneciklikler için bir çatı oluşturur.

Çoklu güvenlik servisleri sağlamaktaki amaç, herkesin her zaman her güvenlik servisine ihtiyaç duymamasıdır. Sözgelimi bir uygulama için, bütünlük ve asıllama servislerine ihtiyaç duyulurken , gizlilik servisi gereksiz olabilir.

Çoklu algoritmaların kullanılabilmesine destek verilmektedir çünkü bugün güvenli olduğu düşünülen bir algoritma gelecekte kırılarak güvensiz hale gelebilir. Böylece bazı algoritmalar zamanla güvensiz hale gelse de IPSec çatısı güvenilirliğini sürdürür. Ağ ortamında yüksek başarım ihtiyacı nedeniyle genellikle simetrik algoritmalar kullanılır[7].

Çoklu tanecikliğe destek verilmesi ile sadece tek bir TCP bağlantısının, iki konak arasındaki tüm trafiğin ya da iki yönlendiriciden geçen tüm trafiğin korunması gibi değişik parçacıklıkta güvenlik hizmetleri verilebilir.

IP bağlantısız olmasına rağmen IPSec bağlantı tabanlıdır. Çünkü güvenliğin sağlanabilmesi için en azından bir anahtarın kurulmasına ve belirli bir süre kullanılmasına ihtiyaç vardır. IPSec bağlamında, "bağlantı" Güvenlik Birliği-Bağlantısı (Security Association - SA) olarak adlandırılır. SA iki uç arasında tek yönlü bir bağlantıdır ve her SA bir güvenlik tanımlayıcısına sahiptir. Eğer her iki yönde güvenlik isteniyorsa, iki güvenlik birliğine ihtiyaç duyulur. Güvenlik tanımlayıcıları güvenli bağlantılar üzerinde akan paketler içinde taşınırlar ve güvenli paketin işlenmesi için gerekli anahtar gibi güvenlik parametrelerinin bulunmasında kullanılırlar.

IPSec temelde iki ana parçadan oluşur. İlk parça, pakete eklenecek, güvenlik tanımlayıcısı, bütünlük kontrol değeri ve diğer bilgileri taşıyan iki başlık yapısını tarif eder. İkinci parça ise anahtarların kurulması ile ilgilenen ISAKMP[8]' dir.

IPSec iletim kipi(transport mode) ve tünel kipi(tunnel mode) olmak üzere iki kipte kullanılabilir. İletim kipinde IPSec başlığı IP başlığından hemen sonra gelir ve IP başlığındaki Protokol alanı IPSec başlığının takip ettiğinin gösterir. Tünel kipinde gerçek tüm IP paketi (IP başlığı , IPSec başlığı vs) tamamen yeni bir IP başlığı tarafından kapsüllenir edilir. Tünel kipi, tünelin son uca varmadan önce başka bir düğümde bitmesi ile ayrı bir IPSec güvenlik geçidinde gerçeklemeye uygundur. Böylece yerel ağdaki diğer bilgisayarlar IPSec' ten habersiz olarak güvenli haberleşebilirler. Tünel kipi ile birlikte yerel ağdaki iç ağ adreslerinin saklanması

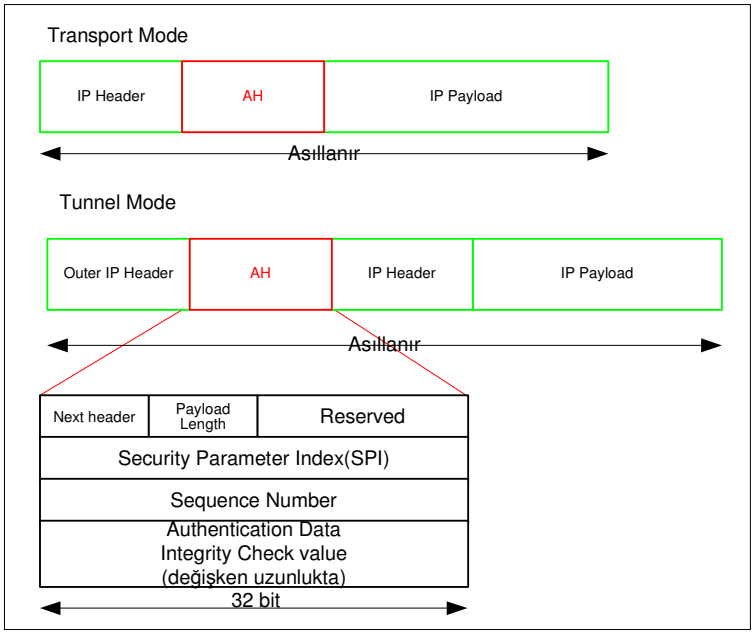
sağlanır böylece güvensiz bölgede hangi iki bilgisayarın birbiriyle ne kadar iletişim kurduğu anlaşılamaz ve kısmi trafik akış gizliği servisi sağlanmış olur.

2.2 Güvenlik Protokolleri

IPSec veri düzleminde(data plane) 2 protokol tanımlar. Bunlar ESP[9] ve AH[10] protokolleridir. AH ve ESP VPN geçitleri arasındaki gerçek veri trafiğini korumak amacıyla kullanılır.

2.2.1 AH

AH (Authentication Header) IP paketinin bütünlüğünü doğrulamak ve asıllamak için kullanılan bir protokoldür. IP paketinde bulunan veriden MAC (Mesaj asıllama kodu) oluşturmak için kriptolojik özet fonksiyonları (HMAC[11]) kullanır. Elde edilen MAC, karşı tarafa, mesajın bütünlüğünün korunduğunun anlaşılması için orijinal paket ile birlikte iletilir. AH Şifreleme yapmaz. AH protokolü IP paket verisinin yanı sıra dış IP başlığının da bütünlüğünü doğrular. IP başlığının, TTL gibi aktarım sırasında değişen kısımlarını bütünlük hesabına katmaz. AH protokolü, IP başlığından sonra AH başlığı yerleştirir. Şekil 2.2 'de AH paket yapısı ayrıntısıyla verilmiştir.



Şekil 2.2: AH paket yapısı

Next Header: AH başlığından sonraki protokol alanı değerini içerir. Orijinal IP başlığındaki protokol değeri AH' ın protokol değeri(51) olarak değişeceğinden , gerçek değerin bu şekilde saklanmasına ihtiyaç vardır. Örneğin aktarılan paket TCP paketi ise burada 6 değeri yazacaktır

Payload Length: AH başlık uzunluğunun 32-bitlik sözcük uzunluğu – 2 değerini taşır. Örneğin bütünlük kontrol değeri 12 sekizli ise , bu değer (3+3) –2, 4 olacaktır.

Reserved: Ayrılmış 2 sekizli alan, bu alan her zaman 0 olmalıdır.

Security Parameter Index: Güvenlik bağlantı tanımlayıcıdır. Bağlantının alıcı veri tabanında bulunmasında gönderici IP adresi ile kullanılan 32 bitlik değerdir.

Sequence number: Tekrarlama ataklarına karşı koruma için 32 bitlik sıra numarası. İlgili SA' dan gelen her pakete ayrı bir numara verilir. Sıra numarası taşar ise yeni bir SA kurulmalıdır. IPSecv3 ile 64-bitlik genişletilmiş sıra numarası desteği vardır. Bu durumda sıra numarasının düşük anlamlı 32 biti paketle birlikte aktarılmaktadır. yüksek anlamlı 32 biti her iki taraf kendi içinde tutmakta ve yüksek anlamlı 32 bit bütünlük hesabında işleme katılmaktadır.

Authentication Data- Integrity Check Value: Verinin sayısal imzasını taşıyan değişken uzunlukta alandır. Bu alanın uzunluğu kullanılan algoritmaya bağlı olarak değişir. SA kurulduğunda iki taraf kullanılacak algoritma ve anahtar üzerinde anlaşırlar. Genellikle açık anahtarlı kriptolojik algoritmalar kullanılmaz. Hızlı olması açısından HMAC[11] yöntemi ile özet hesaplama işlemine ortak anahtar katılır ancak anahtar paketle birlikte aktarılmaz. Böylece paket hem asıllanmış hem de bütünlük kontrolü yapılmış olur.

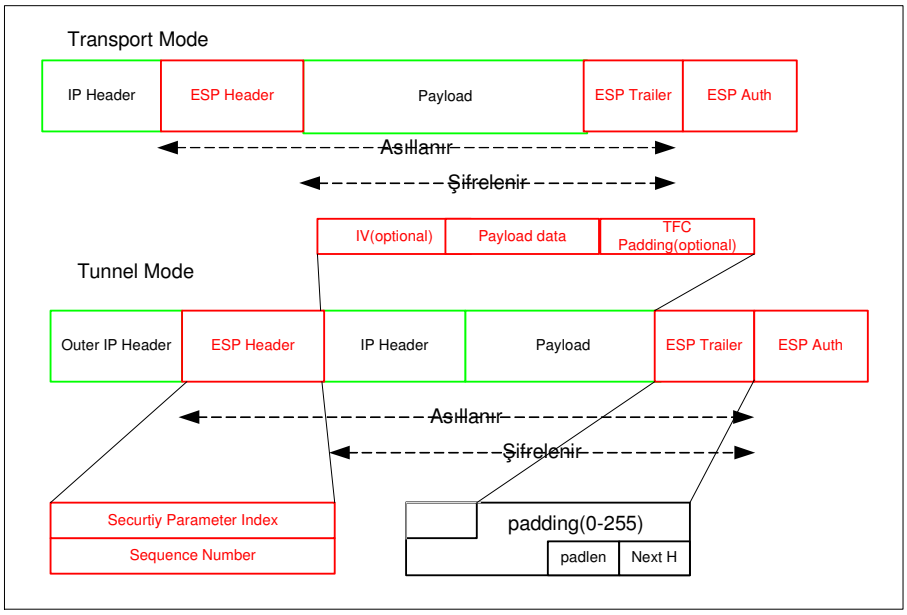
2.2.2 ESP

ESP (Encapsulating Security Payload) protokolü IP paketinin hem şifrelenmesi hem de asıllanması için kullanılır. ESP protokolü IP paketinin sadece şifrelenmesi veya sadece doğrulanması için de kullanılabilir. Opsiyonel olarak veri kaynağının doğrulama, veri bütünlüğünü sağlama ve paket tekrarlamalarına karşı koruma servislerini verir. ESP protokolü IP başlığından sonra ESP başlığı yerleştirir. ESP başlığından sonraki tüm veri şifrelenmektedir ve/veya asıllanmaktadır. AH protokolünden farkı; ESP IP paketinin şifrelenmesini sağlamaktadır.

ESP paket yapısı Şekil 2.3'de ayrıntılı olarak gösterilmiştir. ESP başlığı iki 32-bitlik sözcük içerir, SPI ve sıra numarası, bunların kullanımı AH protokolündeki ile aynıdır. Genellikle bu alanı İlklendirme Vektörü(Intialization Vector-IV) izler. IV ESP başlığının bir parçası değildir ve şifreleme algoritmasının CBC kipte çalıştırıldığı durumlarda kullanılır. IV' nin kullanılıp kullanılmayacağı ve kullanılacaksa boyu şifreleme algoritmasına ve algoritma kipine bağlıdır.

Bazen aktarılan veri kadar verinin boyutu da akan trafik hakkında bilgi verebilir. Böyle durumlar için ESPv3'te ek olarak Trafik Akış Gizliği doldurması (Traffic Flow Confidentiality-TFC padding) ile gerçek paket boyu değiştirilerek ek bir koruma getirilir.

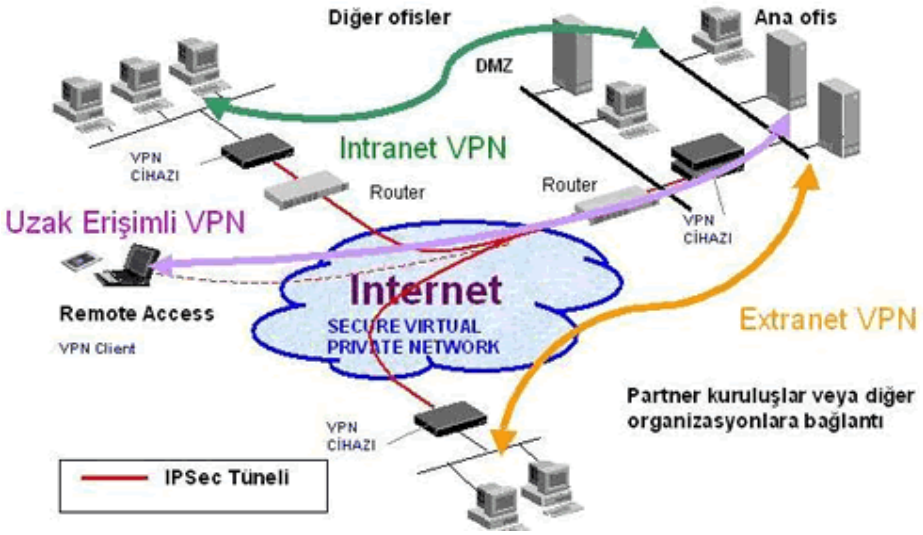
Şifreleme algoritmasının blok kipte çalışması durumunda, verinin şifreleme algoritmasının blok boyunun tam katı olması için paketin sonuna doldurma yapmak gerekebilir. Bu durumda sona doldurma eklenir ve doldurma boyu ESP kuyruk alanına yazılır. Doldurma miktarı ve doldurma şekli kullanılan şifreleme algoritmasına bağlıdır. Sonraki başlık (Next Header), alanı AH' dekine benzer şekilde ESP başlığından sonraki protokol alanı değerini içerir. ESP de AH gibi HMAC bütünlük ve asıllama kontrolünü destekler. Ancak AH' dan farklı olarak, asıllama kontrol değeri paketin en sonuna koyulur böylece donanım tabanlı gerçeklemelerde paket ağdan bit bit geldikçe HMAC değeri hesaplanır ve paketin en sonuna koyulur.



Şekil 2.3: ESP paket yapısı

ESP, AH' nin verdiği hizmetlerin tümünü verdiği halde neden AH protokolün ihtiyaç olduğu akla gelebilir. IPSec ilk çıktığında AH sadece asıllama bütünlük kontrolü için ESP ise sadece veri gizliliği için tasarlanmıştır. Daha sonra ESP 'ye bütünlük ve asıllama hizmetlerinin de katılmıştır. AH' nin tasarlayıcıları AH' nin hala kullanılması için dış IP başlığının bütünlük kontrolüne alınmasını ve AH şifreleme içermediği için lisans ihracatı sırasında daha az sorun çıkarmasını öne sürseler de bu argümanlar çok gerçekçi bulunmamaktadır. IPSec güvenlik mimarisi yoruma çağrı dokümanında [6] (RFC4301) , IPSec uyumlu cihazlar için AH' nin desteklenmesinin zorunlu olmadığı belirtilmiştir.

2.3 IPSec Sanal Özel Ağ Uygulaması



Şekil 2.4: IPSec Sanal Özel Ağ uygulamaları

IPSec' in esnek mimarisi ihtiyaçlara göre farklı şekilde uygulamalara destek verebilir. Örnek sanal özel ağ(VPN) uygulamaları Şekil 2.4' de gösterilmiştir.

Uçtan-Uca(Site-to-Site (Intranet) VPN:

Bir kuruluşun merkez ofis ve bölge veya şubelerinin olduğu yerel ağları VPN ile güvenli bir şekilde birbirine bağlanmasını sağlar .

Dış Ağ (Extranet VPN) :

Bir kuruluşun iş ortakları, iştirakler, ortak çalışılan şirketler ile güvenli bağlanmasını sağlar.

Uzak VPN (Remote VPN) :

Mobil kullanıcıları, küçük/ev uzak ofisleri (SOHO) merkeze uzaktan olarak güvenli bir şekilde bağlanmasını sağlar.

2.4 Kriptolojinin Kullanımı

Kriptolojik işlemlerde açık anahtarlı algoritmaların yavaşlığı nedeniyle paylaşılan gizli anahtara dayalı algoritmalar kullanılır. Paylaşılan anahtarların güvenli bir şekilde dağıtılması için elle (manual) güvenli kanaldan iletim ya da otomatik olarak açık anahtarlı kriptoloji kullanılır.

Kimlik doğrulama için, HMAC anahtarlı kimlik doğrulama algoritmaları kullanılır. Tavsiye edilen kimlik doğrulama algoritmaları şunlardır [12]: AES-XCBC-MAC-96 (RFC3566[13]), HMAC-SHA1-96 (RFC2404)[14], HMAC-MD5-96 (RFC2403 [15])

Şifreleme için, simetrik algoritmalar kullanılır. Tavsiye edilen şifreleme algoritmaları şunlardır: AES-CBC(RFC3602[16]), AES-CTR(RFC3686 [17])

Bu algoritmaların dışında standart olarak kullanılabilecek algoritma grupları [18]'de verilmiştir.

2.5 Çalışma İlkeleri

İki uç arasında IPSec ile güvenli iletişim güvenlik birliğinin(SA) kurulmasına dayanır. SA güvenli iletişim kurmak için gerekli tüm bilgileri içerir. SA iki uç arasında tek yönlü bir bağlantıdır ve her SA bir güvenlik tanımlayıcısına sahiptir. Güvenli iletişim yapacak iki uç şu noktalarda anlaşmalıdırlar:

- Güvenlik mekanizmalarının seçimi:
    - ESP ya da AH koruması
    - Şifreleme algoritması
    - Bütünlük-asıllama algoritması
    - Kimlik denetimi yöntemi
- Güvenlik birliği hedef ve varış adresi
- Şifreleme ve kimlik doğrulama anahtarları

Birlikte çalışabilirlik için RFC 4301 iki tür güvenlik veri tabanı tanımlamaktadır: Güvenlik Birliği Veritabanı-GBV (Security Association Database-SAD) ve Güvenlik Politika Veritabanı GPV (Security Policy Database-SPD)

Güvenlik Birliği Veritabanı (Security Association Database-SAD): Sistemdeki tüm etkin güvenlik birliklerini içerir. Güvenlik parametreleri indeksi (SPI) , dış IP başlığı hedef IP adresi, Güvenlik Protokolü (ESP ya da AH), güvenlik birliği veritabanında bir güvenlik birliğini tanımlar. Veritabanındaki her bir güvenlik birliği kaydı, kimlik denetimi algoritmaları ve anahtarları, şifreleme algoritmaları ve anahtarları, ömür (KB, sn), güvenlik protokolü kipi (tünel ya da iletim), tekrar önleme hizmeti GPV' deki ilgili politikalara bağlantı vs. içerir.

Güvenlik Politika Veritabanı (Security Policy Database-SPD): Hangi trafik akışlarına hangi güvenlik servislerinin uygulanacağını belirler. Güvenlik politika veritabanındaki kayıtlar kural olarak adlandırılır. Her bir kural trafik akışını tanımlayacak seçicileri içerir. Seçiciler şu şekilde olabilir.

- Varış ve Hedef IP adresleri (asıl IP paketindeki)
- İletim katmanı protokolü (TCP, UDP,..)
- Varış ve hedef portları (FTP, SMTP, HTTP...)

Seçiciler ilgili alan için eşleşen tek bir değeri ifade edebileceği gibi, belirli bir aralıktaki tüm değerleri (en küçük≤ değer ≤ en büyük) ya da olası tüm değerleri (*) ifade edebilir. Bir paket birden fazla kurala uyabilir bu durumda en öncelikli kural geçerlidir. Güvenlik politika veritabanında arama problemin karmaşıklığı nedeniyle IPSec başarımında ciddi rol oynamaktadır. Eşleşen her bir trafik akışı için ilgili kurala bir karar atanmıştır. Karar şu 3 seçenekten biri olabilir:

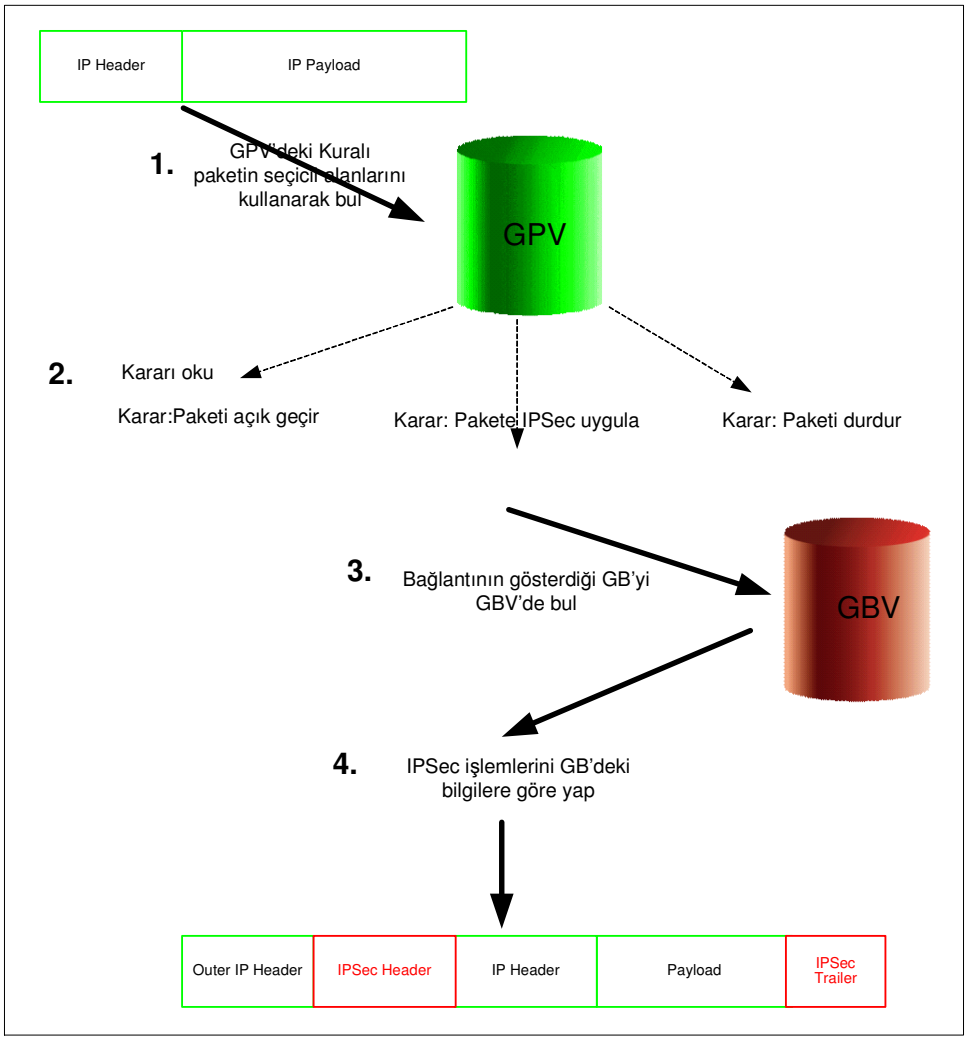
- Açık geçir: Eşleşen paket hiçbir güvenlik hizmeti uygulanmadan geçirilir.
- Durdur: Eşleşen paketin IPSec sınırını geçmesine izin verilmez.
- IPSec uygula: Eşleşen pakete IPSec güvenlik koruması uygulanır. GBV' deki, hangi güvenlik birliğinin kullanılacağı da bu kararda belirtilir.

Tablo 2-1'de örnek bir Güvenlik Birliği Veritabanı verilmiştir.

Tablo 2-1: Örnek güvenlik birliği veritabanı

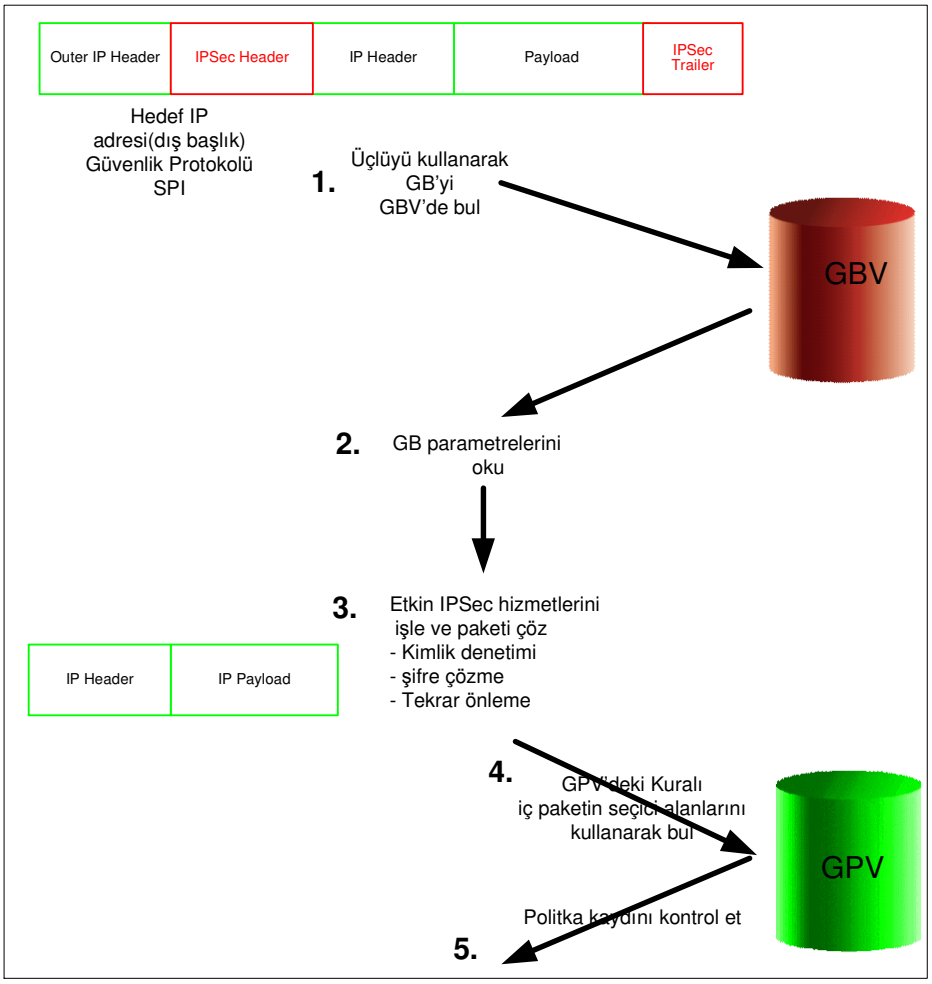
Kural	Ağ katmanı ip varış adresi (adres /maske)	Ağ katmanı ip kaynak adresi (adres /maske)	Ulaşım Katmanı protokolü	Ulaşım katmanı kaynak portu	Ulaşım katmanı varış portu	Karar
R1	192.53.90.47 / 32	192.63.80.11/ 32	*	*	*	durdur
R2	192.58.100.0 / 24	192.53.10.47/ 32	udp	= 80	*	açık geçir
R3	10.1.1.0 / 24	212.26.1.0 / 24	tcp	0>=, <=1024	eşit 80	ipsec_tünel1
R4	*.*.*.*	*.*.*.*	*	*	*	durdur

Giden Paketin işlenmesi: İlk olarak Paketin seçici alanları kullanılarak Güvenlik birliği Veritabanında paketin eşleştiği kural bulunur. Kuralın kararı açık geçir ise pakete IPSec işlevi uygulanmadan güvenli bölgeden çıkmasına izin verilir. Karar durdur ise paket atılır. Karar IPSec uygula ise bağlantının gösterdiği Güvenlik birliği GBV' da bulunur. Güvenlik Birliğindeki bilgilere göre IPSec işlemleri uygulanır ve IPSec paketi hazırlanır. Çıkan paket işleme adımları Şekil 2.5'de gösterilmiştir.



Şekil 2.5: IPSec giden paket işlemi

Gelen Paketin işlenmesi: Gelen IPSec paketinin hedef IP adresi , Güvenlik protokolü ve SPI değerleri okunur. Bu üçlüye göre Güvenlik Birliği GBV' da bulunur. GB parametreleri okunur ve bu bilgilerle, etkin IPSec hizmetleri (Şifre çözme,kimlik denetimi, tekrar önleme) işlenir ve IPSec paketi çözülür. İç paketin seçici alanları kullanılarak paketin eşleştiği kural bulunur ve kararın doğru GB' ni gösterdiği kontrol edilir. Karar farklı ise paket atılır. Gelen Paket IPSec paketi değil ise, pakete ilişkin karar doğrudan GPV' de bakılır. Karar açık geçir ise paket açık bir şekilde geçirilir aksi halde paket atılır.



Şekil 2.6: IPSec giren paket işleme

2.6 Sonuç

IPSec IP ağlarının güvenliği için IETF tarafından belirlenmiş standart bir güvenlik mimarisidir. Güçlü şifreleme, esneklik, saydam çalışma, uygulamaları etkilememe, fiziksel alt yapıdan bağımsızlık,kolay yönetilebilirlik gibi özellikleri kullanımının

yaygınlaşmasında büyük rol oynamaktadır. Ancak sağladığı güvenliğin elbette bir performans bedeli de vardır. Şifrelemenin kullanılması ve pakete katılan ek alanlar ağ geçirimin düşmesine neden olur. TCP/IP ağlarının tüm güvenlik sorunlarının da IPSec ile çözülemeyeceği unutulmamalıdır. IP katmanı üst katmanları yorumlayamaz ve IP hızlı çalışmak zorundadır. Bu nedenle IPSec diğer güvenlik çözümleri(örneğin Güvenlik Duvarı) ile birlikte kullanılmalıdır.

3 CLICK

3.1 Giriş

Click (Click Moduler Router) [19,20] modüler yönlendirici projesi MIT PDLA laboratuarlarında Eddie Kohler tarafından doktora tezi olarak geliştirilmiştir. Click projesinin temelinde ağ ortamında paketlere yapılan işlemlerin basit, modüler ve esnek nesneler halinde tanımlanması ve tüm sistemin bu nesnelerin birleştirerek oluşturmak yatmaktadır. Bu sayede; kolayca genişletilebilen, modüler, hata yönetimi kolay bir ağ cihazı mimarisi ortaya çıkartılmıştır.

Click projesi kapsamında tanımlanan nesneler kullanıcı tarafından çeşitli şekillerde ve sıralarda birbirlerine bağlanarak ardışıl bir paket akış yolu oluşturulabilmektedir. Bu sayede karmaşık bir ağ cihazı sadece gerekli nesnelerin birbirlerine doğru bir biçimde bağlanmasıyla kolaylıkla oluşturulabilmektedir. Click projesinde ortaya atılan basit nesnelerin ardışıl bir biçimde bağlanarak paketleri işlemesi fikri sayesinde standart protokollerin ve gerekli algoritmaların gerçeklenmesi çok daha kolay ve anlaşır bir hale gelebilmektedir.

3.2 Mimari

Click yazılımı nesneye dayalı biçiminde C++ programlama dili kullanılarak gerçeklenmiştir. Nesnelerin yanı sıra yazılımda yardımcı birçok fonksiyon ve makro da bulunmaktadır. Click Linux ve FreeBSD işletim sistemi üzerinde hem kullanıcı hem de çekirdek seviyesinde çalışacak şekilde hazırlanmıştır. Kullanıcı düzeyinde hata ayıklama ve geliştirme kolaylıkla yapılabilirken, çekirdek seviyesinde ise yüksek performans gerektiren son ürüne yönelik yazılımlar ortaya çıkartılabilmektedir. Ancak yazılımın işletim sistemine bağlı tarafları başka işletim sistemlerine de rahatlıkla adapte edilebilecek modüler bir yapıdadır.

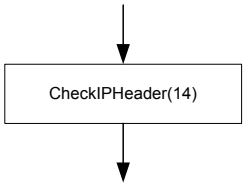
Aynı zamanda güncel simülasyon araçlarından NS-2 ile entegrasyonu sayesinde gerçek ağ cihazları için hazırlanan yazılımlar, simüle edilmiş büyük ağlarda

kolaylıkla denenebilmektedir. Bu sayede simülasyon ortamında denenen yazılımlar kısa zamanlarda çok küçük değişiklikler ile gerçek ağ ortamlarına taşınabilmektedir.

Sıralan bölümlerde Click yazılımının mimari açıdan en temel parçalarından bahsedilecektir.

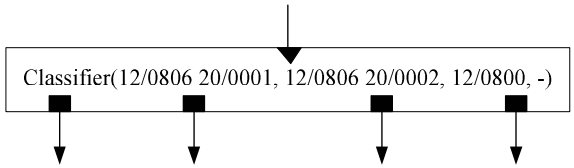
3.2.1 Elemanlar

Elemanlar (Elements) ağ paketlerini kullanan veya paketler üzerinde basit işlemler yapan nesnelerdir. Eleman nesnelerinin birçok girişi veya çıkışı olabilmektedir. Bu giriş ve çıkış birimleri kapı (*Port*) olarak adlandırılmaktadır ve her elemanın en az bir tane giriş kapısı bulunmaktadır. Elemanlar ağ paketlerini giriş kapılarından alır ve gerekli işlemleri gerçekleştirdikten sonra gerekiyorsa çıkış kapılarına verirler. Şekil 3.1' de örnek olarak "CheckIPHeader" elemanı verilmiştir. Bu eleman giriş kapısından IP paketlerini alarak IP başlık yapısını RFC 791'e göre kontrol etmekte ve sadece başlık yapısı doğru olan paketleri çıkış kapısına göndermektedir.



Şekil 3.1: CheckIPHeader elemanı

Kapıların özellikleri ve elemanların başlangıç atamaları konfigürasyon cümleleri ile yapılmaktadır. Click yazılımı tarafından bu cümleler sırayla okunarak eleman nesnelerinin gerekli yapıcı (Constractor) fonksiyonları bu tanımlama cümlelerinden okunan parametrelere göre çağrılarak eleman nesneleri oluşturulur. Şekil 3.2'deki örnek eleman standart bir sınıflandırma elemanı olup, "Classifier(12/0806 20/0001, 12/0806 20/0002, 12/0800, -)" tanımlama cümlesi ile oluşturulmaktadır. Buna göre bir adet giriş kapısı bulunan elamanın dört adet çıkış kapısı bulunmaktadır. Bu eleman giriş kapısı gelen Ethernet paketlerinden ARP isteklerini 0. çıkış kapısına, ARP cevaplarını 1. çıkış kapısına, IP paketlerini 2. çıkış kapısına ve diğer paketleri 3. çıkış kapı yönlendirerek paketleri sınıflandırmaktadır.



Şekil 3.2: Classifier elemanı

3.2.2 Paketler

Paket (Packet) nesneleri Click yazılımındaki diğer önemli temel nesne grubunu oluşturur. Click Mimarisinde Paketler gerçek ağ paketlerini yazılım olarak temsil eder. Paketler iki kısımdan oluşur, birinci kısım gerçek ağ paket verisini taşıyan bellek alanı, ikinci kısım ise bu bellek alanına işaret eden bazı kontrol başlıklarını içerir. Gerçek paketin bellekte bulunduğu alanın başı, sonu, IP başlığının başı, ulaşım katmanı başlığının yeri başlıca kontrol verileridir. Paketler temel olarak elemanlar tarafından kullanılan veya üzerlerinde işlem yapılabilen bit dizilerini tutan nesneler olarak tanımlanabilir. Bu bit dizilerine Ethernet, IPv4 veya IPv6 paketleri örnek olarak verilebilir ancak her hangi bir bit dizisi yani herhangi bir protokolün paket formatı da kolaylıkla paket olarak tanımlanabilmektedir. Click projesi kapsamında birçok yaygın protokolün paket formatı tanımlı bulunmaktadır. Aynı zamanda tanımlanan paket formatları üzerinde işlem kolaylığı sağlayacak yaratma, ekleme, silme gibi birçok fonksiyon da hali hazırda bulunmaktadır.

Paket nesnelerinin en önemli ve işlem performansını arttıran özelliği ek açıklama alanlarının (*Annotations*) olmasınıdır. Ek açıklamalar gerçek paket bellek alanının dışında Paket nesneleri içinde tutulurlar. Elemanlar arasında veri akışı bu ek açıklamalar üzerinden hızlı bir şekilde sağlanabilir. Örneğin *IPLookUp* elemanı sonraki sekme (*nextHop*) ek açıklamasını belirler, *ARPQuerier* elemanı ise bu ek açıklamaya bakarak, sonraki sekme IP adresi için ikinci katman adresini bulur. Ek açıklamalarda paketle ilgili bazı kontrol bilgileri tutulur. Ek açıklamalar pakete işlemler sonucunda ekilebilecek verilerin önceden belirlenip paketlerin veri dizilerinin önceden bu öngörüye göre oluşturulmasını sağlayan özelliktir. Örneğin bu özellik kullanarak bir Ethernet paketi yaratılırken ona IP başlığının ekleneceği önceden belirlenebilir ve paketin veri kısmı ona göre yaratılabilir ki bu paketlerin tekrar tekrar bellekte yeniden yaratılması önler ve işlem performansı arttırır. Yine ek açıklama özelliği kullanılarak paketler üzerinde çeşitli verilerin taşınabilmesi için

ekstra veri bölgeleri de oluşturulabilir. Örneğin paketler öncelik değerlerine göre ek açıklamalarla ayrılan bir veri kısmı sayesinde işaretlenebilir ve elemanlar tarafından bu öncelik değerlerine göre işlenmeleri sağlanabilir.

3.2.3 Bağlantılar

Çalışan bir Click yazılımı birbirine çeşitli biçimlerde bağlı birçok elemandan oluşur. Bağlantı özetle kaynak elemandan varış elamanına paket akışını temsil eder. Her bir bağlantı paket akışı için alternatif bir yolu belirtir. Paketin elemanlar arasındaki aktarımına göre iki tür bağlantı vardır. Birincisi *Push*(itiş) bağlantısı diğeri *Pull*(Çekiş) bağlantısı. *Push* bağlantı da paket akışını kaynak eleman başlatır, *Pull* bağlantıda ise varış elemanı paket akışını başlatır yani kaynak elemanını yeni bir paket göndermesi için sorgular eğer hali hazırda bir paket yok ise geriye boş işaretçi döner. Elemanlar birbirlerine giriş ve çıkış kapılarından çeşitli biçimlerde bağlanarak bir paket akış yolu oluşturulur. Bağlantını türüne elemanların kapılarının durumu belirler. Elemanlar için üç temel kapı biçimi tanımlıdır;

İtmeli Kapı*(Push Port)*: Elemanın aktif olarak kullandığı kapılardır. Paketler Elemanlar tarafından bu kapılara hiçbir ön koşul gerekmeden yollanabilmektedirler.

Çekmeli Kapı*(Pull Port)*: Elemanların ancak başka bir elemandan gelecek istek üzerine kullanabildiği pasif kapılardır.

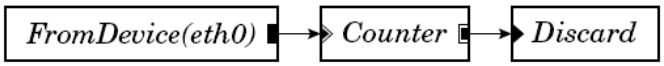
Yönsüz Kapı*(Agnostic Port)*: Hem itmeli hem de çekmeli kapı gibi çalışabilen hibrit kapılardır.

Kapıların biçimi elemanları birbirlerine bağlarken dikkat edilmesi gereken önemli bir noktadır. Buna göre itmeli tipindeki kapılar yalnızca çekmeli veya yönsüz tipindeki kapılara bağlanırlar ve oluşan bağlantı çekmeli bağlantı olur, itmeli tipindeki kapılar ise yalnızca itmeli veya yönsüz tipteki kapılara bağlanabilir ve oluşan bağlantı itmeli bağlantı olur.

Tasarım gereği elemanların çekmeli veya itmeli biçimindeki kapı tiplerini birbirlerine dönüştürmek zorunlu bir hale gelirse Click yazılımında hali hazırda gerçeklenmiş olan *Queue*(*Push* -> *Pull*) veya *Unqueue* (*Pull* -> *Push*) dönüştürücü elemanları kullanılabilir.

3.2.4 Konfigürasyon

Konfigürasyon elemanların tanımlama ve bağlantı biçimini ifade eden bilgi parçasıdır. Yönlendirici konfigürasyonu, bağlantılar ayrıt ve elemanlar düğüm olmak üzere yönlü çevresel olmayan bir graftır. Tanımlama elemanların ilk değerlerinin atanmasını sağlarken bağlantı biçimi paket akış yolunu belirler. Aşağıda bulunan Şekil 3.3ve Şekil 3.4‘ te örnek bir konfigürasyon dosyası ve eleman bağlantı şeması bulunmaktadır.



Şekil 3.3: Tüm paketleri atan bir yönlendirici konfigürasyonu

```
//3 eleman bildirir...
src :: FromDevice(eth0);
ctr :: Counter;
sink :: Discard;
// elemanlari birbirine bağlar
src -> ctr;
ctr -> sink;

//aynı konfigürasyonun başka şekilde de  ifade edilebilir...
FromDevice(eth0) -> Counter -> Discard;
```

Şekil 3.4: Şekil 3.3’deki yönlendirici konfigürasyonunun Click-dilindeki ifadesi

Konfigürasyon bilgisi Click yazılımı tarafından TCL kullanılarak belirlenmiş olan gramer kuralları uyarınca yazılır. Yazılan konfigürasyon bilgisi Click yazılımı çalışmaya başladığında yazılım tarafından okunarak konfigürasyonda belirtilen elemanlar ve bağlantılar buradaki bilgilere göre dinamik bir şekilde oluşturulur.

3.2.5 Yönlendirici

Yönlendrici(*Router*) elemanların oluşturulmasından ve bağlantıların kurulmasından sorumlu olan temel nesnedir. Konfigürasyon bilgisi bu nesne tarafından okunup doğrulanmaktadır. Okunan konfigürasyon bilgisindeki elemanların bağlantı biçimindeki mantık hatalarını bulunması ve performansı arttırmak için optimizasyon yapmak gibi ileri düzey özellikleri de bulunmaktadır.

3.2.6 İş Zamanlama

Paket işleme esnasında elemanlar arasında bir iş paylaşımının yapılması gerekmektedir. Click yazılımında bu işi Zamanlayıcı(*Scheduler)* isimli nesne yapmaktadır. *Router* nesnesi tarafından oluşturulan elemanlar kendisini *Scheduler* nesnesine kayıt ederek belirli bir işlem zamanını kendilerine ayırtmış olurlar. Bu işlem zamanları varsayılan başlangıç değerleri olabileceği gibi konfigürasyon dosyasında kullanıcı tarafından da belirlenebilir. Yine bu işlem zamanları çalışma esnasında yazılım tarafından dinamik olarak da değiştirilebilmektedir.

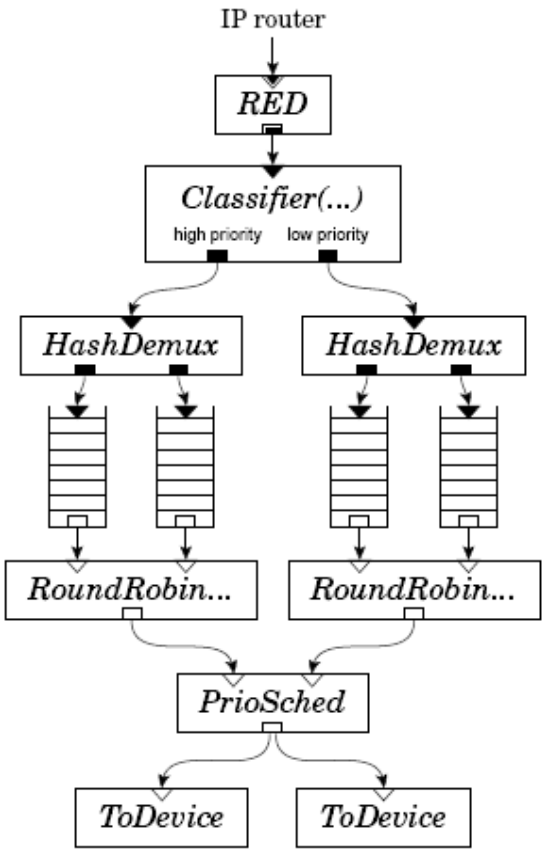
İş paylaşımı için Click yazılımı bir çok iş paylaşım tekniğini kullanabilmektedir bunlardan bazıları; Dönüşümlü Zamanlama (*RoundRobin Scheduling*), (Öncelik Zamanlama) *Priority Scheduling* ve Determinist Zamanlama (*Stride Scheduling*) dir.

3.3 Örnekler

Bu bölümde Click yazılımının yeteneklerinin daha iyi anlaşılabilmesi için kimi standart ağ cihazlarının ve ağ teknolojilerinin Click yazılımı ile gerçeklenmesi örnekler verilerek anlatılmıştır.

3.3.1 İleri Düzeyde Paket Kuyruklama ve Sınıflandırma Gerçeklemesi

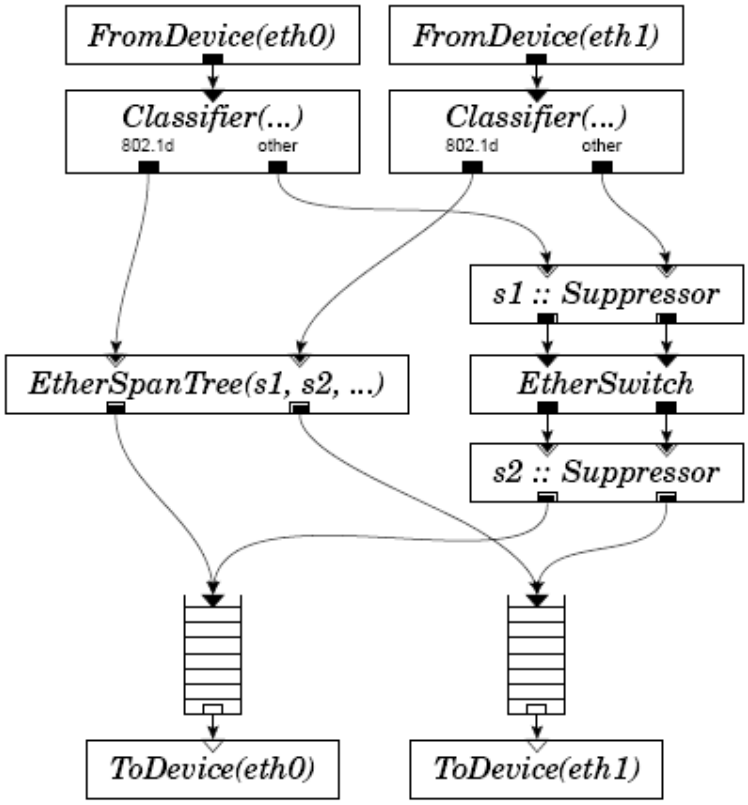
Şekil 3.5'de gösterilen Click konfigürasyonu uyarınca gelen paketler öncelikle *RED* elemanı tarafından kuyrukların doluluk seviyesine göre rasgele düşürülmektedir. RED tarafından düşürülmeyip geçirilen paketler *Classifier* elamanı tarafından öncelik değerlerine göre yüksek öncelikli ve düşük öncelikli olmak üzere ikiye ayrılırlar. Bu paketler daha sonra *HashDemux* elemanı tarafından kullanıcının belirleyeceği bir çırpı tablosu uyarınca kuyruklanmaktadır. En nihayetinde *PriSched* Elemanı tarafından kontrol edilen *RoundRobin* iş paylaşım elemanı bu kuyruklardan paketleri önem sıralarına göre alarak hedef arayüzleri olan *ToDevice* elemanına yollar. Bu sayede çok karmaşık gibi görülen bütün bu işler sadece basit işler yapmak üzere tasarlanmış elamanların uygun bir biçimde bağlanmasıyla kolaylıkla gerçeklenmiş olur.



Şekil 3.5: QoS destekleyen bir konfigürasyon

3.3.2 Ethernet Anahtar Gerçeklemesi

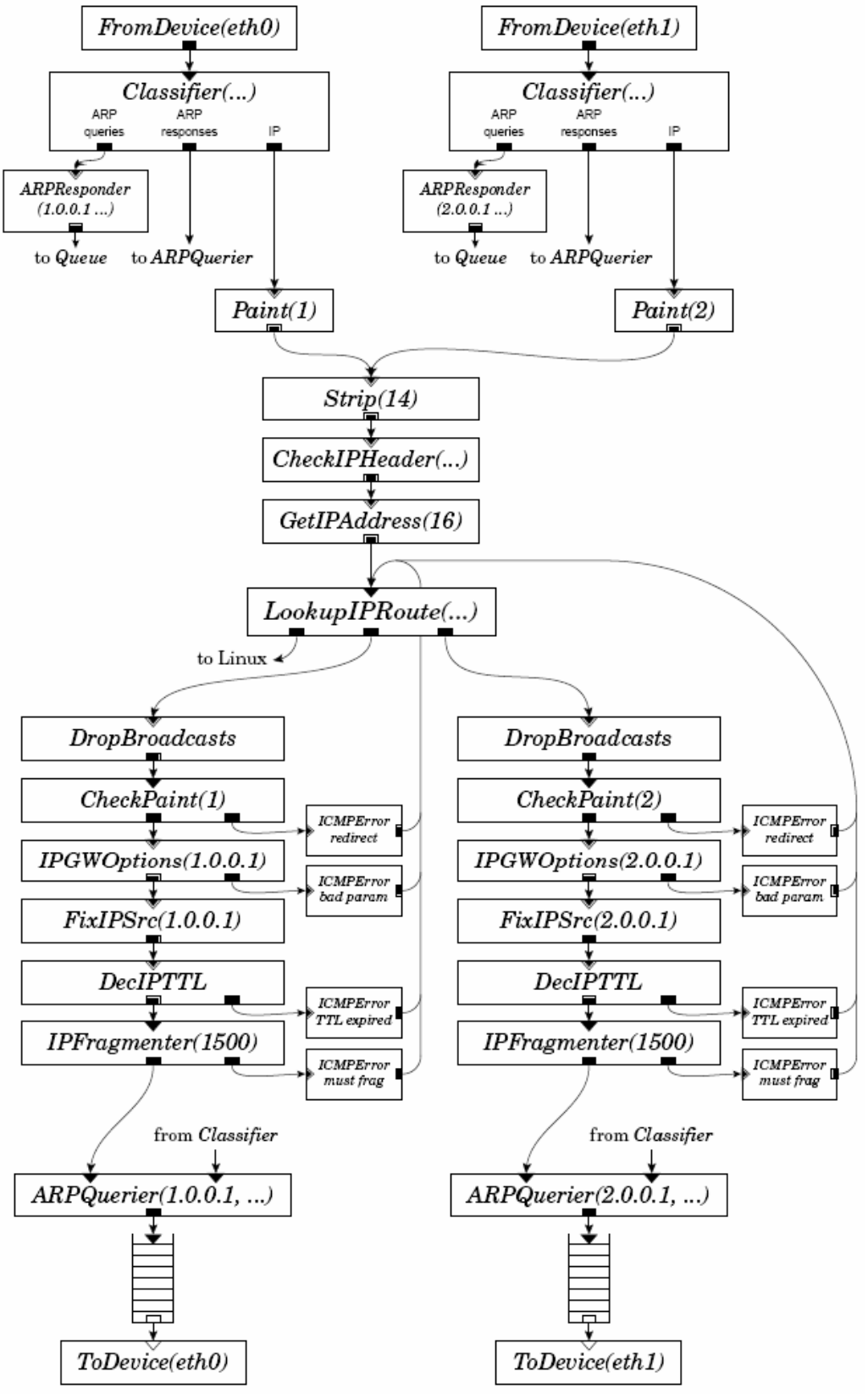
Şekil 3.6'da gösterilen Click konfigürasyonu ile iki kapılı basit bir Ethernet anahtarının gerçeklenmesi gösterilmiştir. *EtherSpanTree* elemanı tarafından gerçeklenen 802.1d protokolü bir çok anahtarın birlikte çalışması durumunda ağda oluşabilecek sonsuz döngülerin önlenmesini sağlarken. *EtherSwitch* elemanı paketlerin uygun arayüzlere gönderilmesini sağlamaktadır ve bunu yaparken basit ve öğrenebilen bir anahtarlama algoritması kullanmaktadır.



Şekil 3.6: Ethernet anahtar konfigürasyonu

3.3.3 IPv4 Yönlendirici Gerçeklemesi

Şekil 3.7'de gösterilen Click konfigürasyon ile RFC ye tamamen uyumlu iki kapılı standart bir yönlendirici yazılımının gerçeklemesi gösterilmiştir. Buna göre yönlendirici ARP, ICMP gibi protokollerini gerçeklemenin yanında IP paketleri üzerinde başlık sınaması, TTL işlemesi ve Parçalama/Birleştirme işlemlerini de gerçeklemektedir.



Şekil 3.7: IPv4 yönlendirici

3.4 Avantajları

Bu bölümde Click yazılımının avantajları tartışılmıştır.

3.4.1 Modülerlik ve Esneklik

Click yazılımı nesneye dayalı tasarımı ve gerçeklenmesi sayesinde oldukça modüler ve esnek bir yapıdadır. Click üzerinde çalışan bir araştırmacı hali hazırda bulunan elemanları bir Lego oyuncağının parçası gibi kullanarak kafasında tasarladığı ağ teknolojisini rahatlıkla gerçekleyebilir.

3.4.2 Genişletilebilirlik

Click yazılımı her şeyden önce kolaylıkla genişletilebilmek ve geliştirilmek üzere yazılmıştır. Bu yazılıma olmayan yeni bir elemanın eklenmesi veya olan elemanların değiştirilmesi rahatlıkla yapılabilmektedir. Bu da yeni teknolojilerin veya araştırma konularının denenmesini kolaylaştırmaktadır.

3.4.3 Test

Click yazılımı kullanılarak tasarlanan sistemler hem NS-2[21] gibi büyük simülasyon ortamlarında hem de gerçek ağ üzerinde denenene bilmektedir. Bu amaçla NS-Click çalışması Click projesine katılmıştır[22].Bunların yanı sıra çalışma esnasında kullanılan elemanların durumları gözetlenebilir ve çeşitli kayıtlar tutulabilir. Bu özellikleri sayesinde yapılan çalışmaların sonuçları çok kısa zamanda elde edilebilmektedir.

3.4.4 Açık Kaynak Desteği

Click yazılımı MIT tarafından koordine edilmekte ancak dünyanın hemen her ülkesindeki bir çok araştırma grubu veya kişi tarafından kullanılmaktadır. Bu kişileri bir forum altında birleştiren asıl geliştirme gurubu, bu kişilerin sürekli iletişim ve işbirliği halinde olmalarını sağlamıştır. Bu sayede karşılaşılan problemler ile ilgili kolaylıkla bir çok uzman kişinin görüşü alınabilmekte. Ve yapılan tüm dünya çapında çalışmalar paylaşılarak gereksiz yere iş tekrarı önlenmektedir [23].

3.5 Sonuç

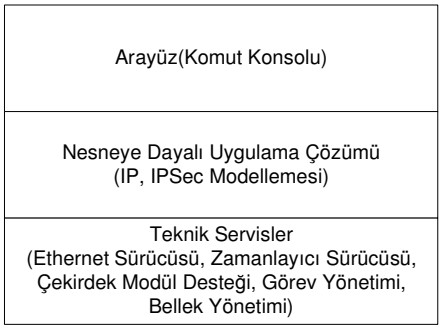
Click yazılım mimarisi genişletilebilir, modüler ve esnek yapısı ile ağ teknolojilileri üzerine araştırma yapan insanlar için çok ideal bir geliştirme platformdur. Araştırmacılar kolaylıkla hali hazırda bulunan elemanlar ile bir Lego oyuncağı misali oynayarak hem simülasyon hem de gerçek ağ ortamda çalışabilecek tasarımlar ve sistemler ortaya çıkartabilir. Yine araştırmacılar kolaylıkla yazılıma yeni özellikler katılarak kendi çalışmalarını hızlı bir biçimde gerçekleştirebilirler. Bugün Click yazılımı kullanılarak yapılmış ve yapılmakta olan bir çok bilimsel ve endüstriyel araştırma bulunmaktadır [24-28].

Tüm bu avantajları nedeniyle bu tez kapsamında yapılan araştırma ve geliştirme faaliyetlerinde temel altyapı olarak Click yazılım mimarisi kullanılmıştır.

4 NESNEYE DAYALI MODELLEME VE TASARIM

Tez çalışması kapsamında IPSec sanal özel ağ yazılımı tasarlanmaya çalışılmıştır. Amaç, yazılımın nesneye dayalı yönteme göre analiz edilerek modellenmesi ve gerekli olan yazılımın kalite kriterleri göz önünde bulundurularak tasarlanmasıdır. Tasarımda nesneye dayalı yöntemin tasarım ve kalıpları en verimli şekilde kullanılmaya çalışılmıştır. Tasarımda genel amaçlı GRASP [29,30], GoF[31] kalıpları ve gerçek zamanlı tasarım kalıpları[32] etkin bir biçimde kullanılmıştır. Böylece modüler, hata giderilme maliyeti düşük, değişikliklere elverişli, kolayca geliştirilebilir ve genişletilebilir bir sistem ortaya konulmuştur.

Proje gerçekleştirilirken tümleştirilmiş sürece (Unified Proccess-UP) uygun şekilde yinelemeli, risk güdümlü ve artırmalı bir çalışma sistemi izlenmiştir. Başlangıç, bölümünde hangi kullanım senaryolarının yazılacağını belirlenmiş ve bunlardan biri ayrıntılı olarak yazılmıştır. Sistem genel olarak 3 katmanlı bir yapıda tasarlanmıştır.[1]



Şekil 4.1: Üst düzey yazılım katmanları

Arayüz bölümünde kullanıcı ile ilgili etkileşim, kullanıcının cihazı yönetebilmesi, cihaz bilgileri görebilmesi için gerekli arayüzler belirlenmiştir. Nesneye dayalı uygulama çözümünde cihazın asıl görevi olan IP, IPSec işlevlerini yerine getireceği alt sistem nesneye dayalı olarak tasarlanmıştır. Teknik servisler bölümünde ise, cihazın farklı donanım platformlarında çalışabilmesi için gerekli donanım soyutlama katmanı tasarlanmıştır.

[1] Katman kalıbı (Layer Pattern)

4.1 Başlangıç

Tasarımın başlangıç aşamasında projenin gerçeklenebilirliği, hangi araç ve yöntemlerin kullanılacağına karar verilmiştir. Yazılım mimarisi olarak Click modüler yönlendirici mimarisi temel alınmış ve hedef sistemin Linux işletim sisteminde Çekirdek görevciği (kernel thread) olarak çalıştırılmasına karar verilmiştir.

Kullanılan Araçlar:

Programlama dili: C++

Faydalanılan Açık Kaynak kodları: Click, Linux Ethernet kartı aygıt sürücüleri

Geliştirme Ortamı: Eclipse ,CDT , CVS:

Tasarım ve Dökümantasyon Araçları:

- Rational Software Architecture : UML Destekli Yazılım Geliştirme Aracı
- Visual Paradigm: UML Destekli Yazılım Geliştirme Aracı
- Doxygen: Koddan dökümanların oluşturulması.

4.2 İsteklerin Çözümlenmesi

Bu bölümde ilk olarak risk önceliklerine göre hangi senaryoların yazılması gerektiği belirlenmiştir ve IPSec RFC standartlarına uygun şekilde kullanım senaryoları yazılmıştır. Gerekli görülen senaryolar için sistem ardışıl diyagramları da çizilmiştir. Kullanım senaryoları yazılırken, senaryolar arası ilişkiler IPSec RFC' lerindekine benzer şekilde kurulmuştur ve diğerlerine temel teşkil eden riskli kısımlar önceliklerine göre yazılmıştır.

Yazılan Kullanım senaryolarının listesi aşağıda verilmiştir:

- KS1: Ayarların yapılması
    - KS1_alt_1: Kural eklemek
    - KS1_alt_2: Kurala seçici eklemek
    - KS1_alt_3: Güvenlik birliği (SA) eklemek
    - KS1_alt_4: Kurallara karar atamak
    - KS1_alt_5: Anahtar tablosuna anahtar eklemek

- KS2: Güvenli arayüzden alınan paketi işlemek
    - KS2_alt_1_gen: SA' ya göre IPSec ESP tünel paketi oluşturma
        - KS2_alt_1_ger : AES-CBC Şifreleme
        - KS2_alt_2_ger : 3DES-CBC Şifreleme
        - KS2_alt_3_ger : NULL-Şifreleme
        - KS2_alt_4_ger : HMAC-SHA-1-96-Asıllama
        - KS2_alt_5_ger :AES_XCBC_MAC-96 Asıllama
    - KS2_alt_2: Çıkan Güvenlik Politika Veri tabanına göre paketi açık geçirme
    - KS2_alt_3: Çıkan Güvenlik Politika Veri tabanına göre paketi atma
- KS3: Güvensiz arayüzden alınan paketi işlemek
    - KS3_alt_1: SA' ya göre IPSec ESP tünel paketi çözme
        - KS3_alt_1_ger : AES-CBC Şifreleme
        - KS3_alt_2_ger : 3DES-CBC Şifreleme
        - KS3_alt_3_ger : NULL-Şifreleme
        - KS3_alt_4_ger : HMAC-SHA-1-96- Asıllama
        - KS3_alt_5_ger :AES_XCBC_MAC-96 Asıllama
    - KS3_alt_2: Giren Güvenlik Politika Veri tabanına göre paketi açık geçirme
    - KS3_alt_3: Giren Güvenlik Politika Veri tabanına göre paketi atma

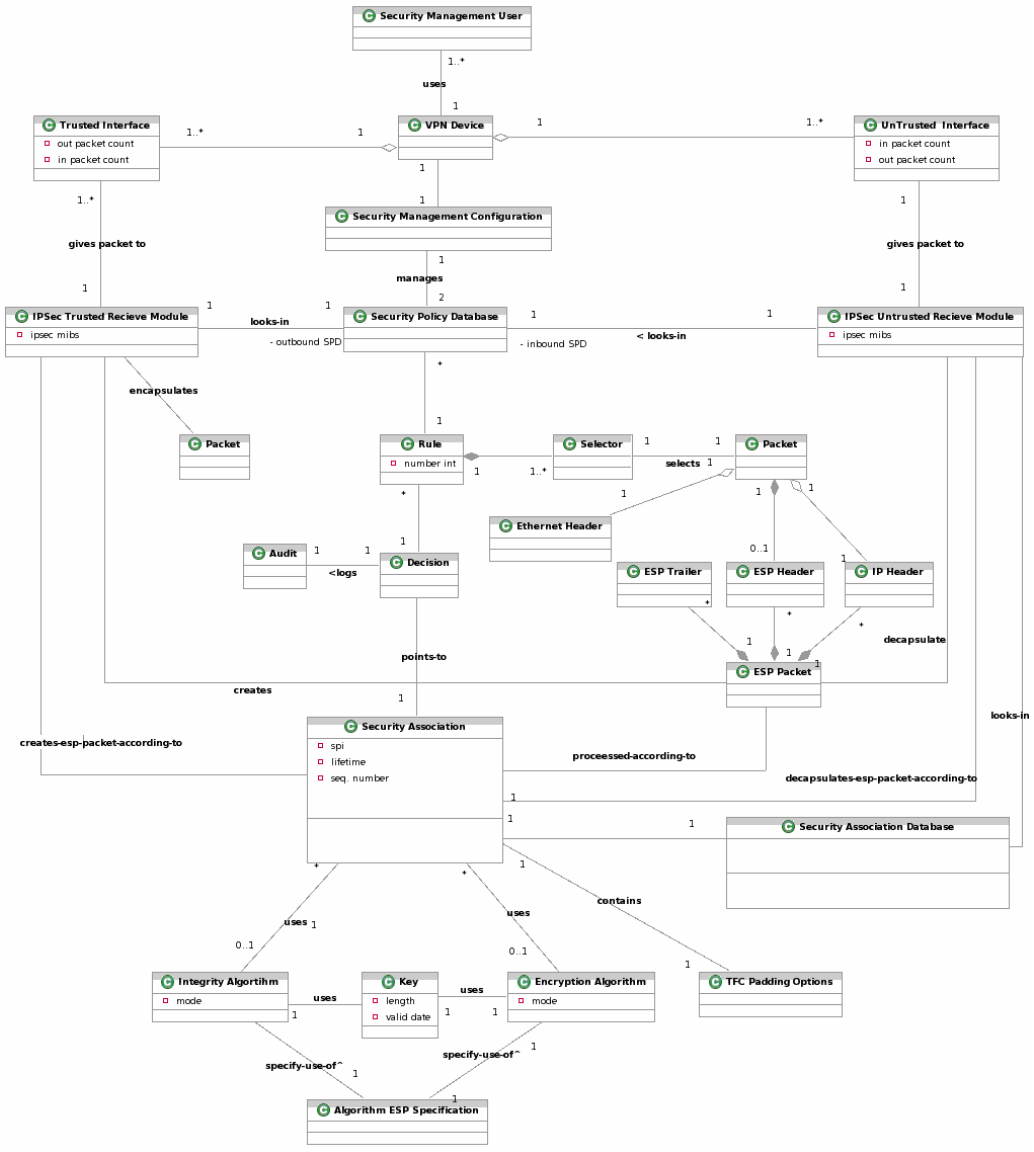
EK-A' da KS2 ve KS2_alt_1_ger senaryoları verilmiştir. Tasarım, örnek olması açısından bu senaryolar üzerinde anlatılacaktır.

4.3 Uygulama Domeninin Modellenmesi

Bu aşamada uygulama domeni kavramsal sınıfları belirlenmiştir. Kullanım senaryolarındaki isim ve isim tamlamalarının listesi çıkartılmış ve fazlalık, belirsiz, ilgisiz sınıflar, işlemler, roller, gerçekleme unsurları uygulama domeni sınıflarından elenmiştir. Kavramsal sınıfları gerekli elemelerden geçirdikten sonra bu sınıflar arası

bağlantılar ve kavramsal sınıfların niteliklerini bulunmaya çalışılmıştır. Tüm bunlara ilişkin UML Sınıf diyagramı Şekil 4.22'de gösterilmiştir. Bakış açısına göre kavramları yorumlamanın ya da kavramları adlandırmanın kişiden kişiye değişebileceği düşünülerek, hem de uygulama modelinin daha iyi anlaşılması amacıyla bir sonraki paragrafta, diyagramlardaki sınıfların ve ilişkilerin açıklaması için sözlük yazılmıştır. Uygulama modelinde son aşama olarak gerekli görülen sözleşmeler yazılmıştır.

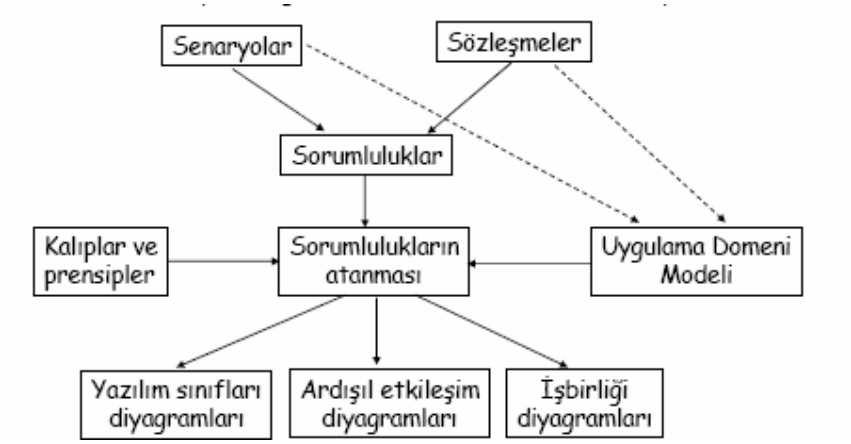
VPN Device, tasarladığımız IPSec sanal özel ağ cihazının kendisidir, VPN cihazını güvenlik yönetimi kullanıcısı (*Security Management User*) kullanır. VPN cihazının Güvenli(Kırmızı) ve Güvensiz(Siyah) arayüzleri vardır.(*Inbound(Red)Interface OutBound(Black)Interface*). VPN cihazı bir Güvenlik yönetimi Konfigürasyonu vardır bu konfigürasyonda güvenlik ile ilgili ayarlar yapılır. Güvenlik yönetimi konfigürasyonu, güvenlik politika veri tabanını (*Security Policy Database*) yönetir. Güvenlik Politika veri tabanını kuralları (*Rule*) içerir. Her kural bir ya da daha çok sayıda seçici içerir, bir IP Paketinin (*Packet*) bir kurala uyması için, kurallın içerdiği tüm seçicilere uyması gerekir. Her kuralın bir de kararı (*Decision*) vardır. Karar seçilen pakete ne yapılacağını belirler. Karar açık geçir, durdur ya da IPSec uygula olabilir. Belirli durumlarda alarm verilebilir ve bu alarmların kayıtları sistemde tutulur (*audit-log*). Internet paketi, Ethernet başlığı (*Ethernet Header*), IP başlığı (*IP Header*) , ESP başlığı (*ESP Header*) içerebilir. ESP Paketi ESP başlığı, IP başlığı ve ESP kuyruk alanlarını içerir. Güvenli arayüz paketi IPSec giriş modülüne(*IPSec-Outbound*), güvensiz arayüz paketi IPSec çıkış modülüne(*IPSec-Inbound*) verir. (*IPSec-Outbound*) modülü IP paketini kapsüller ve ESP paketini oluşturur. (*IPSec-Inbound*) modülü ise, ESP paketini açar (decapsule) ve iç IP paketini çıkarır. ESP paketleri, Güvenlik Birliklerine (*Security Association-SA*) göre açılır ya da oluşturulurlar. SA'lar güvenlik birliği veri tabanında(*Security Association Database-SAD*) tutulur. Güvenlik birlikleri şifreleme algoritmaları, bütünlük algoritmaları ve bu iki işlemi birden yapabilen birleşik algoritmaları kullanırlar. Algoritmalar şifreleme ve asıllama işlemleri için anahtar (*Key*) kullanırlar. Her algoritmanın ESP ile nasıl kullanılacağını belirten algoritma ESP şartnameleri (*Algorithm ESP Specifiacation*) vardır. ESP paketini açmak ve oluşturmak için gerekli tüm bilgileri güvenlik birliği (*SecurityAssociation*) tutar.



Şekil 4.2: Uygulama domeni UML sınıf diyagramı

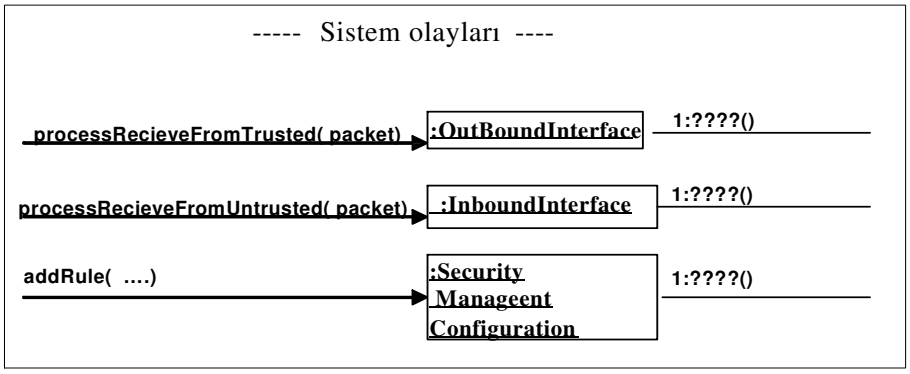
4.4 Tasarım Modelinin Oluşturulması

Uygulama modelini belirledikten sonra elimizdeki senaryolar, uygulama domeni modeli, ve sözleşmelerden yararlanarak, senaryoların birbiriyle etkileşimde olan nesneler şeklinde tasarlanması gerçeklenmiştir[33]. İzleyen sayfalarda modelleme ve tasarıma ilişkin modeller ile açıklamalar buradaki anlatılma sırasına uygun şekilde yer almaktadır.



Şekil 4.3: Tasarıma geçiş [33]

Tasarıma sistem olaylarının çıkartılması ve senaryolar üzerinden gidilmesi ile başlanılmıştır. Temel sistem olayları Şekil 4.4'de gösterilmiştir. Sistem olaylarını denetlemek için bir denetçilere ihtiyacımız var. Bu iş için senaryo denetçisi seçmeyi uygun gördük.[1] Bu iş için en uygun gözüken nesneler, kural eklemek için Güvenlik Yönetim Konfigürasyonu *(Security Management Configuration)*, güvenli arayüzden alınan paketleri işlemek için, *Red Interface*(Kırmızı arayüz), güvensiz arayüzden alınan paketleri işlemek için *Black Interface* senaryo denetçisi olarak seçilmiştir.



Şekil 4.4: Sistem olayları

Adım 1- Kırmızı arayüzden alınan paketi işlemek:

Senaryo denetçisi *RedInterface* 'i hemen yazılım domenine ekliyoruz.

"sistem Ethernet paketini, sistem paketin tip (*type*) alanına bakarak sınıflandırır..(kim sınıflandırır? Ethernet paket sınıflandırıcı...)" bu iş için en uygun nesne, *EthernetClassifier* yapay nesnesidir.[2] Çünkü *RedInterface* sınıfına bu

[1] Denetçi kalıbı
[2] Yapay nesne kalıbı

sorumluluğu atamak iyi uyumu bozacaktır[1]. Bu durumda *RedInterface*, *EthernetClassifier* ve *Packet* nesnelerini tasarım modelimize ekliyoruz. Paket sınıfı bize Ethernet kartından alınan paketi yazılım domeninde temsil edecektir ve tasarımın bundan sonraki kısmında temel bir eleman olarak kullanılacaktır.

"sistem paketin Ethernet başlıkları kontrol eder ve paketin Ethernet başlığı kısmını atar.(ilerletilir IP başlığına ötelenir)"

Paketin Ethernet kısmını ayırmayı *Strip* element nesnesine atıyoruz. Farklı 2. katman protokolleri IP Paketlerini farklı şekilde kapsüllenebilir. *Strip* elemanı bu ayırımı yapacaktır.

"sistem IP başlığını RFC 791', e göre kontrol eder." sorumluluğunu *CheckIPheader* nesnesine atıyoruz. Bozuk IP paketlerini bu nesne atacaktır.

Sistem Paketin içindeki varış IP adresi(*dest_ip*), kaynak IP adresi (*src_ip*), ve akış ile ilgili bilgileri(protokol, varsa varış kaynak portları vs...) belirler sorumluluğunu *GetFlowIds* element nesnesine atıyoruz.

sistem paketi IPSec kırmızıdan paket alma(*IPSecRedRecieve*) modülüne verir.

Buraya kadar ki akış tamamen sırasal idi. Click tasarım kalıbındaki elementlerden yararlanarak, *IPSecRedRecieveModule* element sınıfını tasarımıza ekliyoruz. *IPSecRedRecieveModule*'ünü Element sınıfından türetiyoruz.[2]

"IPSec Kırmızıdan alma modülünde, Pakete ilişkin **karar**, **çıkan politika kural veri tabanı**na (outbound *Security Policy Database* SPD) sırayla bakılarak bulunur."

Kuralların tutulduğu *Security Policy Database* sınıfını , tasarım domenine ekliyoruz.

"IPSec Kırmızıdan alma modülünde, Pakete ilişkin **karar** , **çıkan politika kural veri tabanına** (outbound policy, **Security Policy Database SPD**) sırayla bakılarak bulunur.

"Sistem **Kural**lara sırayla tek tek yukarıdan aşağıya kuralda belirtilen **seçici**lere uyup uymadığı belirlenerek bakar "

"Seçiciler paket içindeki belirli alanlara(seçici alanları diyebiliriz) bakarlar. Paketin bir kurala uyması için, tüm seçicilere uyması gerekir. Seçici alanlar varış adresi, kaynak adresi , protokol numarası, varış ve kaynak portu vs olabilir, Seçiciler

[1] İyi uyum kalıbı
[2] Çok şekillilik kalıbı

tiplerine göre tek bir değeri (exact match), bir aralığı(range match) ya da olabilecek her değeri kapsayabilir (any match,*). Bir kuralda tüm bu seçicilerin olması zorunlu değildir.“

Security Policy Database sınıfı kuralları tutar. Dolayısıyla *Rule(*kural*)* sınıfını da tasarım modeline ekliyoruz. Her kuralın belirli miktarda seçici taşıdığı senaryoda bahsedilmişti.

IPSecRecieve modülü , pakete ilişkin kararın ne olduğunu *Security policy database* sınıfına sorar, S*ecurityPolicyDatabase* sınıfı da tuttuğu kurallara tek tek sorar, eğer bir paket kuralın tüm seçicilerine uyuyorsa, geriye o kuralın kararı döndürülüyor. Bu amaçla tasarım modeline Kural(*Rule*) nesnesini de ekliyoruz. Her kural belirli miktarda seçici taşıyordu, dolayısıyla tasarım modelimize **seçici grubu** (*SelectorGroup*) ve **seçici** (*Selector*) nesnelerini de eklememiz faydalı olacaktır. Bilindiği gibi en az beş çeşit seçici mevcut dolayısıyla tüm seçicileri bu ana seçiciden türetip, paketin bunlara uyup uymadığı bilgisi yoklanabilir. Ayrıca senaryo da yeni seçici tiplerinin de eklenebileceği önceden öngörülmüş gezginlik(mobility) ve ICMP mesaj tipleri gibi).[1] seçici grubu bu seçicilerin iç yapılarını bilmeksizin bu nesnelerin *isSuit* metotlarını çağırır, ve seçici istediği alana bakıp kendisine uygun olup olmadığını geri döndürür. Tüm seçicileri ortak bir sınıftan türeterek Seçici grup ve kural sınıflarının bundan etkilenmemesi sağlamış oluyoruz.

Security policy database sınıfının işlevini, *ipsecredrecieve* modülü sınıfına da bırakabilirdik. Ancak iyi uyum prensibi ve daha hızlı yeni kural arama algoritmalarının geliştirilebileceği düşüncesi ile bu sınıfı ayırmaya karar verdik.[2] Böylece yeni bir kural arama algoritması gerçekleyeceğimiz zaman bu sınıfı *security policy database(SPD)* sınıfından türeteceğiz ve tasarımımız geri kalan kısmı bundan etkilenmeyecektir. Aynı nedenle *SPD* sınıfı pakete ilişkin kararı(*Decision*) kendi üzerinde gerçekleyebilirdi, biz ayrı bir karar nesnesi oluşturarak bu bağımlığı azalmış olduk[3]. Ayrıca ilerde NAT[34] ve gezginlik (mobility) desteği ile daha karmaşık kararlar sisteme eklenebilir bir alt yapı oluşturmuş olduk. Yani karar bir güvenlik kararı olabileceği gibi NAT yapma kararı da olabilir, ip_in_ip kapsülleme kararı da olabilir. Böylece sistemimizi esnek ve sağlam bir şekilde tasarlamış olacağız.

[1] Çok şekillilik kalıbı
[2] İyi uyum kalıbı
[3] Az bağımlılık kalıbı

Kullanım senaryolarında belirli durdurulan paketler için alarm (*audit*) verilebileceği belirtilmiş. Durdurma kararlarını *DiscardDecision* sınıfına atadık. Ancak verilecek alarm ve bu alarma karşı verilecek tepkiler daha çok kullanıcı arayüz kısmını ilgilendirmektedir. Alarmlar ekranda gösterilebilir, bir ağ sunumcusuna gönderilebilir ya da sesli uyarı ile sistem yöneticisine bildirilebilir. *DiscardDecision* nesnesi sadece alarm durumunun oluştuğunu anlayabilir. Burada gözlemci/yayıncı-abone kalıbı kullanarak, gözlemcilerin tümünü *DiscardListener* üst sınıfından türetiyoruz. Her bir gözlemci sınıfı *OnDiscardEvent* metodunu gerçekler, *DiscardDecision* (yayıncı) nesnesi de bu gözlemcileri kendi listesine kaydeder. Bir olay meydana geldiğinde bu abone nesnelere bildirir. [1]

Kullanım Senaryoları KS_2_alt1: SA'ya göre ESP tünel paketi oluşturma ayrı algoritmalar – genelleştirme, incelenirse, ESP tünel paketinin oluşturulması için SPI ve sıra numarası bilgilerine ihtiyaç vardır, uygulama domeninde bu bilgilere sahip olan nesne güvenlik birliği (Security Association)'dır. Dolayısıyla tasarım modelinde bu görevleri tutması için *SecurityAssociation* sınıfını tasarım modelimize ekliyoruz. [2]

"Sistem sonuçta **oluşan paket alanlarını** (payload+padding+ESP trailer), SA'da belirtilen **şifreleme algoritması, algoritma kipi** , anahtar, ve algoritma şartnamesinde belirtilen(RFC_ enc_ALGX_MODEY) dış ve iç kriptolojik senkronizasyon verileri ilere beraber şifrelenmek üzere **şifreleyiciye** verir." İfadesinden de anlaşılacağı gibi hangi algoritmaların kullanılacağı bilgisine de *SA* sahiptir. Ancak *SA*'nın algoritmaların ESP ile ilgili kullanımında ayrı algoritma şartnamelerine uyulur. Dolayısıyla şifreleme işlemlerini algoritma şartnamesi nesnelerine bırakmak en doğrusudur.[3] SA algoritma şartnamesine paketin başı sonu, şifrelenecek alanın başı sonu, asıllanacak alanın başı sonu ve anahtarları sağlar. Sonuçta ESP paketini algoritma şartnamesi oluşturur gerekli senkronizasyon verilerini de algoritma şartnamesine verir.

Şifreleme algoritması için dış doldurma (*explicit padding*) uygulanır, bu doldurma bilgisine sahip olan şifreleme algoritmasıdır.[4] Ancak ESP Şifreleme algoritması şartnamesi, algoritmaya özel yazıldığı için ona bu sorumluluk atanabilir, ayrıca doldurma biçimi de şifreleme algoritma şartnamesinde belirtilir.

---

[1] Gözlemci/yayıncı-abone kalıbı
[2] Bilginin uzmanı kalıbı
[3] Adaptör kalıbı
[4] Bilginin sahibi kalıbı

Şifreleme algoritma şartnamesine, iç IP paketinin başına işaretçi, sonuna işaretçi, ve uzunluk verilir. Algoritma şartnamesi uzunluğa bakarak paketin sonuna doldurma(*padding*) ekleme sorumluğunu atadık. Ayrıca gerekli senkronizasyon verileri üretmekten ve bunları paketin gerekli yerlerine yerleştirmekten de algoritma ESP şartnamesi sorumludur.

Şifreleme şartnamesi paketi, gerekli senkronizasyon verilerini (örneğin *IV*) ve şifreleme anahtarını paketi şifrelemesi için şifreleme algoritmasına verir. Şifreleme algoritması sadece şifreleme yapar.

Algoritmaya her seferinde şifreleme anahtarını vermek yerine bağlantı kurulduğunda bir kez vermek daha mantıklıdır. Ancak *IV*, her şifreleme de değiştiği için, *IV*' yi her bir şifreleme işlemi için ayrıca vermek gerekecektir.

Uygulama domeninde her algoritmanın kipi ayrı bir özellik olarak algoritmalarda belirtilmişti, ancak tasarım modelinde her bir algoritma kipinde ayrı bir şartnamenin yazılması gerekeceğinden böyle bir özelliği tasarım modeline eklemiyoruz. Farklı algoritma kipleri için farklı ESP algoritma şartnamesi nesneleri kullanılmalıdır. Farklı algoritma kipleri algoritmalara farklı senkronizasyon verileri vermektedir. Bu işlemi de yapacak olan yine algoritma şartnamesidir, böylece SA sınıfı da bu değişikliklerden etkilenmemiş olacaktır. Farklı algoritma şartnamelerini ortak bir algoritma şartnamesinden türeterek, SA' ya ortak bir arayüz sunmuş olacağız. Zaman içinde yeni algoritmalar tasarıma eklendiğinde tasarımın kalan kısmı bundan etkilenmeyecektir. Yapılması gereken algoritmayı gerçekleyip ortak arayüzü tasarıma katmak olacaktır.

ESN(genişletilmiş sıra numarası) opsiyonun seçili olup olmadığı seçili ise, değerini tutma sorumluğu yine SA nesnesine atanmıştır.[1]

Paketin başına dış IP'yi ekleme sorumluluğu SA nesnesine atanmıştır. SA ise bu iş için *IPSecOuterIPGenerator* yapay sınıfını kullanır. Böylece ek bir esneklik getirilerek IPv4 paketinin IPv6 tünellemesi, IPv6-IPv4 paketi tünellemesi ileriki aşamalarda kodda bir değişiklik yapmadan gerçeklenebilir.

Paketin sonuna trafik akış gizliliği doldurması (*tfc padding*) ekleme sorumluluğu SA nesnesine atanmıştır. İlerde karmaşık trafik akış gizliliği doldurması ekleme

[1] Bilginin uzmanı kalıbı

algoritmaları öngörülür ise bu iş için bir sınıfı tasarımıza eklemek faydalı olacaktır. Şuan için böyle bir yöntem öngörülmediğinden bu görevi SA' da bırakıyoruz.

Sıra numarasının taşıp taşmadığını kim kontrol edecektir? Tabi ki bu bilgiye sahip olan nesne SA. Sıra numarasının kontrolü ve arttırılması şifreleme ve bütünlük hesabından önce yapılır.[1]

Birleşik algoritmalarla ilgili elimizde şuan gerçek bir senaryo olmadığı için onunla ilgili işlemleri atlıyoruz. İleride birleşik bir algoritma gerçeklenir ise tasarımımızda gerekli değişiklikleri ileri iterasyonlarda yapılacaktır. Mevcut tasarım, diğer algoritmaların da kolayca eklenebileceği modüler yapıdadır

SA' lar tek yönlü, öyleyse SA nesnelerini de tek yönlü yapmak mantıklıdır. Çünkü aynı nesne aynı anda hem yollama hem de alma işlemlerinde kullanılmıyor. Örneğin göndericinin SA' sı sıra numarası kontrolü yapmazken, alıcı SA yapıyor. Böylece SA sınıfının aşırı derecede büyümesi de önlenmiş olur ve iyi uyum kalıbına sadık kalınmış olur.

Öyleyse bir önceki adımda oluşturduğumuz gönderici SA' ya, *SenderSA*, alıcı SA' ya *RecieverSA* demek tasarım modelinde daha mantıklı olacaktır. Eğer istersek bunları ortak bir SA sınıfından da türetebiliriz.[2]

Adım 2- siyahtan alınan paketi işlemek:

Kullanım senaryosu KS3'e bakarsak, kullanım senaryosu KS2 ile birçok ortak noktası olduğunu görürüz, burada kullanacağımız bir çok nesne KS2 senaryosu gerçekleştirilirken tespit edilmişti, bu adımda var olan nesneleri gerekli yerlerde yeni özellikler ekleyerek kullanacağız ve gerekli yeni nesneleri ekleyeceğiz.

İlk olarak tasarım modelimize siyah arayüz "*BlackInterface*" nesnesini ekliyoruz. UC3 senaryosunun adım 5'e kadar olan kısmında , *EthernetClassifier*, *Paint*, *CheckIpHedaer, GetFlowIds*, nesnelerini kullanıyoruz.

4.madde için, *IPSecBlackRecieveModule*, sınıfını tasarım modelimize ekliyoruz. *IPSecBlackRecieveModule* ' paketi açar ve açık paketi üretir.

[1] Bilginin uzmanı kalıbı
[2] Çok şekillik kalıbı

Alınan ESP paketinin, hangi SA ile açılacağını bilinmesi için bir SA veritabanına ihtiyaç vardır. Bunun için *SecurityAssociationDatabase(SAD)* sınıfını tasarım modelimize ekliyoruz.

Paketin açılacağı SA' yı bulma sorumluluğunu *SAD*'ye veriyoruz.

Alınan paketin sıra numarası kontrolü yapılması gerekir bu sorumluluğu SA sınıfına veriyoruz.

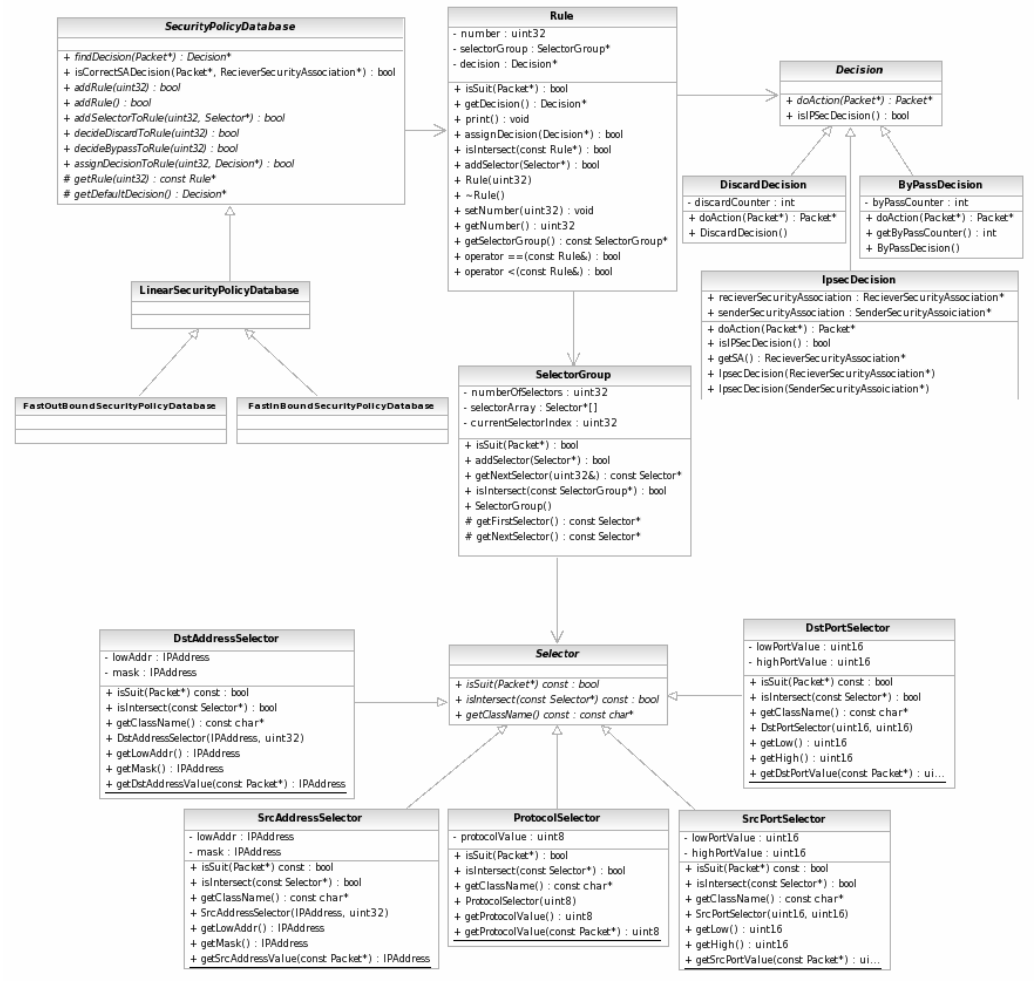
Senaryoda paketin bütünlüğünü hesaplamadan önce, bütünlük sınama değeri (*ICV)'* nin saklanması gerektiği ve eğer genişletilmiş sıra numarası kullanılıyor ise sıra numarasının yüksek anlamlı 32 bitini paketin sonuna eklenmesi gerektiği belirtilmiştir. Bu iş için en uygun nesne *RecieverSA* nesnesidir. Çünkü bu işlemler için gerekli bilgilere bu nesne sahiptir.

Daha sonra paketin bütünlük değerinin hesaplanması gerekir, bu iş için en uygun nesne, bir önceki adımda bu sorumluluk atanan *IntegrityAlgorithSpecification* sınıfından nesnelerdir, bu nesneler, gerekli iç doldurmayı (implicit padding) paketin sonuna eklerler, bütünlük hesabı yapılacak alanı işaretlerler ve paketi *ICV* değeri hesaplanmak üzere, *integrityalgortihm* sınıfından bir nesneye verirler. *ICV* değeri hesaplandıktan sonra, *integrityAlgorithSpecification* eklediği iç doldurmaları(*implicit pad*) paketin sonundan siler ve geriye *ICV* nesnesini döndürür, geriye neden bir *ICV* nesnesi döndürür, çünkü *ICV* değeri değişken uzunlukta bir alandır ve pakette yazılan *ICV* değeri ile hesaplanan *ICV* değerinin tuttuğunu anlamak için karşılaştırma işlemi yapmak yetmeye bilir, örneğin bazı algoritmalarda hesaplanan sonuç ile paketteki değer 1'e tümleyen aritmetiğinde toplandığında sonucun sıfır olması gerekiyor olabilir. Ayrıca bazı algoritmalar hesaplanan *ICV* değerin belirli bir uzunluğunu kullanmaktadırlar. *ICV* değerini *RecieverSecurityAssociation* nesnesi karşılaştırır. Genişletilmiş sıra numarasını siler, asıllama işlemi başarılı ile sıra numarası penceresini günceller ve paketi çözmesi için *Encryption Algoritmspecification* nesnesine yollar, *encryption algoritmspecification* şifre çözme alanlarını işaretler, paketi şifreleme algoritmasına (*EncryptionAlgorithm)* verir, çözülmüş paketi geri yollar. *Encryptionalgoritmspecification,* ESP kuyruk alanlarını ve doldurmaları (ESP trailer , pad) siler, çözülmüş paketi, *RecieverSA* nesnesine verir, son olarak *RecieverSA* nesnesi paketin ESP başlığını siler ve paketi geri döndürür. Farklı şifreleme algoritmaları ve algoritma ESP şartnameleri farklı

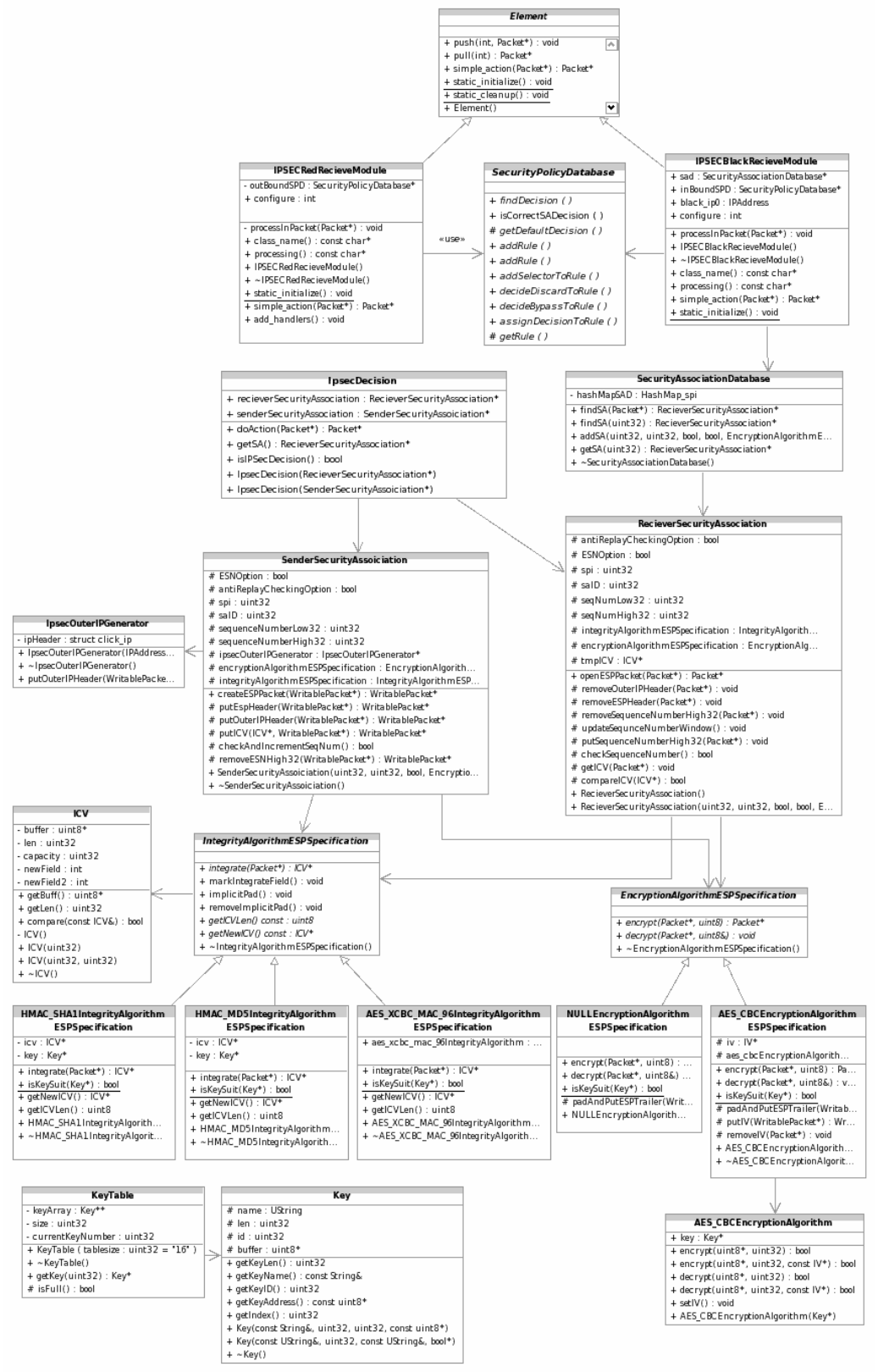
davranışlar gösterebilir, örneğin AES CBC kipi için, *IV* değeri veri (*Payload*)alanından okunup algoritmaya girdi olarak verilecek ve en son paketten çıkartılacaktır.

Uygulama domeninde algoritma kipi, algoritmanın bir özelliği idi, ama tasarım modelinden kip algoritmanın bir özelliği olarak düşünmedik ve her algoritma kipi için ayrı nesneler oluşturduk. Neden? Çünkü algoritma kiplerinin ESP ile kullanımı için ayrı algoritma ESP şartnameleri RFC olarak yazılıyor, o açıdan var olan algoritmanın bir özelliği yerine ayrı bir nesne olarak düşünmek tasarım açısından daha uygun görülmüştür.

Tasarım domeni sınıf diyagramlarından bir kısmı örnek olarak Şekil 4.5 ve Şekil 4.6'de gösterilmiştir.



Şekil 4.5: Tasarım domeni UML sınıf diyagramı örnek-1



Şekil 4.6: Tasarım domeni UML sınıf diyagramı örnek-2

3.adım ayarların yapılması:

Ayarların yapılmasında belirlenene sistem olayları:

Kural eklemek, Kurala seçici eklemek, Anahtar eklemek, Kurala karar atamak

Güvenlik Birliği(SA) eklemek:

Ayarlarla ilgili sistem ayarlarını atmak için bir senaryo denetçisine[1] ihtiyacımız var. Sistem olaylarının gerçekleştirilmesi ile ilgili bilgilere farklı sınıflar sahip olduğundan, sistem olaylarının iş için elimizdeki sınıfları kullanmak iyi uyum ve az bağımlılığı bozacaktır, Bu açıdan sorumluluğu *SecurityManagementConfiguration* yapay sınıfına veriyoruz.[2] Ayarların kaydedilmesini ve sistem açıldığında tekrar yüklenmesini de bu sınıfa yerine getirecektir. *SecurityManagementConfiguration* sınıfı sistem olaylarının sırasını denetlemekten ve işlemleri ilgili nesnelere yönlendirmekten sorumludur.

Kural eklemek:

Kuralları, güvenlik politika veri tabanı (*SecurityPolicyDatabase*) nesneleri tutmaktadır bu nedenle kuralları yaratıp, ekleme sorumluluğu yaratıcı kalıbı gereği, *SecurityPolicyDatabase* sınıfına atanmıştır[3]. Kural(*Rule*) nesnesi seçicileri tutan seçici grubu(*SelectorGroup*) nesnesini yaratır.[4] *SecurityManagementConfiguration (SMC)* kural ekleme görevini giren kurallar için *inBoundSecurityPolicyDatabase* nesnesine, çıkan kurallar için *outBoundSecurityPolicyDatabase* nesnesine aktarır.

Kurallara seçici eklemek:

Şuan için sistemde tanımlanmış 5 tür seçici vardır , ve kullanım senaryolarında yeni tür seçicilerin eklenebileceği belirtilmiştir. Seçici ekleme işlevi karmaşık bir işlemdir bu nedenle seçici ekleme sorumluğu *SelectorFactory* tekil fabrika sınıfına atanmıştır[5]. Kural bilgilerine *SecurityPolicyDatabase* sınıfı sahiptir. Seçicileri ise *selectorGroup* nesneleri tutmaktadır. *SMC* , seçici yaratma görevini, *SelectorFactory* sınıfına aktarır ve yaratılan seçiciyi kurala ekleme görevini *SPD* nesnesine aktarır, SPD, ilgili kural nesnesini bulur ve o kurala seçici eklemesi mesajını gönderir, kural ise seçiciyi *SelectorGroup* nesnesine eklettirir.

[1] Denetçi kalıbı
[2] Yapay sınıf kalıbı
[3] Yaratıcı kalıbı
[4] Yaratıcı kalıbı
[5] Tekil sınıf, fabrika kalıbı

Anahtar Eklemek:

Kullanım senaryolarında algoritmaların anahtarları kullanarak işlem yaptıkları ve anahtarların anahtar tablosunda sakladığı belirtilmiştir. Tasarıma anahtar(*Key*) ve anahtar tablosu (*KeyTable*) sınıflarını da ekliyoruz. Anahtarlar, anahtar tablosunda saklandıkları için *SCM* , anahtar nesnesi yaratma ve ekleme görevini Anahtar tablosuna aktarır. Sistemde tekbir anahtar sınıfı olması gerekir ve algoritmalarda tekbir erişim noktasına ihtiyaç duyduklarından. Anahtar tablosu (*KeyTable*) da tekil nesne olmalıdır.[1]

SA Eklemek: Güvenlik birlikleri (*SA*) güvenlik birliği veri tabanlarında(*SAD*) tutuluyordu. SCM *SA* ekleme sorumluluğu *SAD'* ye aktarır. Ancak SA eklerken, ilk olarak algoritma ESP şartnamesi nesnelerinin yaratılması gerekmektedir. ESP-şartnameleri ortak bir sınıftan türetilmiş ve yeni şartnamelerin zamanla sisteme ekleneceği belirtilmiştir. ESP şartname nesnelerinin yaratma görevini *SA* ya da *SPD* sınıflarına verilmesi, ilerde yeni şartnameler eklendiğinde bu sınıfların değiştirilmesine neden olacağından az bağımlık kalıbına da uyularak, tekil fabrika *EncryptionAlgorithmESPSpecificationTable* ve *Integrit_ AlgorithmESPSpecificatio - Table* sınıflarına bırakılmıştır. *SCM* ilk olarak anahtar kimlik numarası ile anahtar tablosundan ilgili anahtarı alır. *Encryption AlgorithmESPSpecificationTable* ve *IntegrityAlgorithmESPSpecificationTable* bu anahtarların algoritmalar tarafından kullanılıp kullanılmayacağını sorar.[2] SCM, *IntegrityAlgorithmESPSpecification* ve *EncryptionAlgorithmESPSpecificationTable* nesnelerini yaratma görevini tablo nesnelerine verir ve *SAD*' ye bu nesneleri de vererek SA ekleme mesajını yollar.

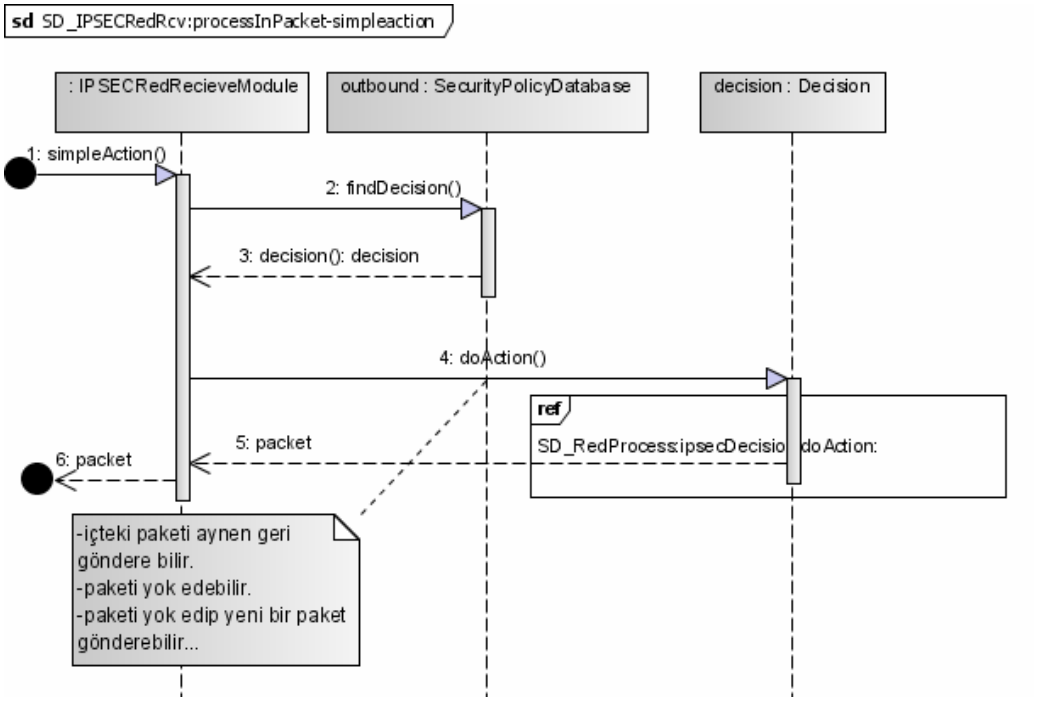
Kurala karar atamak:

Kuralları *SPD* nesneleri tutmakta idi. *SCM* karar nesnesini yaratır. Kural ekleme görevini ilgili *SPD* nesnelerine aktarır *SPD* nesneleri uygun kuralı bulur ve kurala kararı ekletir. Tüm kurallar tek bir karar sınıfından türetildiği için *SPD* ve kural kararın tipinden bağımsız olarak çalışırlar. Böylece sisteme yeni karmaşık kurallar ilerleyen aşamalarda kolayca eklenebilir.

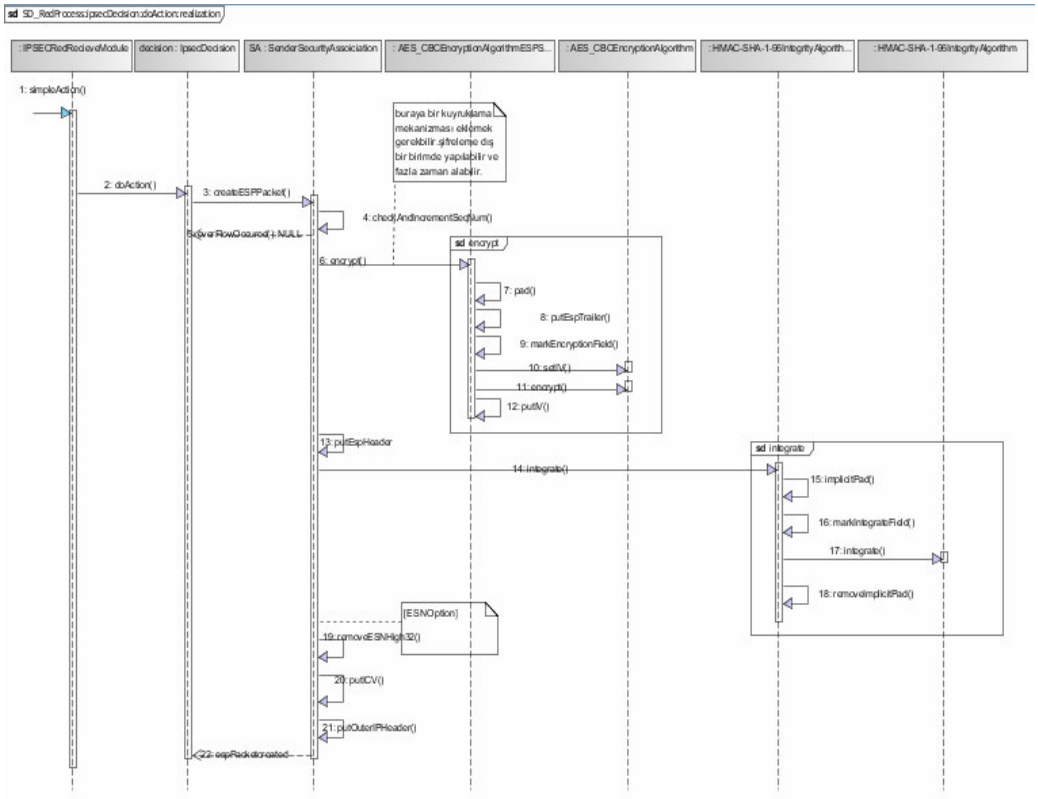
Son olarak güvenli arayüzden alınan paketi işleme ve ESP tünel paketi oluşturmaya ilişkin örnek UML ardışıl diyagramları Şekil 4.7 ve Şekil 4.8'de gösterilmiştir.

[1] Tekil nesne kalıbı

[2] Anahtarın uzunluğu algoritma için uygun olmaya bilir ya da anahtar algoritma için zayıf bir anahtar olabilir



Şekil 4.7: IPSec güvenli arayüzden alınan paketi işleme UML ardışıl diyagramı



Şekil 4.8: ESP paketi oluşturma UML ardışıl diyagramı

5 ÇOK BOYUTLU PAKET SINIFLANDIRMA

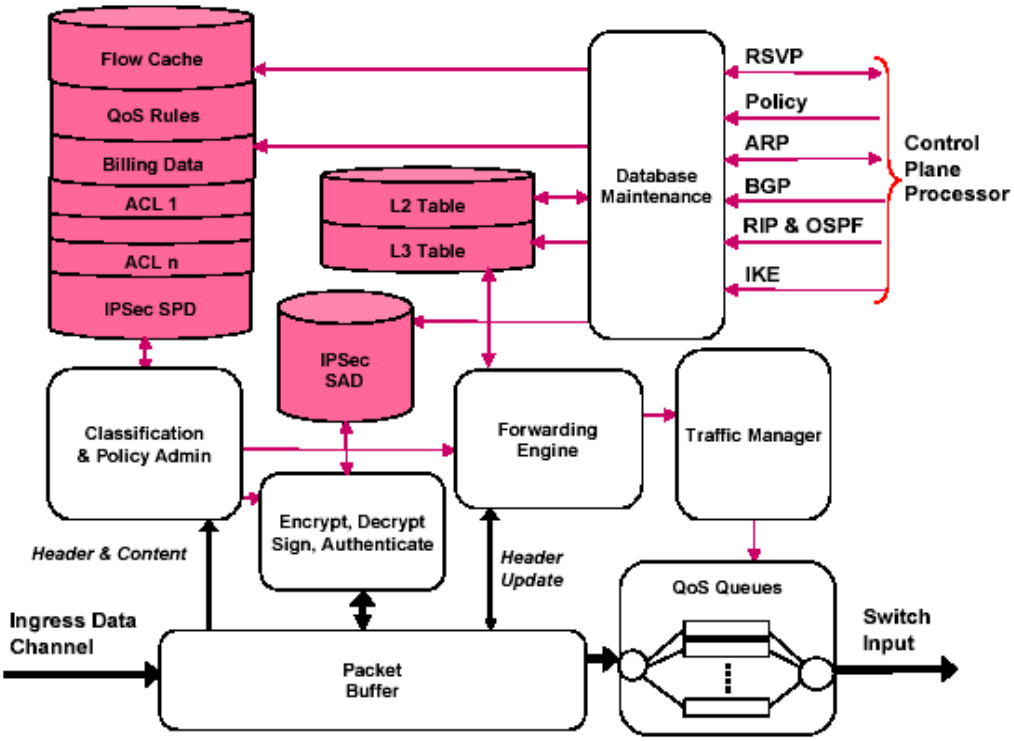
5.1 Giriş

Paket sınıflandırması, IPSec sanal özel ağ geçidi, güvenlik duvarı, NAT [34], (diff. Services), Hizmet kalitesi(QoS), IP üzerinden ses iletimi (VoIP), yük dengeleme (load balancing), politika tabanlı yönlendirme gibi uygulamalar için büyük önem teşkil etmektedir. Tüm bu uygulamalarda paketlerin uyduğu trafik akışı belirlenir ve bu akışa uyan tüm paketlere ilişkin belirlenen kararlara göre paket üzerinde işlemler yapılır. Paketin akışı, paket başlık alanlarına bakılarak belirlenir. Bir paket kural veri tabanındaki birden fazla kurala uyabilir, bu durumda en yüksek öncelikli kuralın kararı pakete uygulanır. Genellikle öncelik kural sırası olarak belirlenir (örneğin IPSec güvenlik politika veritabanı, güvenlik duvarı veri tabanı). Bazen de, IP yönlendirme aramasında olduğu gibi öncelik, en uzun eşleşme de olabilir. Genel olarak paket sınıflandırmanın yönlendirici mimarisindeki yeri Şekil 5.1'de gösterilmiştir. Paket üzerinde yapılacak işlem uygulamadan uygulamaya çok farklılık gösterse de, paket akışlarını sınıflandırma gereksinimi tüm bu uygulamalar için ortaktır. Paket sınıflandırılmadan paket üzerinde daha fazla işlem yapılamayacağı için paket sınıflandırılması ağ cihazının performansını doğrudan etkiler.

Ağ katmanı cihazları için, hat hızında paket iletebilme, en küçük boydaki paketleri verilen zaman aralığında iletebilmeyi gerektirmektedir, eğer bu sağlanamazsa ağ cihazı önemli bir paketi, daha önemli olup olmadığını bile belirleyemeden atacaktır[35]. Internet kullanımının 6 ayda 2 katına çıkmasıyla yüksek hızlı hatlar projelendirilmeye başlamıştır. Bugün kenar ağlarda bile hat hızı 10Mbit/sn Ethernet ağından, Gigabit Ethernet ağına çıkmıştır.

Band genişliği gereksinimiyle beraber ağdaki uç düğüm sayısı da hızla artmaktadır. Bu nedenle sınıflandırıcılardaki kural sayıları da her geçen gün artmaktadır. Günümüzde ticari IPSec güvenlik geçitlerinde 8000 - 16000 arasında tünel tanımlanmaktadır. IPv6 protokol yığınında IPSec 'in zorunlu gerçeklenmesi gereken protokol olarak kabul edilmesi, IPv6 ile birlikte gezginliğin artmasına paralel olarak

uç düğüm sayısının artması, IPv4'teki NAT , gezginlik gibi IPSec' in kullanılmasını zorlaştıran etkilerin IPv6 ile ortadan kalkmasıyla yakın gelecekte IPv6 ile birlikte IPSec' in kullanımının ve IPSec politika tablo boylarının daha da artması beklenmektedir.



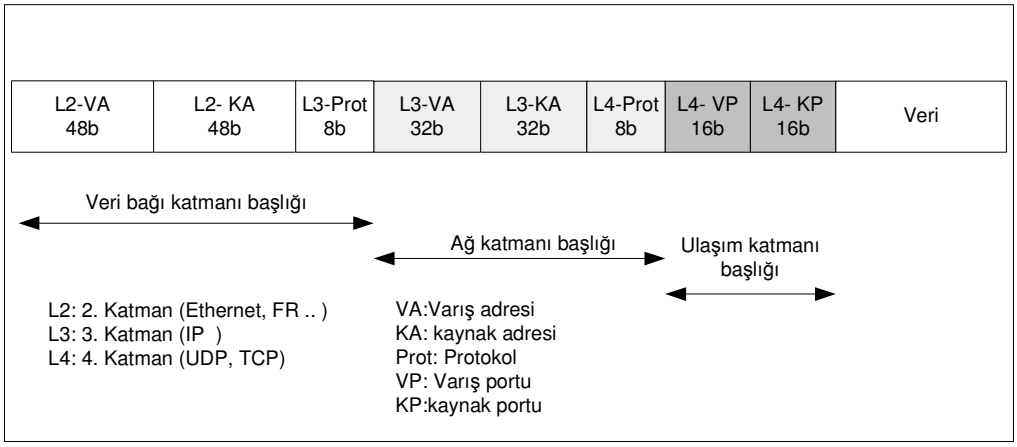
Şekil 5.1: Genel olarak bir ağ yönlendiricisinin mimarisi[36]

Literatürdeki paket sınıflandırma algoritmaları, kural veri tabanı boyu büyüdükçe, zaman ya da bellek miktarı açısından ölçeklenebilir değildir. TCAM [37] (içerik adresli bellek) gibi donanım çözümleri de veri tabanı boyu büyüdükçe yetersiz kalmaktadır.

Kural veri tabanı kuralların sonlu sıralı kümesidir. S = { R_1, R_2, R_3 …. R_N }. Her Kural k tane seçici parçadan oluşur. R[j] , paketin j. alanına ilişkin seçicidir. Her seçici paketin başlıklarındaki belirli bir alanın düzenli ifadesidir. Paket P' nin kural R ile eşleşiyorsa, $\forall$ i için , P' nin i. başlık alanı, R[i] düzenli ifadesini sağlar.

Pratikte seçiciler genel bir düzenli ifade olarak yer almazlar ve genellikle gösterimle sınırlı adres/maske , operatör / değer(ya da değerler) olarak ifade edilir. Örneğin "1.1.1.0 255.255.255.0" , "eşit 6" , "aralık 11-99" şeklinde gösterilir. Farklı ifade

şekilleri birbirilerine dönüştürülebilir, örneğin "1.1.1.0 255.255.255.0" , "aralık 1.1.1.0- 1.1.1.255" şeklinde de ifade edilebilir.



Şekil 5.2: Paket başlık alanları ve sınıflandırılmada kullanımı

Şekil 5.2'de paketin sınıflandırılmada kullanılabilecek başlık alanları gösterilmiştir. Uygulamalara göre farklı katmanlarda, farklı başlık alanlarına bakılarak sınıflandırma yapılabilir. Örneğin IPSec politika veri tabanlarında 3. katman IP kaynak ve varış adresleri, 4. katman protokolleri, kaynak-varış port başlık alanları sınıflandırmada kullanılır. Tablo 5-1 ' de örnek bir IPSec veri tabanı gösterilmiştir, IPSec seçici alanların gösterimi uygulamaya özgü olarak, IP adresleri için adres/maske uzunluğu, port değerleri de operatör/değer(ler) şeklinde ifade edilmiştir. Her kurala uyan paketler için bir karar da bu kural veri tabanlarında belirtilmiştir, sınıflandırma adımından sonra, kuralda belirtilmiş karara göre paketler işlenmeye devam edilir. Buna göre Tablo 5-2'de paketler ve paketlerin hangi kuralara uyacağı gösterilmiştir. Tablolardan Görüldüğü gibi kurallar arasında örtüşme de olabilir, bu durumda en öncelikli olan kural (sıra numarası küçük olan) seçilir. Eğer bir paket kural veri tabanındaki hiçbir kurala uymaz ise, pakete uygulamaya göre ön tanımlı bir karar uygulanır. Örneğin IPSec protokolü için durdur kararıyla paket atılır.

Tablo 5-1: 5 boyutlu örnek bir kural veri tabanı

Kural	Ağ katmanı ip varış adresi (adres /maske)	Ağ katmanı ip kaynak adresi (adres /maske)	Ulaşım Katmanı protokolü	Ulaşım katmanı kaynak portu	Ulaşım katmanı varış portu	Karar
R1	192.53.90.47 / 32	192.63.80.11/ 32	*	*	*	durdur
R2	192.58.100.0 / 24	192.53.10.47/ 32	udp	eşit 80	*	açık geçir
R3	10.1.1.0 / 24	212.26.1.0 / 24	tcp	büyük 1024	eşit 80	ipsec_tünel1
R4	*.*.*.*	*.*.*.*	*	*	*	durdur

Tablo 5-2: Örnek sınıflandırma sonuçları

Paket başlık alanları	Ağ katmanı ip varış adresi	Ağ katmanı ip kaynak adresi (adres /maske)	Ulaşım Katmanı protokolü	Ulaşım katmanı kaynak portu	Ulaşım katmanı varış portu	Eşleşen Kurallar
P1	10.1.1.24	212.26.1.29	TCP	2408	80	R3
P2	192.53.90.47	192.63.80.11	UDP	3509	500	R2
P3	192.26.5.1	212.4.5.1	UDP	23	204	R4

Sınıflandırma Probleminin Matematiksel Tanımı:

N: Kural veri tabanındaki kural sayısı

d : boyut, kural veri tabanının seçici alan sayısı

w_i : paketteki i. seçici başlık alanın genişliği

$R_{j:}$ kural veri tabanındaki j. kural

P≈Ri : P paketinin kural ile j. kural ile eşleşmesi

$S = \{R_i\}$ kuralların sonlu sıralı kümesi i =0,1…N; R_i . i. kuralı göstermektedir.

$\forall R_i$ kuralı $R_i[j]$ seçicilerinin kümesini içerir j = 1, 2,…d ; $R_i[j]$. i. kuralın j. seçicisi.

$\forall$ Seçici bir $[l_{ij}, u_{ij}]$ aralığını belirtir. $0 \le l_{ij}, u_{ij} \le U_j$ $U_j = 2^{w_i} - 1$

Paket P , p_j P'nin j. başlık alanındaki değer ; $0 \le p_j \le U_j$ j= 1,2… d tane paket başlık değerini taşır.

Problem: Paket P için öyle bir $R_i \in S$ bul ki ,

$P \approx Rj \ \forall j \ \ l_{ij} \le p_j \le u_{ij} \wedge (P \approx R_k \Rightarrow j \le k)$; j,k=1,2…d

Matematiksel olarak çok boyutlu paket sınıflandırılması, d boyutlu bir uzayda, bir noktanın yerinin bulunması problemidir. Hesaplamalı geometride, bu sorgulanan noktayı içeren nesnenin bulunması demektir.

N: kural sayısı, d sınıflandırılma boyutu (kullanılan seçici alan sayısı) olmak üzere, teorik olarak paket sınıflandırması alana göre optimize edilirse kurallar arası örtüşme olmadığı durumda $O(\log^{d-1} N)$ zaman ve O(N) lineer bellek miktarı, ya da zamana göre optimize edilirse logN zaman ve $O(N^d)$ bellek miktarı gerektirmektedir[38]. sadece 1000 kural ve 3 seçici alan için d=3, N = 1000 : $N^{d=}$ 1GB bellek ve $\log^{d-1} N$ =100 bellek erişimine denk düşer ki bu seçiciler arasında hiçbir örtüşmenin olmadığı durumda böyledir.

Teorik limitlerden çıkarılan sonuçlar, problemin en kötü durumda pratik bir çözümünün olmadığı ve tek bir algoritmanın tüm durumlar için etkin sonuç veremeyeceğidir[39].(Örneğin basit bağlantılı liste ya da donanım çözümleri küçük veri tabanları için etkin çözüm olurken büyük veri tabanlarında yetersiz kalırlar) Problemin en kötü durumda çözümünün pratik olmaması, uygulamanın karakteristiğini ve yaygın kural veri tabanları incelenerek, sezgisel bir çözüm bulunmasını zorunlu hale getirmektedir [40,41]. Bu çalışmada biz, gerçek dünyadaki IPSec güvenlik veri tabanlarında pratik bir çözüm olan bir algoritma geliştirmeyi hedefledik.

Paket sınıflandırma problemi tüm uygulamalarda aynı amaçla kullanılsa da bu uygulamaların kendilerine has kimi özellikleri vardır. Örnek olarak kaynak yetersizliği , hız ve bellek kullanımı sınırı, veri tabanı boyu, seçici olarak kullanılacak alanlar, kural seçicilerin kendine has özellikleri, veri tabanını güncelleme sıklığı, en kötü durum, ortalama durum performansı verilebilir[42].

Pratikte kullanılan veri tabanlarına bakıldığında IPSec veri tabanlarının şu karakteristiklere sahip olduğu gözlenmektedir:

- IPSec veri tabanlarında bugün için kural sayısı 8K-16K arasındadır, IPv6 protokol yığınında IPSec' in zorunlu gerçeklenmesi gereken IP güvenlik mimarisi olarak kabul edilmesiyle ve Internet'in büyümesiyle doğru orantılı bir şekilde politika veri tabanlarının büyümesi beklenmektedir.
- IPSec güvenlik veri tabanlarında , 5 alana bakılarak paketler seçilmektedir. 32 bit IP varış ve kaynak adresi (aralık ya da ağ adresi olarak), 8 bit protokol değeri, 16 bit TCP, UDP portları (aralık olarak)
- Güvenlik duvarı veri tabanlarının aksine, kurallar arasında örtüşme durumu oldukça azdır.
- En belirleyici alanlar sırasıyla uzak ağın IP adresi , yerel ağın IP adresi daha sonra protokol ve port değerleridir
- Kural veri tabanı üzerinde güncelleme genellikle yönetim merkezlerinden ya da güvenlik operatörleri tarafından elle yapıldığından , veri tabanı üzerinde güncelleme, sık değildir.

5.2 Tek Boyutlu Paket Sınıflandırma Algoritmaları

Tek boyutlu paket sınıflandırma algoritmaları, çok boyutlu paket sınıflandırma algoritmalarının temelini oluşturmaktadır.Bu nedenle bu bölümde ilk olarak tek boyutlu paket sınıflandırma algoritmaları anlatılacaktır.

Tek boyutlu paket sınıflandırma uygulamaları iki çeşittir:

- Tablodaki kayıtlar tek bir değeri içerir. Örneğin, ARP adres tablosunda tek bir 32 bitlik IP adresine karşılık düşen 48 bitlik MAC adresinin bulunması.
- Tablodaki kayıtlar bir çok değeri içerebilir ve bir değer birden fazla kayıtla eşleşebilir. Örneğin, yönlendirme tablosunda IP adresi için en uzun eşleşmenin bulunması. Tablodaki kayıtlar genellikle bir değer aralığını içerir. (v_1,v_2)

Birinci tip uygulamalarda genellikle çırpı fonksiyonları ile doğrudan eşleşen kayıt hızlı bir şekilde (O(1)) bulunurken, ikinci. tip uygulamalarda çözüm bu kadar kolay olmamaktadır. Bu bölümde 2. tip uygulamalar için geliştirilmiş sınıflandırma algoritmaları üzerinde durulacaktır.

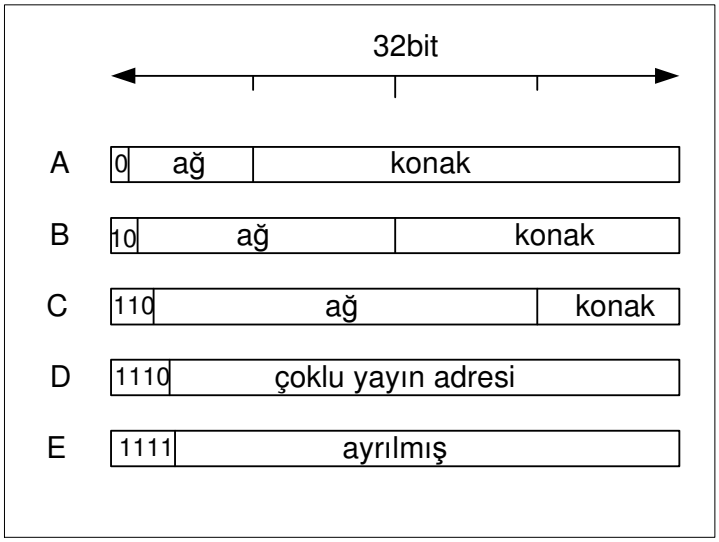
Algoritmaların çalışması da ağ. katmanı cihazlarında sıkça kullanılan IP adres araması üzerinde anlatılacaktır.

IP adres araması: Tüm yönlendiricilerin yapması gereken ilk görev IP yönlendirmesidir. Yönlendiriciler, paketin başlığında bulunan varış adresi alanına bakarak paketi bir sonraki yönlendiriciye iletir. Bu işlem paket hedefine varıncaya kadar devam eder. Yönlendiriciler bu işlem için yönlendirme tablosu tutarlar. Yönlendirme tabloları ağ yöneticisi tarafından elle ya da yönlendiriciler de koşan yönlendirme protokolleri tarafından güncellenir.

IP Adres Yapısı: IP adresleri[1] 32 bit uzunluğundadır ve iki parçadan oluşmaktadır. İlk parça ağı, ikinci parça ağ içindeki sistemleri temsil eder.(konak bilgisayarlar, ağ cihazları vs..). IP adresi bir konağı adreslemez, ağ arayüzünü adresler. Internet'in ilk çıktığı yıllarda IP adresi 5 kategoriye bölünmüştü. Sınıflandırmalı adresleme olarak adlandırılan bu adresleme yöntemi Şekil 5.3' de gösterilmiştir. Ancak sınıflandırmalı adresleme IP adreslerinin verimsiz kullanılmasına neden olmaktadır. Bir çok kuruluş için A sınıfı ağ çok büyük iken, C sınıfı adres çok küçüktür. Bir B

[1] Yazıda ek bir ifade kullanılmadığı sürece IP adresi , IPv4 adresini ifade etmektedir.

sınıfı ağ ise 65536 konak adresi içerir. Çalışmalar B sınıfı ağların yarısından fazlasının 50'den daha az bilgisayar içerdiğini göstermiştir.



Şekil 5.3: Sınıflandırmalı IP adres formatı

Zamanla IP'nin yaygınlaşması ile birlikte IP adresleri yetersiz kalmış ve çözüm olarak sınıfsız adresleme şekli getirilmiştir.(CIDR). Bu yöntemle ağ adresi kısmının uzunluğu değişken olabilmektedir. Böylece IP adresleri de değişken uzunlukta bloklar halinde dağıtılabilir. Sözgelimi 2000 adrese ihtiyaç duyan bir kuruluş 65536 yerine 2048'lik bir adres bloğu alabilmektedir. Tablo 5-3' de 172.34.0.0 / 16 ağ adresinin, CIDR ile farklı alt ağlara bölünmesi gösterilmiştir.

CIDR IP adres kıtlığına bir çözüm sunarken, IP yönlendiricilerinin işini de zorlaştırmaktadır. Sınıflandırmalı adresleme yapısında yönlendirici ilk 4 bite bakıp, adresin sınıfını belirlemekte ve daha sonra sabit uzunlukta adresi A, B ya da C yönlendirme tablosunda arar. A ve B sınıf ağlar için doğrudan indeksle erişim yapılırken, C sınıf için çırpı tabloları yardımıyla aranan kayıt oldukça kolay bulunur. Ancak bu basit algoritma CIDR ile beraber kullanılamaz çünkü her kayıtla beraber 32 bitlik bir ağ maskesi tutulmasına gerek vardır.(IP adresi , ağ maskesi, sonraki sekme). Bir paket geldiğinde onun varış adresi bulunur ve yönlendirme tablosu (kavramsal olarak) kayıt kayıt taranarak eşleşme aranır. Birden fazla kayıtla eşleşme olabilir, bu durumda ağ maskesi en uzun olan kayıt kullanılır. Arama işleminin iki parametreye dayanması (adres değeri ve uzunluk) karmaşıklığı arttırmaktadır. Bu bölümde IP adres araması problemine çözüm olabilecek, bellek ve hız açısından bazı temel algoritmalar anlatılmaktadır.

Ağ	İlk Adres	Son Adres	Uç sayısı	Gösterim
Ağ1	172.34.0.0	172.34.7.255	2048	172.34.0.0/21
Ağ2	172.34.8.0	172.34.11.255	1024	172.34.8.0/22
Ağ3	172.34.12.0	172.34.15.255	1024	172.34.12.0/22
Ağ4	172.34.16.0	172.34.31.255	4096	172.34.12.0/20

Tablo 5-3: CIDR adreslemesinin kullanımı

5.2.1 İkili Trie Ağacı

Trie[1] temelde bir ağaç veri yapısıdır. Ancak normal ağaçlardan farklı olarak düğümler anahtar değerleri tutmazlar. Düğümlerin ağaçtaki konumları ifade ettikleri değeri belirtir [43]. Şekil 5.4 ' de ikili trie ağacının çalışma biçimi gösterilmiştir. 1011* gösterimi, ön ekin(ağ adresi) '1011' ve maske uzunluğunun 4 olduğunu belirtmektedir. Paketin seçici alanındaki değer bit bit gezilerek ağaç üzerinde aşağıya doğru inilir. Örnekte '1' olan bitler için sağa, '0' için sola doğru olan dallar üzerinde ilerlenmektedir. En son gelinen düğüm en uzun eşleşen kuralı belirtir.

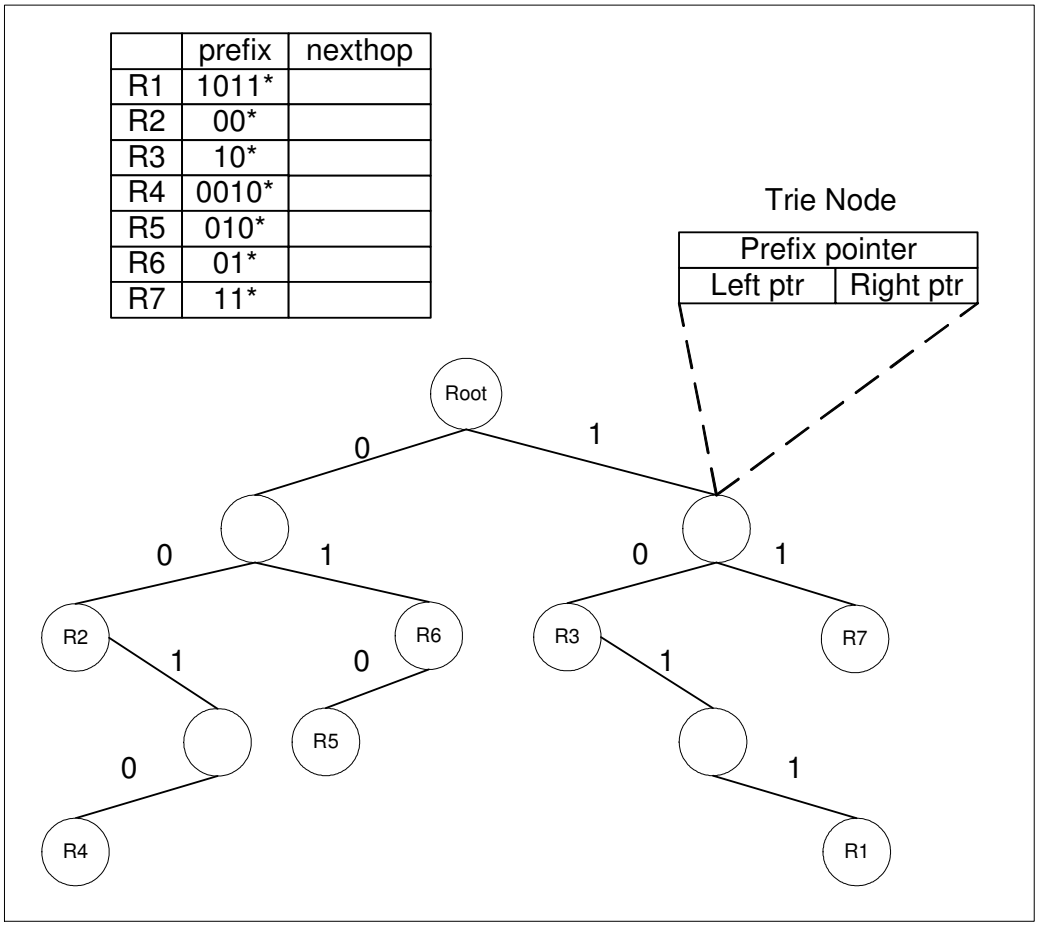
Arama işlemi için, en kötü durumda seçici alanın genişliği(W) kadar bellek erişimi yapılır; IPv4 adres araması için en fazla 32 bellek erişimi yapılarak hedef kural bulunur. Güncelleme işlemi de ilk olarak arama işlemi gerektirdiğinden en fazla 32 bellek erişimi ile güncelleme yapılabilir. N tablodaki kayıt sayısı ve W adres genişliği olmak üzere:Bir kayıt eklemek, trie ağacı üzerinde en fazla W tane ara düğüm oluşturabileceğinden bellek ihtiyacı NxW^2. En kötü durumda algoritmanın karmaşıklıkları şöyledir:

Arama karmaşıklığı : O(W)

Güncelleme karmaşıklığı : O(W)

Bellek Gereksinimi : O(NW)

[1] "Trie" kelimesi İngilizce "re**trie**val" kelimesinden gelmektedir ve "tray" şeklinde okunmaktadır
[2] Düğümler farklı kayıtlar tarafından paylaşılacağından gerçekte bu üst limit N değeri büyüdükçe azalacaktır



Şekil 5.4: İkili trie ağacının çalışma biçimi

5.2.2 Çok Bitli Trie Ağacı

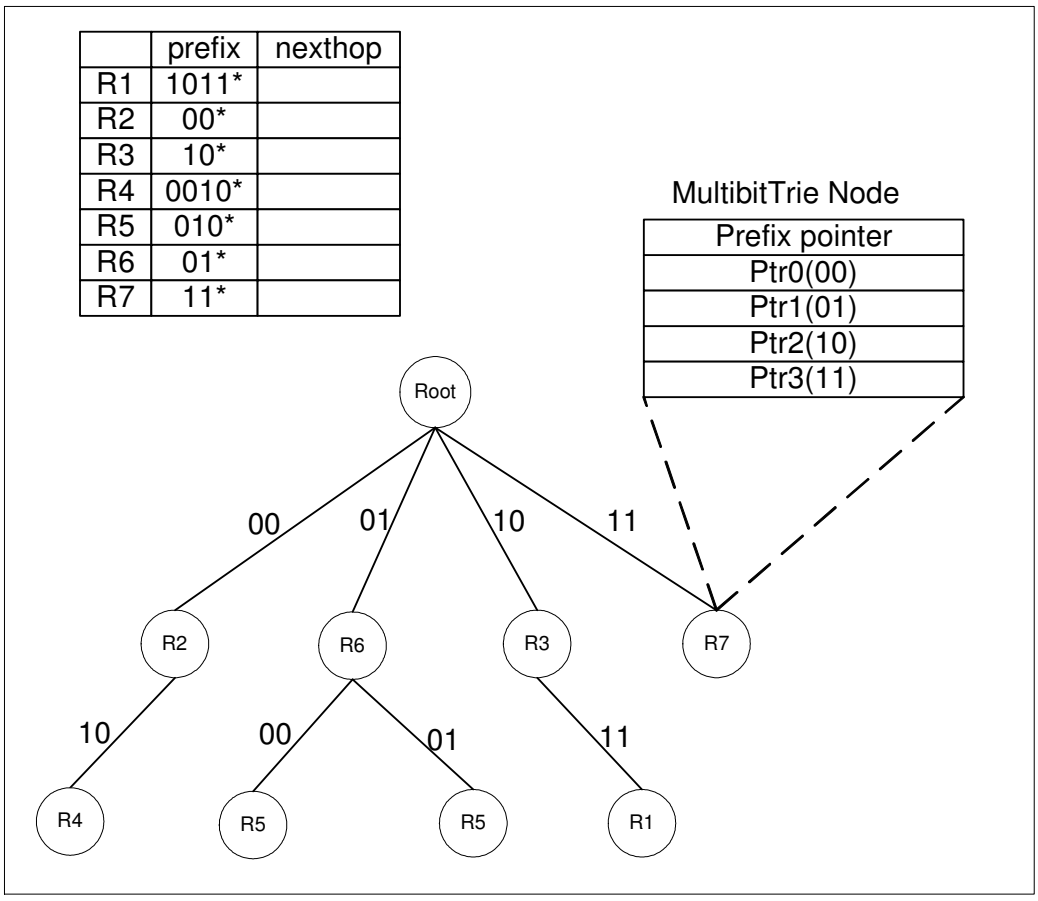
İkili trie ağacında her bir adımda tek bir bite bakılmakta idi. Çok bitli trie ağacında ise bir seferde birden fazla bit ilerlenerek bellek erişimi azaltılmaktadır. İlerleme miktarı sekme katsayısı(k) olarak adlandırılır. Sekme katsayısı arttıkça bellek tüketimi de artar. Her seviyede farklı sekme katsayısı da kullanılabilir.(k_1-k_2..k_L Trie)

Pratikte 8-4-4-4-4-4-4 IP adres araması için çok bitli trie ağacı kullanımı yaygındır. Bu durumda en fazla 7 adımda eşleşen kural bulunur. Sekme katsayısı sabit 2 için Şekil 5.5' de çok-bitli trie ağacının çalışma biçimi gösterilmiştir. Sabit sekme katsayısı seçilirse, arama W/k adımda bulunabilir. Güncelleme ise en kötü durumda, önce arama ve 2^{k-1} düğümde değişiklik gerektirebileceğinden, W/k + 2^k bellek erişimi gerektirir. Bellek tüketimi, k ile beraber üstel bir şekilde artar[42,44]. Bir düğüm eklemek W/k düğüm ve her düğümde 2^k tane alt trie ağacı oluşturabileceğinden bellek tüketimi en kötü durumda 2^k NW/k olur.

Arama karmaşıklığı : O(W/k)

Güncelleme karmaşıklığı : O(W/k + 2^k)

Bellek Gereksinimi : O(2^k NW/k)



Şekil 5.5: Çok bitli trie ağacı çalışma biçimi

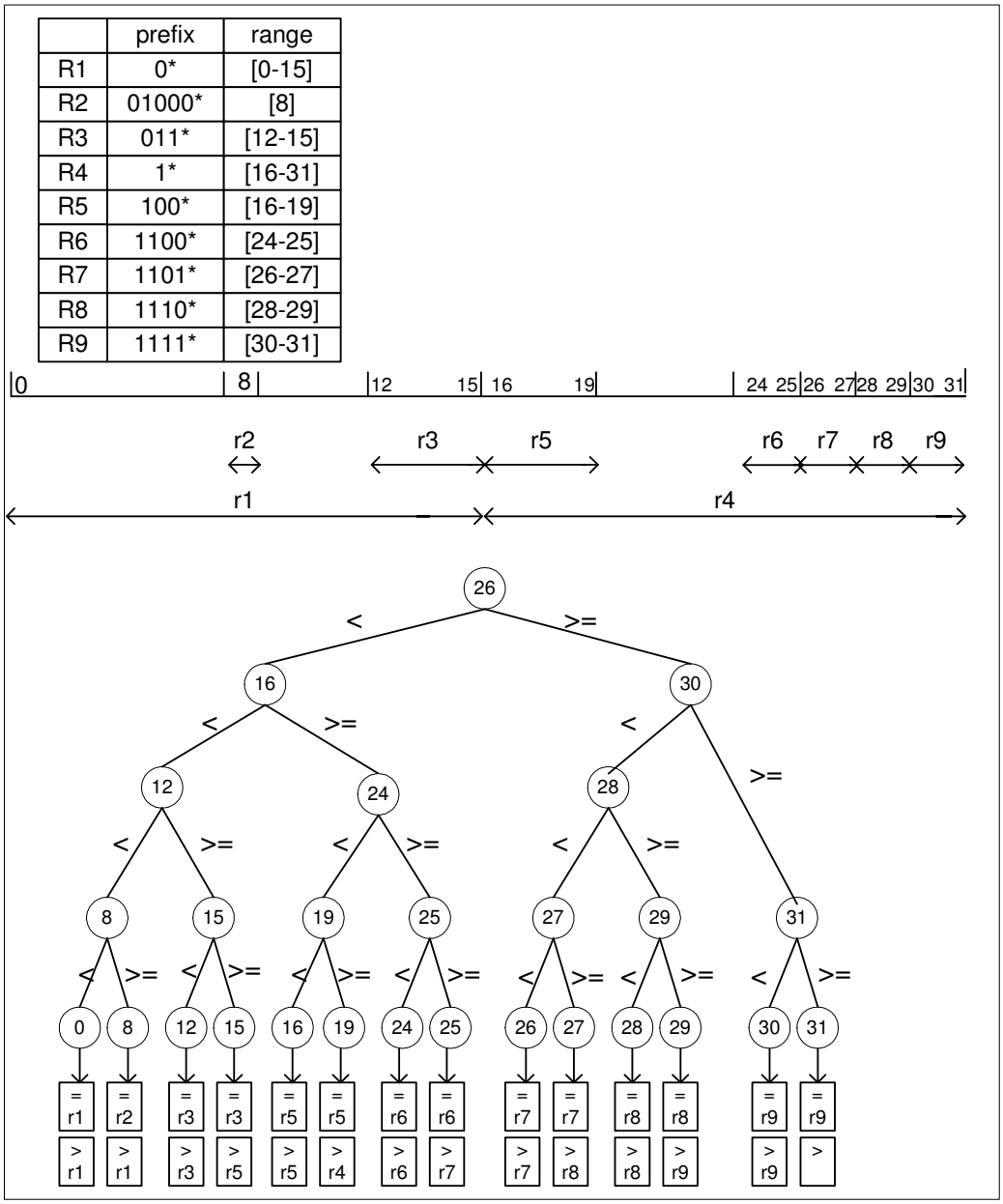
Trie ağaçları üzerinde yol-sıkıştırmalı Trie, seviye-sıkıştırmalı Trie [45], Lulea Trie [46] gibi bellek ve hız açısından iyileştirme yapılmış algoritmalarda bulunmaktadır.

5.2.3 Aralık Ağacı

Adres/maske-uzunluğu gösterimi ardışık adres aralıklarını tanımlar. Örneğin 5-bit uzunluğunda adres için 1*, [16,32] adres aralığını belirtir. Eğer aralıkların uç noktalarını bir ikili dengeli ağaç yapısında sıralı olarak tutarsak, eşleşen aralığı ikili ağaç üzerinde ilerleyerek bulabiliriz. Ancak bu yöntem aralıkların örtüştüğü durumda çalışmayacaktır. Eğer gerçek aralıklar, birbirinden ayrık aralıklara dönüştürülürse ve bu ayrık aralıklara göre ikili ağaç oluşturulursa bu problem çözülebilir.

Örneğin kural kümesi r1=1*,r2=10* olursa , aralıklar [16,32] , [16,23] olur. Bu durumda ayrık aralıklar [16,23] ve [24,32] olur. Örtüşme durumlarında ayrık aralığa, bu aralığı kapsayan en küçük aralığın (en uzun eşleşme) kuralı atanır.

Şekil 5.6 örnek bir adres tablosu için oluşturulan aralık ağacı gösterilmiştir. Örneğin 22 değeri aranıyorsa ilk 26 ile karşılaştırılır, küçük olduğundan sola dallanılır, daha sonra 16 ile karşılaştırılır ve sağa dallanılır. 24'le karşılaştırılır ve 19' a ulaşılır. 22 19'dan büyük olduğundan eşleşen kural r4 olarak bulunur.



Şekil 5.6: Aralık ağacı algoritmasının çalışması

Aralık ağacı yaklaşımı, problemi uzunluk boyutundan kurtararak, aramayı sınırları ayrılmış parçalı aralıklar içinde yapmaktadır. Ana aralıkların sayısı, kayıtlar arasındaki örtüşmeye bağlıdır. Ancak en kötü durumda 2N aralık olabilir. Ağaçta ikili arama yapıldığından arama karmaşıklığı $O(\log_2 N)$. Arama algoritmasında aralık sınırları tutulduğundan bellek gereksinimi karmaşıklığı O(N). Her ana aralık için, en iyi eşleşmenin önceden hesaplanması gerektiği için güncelleme karmaşıklığı O(N) olur.

Arama karmaşıklığı : $O(\log_2 N)$

Güncelleme karmaşıklığı : O(N)

Bellek Gereksinimi : O(N)

5.3 Çok Boyutlu Paket Sınıflandırma Algoritmaları

Daha önceki çalışmalar göstermiştir ki çok boyutlu paket sınıflandırma problemi karmaşıklığı yüksek bir problemdir. Pratik olarak bilinen cihazlar, ya lineer arama yapmakta ya da TCAM tarzı donanımsal çözümler kullanmaktadır[40].

Bu bölümde çok boyutlu paket sınıflandırma algoritmalarından bazıları anlatılmaktadır.

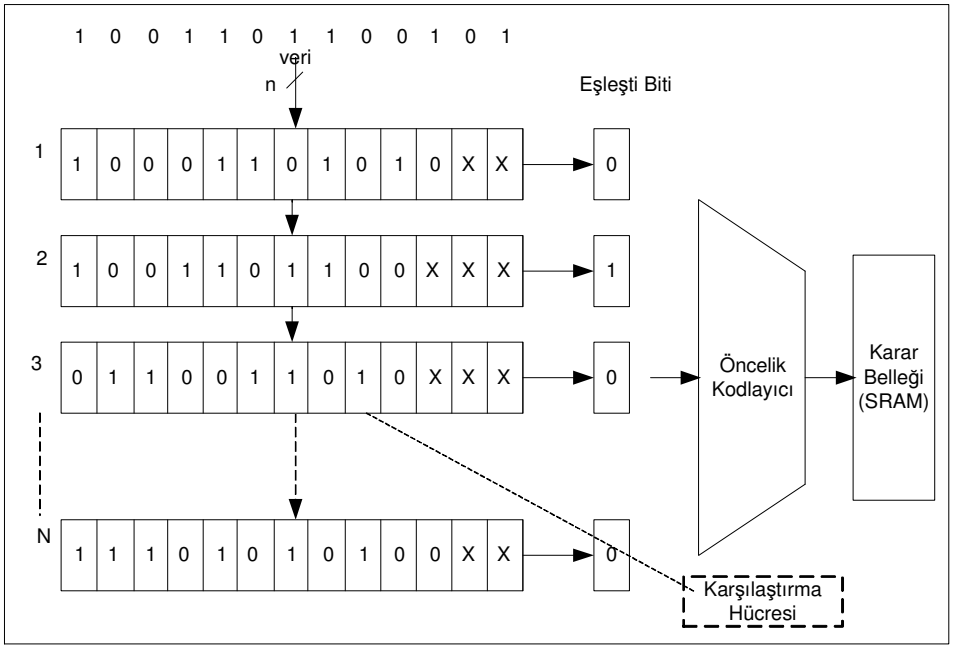
5.3.1 TCAM

TCAM donanımsal olarak içerikle adreslenebilir bellek sunmaktadır. Klasik CAM donanımlarından olarak bir TCAM hücresi 3 değeri tutabilir: 0,1,X. Burada X adreslemede dikkate alınmayacak alanları maskelemek için kullanılır. TCAM donanım çalışma mantığı Şekil 5.7' de gösterilmiştir. Her bir satırda bir karşılaştırma hücresi bulunur, karşılaştırma hücresindeki değer (0, 1, X) ile gelen veri karşılaştırılır, eğer bir satırdaki tüm hücrelerin karşılaştırma sonucu bir ise eşleşme biti 1 yapılır. Birden fazla satırda eşleşme olabileceği için öncelik kodlayıcı ile bunlardan uygun olan belirlenir. Her satırda karşılaştırma paralel olarak yapıldığında algoritmanın karmaşıklığı O(1) olur. TCAM oldukça yaygın olmasına rağmen,

- normal belleklerden oldukça fazla sayıda transistor içerirler. Karşılaştırma lojik devresi karmaşıklığı artırmaktadır. SRAM' de bit başına 4-6 transistor kullanılmaktadır TCAM donanımında bit başına 11-15 transistor vardır[47].
- daha pahalıdır.

- daha çok güç harcamaktadır [47]
- büyük veri tabanları için ölçeklenebilir değildir.
- En önemlisi TCAM kul PC-tabanlı yönlendiriciler [23]için uygun değildir.

Ancak TCAM teknolojisinin ilerlemesi ile birlikte gelecekte büyük veri tabanları için de etkin çözüm olabilir.



Şekil 5.7: TCAM çalışma biçimi

TCAM diğer algoritmalarla birlikte çok erişilen kuralları saklanmasında kullanılabilir. Ancak çok erişilen akışlar için cep kullanımı da kısa süreli akışların sıklığı, ıskalama oranın yüksek olması nedeniyle pekte uygulanabilir değildir.

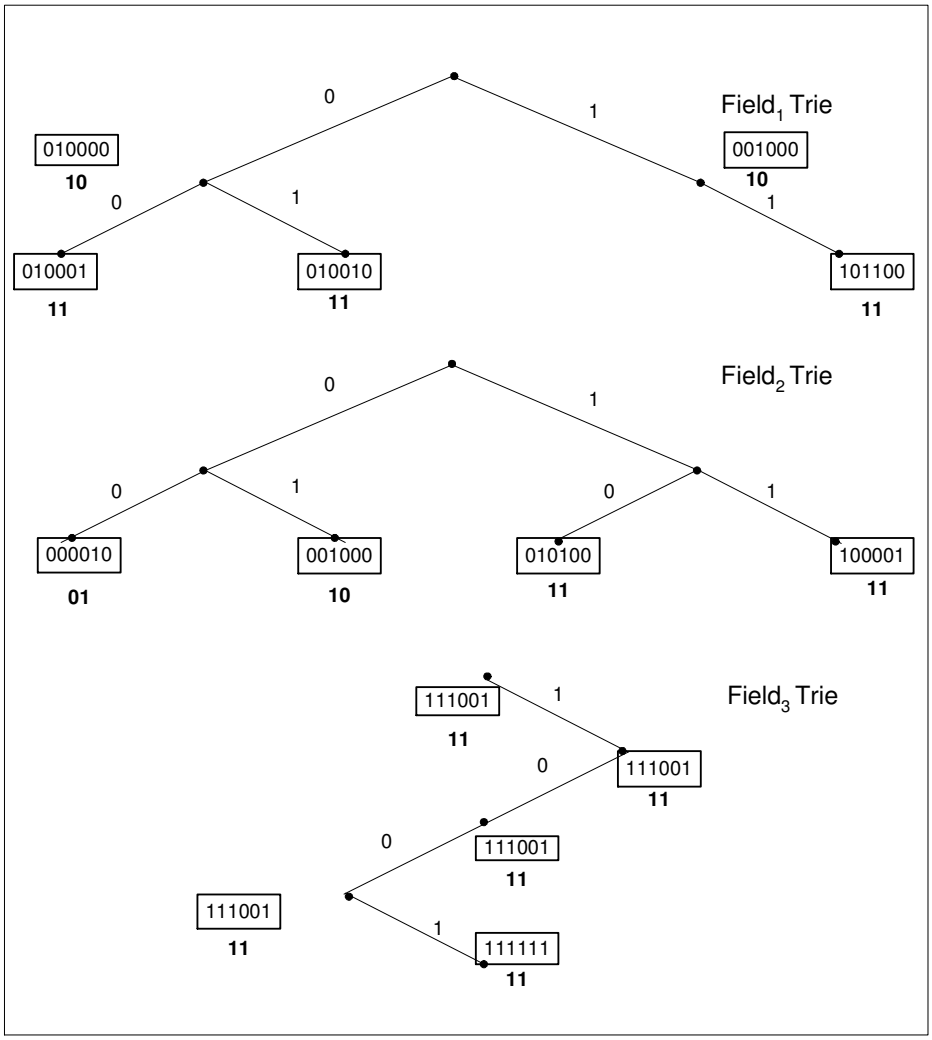
Tablo 5-4: 3 boyutlu 6 kurallı örnek veri tabanı

Rule	$Field_1$	$Field_2$	$Field_3$
R_1	11*	11*	*
R_2	0*	10*	*
R_3	1*	01*	*
R_4	11*	10*	1001
R_5	01*	00*	1001
R_6	00*	11*	*

5.3.2 Bit Vektör Algoritması

Bit vektör algoritması [35] problemi d alt probleme böler ve sonuçları birleştirir. Bu nedenle ilk olarak k tane tek boyutlu trie ağacı, veri tabanındaki her bir alan için

oluşturulur. Burada aralık ağaçları yerine trie ağacları kullanılmaktadır. Aralıkları, maskeli gösterime çevirme [48] da gösterilmiştir. Trie ağacının her bir düğümüne N-bitlik bit vektörü atanır. Vektörün soldan j. biti , j. kurala eşleşiyorsa 1dir. Bir paket geldiğinde p_1, p_2,..p_j,.. p_d her bir alan için ayrı ayrı Trie ağaçlarında en uzun eşleşme araması gerçekleştirilir. pj alanı Tj trie ağacında aranır ve düğümdeki M_j, bit vektörü elde edilir.



Şekil 5.8: Bit vektör ve sıkıştırılmış bit vektör algoritmalarının çalışma biçimi

Tüm alanların eşleşmelerini elde etmek için tüm bit vektörleri lojik VE'leme işlemine sokulur $S(V_j) = V_1$ & V_2 & …. V_j &…V_d Eşleşen kuralı bulmak için ilk 1 olan bite karşılık düşen kural seçilir.

Tablo 5-4' deki örnek veri tabanı için Şekil 5.8 bit vektör algoritmasının veri yapıları gösterilmiştir. Her düğümün yanında çerçeve içinde 6-bitlik bit vektörleridir.

Örneğin paket (0101,00101,10010) için V_1=001000 V_2=001000 V_3 =111001 olur ve sonuç

001000 & 001000 & 111001 = 001000 olur.Bu durumda paket R3 kuralına eşleşir.

BV algoritmasının karmaşıklığının hesaplanması:

1-d alan için ayrı ayrı trie ağacı araması yapmak için, $\sum_{j=1}^{d} w_j$, bellek erişimi gerekir. eğer tüm alanların genişliği eşitse $w \times d$, O(wd)

2-VE'leme işlemleriyle eşleşen kuralın bulunması, O(N) lojik VE işlemi gerektirir. en kötü durumda $\frac{N \times d}{M}$ bellek erişimi gerekir, M burada bir seferde erişilen bellek sözcüğü genişliğidir.

-Bit Vektör algoritması bellek erişimine belirli bir ölçüde iyileştirme getirmektedir. Örneğin 32-bitlik işlemciler de bellek erişimini 32 kat azatlasa da N ye lineer olarak bağlılığı değiştirmemektedir.

-arttırmalı güncellemeyi desteklememektedir, bu yüzden güncelleme karmaşıklığı oldukça yüksektir.

5.3.3 ABV Algoritması:

Bit vektör algoritmasında bellek erişimini azaltmak için sıkıştırılmış bit vektör algoritması[49] geliştirilmiştir. Veri tabanlarında tüm alanlarda eşleşmelerin düşük sayıda olduğu fark edilmiştir. Bir bit vektöründeki 1'lerin yoğunluğunun oldukça azdır. Önce A sıkıştırma boyu belirlenir. Ardışık A bit'i temsil etmek için, sıkıştırılmış bit vektörü (Aggregated Bit Vector) ayrıca oluşturulur,

Trie ağaçlarındaki her bir düğüm için , N/A-bitlik sıkıştırılmış bit vektörü oluşturulur :VA_j. VA_j'nin bir biti eğer $[V_{A\times j}, V_{A\times(j+1)})$ aralığında en az bir biti 1 ise 1'dir aksi halde 0 dır.

Şekil 5.8 ' de A= 3 için sıkıştırılmış bit vektörleri, normal bit vektörlerinin altında koyu yazıyla gösterilmiştir. Örneğin paket (0101,00101,10010) için VA_1=11 VA_2=01 VA_3 =11 olur ve sonuç VA = 11 & 01 & 11 = 01 böylece ilk 3 bit doğrudan atlanabilir.

ABV algoritmasının karmaşıklığının hesaplanması:

1-d alan için ayrı ayrı trie ağacında arama yapmak için, $\sum_{j=1}^{d} w_j$, bellek erişimi gerekir. eğer tüm alanların genişliği eşitse $w \times d$, O(wd)

2-VE'leme işlemleriyle eşleşen kuralın bulunması,

VE'leme işlemi gerektirir .en kötü durumda $\frac{N \times d}{M \times A}$ bellek erişimi gerekir.

Sıkıştırma çoğu zaman hız kazandırsa da bazı durumlar için de yavaşlatma getirebilir. Örneğin V_1 101010…10 , V_2 0101…01 olsun A=2 için VA_1 =11…111

VA_2=11..111 olur, böyle bir durumda Hem VA' bit vektörü algoritma daha da yavaşlayacaktır. Ancak pratikte böyle bir durumun oluşma ihtimali oldukça düşüktür.

- pratikte arama işlemine A kat hızlandırma getirmiştir ama karmaşıklık yine de O(N) ' dir.
- sıkıştırma ile güncelleme işlemi daha da zorlaşmıştır.

5.4 Yeni Yöntem

Tek boyutta arama yapmak, aynı anda çok boyutta arama yapmaktan çok daha kolaydır. Tek bir boyutta arama yapılarak paketle eşleşen gerçek kural bulunamasa da kural kümesi S kümeleri daraltılabilir.

Pratik veri tabanları incelendiğinde bazı alanların diğerlerine kıyasla sınıflandırma da daha ayırt edici olduğu fark edilir. Bazı alanlar için ise veri tabanında hep aynı tür de değer aralıklarına rastlanır. örneğin 8-bitlik protokol alanının bir çok veri tabanında *, =TCP ya da =UDP olarak yer alması gibi. Kısacası "Her alan seçicidir ama bazıları daha seçicidir." denilebilir.

Belirli bir alan için seçilecek etkin sınıflandırma algoritması büyük ölçüde seçici alanın kendine özgü özelliklerine, alan genişliğine(w) ve kural kümesinin büyüklüğüne (N) bağlıdır. Örneğin küçük N değeri için en iyi sonucu lineer arama verecektir.

Algoritmanın akışı:
1-Veri tabanındaki seçici alanlar F_1 F_2 F_3…F_d ayırt ediciliklerine göre sıralanır.

F_1 F_2 F_3…F_d → $F_{o1,}$ F_{o2} ,F_{o3}…,F_{od}

2-F_{o1} (ilk alan) için uygun Sınıflandırma algoritması seçilir ve S kural kümesi bu algoritma tarafından alt kümelere bölünür. S_1, S_2, S_3 …, S_x

3-Her bir alt küme için , bir sonraki alan (F_{oi+1}) için sınıflandırma yapılır ve alt kümenin, alt kümeleri oluşturulur.

4-tüm alanlar için 3. adım yinelemeli olarak tekrarlanır.

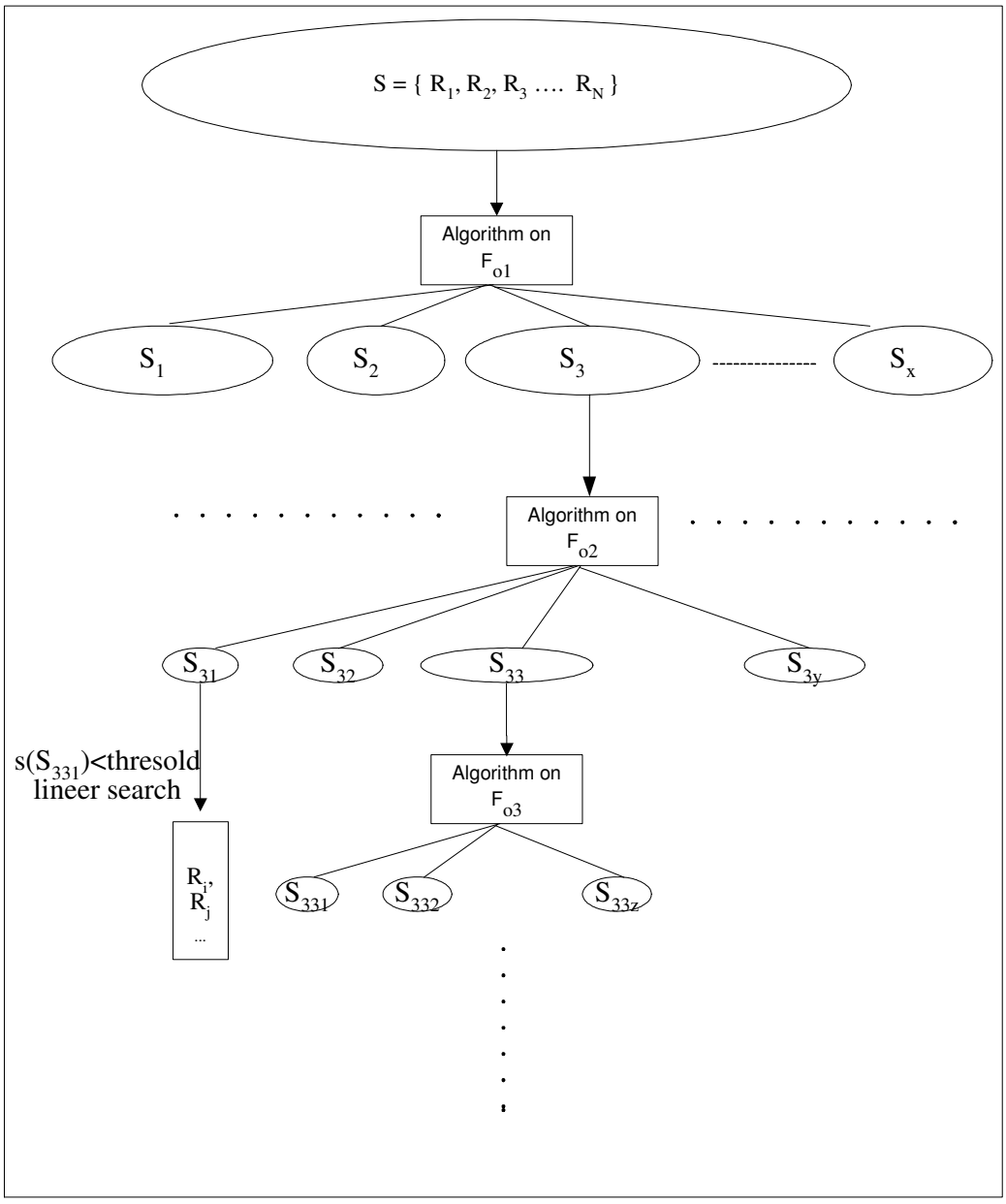
Her adımda, seçici alanın kendi özelliklerine ve kümenin büyüklüğüne uygun sınıflandırma algoritması seçilir. Her algoritma giriş kümesini, daha küçük alt kümelere böler.

Alt kümeler birbirinden ayrık olmak zorunda değildir , kümeler arası ortak elemanlar olabilir.

Sınıflandırma algoritmalarından çıkan her alt küme o alan için pakete uyan tüm kuralları içerir. Ama alt küme içinde paketin o alan için eşleşmeyeceği fazlalık kurallar da bulunabilir. Bu fazlalık kuralların sonraki sınıflandırma algoritma tarafından elenme ihtimali oldukça fazladır. Alt küme içinde bulunan fazla elemanlar ile alt küme sayısı azaltılarak bellek tüketimi ciddi oranda 3azaltılabilir.

Herhangi bir adımda alt kümedeki eleman sayısı tanımlanmış eşik değerinin altına düşerse, bu küme için kurallar arası lineer arama yapılır. Eşik değeri son adımda kalan eleman sayısına göre seçilebileceği gibi, toplam erişilen bellek miktarı göz önüne alınarak ta seçilebilir.

Algoritmanın genel olarak akışı Şekil 5.9‘ de gösterilmiştir.



Şekil 5.9: Genel olarak yeni algoritmanın çalışması

Bir alanın seçici olarak ayırt ediciliğini kuralların alt kümelere dağılımı belirler. Örneğin, Tablo 5-4 incelenirse F_3 seçici alanının en az ayırt edici alan olduğu hemen fark edilebilir. Şekil 5.8' de gösterilen F_3 için oluşturulan trie ağacının, 4 düğümü R_1, R_2, R_3, R_6 kurallarını içerirken, 1 düğüm de R_1, R_2, R_3, R_4, R_5, R_6 kurallarını içerir. F_2 alanı için oluşturulan trie ağacı ise Toplam 4 düğüm içerir ve düğümlerin oluşturduğu alt kümelerde ortak eleman yoktur. Bu durumda alanlar ayırt ediciliklerine göre sıralanırsa.F_2,F_1,F_3 olur.

Sınıflandırma algoritması ne olursa olsun bellek ve hız performansı algoritmaya giren kural sayısına bağlıdır. Bu yüzden kuralların alt kümelere düzgün dağılımı önemlidir.

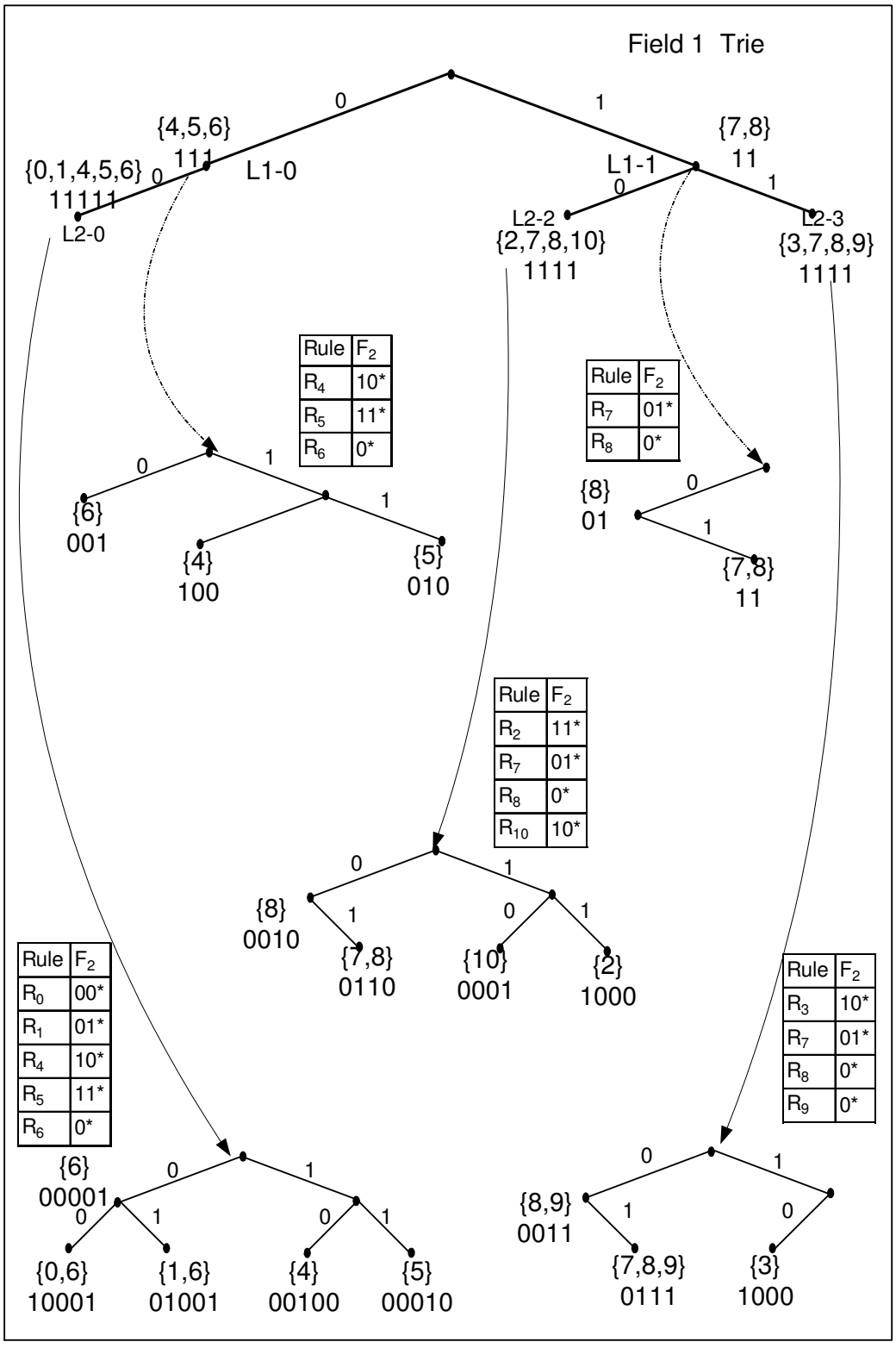
- Sınıflandırma sonucu oluşan alt kümelerdeki toplam eleman sayısı azlığı
- Alt kümelerdeki ortak eleman sayısının az olması
- Her bir alt kümedeki eleman sayısının az olması seçici alanın ayırt ediciliğini arttırır.

Tablo 5-5 'de 2 boyutlu 11 kurallı örnek bir veri tabanı verilmiştir. Bu veri tabanı için sınıflandırıcıların oluşturulması Şekil 5.10' da gösterilmiş. İlk olarak F_1 alanı için oluşturulan trie ağacı S kümesini 5 alt kümeye bölmüştür.

$S_1=\{R_4,R_5,R_6\}$, $S_2=\{R_7,R_8\}$, $S_3=\{R_0,R_1,R_4,R_5,R_6\}$, $S_4=\{R_2,R_7,R_8,R_{10}\}$, $S_5=\{R_3,R_7,R_8,R_9\}$. Alt küme sayısı 5, alt kümelerdeki toplam eleman sayısı $\sum_N S_i=18$. F_2 trie ağaçları her bir küme için ayrı ayrı oluşturulmuştur. Burada sınıflandırıcı olarak her alanda trie ağacı seçilmiştir ancak pratikte her alan ve küme büyüklüğü için farklı algoritmalar kullanılabilir. En kötü durumda 4 adımda aranan sonuç bulunurken, kullanılan bellek miktarı yani ağaçlardaki toplam düğüm sayısı da 32'dir. Aynı veri tabanı için BV algoritması kullanılsa idi toplam ağaçlarda 13 düğüm ve her düğümde 11-bitlik bit vektörü olacaktı. BV algoritmasında arama işlemi ise en az 5 en fazla 6 bellek erişimi gerektirecekti.

Tablo 5-5: 2 boyutlu örnek veri tabanı

Rule	F_1	F_2
R_0	00*	00*
R_1	00*	01*
R_2	10*	11*
R_3	11*	10*
R_4	0*	10*
R_5	0*	11*
R_6	0*	0*
R_7	1*	01*
R_8	1*	0*
R_9	11*	0*
R_{10}	10*	10*

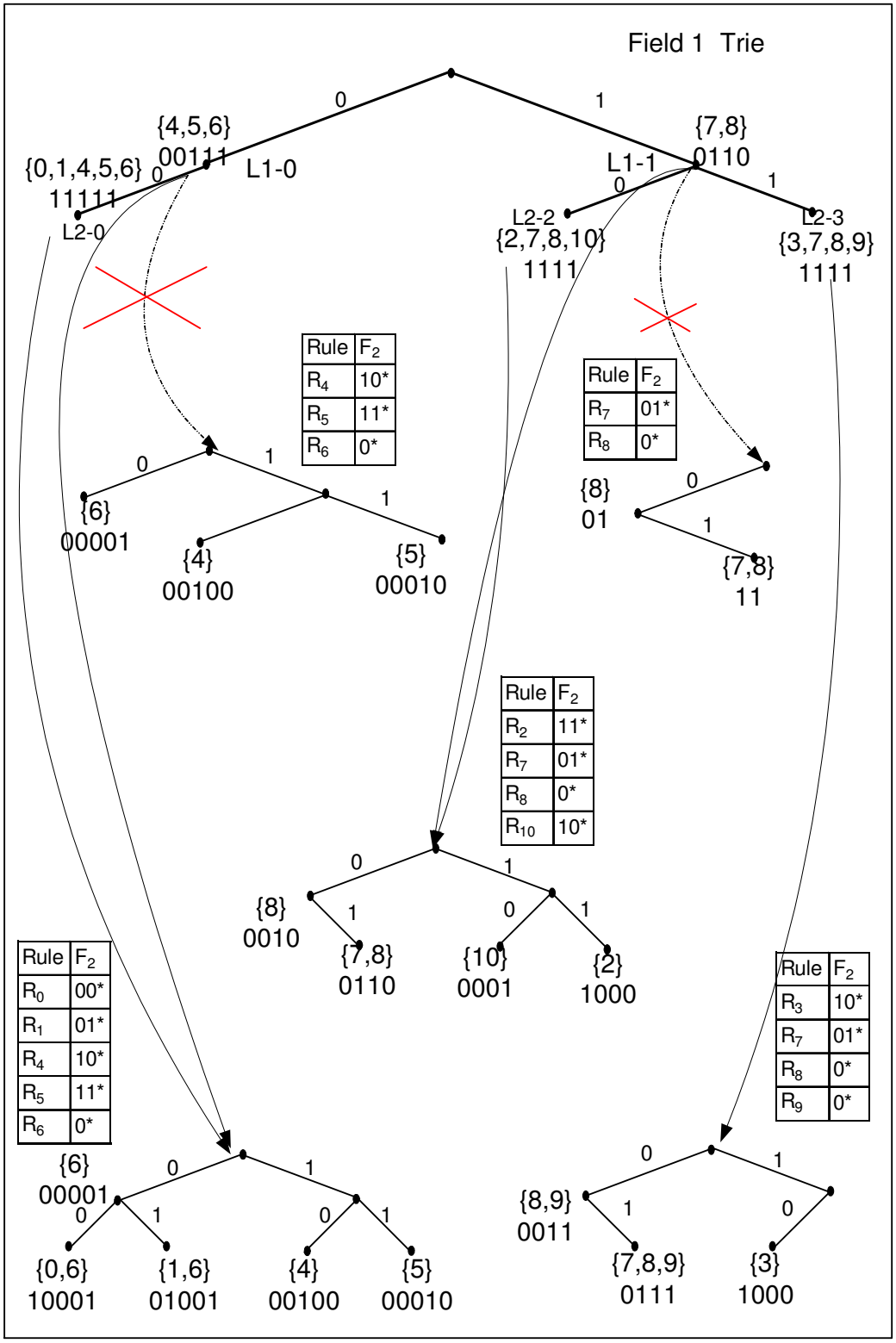


Şekil 5.10: Tablo 5-5 için yeni yöntemin çalışması -1

Yeni yöntem tek boyutta sınıflandırma algoritmalarına, alt kümeler içinde pakete uyuşmayan kurallar olmasına izin vererek bir esneklik tanıyordu. Böylece farklı kümeler üzerinde birleştirme yapıp ortak ağaçlar kullanılarak bellek kullanımı azaltılabilir. Her kümeye belirli miktarda pakete uymayan fazlalık kural eklenecektir. Eklenen bu fazlalık kuralların sonraki adımlarda elenme olasılığı oldukça yüksektir. Bu özellikten yararlanılarak Şekil 5.10'daki trie ağacının bazı düğümleri için oluşturulan alt kümeler birleştirilebilir. Çünkü trie ağacının bir düğümündeki kural kümesi, tüm çocuk düğümlerin kural kümesinin alt kümesidir. Şekil 5.11' L1-0 düğümü ile L2-0 düğümü, L1-1 ile L2-2 düğümü birleştirilmiştir ve bu kümeler için bir sonraki adımda aynı trie ağaçları kullanılmıştır. Böylece toplam bellek miktarı 2 trie ağacı azaltılması ile 8 düğüm azaltılmıştır. Örneğin gelen paket (011,101) ise 3 adımda 4 numaralı kuralın paketle eşleştiği bulunur. (011,000) ise geriye {0,6} kuralları kalır ve bundan sonra lineer arama yapılarak 2 adımda paketin 6. kuralla eşleştiği bulunur. Kümeleri etkin şekilde birleştirme algoritmanın bellek gereksinimini oldukça azaltabilir. Burada dikkat edilmesi gereken kümeleri birleştirirken pakete uymayan kuralların oranını fazla arttırmamaktır. Bu oranın çok büyük seçilmesi bellek erişimini arttırır. Kümeler birleştirilirken farklı yöntemler izlenebilir. Örneğin L2-2 düğümü ile L2-3 düğümü birleştirilebilirdi.

Sınıflandırma algoritması gereksiz alt kümelerin oluşmasına da neden olabilir. Örneğin gelen bir paketin L1-1 düğümünde kalması olanaksızdır. Çünkü L1-1 düğümünden hem sola hem sağa dallanma vardır. Bu durumda L1-1 düğümü için oluşturulan alt küme , alt kümeler listesinden silinebilir.

Son olarak büyük kümeler için lineer arama yerine birleştirilen ağaçlarda bit vektör algoritması da kullanılabilir. Böylece son adımda lineer arama işlem süresi de azaltılabilir. Şekil 5.11'de bit vektör değerleri kümelerin altında gösterilmiştir. Örneğin gelen paket (010,011) ise L1-0 düğümünde bit vektörü 00111, ve bir sonraki trie ağacında arama donucu bit vektörü 01001 olarak bulunur. Sonuç bit vektörü 00001 olur ve 6 numaralı kuralın paketle eşleştiği bulunur.



Şekil 5.11: Tablo 5-5 için yeni yöntemin çalışması -2

IPSec yerel ağları güvenli bir şekilde birbirine bağlamak için kullanılır, bu nedenle kural veri tabanları incelendiğinde şu özellikler dikkat çekmektedir.

- giren politika veri tabanı için kaynak IP adresi seçici alanında kurallar arası örtüşme oranı diğer alanlara kıyasla oldukça düşüktür.
- çıkan politika veri tabanı için varış IP adresi seçici alanında kurallar arası örtüşme oranı diğer alanlara kıyasla oldukça düşüktür.
- protokol ve port alanlarında kurallar arası örtüşme oranı ise oldukça yüksektir.
- protokol seçicisi TCP, UDP ya da * olmaktadır.
- port seçicileri değerleri genellikle 1024'ten küçük bilinen port değerlerini içermektedir.
- Kurallar arasında tüm alanlarda örtüşme oranı oldukça düşüktür.
- Varış ve kaynak IP adresi seçimi için adres / maske uzunluğu gösterimi kullanılmaktadır.

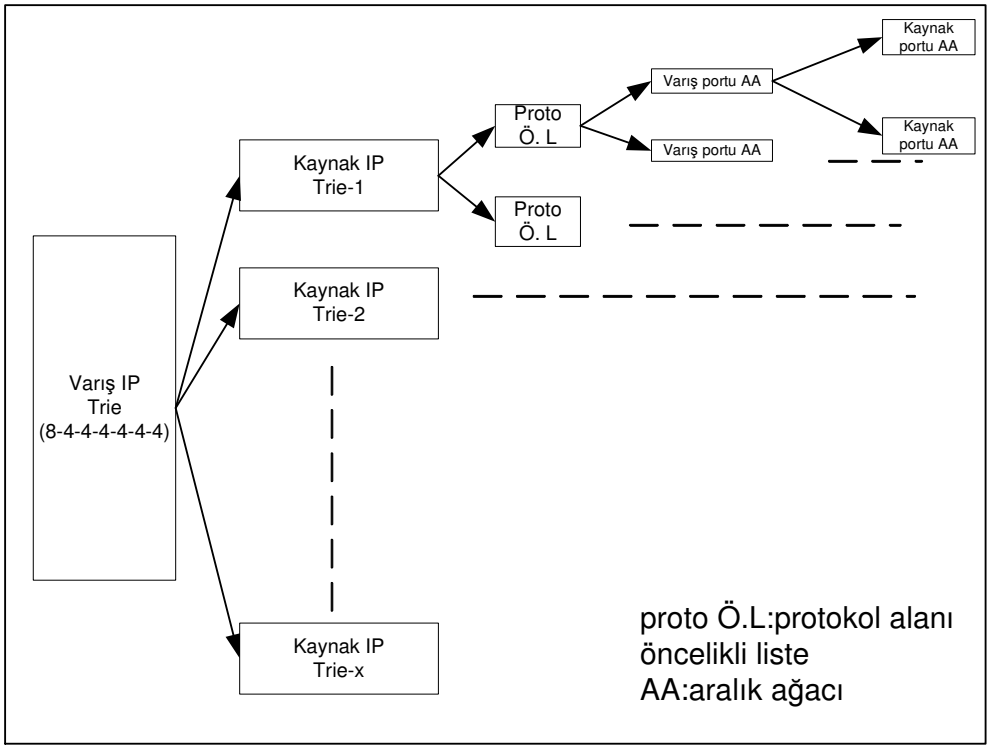
Tüm bunlardan faydalanılarak IPSec kural veri tabanları için yeni yöntem ile etkin bir arama yapısı geliştirilebilir.

Çıkan güvenlik politika veri tabanlarında, seçici alanların öncelik sırası varış IP adresi- kaynak IP adresi-protokol-varış portu – kaynak portu olarak sıralanabilir.

Giren güvenlik politika veri tabanlarında, seçici alanların öncelik sırası ise kaynak IP adresi- varış IP adresi – protokol - varış portu – kaynak portu olarak sıralanabilir.

Varış ve kaynak IP adresi en seçici alanlar olduğu için bu kısımlarda daha hızlı algoritmaların kullanılması genel performansı arttıracaktır. Bunun için en uygun algoritma çok bitli trie ağacı olarak belirlenmiştir. Internet için 8 bitten daha küçük IP maskesi olmadığından 8-4-4-4-4-4-4-4 şeklinde bir çok bitli trie ağacı kullanılarak en kötü durumda 7 adımda ilk arama ağacı tamamlanmış olur. protokol seçicisi TCP, UDP ya da * olduğuna göre bu alan için en ideal çözüm öncelikli bir listedir. İlk düğüm TCP, ikinci düğüm UDP, ve geri kalan protokol değerleri sırayla eklenerek, listenin sonuna * seçicisi için bir düğüm koyulabilir. Port seçici alanı 16 bit uzunluktadır ve genellikle bilinen port değerlerini yani 1024'den küçük değerleri içine almaktadır 3 alanda eleme yaptıktan sonra geriye az sayıda port seçicisi

kalacağı düşünülürse burada arama karmaşıklığı (log_2N) ikili dengeli aralık ağacı kullanmak hem bellek tüketimi hem de arama hızı açısından ideal bir çözüm olacaktır. Yeni yöntemin IPSec güvenlik politika veri tabanları için uyarlanması Şekil 5.12 ‘ de gösterilmiştir.



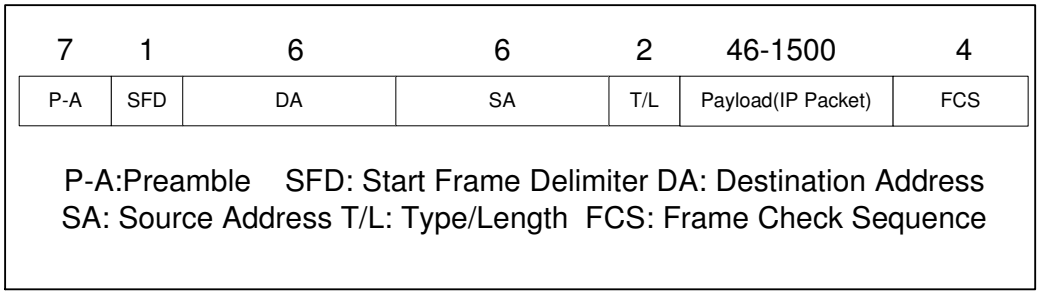
Şekil 5.12: IPSec güvenlik veri tabanları için yeni yöntemin uyarlanması

5.5 Gigabit Ethernet Ağı İçin Gerekli Paket İşleme Hızının Hesaplanması

Geçirim, Bir ağ cihazının birim zamanda işleyebileceği veri miktarını ifade etmektedir. Geçirim genellikle Mbps(Mb/sn) cinsinden ifade edilir, bazı üreticiler geçirimi pbs (paket /sn) cinsinden ifade etmeyi de tercih etmektedirler. Bazı cihazlar için de hat hızında paket işleme terimi kullanılmaktadır. Peki terimler performans açısından gerçekte ne ifade etmektedir? Bir ağın teorik veri taşıma limitleri nelerdir? Gerçekte 100Mbps’lik bir ağda , saniyede gerçekten 100Mbit veri iletmek ne derece mümkündür? IPSec cihazlarının performansını nasıl ölçebiliriz ve birbirleriyle karşılaştırabiliriz? Bu bölümde tüm bu sorulara açıklık getirmeyi amaçlamaktayız.

Tüm bunları anlamak için, ilk olarak verinin ağ ortamında nasıl taşındığını bilmeliyiz. IPSec VPN cihazları genellikle Ethernet ağ arayüzlü cihazlardır, IP Paketleri, boyu 64 ile 1518 sekizli arasında değişen Ethernet çerçeveleri içinde

taşınırlar.[1] Her çerçeve 46 ile 1500 sekizli arası taşınan gerçek veriyi(payload) içerir. Ethernet çerçeve yapısı Şekil 5.13 'de gösterilmiştir.



Şekil 5.13: Ethernet çerçeve yapısı[51]

Ethernet çerçevesi şu alanlardan oluşmaktadır:

Öntakı-preamble (7 sekizli): Ethernet çerçevesini ilk 7 sekizlisi öntakı olarak senkronizasyon amacıyla kullanılır. Bu alan sabit 10101010…..11 bit dizisi olup, gönderici ve alıcının saatlerinin senkronize olmasını sağlar.

Çerçeve Başı Ayıracı- Start Frame Delimeter (1 sekizli): ön takıyı 1 sekizli uzunluğunda sabit çerçeve başı ayıracı vardır.

Varış Adresi – Destination Address (6 sekizli):

Alıcı ağ arayüz kartının MAC adresini içerir.

Kaynak Adresi – Source Address (6 sekizli):

Gönderici ağ arayüz kartının MAC adresini içerir

Tür- Type (2 sekizli):

Aktarılan çerçevenin hangi 3. katman protokolüne ait olduğunu belirlemek için kullanılır.

Veri- Data (46-1500 sekizli):

Aktarılan üst katman verisini içerir, uzunluğu 46 ile 1500 sekizli arasında değişebilir. Eğer 46 sekizliden daha az olursa hatta oluşacak çakışmalar sezilemeyeceği için daha az veri olması durumunda dolgu kullanılır.

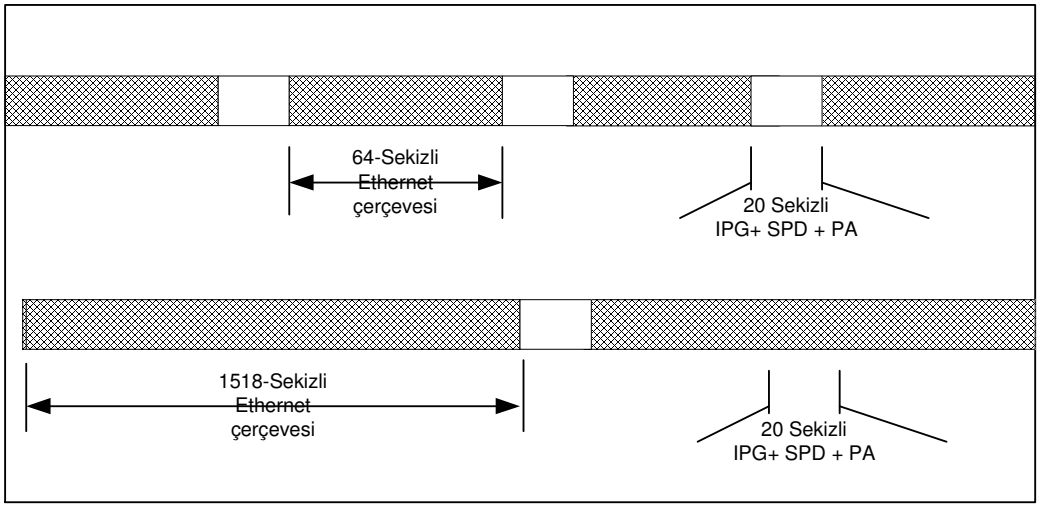
Çerçeve Hata Sınaması – Frame Check Sequence (4 sekizli):

[1] 4000 ve 9000 sekizli gibi 1518'den büyük boyutda jumbo Ethernet çerçeveleri de vardır..

Bu alana çerçevede aktarım sırasında oluşabilecek hataların sezilebilmesi amacıyla 32 bitlik CRC değeri konulur.

Ethernet Standardında yola erişim standardı olarak CSMA/CD kullanılmaktadır. Bir Ethernet ağ arayüz kartı hatta veri çıkartmadan önce hattı dinler, eğer hat başka bir düğüm tarafından kullanılıyorsa, hatta bir taşıyıcı olduğunu sezer ve hatta verisini o anda çıkarmaz ve bir süre bekler aksi halde hatta çakışma olur. Gönderen tarafta veriyi hatta çıkardıktan sonra belirli bir süre hattı dinler ve çakışma olmadığından emin olur. Bu nedenle Ethernet arayüz kartları , paketleri artarda gönderemezler, IPG (Inter-packet Gap – Paketler arası boşluk) kadar süre beklemek zorundadırlar. IPG boyu Ethernet standardında 12 sekizli olarak belirlenmiştir.

Ethernet çerçevesine atanmış 20 sekizlilik(12 IPG + 7 önek + 1 çerçeve başı) bir başlık alanı maliyeti söz konusudur. Şekil 5.14' de Ethernet çerçevelerini hatta çıkarılması gösterilmiştir. Bu maliyet nedeniyle 100Mbps'lık bir ağda, 100Mbps veri geçirimi mümkün değildir.



Şekil 5.14: Ethernet çerçevelerinin artarda hatta çıkarılması

Herhangi bir çerçeve boyu için , Teorik veri geçirimini şu şekilde hesaplayabiliriz:

Paket işleme hızı (pbs) = ağ hızı ÷ ((çerçeve boyu + 8 + 12) × 8)

Veri Geçirimi (Mbps) = Paket işleme hızı × çerçeve boyu × 8

1Gbps'lık bir ağda ,1518 sekizli boyutundaki çerçeveler için teorik limitler

Paket işleme hızı (pbs) = 10^9 ÷ ((1518 + 8 + 12) × 8) = 81,274 pbs

Veri Geçirimi (Mbps) = 81,274 × 1518× 8 = 986,99 Mbps

Eğer bir ağ cihazı bu performansı sağlıyorsa Ethernet band genişliğinin %100'nü kullandığı için, 1Gbps (hat hızında) veri geçirimi olduğu kabul edilebilir. Eğer bu hızda aynı anda hem veri alıyor ve gönderebiliyorsa , Cihazın performansı 1Gbps çift yönlü(full dublex) ya da 2 Gbps toplam(Aggregate) geçirimi var şeklinde ifade edilir.

Ancak başlık maliyetlerinin küçük çerçevelerde daha fazla etki yapacağı açıktır. Küçük paketler aynı zamanda cihazın saniyede, çok daha fazla paket işlemek zorunda kalmasına neden olmaktadır. 64 sekizlik çerçeveler için aynı hesaplamaları yaparsak:

Paket işleme hızı (pbs) = $10^9 \div ((64 + 8 + 12) \times 8) = 1.480.895$ pbs

Veri Geçirimi (Mbps) = $1.480.895 \times 64 \times 8 = 761{,}904$ Mbps

Tablo 5-6 ' de 1Gbps Ethernet ağı için, farklı çerçeve boylarında maksimum teorik geçirimi değerleri verilmiştir. Eğer bir ağ cihazı saniyede 74,000 paket iletebiliyorsa, bu cihaz 64-sekizli teorik maksimum geçirimin sadece %50si kadar performansa sahip demektir.

Tablo 5-6: 1 Gbps ağ için maksimum teorik geçirim değerleri

Paket Boyu (Sekizli)	**Veri Geçirimi (Mb/s)**	**Paketişleme hızı(pbs)**
64	761.90	1488095
128	864.86	844594
256	927.54	452898
512	962.40	234962
1024	980.84	119731
1280	984.61	96153
1518	986.99	81274

Ayrıca, her IP paketi , 20 sekizli IP başlığı, 8 sekizli UDP başlığı ya da 20 sekizli TCP başlığı ve belki daha üst katmanların protokol başlıklarını da içerebilir, Bu nedenle küçük Ethernet çerçeveleri band genişliğinin oldukça verimsiz kullanılmasına neden olmaktadır [52].

IPSec için teorik performans limitleri: IPSec performans limitleri sıradan bir yönlendirici ve anahtara göre farklıdır. Çünkü ilk olarak, IPSec işlemi paket boyunu büyütmektedir. Örneğin tünel kipinde çalışan bir IPSec cihazı, pakete dış IP başlığı, ESP başlığı, ESP kuyruğu eklemektedir ve bu alanların bazıları kullanılan algoritmalara göre farklılık göstermektedir. Şekil 5.15 'de şifreleme algoritması olarak 128-bit AES şifreleme algoritması CBC kipinde ve bütünlük sınama

algoritması olarak AES-XCBC-MAC-96 algoritması kullanılması durumunda IPSec tünel paketi formatı gösterilmiştir. Paketin başına 20 sekizli tünel paketinin dış IP başlığı, 8 sekizli ESP başlığı, CBC kip için gerekli, algoritma blok boyu (16 Sekizli) kadar ilklendirme vektörü, uzunluğu 2-17 sekizli arasında değişen doldurma ve sonraki başlık bilgileri, ve AES-XCBC-MAC-96 kullanılması durumunda 12 sekizli bütünlük sınama değeri pakete eklenir. Bu durumda paketin boyu 58-73 sekizli kadar artar. Bu durumda IPSec cihazının parçalamadan iletebileceği en büyük çerçeve boyu 1518 sekizliden, 1460 sekizliye düşer. Bu durumda 1460 sekizlilik çerçeveler için maksimum veri geçirimi:

Paket işleme hızı (pbs) = $10^9 \div ((1460 + 58 + 8 + 12) \times 8) = 81.274$ pbs

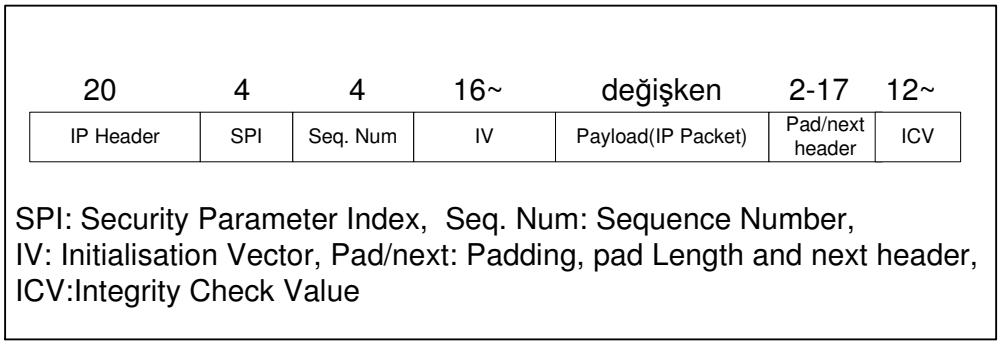
Veri Geçirimi (Mbps) = $81.274 \times 1460 \times 8 = 949{,}280$ Mbps

64 sekizlilik çerçevelerin için maksimum veri geçirimi:

Paket işleme hızı (pbs) = $10^9 \div ((64 + 58 + 8 + 12) \times 8) = 880.281$ pbs

Veri Geçirimi (Mbps) = $880.281 \times 64 \times 8 = 450{,}704$ Mbps

Görüldüğü gibi 64 sekizli çerçeveler için , 1 sn de en fazla 880.281 çerçeve geçirilebileceği için, Gigabit hızlı bir IPSec cihazının saniyede 880.281 paketi işleyebilirse teorik limitin %100' ünü kullanabiliyor demektir.

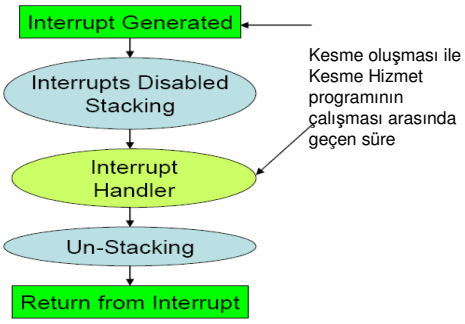


Şekil 5.15: 128-bit AES-CBC ve AES-XCBC-MAC-96 için tünel paket formatı

Bir IPSec VPN cihazının performansı sadece paket geçirimi olarak değerlendirilmelidir, paket geçirimiyle beraber gecikme de cihazın performansı açısından önemlidir. Özellikle gerçek zamanlı uygulamalar için (VoIP, gerçek zamanlı ses ve görüntü aktarımı) gecikme performansı da son derece önemlidir. Bu nedenle IPSec VPN cihazı tüm bu IPSec işlemlerini bu ihtiyaçlara cevap verebilecek sürede tamamlayabilmelidir.

6 BAŞARIM İYİLEŞTİRMELERİ

6.1 Kesmeli ve Yoklamalı Çalışma

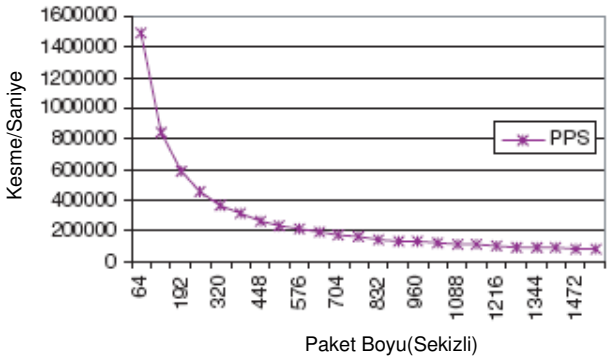


Şekil 6.1: Üst seviyede kesme dallanmasının akışı

Ethernet kartından paket alındığında Ethernet kartı paketi kendi belleğinde bir süre tutar ve DMA ile işlemci belleğine kopyalar. Paket işlemci tarafından işlenmeye hazır olduğunda Ethernet kartı kesme üreterek bunu işlemciye bildirir. Genel olarak Ethernet kartı her bir paket için bir kesme üretir.

Kesme gecikmesi, kesmeyi tetikleyen olayın meydana gelmesi ile kesmenin işlenmesi arasında geçen süre olarak tanımlanır. Kesmenin önceliği paket işleme görevlerinden daha fazladır. Bu nedenle yoğun yük altında sistem canlı-kilitlenmeye girebilir ve paket işlemesi ciddi şekilde düşer.

Bir IP ağ cihazında işlerin büyük çoğunluğu paket boyundan bağımsız olarak paket başına yapılır. Örneğin her paket için ana iş olarak bağlam değiştirme, bellek alanı ayrımı, kart saklayıcılarına PCI erişimi, protokol alanlarını inceleme ve arama işlemleri vs. yapılır. Paket işleme hızı arttıkça bu görevler için gerekli kaynak ihtiyacı ciddi oranda artar. Gigabit Ethernet ağı için Şekil 6.2'de farklı paket boyları için bir saniyede gelebilecek kesme sayıları gösterilmiştir. Eğer her paket için bir kesme üretilirse saniyede 1,5 milyon kesme gelebilir. Böyle bir durumda işlemci sürekli kesme taşkınına cevap vermeye çalışacak ve başka bir iş yapmaya fırsat bulamayacaktır. Bu durum kesmeyle-canlı kilitlenme olarak adlandırılır.



Şekil 6.2: Paket boyuna göre kesme oranı

Kesmelerin sistem üzerindeki maliyetini azaltmak amacıyla iki tür çözüm uygulanabilir.

Kesme birleştirme (Interrupt Coalescing): Bu çalışma biçiminde sistem belirli sayıda tetikleyici olay olduğunda kesme üretir. Böylece kesmenin sistem üzerindeki maliyeti azaltılmış olur. Ancak bu çalışma kesme gecikmesini arttıracağından kesme üretme koşulu olarak bir zaman aşımı da eklenir, zaman aşımı süresince n tane olay meydana gelmese bile kesme üretilir. Örneğin modern Ethernet kartları belirli sayıda paket için sadece bir kesme üretebilmektedir.

Yoklamalı Çalışma (Polling): Bu çalışma biçiminde aygıt sürücüsü, dış donanım aygıtındaki kesme maskeleme saklayıcısını aktif hale getirir ve aygıtın kesme üretmesi engeller. Aygıt sürücü belirli aralıklarla aygıtın kesme saklayıcısını kesme gelmiş mi diye yoklar ve eğer kesme gelmiş ise yapılması gerekli işlemleri başlatır. Bu çalışmada aygıt hiç bir olay için kesme üretmez. Böylece sistemin başarımını ciddi oranda artarken, gecikme de artmamış olur. Yoklamalı çalışma, çok yoğun kesme üreten ve yüksek başarım ihtiyacı duyan determinist sistemler için uygun bir çalışma biçimidir[53].

Bu çalışmada Ethernet kartı hem kesme birleştirmeli hem de yoklamalı çalıştırılarak sistem başarımı üzerindeki olumlu etkileri gözlenmiştir.

6.2 Çekirdek Uzayı ve Kullanıcı Uzayı

Çekirdek uzayı ile kullanıcı uzayı arası bağlam değiştirmeleri de başarım açısından oldukça maliyetlidir. Bağlam değiştirmeleri kesmelerden daha maliyetlidir[54]. Cep

bellek ıskalamaları ilk komutlarda oldukça artar, iş hattında işlenen komutlar boşaltılır, saklayıcıların içerikleri belleğe saklanır ve eski değerleri geri yüklenir. Bellek koruması nedeniyle çekirdek uzayından kullanıcı uzayına veri kopyalamasına ihtiyaç duymak ayrıca ek maliyet getirir. Bunu önlemek için tasarlanan istem Linux işletim sistemi çekirdeğinde bir çekirdek görevciği olarak gerçeklenmiştir.

6.3 Bellek Ayırımı İyileştirmeleri

Paket havuzları: Gerçek zamanlı sistemler için dinamik bellek ayırımı hız açısından yavaş ve kararsız olabilir. Böyle uygulamalar için sistemin ihtiyaç duyabileceği maksimum bellek miktarını belirlenir, sistem açıldığında gerekli bellek alanları statik olarak alınır ve bir havuzda tutulur. Bellek alımları bu havuz üzerinden yapılır.Böylece bellek yönetimi daha basit ve hızlı bir şekilde gerçekleştirilir. Tasarlanılan sistemde havuz kalıbı (Pool Pattern)[32]kullanılarak paketler için gerekli bellekler önceden statik olarak alınmış ve bellek yönetimi hızlandırılmıştır.

Paket tampon belleği ayarları:Paketler protokol yığınları tarafından işlendikçe paketlerin başına ve sonuna belli miktarlarda alanların eklenmesi-çıkarılması gerekebilir. Paket ilk ağ arayüzünden alındığında ayrılan bellek alanının başında ve sonunda yeterli miktarda alan olmaması, yeni bellek alanı alınmasına ve ek kopyalama işlemi gerektirecektir [55]. Örneğin IPSec tünel kipinde paketin başına dış IP başlığı, IPSec başlığı, şifrelemeye göre ilklendirme vektörü, paketin sonuna şifreleme için doldurma, IPSec kuyruk alanları, asıllama değeri gibi alanlar eklenir. Tasarlanan sistemde bu alanlar önceden hesaplanmakta ve bellek ayarı buna göre yapılmıştır.

6.4 Hızlı Yola Odaklanma

Gerçek zamanlı bir sistemde, sistemin tamamının gerçek zamanlı çalışmasına gerek yoktur. Genellikle sistemin ana görevi için birkaç sürecin hızlı bir şekilde kotarılmasına ihtiyaç vardır. Örneğin IPSec güvenlik geçidi için, paket alma ve paketleri işleme görevleri hızlı bir şekilde yerin getirilmesi gerekirken, cihazın yönetimiyle ilgili görevlerin bu hızda çalışmasına gerek yoktur. Sistemin kritik görevleri içinde de istatistik olarak daha fazla karşılaşılacak durumlar söz konusu olabilir. Örneğin aranan bir ARP kaydının tabloda olmama ihtimali tabloda olma

ihtimalinin yanında oldukça düşüktür. Böyle durumların söz konusu olduğu uygulamalarda sistemin kritik görevleri ve bu kritik görevler içindeki hızlı yollar belirlenmelidir [55]. Hızlı yollara odaklanılması, bu kısımların iyileştirilerek sistemin genel başarımının arttırılmasını ve sistemin genel başarımını arttırmayacak kısımlarda gereksiz zaman kaybedilmemesini sağlar.

6.5 Sistem Çağrılarını Azaltma

Sistem çağrılarının işletilmesi (işletim sistemine ve MİB' ne bağlı olarak) 100 nano saniyeden birkaç milisaniyeye kadar gecikmeye neden olabilir[1]. Sistem çağrıları işletim sistemi içinde yazılım kesmesi olarak çalışır. Her sistem çağrısı sistemde kesmeye yol açacağından sistemin başarımını düşürebilir. Sistem çağrılarıyla yapılan işler mümkünse iç kütüphanelerde gerçeklenerek bu ek maliyet azaltılabilir[54].

6.6 Diğer İyileştirmeler

Sistem üzerinde paketleri önceden parçalama, tablo erişimlerini iyileştirme gibi iyileştirmeler de yapılabilir.

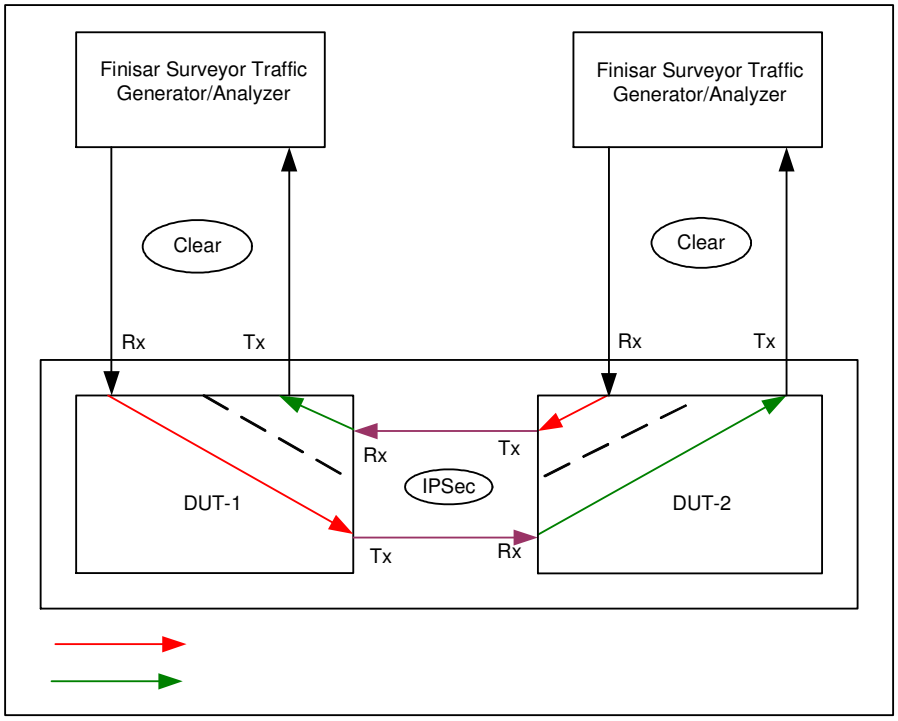
IPSec IP paketlerinin boyutunu büyüttüğü için bazı paketlerin ağa çıkarılmadan parçalanması gerekebilir. Karşı uçta bu paketlerin çözülmesi için önce tüm parçaların birleştirilmesi gerekir. Oysa gönderici tarafta IP paketi, IPSec uygulanmadan önce parçalanır ve karşı uçta paketlerin çözülmesi için birleştirme yapılmasına gerek kalmaz. Böylece birleştirme maliyetini ortadan kaldırılır.

Tablo erişimlerinde ek iyileştirmeler yapmak mümkündür. Ortak erişilen tablolarda senkronizasyonu sağlamak için, tüm tabloyu kilitlemek yerine, tablonun sadece gerekli satırlarını kilitlemek paralel çalışmayı arttıracağından başarımı da arttıracaktır[55]. Eğer paket başına zamanlayıcı kurulması gerekiyorsa, zamanlayıcıyı kurma sıklığı zaman aşımı sıklığından çok daha yüksek olabilir. Zamanlayıcıyı kurma işletim sistemi ve donanımla etkileşimi gerektireceğinden sistem başarımını arttırmak için zamanlayıcı sürekli kurulmadan, değişkenlerde zaman aşımı zamanı tutulursa ve zaman aşımı oluşunca, zaman aşımının geçerliliği değişkenler yardımıyla kontrol edilerek zamanlayıcı kurma maliyeti azaltılabilir.

[1] Monta Vista , LMBench* test sınuçlarına göre 300Mhz Celeron ve Linux: Kesme gecikmesi : 98.998% 2 µs (en kötü durumda 38 µs); Sistem çağruları: 1 µs; bağlam değiştirme: 3 µs (0.55 µs Intel® Pentium III®, QNX)

7 TEST VE BAŞARIM ÖLÇÜMLERİ

7.1 Test Ortamı



Şekil 7.1: Test ortamı

Başarım ölçümleri için Şekil 7.1'deki test ortamı kurulmuştur. Test ortamının ayrıntılı açıklaması aşağıda verilmiştir.

Test Cihazı: 2 adet Finisar Surveyor 7.0 donanım tabanlı ağ trafik analiz ve trafik üreteç cihazı.

Cisco Catalyst 2970 anahtar – paket alma (RX) ve paket gönderme(TX) hatlarını test cihazının farklı portlarını ayırmak için kullanılmıştır.

DUT: (Device Under Test), Test altındaki hedef platformu.

Donanım Özellikleri:

İşlemci: Pentium IV - 3 GHz.

Cep bellek: 16KB L1, 1024KB L2.

Bellek: 512MB DDR-RAM.

Ethernet kartı: 2 adet IntelPro/E1000 Gigabit Ethernet Kartı.

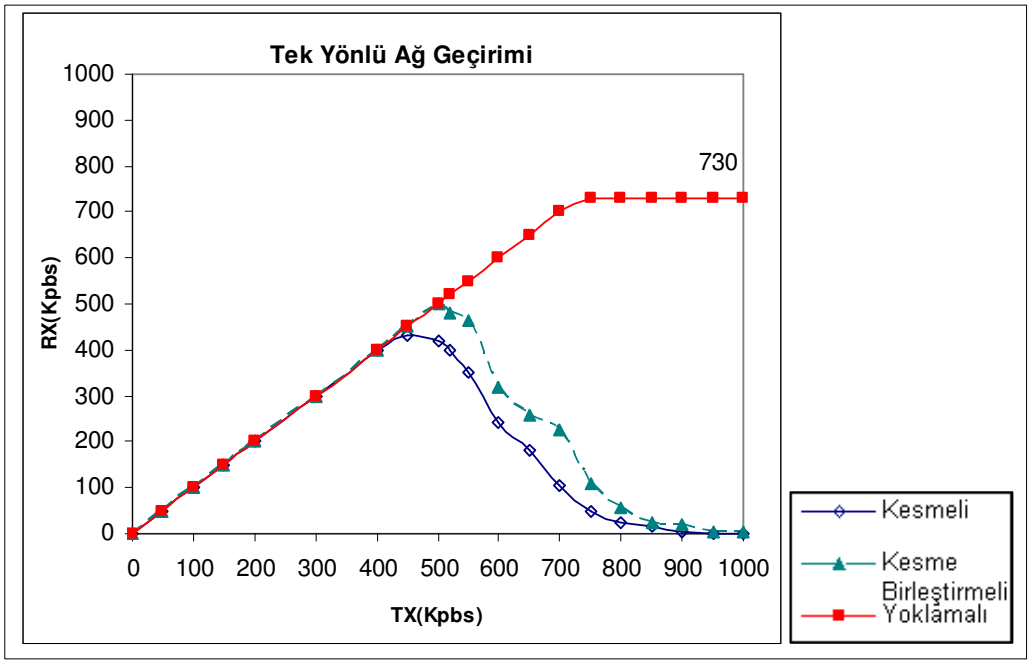
Politika Üretme Benzetim Programı: Testler sırasında güvenlik politika veritabanları üretmek için benzetim programlar yazılmıştır. Politika üretme algoritması olarak Network Processor Forum tarafından önerilen algoritma kullanılmıştır [56].

Paket Üretme Benzetim Programı: Başarım ölçümü için Güvenlik politika veri tabanlarına uyacak şekilde rasgele paketleri üreten ve Finisar test cihazı için uygun trafik dosyalarını hazırlayan benzetim programı yazılmıştır. Testler sırasında sistemi zorlaması açısından, paket boyu olarak, en küçük paket boyu 64 Sekizli olarak seçilmiştir.

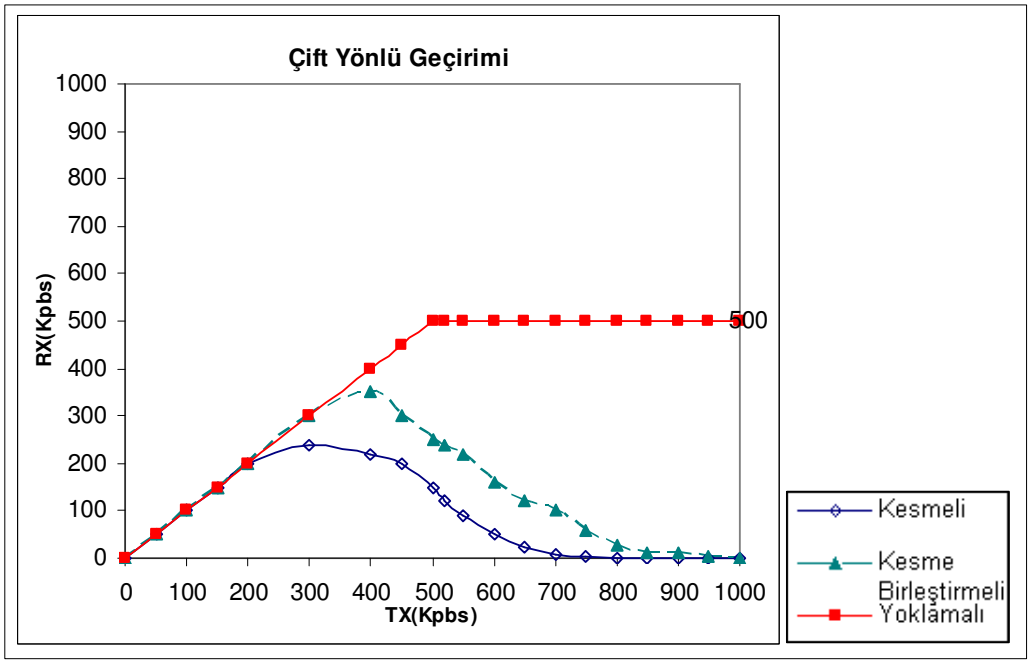
7.2 Başarım Ölçümleri

7.2.1 Yoklamalı Çalışma Başarımı

Bu aşamada Ethernet kartları kesmeli, kesme birleştirmeli ve yoklamalı çalıştırılarak paket iletim başarımları tek yönde ve çift yönde ölçülmüştür. Kesme birleştirmede, paketler için üretilen ardışık iki kesme arasında paket alırken 131 mikrosaniye, alırken 66 mikro saniye gecikme uygulanmıştır. Şekil 7.2 ve Şekil 7.3' de görüldüğü gibi yoklamalı çalışma sistemin başarımını ciddi oranda arttırmaktadır. Yoklamalı çalışma çift yönlü trafikte arayüz başına 500.000 pbs (toplamda 1.000.000 pbs), tek yönlü trafikte 730.000 pbs gibi yüksek bir geçirim başarımı göstermektedir. Kesmeli çalışmada aşırı yüklenme durumunda cihaz canlı kilitlenmeye girmekte ve sistemin başarımı ciddi oranda düşmektedir. Kesme birleştirmeli çalışma başarımı biraz yükseltmektedir. Kesme birleştirmeli çalışma canlı kitlitlenmeyi geciktirse de önleyememektedir. Yoklamalı çalışmada ise sistem en yüksek başarımda ve kararlı bir şekilde çalışmaktadır. Aşırı yüklenme durumlarında bile başarım maksimum değerde sabit kalabilmiştir. Yoklamalı çalışmanın, kesme birleştirmeye kıyasla diğer bir avantajı da gecikmeye ve seyirmeye neden olmamasıdır. Gecikme ve seyirme görüntü ve ses aktarımı gibi gerçek zamanlı trafik akışlarında oldukça önemlidir.



Şekil 7.2: Tek yönde kesme birleştirmeli ve yoklamalı paket geçirim başarımları



Şekil 7.3: Çift yönde kesme birleştirmeli ve yoklamalı paket geçirim başarımları

7.2.2 IPSec Güvenlik Politika Veritabanı Arama Başarımı

Bu aşamada IPSec Güvenlik Politika Veritabanının sistem başarımına etkileri gösterilmiştir. Değişik kural sayılarında algoritmaların başarımları tek ve çift yönlü

trafikte ayrı ayrı ölçülmüştür. Bu aşamadaki tüm testlerde yüksek başarım için Ethernet kartı yoklamalı çalıştırılmıştır.

Geliştirilen politika üretme programıyla değişik kural sayısında kural veri tabanları hazırlanmış ve üretilen politikalar test cihazına yüklenmiştir. Daha sonra bu kurallara uyacak şekilde paketler geliştirilen diğer bir programla oluşturulmakta ve paket üreticiden cihaza yollanarak başarım ölçülmüştür.

Lineer arama, kurallar bağlantılı bir listede tutulur ve lineer bir şekilde karşılaştırılır.

uuv1-yeni yöntemle ikili trie ağacı kullanarak ve eşik değeri kullanmadan sonraki tüm ağaçları gezmeye devam ederek arama yapma:

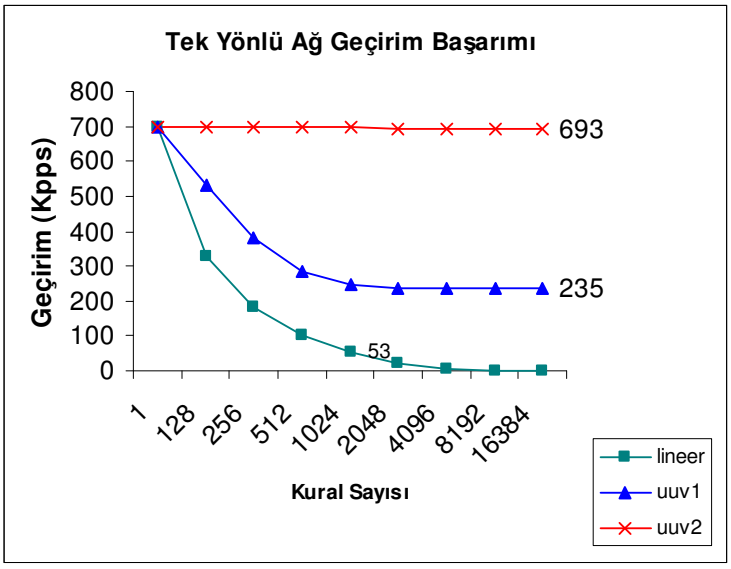
uuv2- ikili trie ağacı kullanarak eşik değerinden sonraki ağaçlarda arama yapmadan kalan kuralları lineer bir şekilde arama. (eşik değeri = 2)

uuv3- ikili trie ağacı yerine çok-bitli trie ağacı kullanarak ağaçları üzerinde arama yapma.

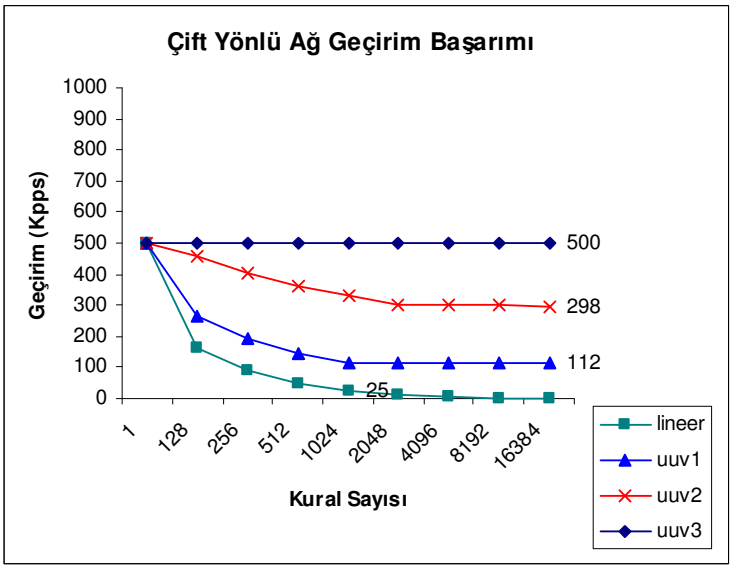
Şekil 7.4 ve Şekil 7.5' de görüldüğü gibi bellek erişiminin paket işleme başarımına etkisi büyüktür. Kural sayısı arttıkça lineer kural arama Gigabit hızında paket işleyebilmek için yetersizdir. Yeni yöntemde yapılan iyileştirmelerle bellek erişimi giderek azaltılmış ve sonunda 16.000 gibi büyük kural sayıları için bile cihazın paket geçirimi sınırı kadar paket geçirilmesi başarılmıştır.

Yeni yöntemde ikili Trie ağacının başarımı ilk olarak belirli bir eşik değerinde daha fazla ağaç gezmeden kural eşleşmesini lineer yaparak iyileştirilmiştir. Bu yöntemle başarımın oldukça arttığı ve tek yönde Ethernet kartının paket iletim sınırına ulaşıldığı gözlenmektedir. Ancak çift yönlü trafikte bu iyileştirme ile en fazla 298.000 pbs başarım ile yetersiz kalmıştır. Yeni yöntemle çok bitli trie ağaçları kullanılmış ve Ethernet kartının başarım sınırlarına çift yönde de erişilmiştir.

Şekillerde dikkati çeken diğer bir sonuç ta, küçük kural sayıları için yeni yöntemde kullanılan algoritmaların bellek erişimleri aynı olmasına rağmen başarımlarının daha yüksek olması ve kural sayısı arttıkça başarımın giderek düşerek belirli bir değerde sabit kalmasıdır. Bunun nedeni ise küçük veri tabanlarında erişilen bellek alanın cep bellekte olma ihtimalinin daha yüksek olmasıdır. Kural veri tabanının boyu büyüdükçe cep bellekte ıskalama oranının artması başarımın düşmesine neden olmaktadır.



Şekil 7.4: Tek yön IPSec güvenlik politika veritabanı arama algoritmaları başarımı



Şekil 7.5: Çift yön IPSec güvenlik politika veritabanı arama algoritmaları başarımı

8 SONUÇLAR VE İLERİKİ ÇALIŞMALAR

Tez çalışmaları kapsamında yeni IPSec versiyon 3 güvenlik mimarisi incelenmiş ve standartlara uygun nesneye dayalı IPSec Güvenlik Geçidi tasarlanmıştır. Tasarlanan sistem sıradan kişisel bilgisayarlar üzerinde, Linux çekirdeğinde çalışan bir yazılım olarak gerçeklenmiş ve başarımı ölçülmüştür. Tasarlanan sistemin yazılım tabanlı olması ucuzluk, kolay güncellenebilirlik, modülerlik gibi avantajları bir arada sunmuştur.

Yazılım tabanlı bir sistemin en büyük dezavantajı olan başarım düşüklüğü problemi tez çalışmaları kapsamında incelenerek, sistemdeki başarım dar boğazları belirlenmiş ve kimi iyileştirmeler yapılmıştır. Çalışmalar sonucu kesmeli çalışmanın ve güvenlik veri tabanlarında paket eşleşmesi arama işlemlerinin sistemin başarımında en önemli dar boğaz noktaları olduğu tespit edilmiştir. Yoklamalı çalışma ve güvenlik veri tabanları için geliştirilen algoritmaların sistem üzerinde olumlu etkileri ölçülmüş ve bu iyileştirmelerle, tasarlanan sistem başarımının bir çok firma için yeterli sayılabilecek düzeye sıradan kişisel bilgisayar donanımı üzerinde koşan yazılımlarla ulaşılabildiği gösterilmiştir.

Yakın zamanda daha hızlı işlemcilerin ve ağ geçirim başarımı daha yüksek Ethernet kartlarının piyasaya sürülmesi ile yazılım tabanlı sistemlerin toplam başarımı daha da arttırılabilir. Günümüzde daha yüksek cep bellekli ve çok çekirdekli işlemcilerin üretimi giderek artmaktadır. Ayrıca modüler tasarım ileride bazı modüllerin donanım destekli olmasına da izin vermektedir. Zaman açısından çok kritik işlemlerin özel donanımlarla gerçekleştirilmesi ile donanımın toplam test ve bakım maliyetleri azaltılabilir.

Tasarımın temelinde Click yazılım mimarisinin kullanması ve onun NS-2 ağ benzetim programı ile çalışabilmesi, teorik çalışmaların sonuçlarının kısa bir sürede elde edilebilmesini sağlamakla beraber, bu çalışmaların günümüz ağ ortamlarına çalışan ürünler veya sistemler olarak aktarılabilmesini de kolaylaştırmaktadır. NS-2

programıyla tasarlanan alt yapı kullanılarak IPSec üzerinde araştırma amaçlı yeni kriptolojik algoritmalar ve protokoller kolayca geliştirilerek sonuçları ölçülebilir.

Tasarlanan sistemin nesneye dayalı esnek ve modüler mimarisi sayesinde sistem üzerinde yeni değişiklikler, eklemeler kolaylıkla yapılabilir. Mevcut alt yapı kullanılarak yeni modüllerin eklenmesiyle, IPv6 ve gezgin VPN desteği kısa zamanda kazandırılabilir. İleriki aşamalarda, şuan tasarım aşamasında olduğumuz, trafik akış gizliliğini arttırmak için yeni bir çalışma konusu olan anonim IPSec VPN geçidini mevcut mimari üzerinde gerçekleyip NS-2 ağ benzetim ortamında denemeyi planlamaktayız.

KAYNAKLAR

[1] **Kleinrock, L.,** 2004**.** Visionary Conference: Facing the Future Istanbul, Turkey July 9.

[2] **Rivest, R.,** The MD5 Message-Digest Algorithm, RFC 1321.

[3] **FIPS 180-1,** 1995. Secure Hash Standart, *NIST*, Gaithersburg, A.B.D.

[4] **Wang, X., Yin, Y.L. and Yu, H.,** "Finding Collisions in the Full SHA-1".

[5] **Wang, X., Feng D., Lai X. ve Yu, H.,** 2004. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, rump session, CRYPTO 2004, Cryptology ePrint Archive, Report 2004/199, second version (August 17), http://eprint.iacr.org/2004/199.pdf.

[6] **Kent S. ve Seo, K.,** Security Architecture for the Internet Protocol, RFC 4301.

[7] **Tanenbaum, A. S.,** 2003. Computer Networks 4/e.

[8] **Maughan, D. ve diğ.,** Internet Security Association and Key Management Protocol (ISAKMP), RFC 2408.

[9] **Kent S.,** IP Encapsulating Security Payload (ESP), RFC 4303.

[10] **Kent S.,** IP Authentication Header(AH), RFC 4302.

[11] **Krawczyk ve diğ.,** HMAC: Keyed-Hashing for Message Auyhentication, RFC 2104.

[12] **Eastlake 3rd Motorola Laboratories.,** Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH), RFC 4305.

[13] **Frankel S., Herbert H**., The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec, RFC3566.

[14] **Madson, C., Glenn, R.,** The Use of HMAC-SHA-1-96 within ESP and AH, RFC 2404.

[15] **Madson, C., Glenn, R.,** The Use of HMAC-MD5-96 within ESP and AH, RFC 2403.

[16] **Frankel S. ve diğ.,** The AES-CBC Cipher Algorithm and Its Use with IPsec, RFC 3602.

[17] **Housley, R.,** Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP), RFC 3686.

[18] **Hoffman, P.,** Cryptographic Suites for IPsec, RFC 4308.

[19] **Kohler, E.,** 2000. The Click modular router. Ph.D. thesis, MIT.

[20] **Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek**, **M.F.,** 2000. ACM Transactions on Computer Systems 18(3), August, pages 263-297.

[21] **The Network Simulator - ns-2,** http://www.isi.edu/nsnam/ns/.

[22] **Neufeld, M., Jain, A. ve Grunwald D.,** 2002. Nsclick::bridging network simulation and deployment, In Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems, pages 74–81. ACM Press.

[23] **The Click Modular Router Project**, http://www.read.cs.ucla.edu/click/.

[24] **The Grid Ad Hoc Networking Project**, http://pdos.csail.mit.edu/grid/.

[25] **MIT Roofnet 802.11b mesh network,** http://pdos.csail.mit.edu/roofnet/doku.php.

[26] **The Click DSR Router Project,** http://pecolab.colorado.edu/DSR.html.

[27] **Palanisamya N. K.,** 2003. Modular Implementation of Temporally Ordered Routing Algorithm, ME Thesis, University of Colorado.

[28] **Braem B.,** 2005. Implementation and evaluation of ad-hoc distance vector routing, ME Thesis, Universiteit Antwerpen.

[29] **Larman C.,** 2005. Applying UML and Patterns , An Introduction to OOA/D and Iterative Development, 3/e, Prentice Hall PTR.

[30] **Shalloway A. and Trott J.,** 2002. Design Patterns Explained: A New Perspective on Object-Oriented Design, Addison-Wesley.

[31] **Gamma, E., Helm, R., Johnson, R., Vlissides, J.,** Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional Computing Series.

[32] **Douglas, B. P.,** 2002. Real-Time Design Patterns: Robust Scalable Architecture for Real Time Systems, Addison Wesley.

[33] **Buzluca, F.,** 2006. Nesneye Dayalı Modelleme ve Tasarım Ders Notları.

[34] **Egevang, K.B. ve Francis, P.,** The IP Network Address Translator (NAT), RFC 1631

[35] **Lakshman T.V. and Stiliadis, D.,** 1998. "High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching," Proc. ACM Sigcomm, pp. 191–202, September.

[36] **Miller, M.,** Optimum Search Methods for Switch/Router Databases in Access and Metro Edge Networks , IDT White Paper.

[37] **Lara Networks**, 2005. http://www.laratech.com.

[38] **Berg, M., and Marc, K.,** 2000. Computational Geometry: Algorithms and Applications, Springer-Verlag.

[39] **Srinivasan, V., Suri, S. and Varghese G.,** 1999. "Packet classification using tuple space search," in Proc. ACM SIGCOMM, pp. 135–146.

[40] **Gupta, P. ve McKeown, N.,** 2000. "Packet Classification using Hierarchical Intelligent Cuttings", IEEE Micro, Vol. 20, No.1, p.34-41.

[41] **Gupta, P. ve McKeown, N.,** 1999. Packet Classification on Multiple Fields" ACM SIGCOMM '99 September.

[42] **Feldmann, A. and Muthukrishnan, S.,** 2000. "Tradeoffs for packet classification," in Proc. IEEE INFOCOM, , pp. 1193–1202.

[43] **Knuth., D.E.,** 1998. The art of computer programming, vol. 3: sorting and searching, Addison-Wesley, 3rd edition.

[44] **Sedgewick, R. and Flajolet, P.,** 1996. An introduction to the analysis of algorithms, Addison-Wesley.

[45] **Nilsson, S. and Karlsson, G.,** 1999. "IP-address lookup using LC-tries," IEEE Journal of Selected Areas in Communications, June vol. 17, no. 6, pages 1083-92.

[46] **Degermark, M, Brodnik, A., Carlsson, S., and Pink, S.,** 1997. "Small forwarding tables for fast routing lookups," Proceedings of ACM Sigcomm, October pages 3-14.

[47] **Shafai, F. ve diğ.,** 1998. "Fully Parallel 30-Mhz, 2.5 Mb CAM" IEEE J. Solid-State Circuits vol. 33, no. 11, November.

[48] **Lampson, B., Srinivasan, V.,and Varghese, G.,** 1998. "IP lookups using multiway and multicolumn search," Proceedings of IEEE Infocom, vol. 3 , pages 1248-56, April.

[49] **Baboescu, F. ve Varghese, G.,** 2001. Scalable packet classification, ACM SIGCOMM '2001 Pp: 199 – 210.

[50] **Overmars, M.H. and Stappen, A.F.,** 1996. "Range searching and point location among fat objects", in Journal of Algorithms, 21(3), pp 629-656.

[51] **Çölkesen R. ve Örencik B.,** 2003. Bilgisayar Haberleşmesi ve Ağ Teknolojileri

[52] **Thales White Paper,** Understanding IP Encryptor Performance, http://www.thales-esecurity.com/DownloadDocuments/-Understanding_IP_Encryptor_Performance.pdf.

[53] **Mano M.,** 1993. Computer System Architecture, 3/e, Prentice Hall.

[54] **Intel Application Note**, Small Packet Traffic Performance Optimization for 8255x and 8254x Ethernet Controllers.

[55] **Sridhar T.,** 2003. Designing Embedded Communications Software CMP Books.

[56] **Network Processing Forum Benchmarking Working Group**, 2004. IPSec Forwarding Application Level Benchmark Implementation Agreement Revision 1.0 July 15.

[57] **Housley, R.**, Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP) , RFC 4309.

[58] **Whiting ve diğ.,** Counter with CBC-MAC (CCM), RFC 3610.

[59] **FIPS 197,** 2001. Advanced Encryption Standart (AES), *NIST*, Gaithersburg, A.B.D.

[60] **Computer Security Resource Center.** 2005. http://csrc.nist.gov/CryptoToolkit/modes.

[61] **Jonsson, J.**, 2002. On the Security of CTR + CBC-MAC, *Ninth Annual Workshop on Selected Areas in Cryptography*, Newfoundland Kanada

[62] **Buzluca, F.,** 2006. Bilgisayar Mimarisi Ders Notları.

[63] **Bilişim Sözlüğü,** 2004. http://www.tbd.org.tr/index.php?module=dict.

EK-A ÖRNEK KULLANIM SENARYOSU:

Kullanım senaryosu KS-2: Güvenli(Kırmızı) arayüzden alınan paketi işlemek

Birincil Aktör(Primary Actor): Güvenli arayüzden alınan paket

İlgililer ve Beklentileri (Stakeholders and Interests):

-sistem yöneticisi – paketle ilgili olayların (MIB'lerin sayılarının güncellenmesini ve alarm kayıtlarının tutulmasını (audit log)ister)

Ön Koşullar(PreConditions): Sistem devrededir ve arayüzler aktif durumdadır.(interface up)

Son Koşullar:Succes Guarantee(PostCondition): paket başarılı bir şekilde işlenir.

Ana Başarılı Senaryo (Main Success Scenario or Basic Flow):

1. **Ethernet kartı** ile **güvenli (kırmızı)arayüzlerin** birinden bir **IP paketi** , **vpn cihazına** gelir.
2. sistem **Ethernet paketini**, **paketin tip** (*type*) **alanına** bakarak sınıflandırır..(kim sınıflandırır? Ethernet paket sınıflandırıcı...)
3. tipi ipv4 olarak belirlenen Ethernet paketinin hangi kırmızı arayüzden alındığı bilgisi, paketle ilgili bilgilerin tutulduğu (..) ayrı bir alanda ek not olarak (*annotation*) işaretlenir,
4. sistem ilgili arayüzden alınan **paket sayısı MIB'ini** günceller,
5. sistem **paketin ethernet başlıklarnıı** kontrol eder ve paketin Ethernet başlığı kısmını atar.(ilerletilir IP başlığına ötelenir)
6. sistem **IP başlığını** RFC 791' e göre kontrol eder
7. sistem **Paketin** içindeki **varış IP adresi**(dest_ip), **kaynak ip adresi** (src_ip), ve **akış ile ilgili bilgileri**(protocol, varsa varış kaynak portları vs...) belirler.
8. sistem paketi **IPSec güvenli paket alma modülüne** verir.
9. IPSec güvenli alma modülünde, Pakete ilişkin **karar** , **çıkan politika kural veri tabanına** (outbound policy, **Security Policy Database SPD**) sırayla bakılarak bulunur.
10. sistem **Kural**llara tek tek yukarıdan aşağıya kuralda belirtilen **seçici**lere uyup uymadığı belirleyerek bakar
11. Seçiciler paket içindeki belirli alanlara(seçici alanları diyebiliriz) bakarlar. Paketin bir kurala uyması için, tüm seçicilere uyması gerekir. Seçici alanlar varış adresi, kaynak adresi , protokol numarası, varış ve kaynak portu vs olabilir, Seçiciler tiplerine göre tek bir değeri (exact match), bir aralığı(maskeli ya da bir aralığı) ya da olabilecek her değeri kapsayabilir. (any). Bir kuralda tüm bu seçicilerin olması zorunlu değildir.
12. Paketin uyduğu karar IPSec tünel uygula ise,sistem paketin tünelleneceği güvenlik birliğini (SA**, Security association**), Güvenlik birliği veri tabanında (**SAD**) bulur
13. sistem SA'daki konfigrasyonlarda belirtirildiği gibi paket şifrelenir, asıllanlanır, **dış IP** eklenir ve **ESP tünel paketini** oluşturur. <ALT SENARYO_SA'ya_gore_ESP paketi oluşturma **KS2_alt_1_ger / KS2_alt_2_ger** >
14. sistem SA ile ilgili yaşamsüresi (**lifetime**) , gönderilen **paket sayısı** miktarı, sayıcılarını günceller.
15. sistem **IPSec mibleri** günceller
16. sistem paketi paket **IP iletim (IP_Forwarding) modülüne** verir
17. sistem **LookUpIPRoute modülü** ile paketin varış adresine göre (dst_ip) , **next_hop** ve arayüzü belirler
18. sistem **Dropbroadcast elementi** ile broadcast paketlerinin aktarımına izin vermez.
19. sistem **painttee elementti** ile, paketin alındığı arayüzden ve alındığı taraftan tekrar yollanması engellenir.
20. sistem **paketin ttl değeri** bir azaltır
21. sistem **Paketin fragmantasyonu** gerekliyse paket parçalanır ve **arp_querier elementine** verir
22. sistem arp_querier elementinden kuyruğa geçirilen paketler ilgili arayüze verir ve hatta yollayarak **güvensiz arayüzden** apketi aktarır.
23. sistem **güvensiz arayüzle ilgili gönderme miblerini** günceller.

Uzantılar(Extensionsor Alternative Flows):

*.
2a. Tip arp paketi
2b tip ipv6 paketi
3a. Paket ipv6 paketi olabilir, ipv6 işleme kısmı gerçeklenmediğinden paket atılır.
3b. Paket arp sorgu(arp queries) paketi olabilir, paket arp cevaplayıcı (arp responder)elemanına verilir. Alt senaryo:< KS_ARP_RESPONDER>
3c. Paket arp sorgu cevabı paketi (arp reply)olabilir. Paket arp sorgulayıcısı (arp querier) elemanına verilir. . Alt senaryo:< KS_ ARP_QUERIER>

6a. Geçerli olmayan IP paketleri atılır, ilgili mibler güncellenir ve bir sonraki paket beklenir. (hangi mibler güncellenir.)
12a. Pakete ilişkin karar durdur olabilir, bu durumda paket atılır, (audit??) ilgili mib güncellenir.pakete gönderelen icmp dest. Prohibited administratively, mesajı gönderilir,
12b açık geçir olabilir.

1. açık geçire ilişkin mibler güncellenir
2. 21' den devam edilir..

12c. Paketin uyduğu hiç bir kural olmaya bilir bu durumda paket atılır, (audit) ilgili mib güncellenir. (pakete gönderen icmp dest. Prohibited administratively, mesajı gönderilir,
12d. Karar ipsec olup paketin cihazın kendisine gelebilir, bu durumda paket atılır.
12e. Aramada hata oluşabilir zaman aşımı olabilir..

17a Yön bulunamazsa paket atılır**, yön bulunamama mib'i** güncellenir. (icmp mesajı yollanır mı?..ileri aşamada..)

21a. Paketin DF biti setli ise, icmp error yollanır.()
Özel İstekler (Special Requirements):

Teknolojik Beklentiler (Tecnology and Data Variation List):
-tarama işlemlerinin hızlı olması için ,

-Sıklık (Frequency of Occurence): vpn cihazının paket işleme hızı, (paket *rate*') kadar.

KS2_alt_1_ger _AES-CBC: ALT SENARYO_SA' ya_ göre_ESP_tünel paketi oluşturma – gerçekleme

Birincil Aktör(Primary Actor): Kırmızıdan alınan , SA işlemeye ulaşan paket
İlgililer ve Beklentileri (Stakeholders and Interests):
-
Ön Koşullar(PreConditions): Sistem

Son Koşullar:Succes Guarantee(PostCondition): tünel paketi başarılı bir şekilde işlenir.

Ana Başarılı Senaryo (Main Success Scenario or Basic Flow):
.....

1. sistem spi değeri SA'dan alır.
2. sistem sıra numarası bir arttırır
3. sistem sıra numarasının taşıp taşmadığı kontrol eder.
4. sistem 32 bit spi ve sıra numrasının düşük anlamlı 32 biti, paketin başına_ ESP başlığı olarak ekler.
5. sistem toplam şifrelenecek toplam veri boyunu hesaplar. (total__enc_len orjinal paket+ESP_trailer(2))
6. sistem şifreleme için 16 byte'lık blok boyuna göre pad uzunluğunu belirler (pad_len= 16-total__enc_len%16)
7. sistem paketin sonuna **tfc padding** 16 byte'in'katı olacak şekilde ekler
8. sistem pad alanını pad alanına sırayla 1,2,...pad_len ile doldurur.

9. sistem paketin sonunda ESP trailer alanlarında biri olan , 8 bit payload length'e pad miktarını yazar
10. sistem paketin sonunda ESP trailer alanlarında biri olan, 8 bit next header alanına tünellenen orjinal paketin protokol numarasını yazar
11. sistem 16 sekizli(byte) IV değeri üretir.
12. sistem Sonuçta oluşan paket alanlarını (payload+padding+ESP trailer), SA'da belirtilen **şifreleme algoritması AES** , algoritma modu cbc, , anahtar(128-192-256 bit olabilir), ve 16 sekizli(byte) iv şifreleyiciye verilir. (RFC_3602_aes_cbc)
13. şifreleyici şifreleme işlemi bitince geriye şifreli paketi döndürür.
14. sistem ESP başlığından sonra , paketin payload alanının başına, 8 sekizli(byte)IV değerini koyar, ardından şifrelenen veriler gelir.
15. .....
16. sistem ESP paketin sonuna sıra numarasının yüksek anlamlı 32 bitini ekler,
17. sistem bütünlük değeri hesaplanacak ESP paketi(sıra numarasının yüksen anlamlı 32 biti de dahil) , (algoritmanın şartnamesinde belirtildiği şekilde)algoritmanın blok boyunun katı olacak şekilde paketin sonuna iç doldurma(mplict padding)eklenir.
18. sistem ESP paketini , asıllama anahtarı ile beraber bütünlük ve şifre bütünlük şifreleme algoritmasına verir.
19. asıllama algoritması ICV değerini geri döndürür.
20. sistem 10. ve 11. adımlarda eklediği kısımlar paketin sonundan siler.
21. sistem paketin sonuna ICV değerini ekler
22. sistem dış IP başlığı oluşturulur.
23. sistem paketin başına dış IP başlığı ekler.

Uzantılar(Extensionsor Alternative Flows):
3a-sıra numarası taşmış olabilir. -----
5a ipv4 için 4 byte ipv6 için 8 byte hizalama yapılır. -hizalamaya gerek kalmadı hem 8 in hem 4 ün katı oluyor.
10a- eğer implicit senkronizasyon verisi gerekli ise, algoritmaya input olarak yine verilir ama payload alanına koyulmaz.
11a. ESN seçil değilse bu adım atlanır.
Özel İstekler (Special Requirements):

Teknolojik Beklentiler (Tecnology and Data Variation List):
-hızlı işleme için, bazı verileri cepte tutma yapılabilir..
-şifreleyici iç bir yazılım birimi de olabilir, dış bir şifreleme donanımı da olabilir.
Sıklık (Frequency of Occurence): vpn cihazının hızı, (paket *rate*') i kadar.

ÖZGEÇMİŞ

Ural Erdemir 19 Ocak 1982' de İstanbul' da doğdu. Lise eğitimini Kocaeli Fen Lisesi'nde tamamladı ve 1998 yılında İstanbul Teknik Üniversitesi Bilgisayar Mühendisliği bölümüne girdi. Lisans eğitimini 2003 yılında tamamladı ve aynı yıl İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği programında Yüksek Lisans eğitimine başladı. 2004 yılından bu yana TÜBİTAK-UEKAE' de araştırmacı olarak çalışmaktadır.