KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

YENİ BİR KABLOSUZ ALGILAYICI AĞ VERİ BAĞI KATMANI GÜVENLİK PROTOKOLÜ TASARIMI

DOKTORA TEZİ

Necla BANDIRMALI

Anabilim Dalı: Elektronik ve Bilgisayar Eğitimi

Danışman: Prof. Dr. İsmail ERTÜRK

KOCAELİ, 2010

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

YENİ BİR KABLOSUZ ALGILAYICI AĞ VERİ BAĞI KATMANI GÜVENLİK PROTOKOLÜ TASARIMI

DOKTORA TEZİ

Necla BANDIRMALI

Tezin Enstitüye Verildiği Tarih: 24 Kasım 2009

Tezin Savunulduğu Tarih: 8 Ocak 2010

Tez Danışmanı	Üye	Üye
Prof. Dr. İsmail ERTÜRK	Prof. Dr. Hüseyin EKİZ	Doç. Dr. Y.Emre ERDEMLİ
(.....)	(.....)	(.....)

Üye	Üye
Doç. Dr. Celal ÇEKEN	Yrd. Doç. Dr. A.Turan ÖZCERİT
(.....)	(.....)

KOCAELİ, 2010

ÖNSÖZ VE TEŞEKKÜR

Kablosuz Algılayıcı Ağlar (KAA'lar), endüstriyel uygulamalardan sağlık uygulamalarına kadar çok değişik alanlarda kullanılmaktadır. KAA'lar, özellikle sağlık ve askeri uygulamalar başta olmak üzere birçok alanda veri gizliliği, tazeliği ve bütünlüğü gibi temel güvenlik gereksinimlerini sağlamak zorundadır. Günümüzde, araştırmacılar, KAA düğümlerinin sınırlı kaynak, enerji ve hesaplama yeteneklerini göz önüne alarak çeşitli güvenlik protokolleri geliştirmektedir. Bir KAA güvenlik protokolü geliştirilirken kullanılacağı uygulama alanın iyi tespit edilmesi gerekmektedir. Çevre ya da endüstri uygulama alanında kullanılacak ise enerji tüketimi, sağlık ya da askeri uygulama alanında kullanılacak ise güvenlik esasları ön planda tutulmalıdır. Sunulan tez çalışmasında esas alındığı gibi, sağlık uygulamaları için geliştirilen bir KAA veri bağı katmanı güvenlik protokolünün, enerji tüketimini fazla arttırmadan üst düzey bir güvenlik sağlaması gerekmektedir. Bunu yaparken de, özellikle sınırlı kaynaklara sahip cihazlarda kullanılmak için tasarlanan güvenli bir şifreleme algoritması ve veri güvenilirliğini arttıran çeşitli şifreleme yaklaşımları kullanmak zorundadır.

Tez çalışmalarım süresince her konuda yardımcı olan ve tezin hazırlanmasında bütün detaylara kadar rehberlik eden, değerli zamanlarını ayıran, bilgi ve deneyimlerini paylaşan, çalışmalarımı yönlendiren ve her zaman destek olan tez danışmanım Prof. Dr. İsmail ERTÜRK'e (Kocaeli Üniversitesi) teşekkürlerimi sunarım. Tez izleme jüri üyesi olan Doç. Dr. Yunus Emre ERDEMLİ'ye (Kocaeli Üniversitesi) ve Yrd. Doç. Dr. Ahmet Turan ÖZCERİT'e (Sakarya Üniversitesi) yardım ve destekleri için çok teşekkür ederim. Tez çalışmalarımın yürütülmesinde her türlü ilgi ve desteği gösteren ve aynı zamanda bir süre tez izleme jüri üyesi de olan Doç. Dr. Celal ÇEKEN'e (Kocaeli Üniversitesi) ve Yrd. Doç. Dr. Mehmet YAKUT'a (Kocaeli Üniversitesi), tez çalışmalarıma başladığım ilk günlerde benden yardımlarını esirgemeyen arkadaşım Yrd. Doç. Dr. Cüneyt BAYILMIŞ'a teşekkürlerimi sunarım. Ayrıca çalışmalarım boyunca desteklerini gördüğüm tüm dostlarıma ve arkadaşlarıma da yardımlarından dolayı teşekkür ederim.

Beni bugünlerime getiren, her konuda destek veren ve yanımda olan çok değerli annem Arife, babam Orhan ve canım kardeşim Buket BANDIRMALI'ya yaptıkları her şey için teşekkürlerimi sunuyorum.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER

C : Şifreli veri (Ciphertext)
P : Şifresiz veri (Plaintext)

Alt indisler

N : Veri blok sayısı
i : Bit sayısı

Üst indisler

ab : Anahtar boyutu
nz : Normalize
vb : Veri blok boyutu

Kısaltmalar

AAP : Attacks Against Privacy
AES : Advanced Encryption Standard
CBC : Cipher Block Chaining
CBC-MAC : Cipher Block Chaining Message Authentication Code
CCM : Counter with Cipher Block Chaining Message Authentication Code
CFB : Cipher Feedback
CTR : Counter
DES : Data Encryption Standard
DoS : Denial of Service
ECB : Electronik Codebook
ECC : Elliptical Curve Cryptography
GCM : Galois/Counter Mode
IEEE : Institute of Electrical and Electronics Engineers
IPSec : Internet Protocol Security
ISM : Industries, Scientific, Medical
IV : Initial Vector
LLSP : Link Layer Security Protocol
MAC : Message Authentication Code
NIST : National Institute of Standards and Technology
NR : Node Replication

RSA	: Ron Rivest, Adi Shamir, and Leonard Adleman
OMAC	: One Key Message Authentication Code
OFB	: Output Feedback
SEA	: Scalable Encryption Algorithm
SNEP	: Sensor Network Encryption Protocol
SSH	: Secure Shell
SSL/TLS	: Secure Sockets Layer/Transport Layer Security
TDES	: Triple Data Encryption Standard
TDMA	: Time Division Multiple Access
TKIP	: Temporal Key Integrity Protocol
WEP	: Wired Equivalent Privacy
WSN	: Wireless Sensor Network

YENİ BİR KABLOSUZ ALGILAYICI AĞ VERİ BAĞI KATMANI GÜVENLİK PROTOKOLÜ TASARIMI

Necla BANDIRMALI

Anahtar Kelimeler: KAA, Veri Bağı Katmanı Güvenlik Protokolü, Yüksek Güvenlik.

Özet: Kablosuz Algılayıcı Ağ (KAA) güvenliği konusunda yapılan araştırmalar genellikle enerji, bellek ve işlemci gibi sınırlı düğüm kaynaklarının etkin kullanımı üzerine odaklanmaktadır. Bununla birlikte, tez çalışmasının başlangıç sebebini oluşturan ve özellikle sağlık ve askeri olmak üzere, kritik KAA uygulamalarında yüksek güvenlik sağlayan protokollere olan ihtiyaç günümüzde artan bir önem kazanmaktadır.

Bu tez çalışmasında, özellikle sınırlı kaynakları bulunan düğümlerde kullanılan SEA blok şifreleme algoritması ile veri gizliliği ve güvenilirliği için kullanılan CTR ve CBC–MAC asıllama/bütünlük denetim yaklaşımlarının bütünleşik kullanımıyla geliştirilen ve kısaca YP (Yeni Bir Kablosuz Algılayıcı Ağ Veri Bağı Katmanı Güvenlik Protokolü) olarak adlandırılan yeni bir protokol sunulmaktadır. YP güvenlik düzeyi, kullanılan yaklaşımlarla dinamik olarak yükseltilirken, bellek kullanım değeri nispî olarak çok az artmaktadır. Bununla birlikte, 96-bit veri blok/anahtar boyutuna sahip güvenlik düzeyinde, TinySEC'e kıyasla, YP'nin KAA düğüm enerji tüketim değerine olumsuz etkisi bulunmazken, 192-bit veri blok/anahtar boyutuna sahip oldukça yüksek güvenlik düzeyinde, ihmal edilebilir bir artış görülmektedir. Arttırılmış KAA güvenliği için kullanılan hemen hemen tüm ek süreçler ve algoritmalar, düğüm işlemcisinin daha fazla kullanımına bağlı olarak, uçtan–uca gecikme içerisinde önemli bir yeri olan şifreleme/şifre çözme süresini olumsuz etkilemektedir. Diğer yandan, toplam gecikmedeki bir artışa da karşılık gelen bu durum, KAA uygulaması tüm bileşenleri ile bir bütün olarak ele alındığında, hedeflenilen yüksek güvenlik sonucu ile dengelenmektedir.

YP'nin geleneksel TinySEC ve LLSP protokolleri ile karşılaştırmalı geçerlilik değerlendirme sonuçlarına ek olarak, YP'nin kullanıldığı basit bir KAA uygulamasının OPNET yazılımı ile modellenmesi ve benzetimi de bu tez çalışmasında sunulmaktadır. Sonuçlar genel olarak ele alındığında, tez çalışmasının özünü teşkil eden veri güvenilirliği arttırılırken diğer somut başarım ölçütleri açısından muhtemel olumsuzlukların, CTR ve CBC–MAC yaklaşımlarının YP içerisinde oldukça etkin bir şekilde uyarlanılması ile ortadan kaldırıldığı tespit edilmiştir.

A NEW DATA LINK LAYER SECURITY PROTOCOL DESIGNED FOR WIRELESS SENSOR NETWORKS

Necla BANDIRMALI

Keywords: WSN, Data Link Layer Security Protocol, High Security.

Abstract: The effective usage of sensor node energy and its other sources is of high importance in Wireless Sensor Network (WSN) security researches as much as those in for example MAC and physical layer studies. As the initial motivation of this thesis work, research on especially increasing security of the WSNs employed in military and health applications recently also receives a remarkable attention.

In this thesis, a new highly secure WSN data link layer security protocol called HighSEC is proposed. It smoothly combines the advantageous aspects of the SEA with the CTR & CBC–MAC approaches for highly increased data confidentiality and authentication & integrity functions, respectively. The security level of the HighSEC can be dynamically boosted up employing these methods with respect to a small increase in memory usage. Using the proposed HighSEC with the 96-bit data block/key size has a trivial effect on the WSN node energy consumption that is ignorably increased compared to that of the traditional TinySEC protocol while employing the HighSEC with the 192-bit data block/key size providing an extremely high level of security. Almost all of the additional processes for increased reliability in WSN data link layer security protocols adversely affect the total encryption/decryption times added to the end–to–end delay, mainly caused by the further exploitation of the microprocessor in the WSN nodes. Considering a complete WSN application including the physical layer components, this negative aspect should be perfectly counterbalanced with the achieved high security and reliability outcome as realized by the proposed HighSEC, well supporting its usability in especially crucial health and military areas.

In addition to the comparative evaluation of the HighSEC and traditional TinySEC and LLSP, modeling and simulation of a simple WSN application employing the proposed HighSEC have been realized using the OPNET software. It has been succinctly concluded that employing the HighSEC utterly increases the data reliability of the WSN application traffic while consequently expected drawbacks such as increased end to end delays and use of limited energy sources are overcome by means of adapting the CTR and CBC–MAC approaches in highly efficient manner.

1. GİRİŞ

Kablosuz Algılayıcı Ağlar (KAA'lar), tabi olaylardan, otomasyon ortamlarındaki üretim seviyesindeki cihazlara, bina güvenliğinden, yerküre hareketlerinin izlenmesine, sağlık ve askeri uygulamalara kadar çok değişik alanlarda kullanılmaktadır. KAA'lar, özellikle askeri uygulamalar başta olmak üzere birçok alanda veri gizliliği, bütünlüğü, tazeliği ve kimlik doğrulaması gibi temel güvenlik gereksinimlerini sağlamak zorundadır. KAA'lar, geleneksel ağlardan farklı olarak özellikle enerji gibi sınırlı kaynaklara sahip olduğundan, geleneksel iletişim güvenlik tekniklerinin doğrudan uygulanması açısından önemli olumsuzluklarla karşı karşıyadır. Günümüzde, KAA düğümlerinin sınırlı kaynak, enerji ve hesaplama yetenekleri göz önüne alınarak çeşitli güvenlik protokolleri geliştirilmektedir.

Güvenlik (ve veri güvenilirliği) konusunda yapılan araştırmalar ve uygulamalar, KAA alanındaki diğer tüm çalışmalarda olduğu gibi, genellikle enerji ve diğer kaynakların etkin kullanımı üzerine odaklanmaktadır. Bununla birlikte, tez çalışmasının başlangıç sebebini de oluşturan ve özellikle sağlık ve askeri olmak üzere, kritik KAA uygulamalarında yüksek güvenlik sağlayan protokollere olan ihtiyaç günümüzde artan bir önem kazanmaktadır.

Günümüze kadar KAA'larda güvenlik protokolleri üzerine birçok çalışma/araştırma gerçekleştirilmiştir. Aşağıdaki alt bölümde bu çalışmalardan birkaçı kısaca özetlenmektedir.

1.1. Literatürde Yapılan Çalışmaların Özetleri

Perrig ve diğ. (2002), kablosuz haberleşme ve sınırlı kaynaklara sahip ortamlar için SPINS (Security Protocols for Sensor Networks) güvenlik protokolünü geliştirmişlerdir. Bu protokol iki güvenlik bloğundan oluşmaktadır: SNEP ve µTESLA. SNEP (Secure Network Encryption Protocol), veri gizliliği, mesaj

asıllaması/bütünlüğü ve veri tazeliği sağlamaktadır. Veri gizliliği, bir blok şifreleme algoritması olan RC5 ve bir blok şifreleme yaklaşımı olan CTR (Counter Mode) kullanılarak gerçekleştirilmektedir. Veri tazeliğinin sağlanması ve tekrar gönderme saldırılarının önlenmesi için sayaç olarak başlangıç vektörü (IV) kullanılmaktadır. Veri asıllaması/bütünlüğü için ise Mesaj Asıllama Kodu (MAK) üreten CBC–MAC (Cipher Block Chaining Message Authentication Code) kullanılmaktadır. Diğer yandan, μTESLA (Micro Timed Efficient Stream Loss–tolerant Authentication) sınırlı kaynaklara sahip ortamlar için asıllanmış yayın (broadcast) sağlayan bir protokoldür. Bu protokolde, baz istasyonu gizli anahtarıyla her paket için bir asıllama kodu üretmektedir. Baz istasyonundan paketi alan düğüm, SNEP protokolüyle paketi onaylamakta ve anahtar baz istasyonu tarafından açıklanıncaya kadar aldığı paketi tamponunda saklamaktadır. Anahtar açıklandığı zaman ise düğüm, tamponundaki paketleri bu anahtarla onaylamaktadır. KAA'lar için geliştirilmiş olan bu güvenlik protokolünün enerji maliyet hesabı incelendiğinde, hesap karmaşıklığından çok, ek veri iletiminin sisteme aşırı yük getirdiği saptanmıştır. Ayrıca bu protokolün hiçbir gerçeklemesi yapılmamıştır. SPINS, şifreleme de kullanılan blok şifreleyici açısından tez çalışması ile farklılık gösterirken, blok şifreleme yaklaşımı ve mesaj asıllaması/bütünlüğü tez çalışmasında kullanılanla benzerlik taşımaktadır.

Karlof ve diğ. (2004), mevcut güvenlik protokollerinin algılayıcı ağ güvenliği için yeterli olmamasından yola çıkılarak TinySEC isimli bir veri bağı katmanı güvenlik mimarisi geliştirmişlerdir. TinySEC, güvenlik için sadece asıllama TinySEC–Auth (Authentication Only TinySEC) ve asıllama ve şifreleme (TinySEC–AE, Authenticated Encryption TinySEC) olmak üzere iki farklı kullanım seçeneği sunmaktadır. Sadece asıllama seçeneği, bir MAK ile tüm paketi doğrularken, veriyi şifrelememektedir. Asıllama ve şifreleme seçeneğinde ise, hem veri şifrelenirken hem de paket (başlık+veri) bir MAK ile doğrulanmaktadır. TinySEC'de şifreleme algoritması olarak SKIPJACK blok şifreleme algoritması, blok şifreleme yaklaşımı olarak ise CBC kullanılmaktadır. TinySEC, şifreleme de kullanılan blok şifreleyici ve blok şifreleme yaklaşımı açısından tez çalışması ile önemli farklılık gösterirken mesaj asıllaması/bütünlüğü için kullanılan yaklaşım tez çalışması ile benzerdir.

Ren ve diğ. (2005), KAA'ların sınırlı enerji kaynaklarını düşünerek enerji etkin bir veri bağı katmanı güvenlik protokolü olan LLSP'yi (Link Layer Security Protocol) gerçekleştirmişlerdir. LLSP'de veri gizliliği, simetrik blok şifreleme algoritmalarından AES ve blok şifreleme yaklaşımlarından CBC'nin bir arada kullanılması ile sağlanmaktadır. AES, güvenliği yüksek ve mevcut kablosuz ağlarda kullanılan bir şifreleme algoritmasıdır. TinySEC'e benzer şekilde, bu protokolde de şifrelemenin güvenliğini arttırmak için bir başlangıç vektörü kullanılmakta ve mesaj asıllaması/bütünlüğü için CBC–MAC yaklaşımı ile bir MAK elde edilmektedir. LLSP, aynı paketlerin tekrar gönderilip gönderilmediğinin denetimini yapmak için kullanılan sayaç değerinin, paket başlığına eklenmesi yerine, gönderici ve alıcı arasında bir kaymalı kaydedici (shift register) kullanmaktadır. Böylece paket ek yükü (overhead) azalmaktadır. LLSP, şifrelemede kullanılan blok şifreleyici ve blok şifreleme yaklaşımı açısından tez çalışması ile farklılık gösterirken mesaj asıllaması/bütünlüğü için kullanılan yaklaşım tez çalışması ile benzerdir.

Zhu ve diğ. (2003), algılayıcı ağlar için bir anahtar yönetim protokolü olan LEAP'i (Efficient Security Mechanism for Large–Scale Distributed Sensor Networks) geliştirmişlerdir. LEAP, algılayıcı ağlarda gizlilik ve asıllama/bütünlük sağlamak için aşağıda kısaca değinilen çoklu anahtar mekanizmasını kullanmaktadır:

- Kişisel anahtar: Her düğüm eşsiz (unique) bir anahtara sahiptir. Bu anahtar, baz istasyonu ile paylaşılarak güvenli haberleşmeyi sağlamaktadır.
- Paylaşımlı çiftli (pairwise) anahtar: Başka bir algılayıcı düğüm ile paylaşılan anahtardır. Bu anahtar ile güvenli iletişim sağlanır.
- Küme anahtar: Birden fazla komşu düğüm ile paylaşılan anahtardır. Küme anahtarlar güvenli yerel mesaj yayımları için kullanılır.
- Grup anahtar: Ağdaki tüm düğümlerle paylaşılan anahtardır. Baz istasyonu tarafından grubun tamamına şifrelenmiş mesaj yayımlarında grup anahtarları kullanılmaktadır.

LEAP güvenlik protokolü çeşitli KAA uygulamalarında farklı güvenlik gereksinimlerini karşılamaktadır. LEAP güvenlik protokolünde kullanılan anahtarlar, her bir düğümün haberleşmesini ve enerji verimliliğini düzenlemektedirler. LEAP baz istasyonu ile etkileşimi en aza indirir ve ağdaki pasif ortaklık ve yerel çakışma

için düğümlerin uzlaşması gibi önemli işlem mekanizmalarını içermektedir. LEAP, KAA'lardaki çoğu saldırıyı önleyebilmekte ya da azaltabilmektedir.

Park ve Shin (2004), sınırlı kaynaklara sahip algılayıcı düğümlerden oluşan, büyük ölçekli kablosuz ağlarda güvenlik çözümü için LISP (Lightweight Security Protocol) protokolünü sunmuşlardır. Bu yöntem, çok sayıda algılayıcı düğümden oluşan ağları ölçeklemek için kümelere ayırmaktadır. Her küme için bir küme başı seçilir ve anahtar sunucu oluşturulur. LISP güvenlik protokolü yeni bir anahtar yönetim mekanizması içermektedir. Anahtar yöntemini, küme başlarını ve anahtar sunucuları kullanarak gerçekleştirilmektedir. Bu yöntemin çeşitli avantajları şunlardır:

- Tekrar gönderim/ACK gerektirmeyen etkili anahtar yayını kullanır,
- Veri mesajına eklenmeksizin oluşturulan doğrulama verisi kullanır,
- Kayıp anahtarları algılama ve düzeltme özelliğine sahiptir,
- Şifrelenmiş/şifresi çözülmüş veriyi etkilemeden (bozmadan) anahtar yenileme yapabilir.

LISP'nin, saldırılara karşı kritik bilgileri korumak için sağladığı faydalar ise şu şekilde özetlenebilir:

- Gizlilik
- Veri bütünlüğü
- Erişim denetimi
- Anahtar yenileme.

LISP protokolü, güvenlikle beraber diğer servisleri de (yönlendirme, veri dağıtımı, konum) birleştirebilmektedir. LISP esnek ve enerji duyarlı bir protokoldür. Ayrıca ACK ve diğer kontrol paketlerine gerek duymadığından DoS saldırılarına karşı oldukça güçlüdür.

Chan ve diğ. (2007), algılayıcı ağlarda güvenli veri toplama için SIA (Secure Information Aggregation in Sensor Networks) adlı yeni bir çalışma önermişlerdir. Bu protokol geniş algılayıcı ağlarda, güvenli hesaplama ve işlem yapılabilmesi için geliştirilmiştir. Protokolde, merkezi düğümler belli sorgulara göre diğer algılayıcılardan gelen bilgileri toplamaktadır. SIA, güvenli hesaplama ve ölçümlerin

ortalamaları, ağ büyüklüğünün tahmini, en az ve en çok algılayıcı okuma bulma, rastgele örnekleme ve lider seçimi için etkili bir protokol yapısı sunmaktadır. Ayrıca bu protokol merkezi düğüm ve kullanıcı arasında sadece doğrusal haberleşme gerektirmektedir.

Luk ve diğ. (2007), düşük enerji tüketimi ve yüksek güvenlik sağlayan güvenli bir ağ katmanı protokolü MiniSec'i (A Secure Sensor Network Communication) tasarlamışlardır. MiniSec, MiniSec–U olarak adlandırılan tek–kaynak haberleşme ve MiniSec–B olarak adlandırılan çok–kaynak yayım haberleşme olmak üzere iki farklı yaklaşımına sahiptir. Her ikisinde de başlangıç vektörü olarak bir sayaç kullanılırken, veri gizliliği ve asıllama sağlamak için OCB şifreleme yaklaşımı kullanılmaktadır. Bu iki yaklaşım, sayaçların yönetimi konusunda farklılık göstermektedir. MiniSec–U'da her gönderici için yerel ayrı bir sayaç tutan alıcının bulunmasını gerektiren senkronize olmuş algoritmalar çalışırken, MiniSec–B'de her gönderici için böyle bir gereksinim bulunmamaktadır. Ancak MiniSec–B'de, tekrarlama saldırılarını önlemek için, veri bir Bloom filtrenin içinde depolanmaktadır. MiniSec, önceki yaklaşımlardan çok daha az enerji gerektiren yüksek seviyeli bir güvenlik sunmaktadır.

Liu ve Ning (2008), KAA'larda ECC (Elliptic Curve Cryptography) işlemleri için ayarlanabilen bir kütüphane yapısına sahip TinyECC (A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks) protokolünü gerçekleştirmişlerdir. Protokolün ilk amacı, algılayıcı ağ uygulamalarına kolaylıkla ayarlanabilen ve bütünleşik ECC tabanlı PKC (Public Key Cryptography) işlemleri için açık kaynak kodlu bir yazılım paketini kullanıma hazır (ready–to–use) olarak sunmaktır. Geliştiricilere büyük esneklik sağlayan farklı optimizasyon birleşimleri, değişik çalışma zamanı ve kaynak tüketimi seçenekleri sunmaktadır. TinyECC'nin MICAz, Tmote Sky ve Imote2 algılayıcı platformlarında, "Barrett Reduction", "Hybrid Multiplication", "Hybrid Squaring", "Projective Coordinate System", "Sliding Window Method", "Shamir's Trick" ve "Curve-Specific Optimization" yaklaşımları kullanılarak bellek kullanımı, çalışma zamanı ve kaynak tüketimi değerlendirmeleri yapılabilmektedir.

1.2. Tez Çalışmasının Amacı ve Başlatılma Sebebi

KAA düğümleri tarafından algılanan verinin gizli tutulmasını ve güvenilir olmasını gerektiren özellikle sağlık ve askeri uygulamalarının yaygın olarak kullanımıyla birlikte, veri güvenilirliği konusu da büyük önem kazanmıştır. Sunulan tez çalışmasının da en önemli başlatılma sebebini oluşturan, KAA sağlık uygulamalarında verinin oldukça güvenilir bir şekilde ağ içinde iletimini sağlamak amacıyla yeni yöntemler geliştirilmektedir. Bu yöntemler, genellikle enerji ve diğer kaynakların da etkin kullanımı üzerine odaklanmaktadır. Bu açıdan ele alındığında KAA'ların sınırlı kaynakları da düşünülerek yüksek güvenlik sağlayan KAA güvenlik protokollerinin geliştirilmesi gerekmektedir.

Güvenlik protokollerinin güvenilirlik düzeylerini arttırmak için literatürde tanımlanmış çeşitli yaklaşımlar bulunmaktadır. Sınırlı kaynaklar için tasarlanmış bir şifreleme algoritması, güvenli bir şifreleme ve mesaj asıllaması/bütünlüğü sağlayan yaklaşımların birlikte kullanılmasıyla, verinin güvenilirliği oldukça üst düzeylere çıkartılabilmektedir.

Özellikle sınırlı kaynakları bulunan düğümlerde kullanılan SEA (Scalable Encryption Algorithm) blok şifreleme algoritması ile veri gizliliği ve güvenilirliğini arttırmak için kullanılan CTR (Counter Mode) ve CBC–MAC (Cipher Block Chaining Message Authentication Code) mesaj asıllama/bütünlük denetim yaklaşımlarının, bu tez çalışmasında yeni bir bütünleşik yöntemle kullanımı ile yüksek güvenlik sağlayan bir KAA veri bağı katmanı güvenlik protokolü geliştirilmesi amaçlanmaktadır. Bununla birlikte, güvenlik düzeyi, kullanılan bu yaklaşımlar ile dinamik olarak yükseltilirken, temel başarım değerlendirme ölçütleri olan enerji, mikroişlemci ve bellek kullanım değerlerinde minimum değişim hedeflenmiştir.

Özetle, bu tez çalışmasının iki ana hedefi bulunmaktadır;

- Veri güvenilirliği arttırılmış yeni bir KAA veri bağı katmanı güvenlik protokolü tasarlamak ve
- Sağlık uygulamaları için yüksek güvenlik sağlamak.

1.3. Tez Çalışmasının Katkıları

Bu tez çalışmasında sınırlı kaynaklara sahip düğümleri bulunan KAA'lar için yeni bir veri bağı katmanı güvenlik protokolü önerilmiştir. Bu protokol KAA'larda veri gizliliği, tazeliği ve asıllaması/bütünlüğünü temin etmektedir. Önerilen bu yeni protokol ile bilime ve teknolojiye iki temel katkı sağlanmıştır;

- Özellikle sınırlı kaynakları bulunan düğümlerde kullanılan ölçeklenebilir bir şifreleme algoritmasının, değişik yaklaşımlarla veri güvenilirlik düzeyi arttırılarak literatürde bulunan veri bağı katmanı güvenlik protokollerine kıyasla yüksek güvenlik sağlayan bir KAA veri bağı katmanı güvenlik protokolü gerçekleştirilmiştir. Bu protokolün güvenlik düzeyi, kullanılan yaklaşımlarla dinamik olarak yükseltilirken, bellek kullanım değeri nispî olarak çok az artmaktadır. Bununla birlikte, 96-bit veri blok/anahtar boyutuna sahip güvenlik düzeyinde, TinySEC'e kıyasla, YP'nin KAA düğüm enerji tüketim değerine olumsuz etkisi bulunmazken, 192-bit veri blok/anahtar boyutuna sahip oldukça yüksek güvenlik düzeyinde ihmal edilebilir bir artış görülmektedir.
- Kısıtlı enerji kaynaklarının etkin kullanımından çok, iletilen verinin güvenilirliğinin (doğruluğunun) temin edilmesini gerektiren askeri ve sağlık uygulamalarında yüksek güvenlik sunulabilmektedir.

Tez çalışmasının yukarıda ifade edilen ana katkılarının yanı sıra bazı ek özellikleri de bulunmaktadır. Bunlar aşağıda maddeler halinde sıralanmıştır;

- Geliştirilen protokolde, mevcut kablosuz ağ uygulamalarında kullanılan bir şifreleme algoritması yerine literatürde sınırlı kaynaklar için sunulmuş ve çeşitli ortamlarda denenerek etkinliği ve güvenliği kanıtlanmış ölçeklenebilir bir şifreleme algoritması kullanılarak KAA düğüm kaynak kullanımı azaltılmaktadır.
- Literatürde tanımlanmış şifreleme yaklaşımları kullanılarak veri güvenilirliği dinamik olarak arttırılabilmektedir.
- Önerilen YP'nin kullanıldığı örnek bir KAA uygulama modeli ve benzetimi OPNET yazılımı ile gerçekleştirilerek araştırmacıların kullanımına sunulmuştur.

1.4. Tez Düzeni

Bölüm 2'de, KAA'lar, olumlu ve olumsuz özellikleri, uygulama alanları, düğüm mimarisi, topolojisi ve protokol mimarisi hakkında temel bilgiler verilmektedir.

Bölüm 3'de, KAA güvenliği anlatılmaktadır. KAA güvenliğindeki engeller, güvenli KAA gereksinimleri, saldırılar ve savunma önlemleri detaylı bir şekilde açıklanmaktadır.

Bölüm 4'de, değişik şifreleme yöntemleri, yaklaşımları ve bunların geleneksel KAA veri bağı katmanı güvenlik protokollerinde kullanımı sunulmaktadır.

Bölüm 5'de, YP (Yeni bir Kablosuz Algılayıcı Ağ Veri Bağı Katmanı Güvenlik Protokolü) ayrıntılı olarak açıklanmaktadır. Ayrıca, YP'nin geçerlilik değerlendirmesi sunularak, YP ve LLSP'nin TinySEC ile normalize edilen bellek kullanım ve enerji tüketim değerleri karşılaştırmalı olarak incelenmektedir. Yine bu bölümde, OPNET yazılımı kullanılarak gerçekleştirilen, YP'nin örnek bir ağ uygulama modeli ve benzetimi sunulmaktadır. Elde edilen benzetim (uçtan-uca gecikme) sonuçları, geleneksel TinySEC yöntemi ile karşılaştırmalı olarak değerlendirilmektedir.

Önerilen KAA veri bağı katmanı güvenlik protokolü YP'nin temel özellikleri, bilime ve teknolojiye sunmuş olduğu katkılar, sonuçlar ve değerlendirmeler Bölüm 6'da ifade edilmektedir. Bu bölümde ayrıca, tasarlanan protokolün geliştirilmesine ve uygulanmasına yönelik öneriler de yer almaktadır.

2. KABLOSUZ ALGILAYICI AĞLAR

2.1. Giriş

KAA'lar, sınırlı mikroişlemci ve bellek kapasitelerine sahip, kısa mesafede kablosuz ortam üzerinden haberleşebilen düşük güçlü, düşük maliyetli ve çok işlevli algılayıcı düğümlerden meydana gelmektedir (Akyildiz ve diğ., 2002). Kurulum sonrası sorunsuz çalışabilmeleri, ek bakım gerektirmemeleri ve çok çeşitli uygulama alanlarına sahip olmalarından dolayı KAA'ların kullanımı gün geçtikçe yaygınlaşmaktadır. Bu bölümde, KAA'ların kullanıcılarına sunduğu ve literatürde üzerinde çok sık durulan önemli olumlu/olumsuz özellikleri, uygulama alanları, KAA düğüm mimarisi, KAA topolojisi ve KAA protokol mimarisi açıklanmaktadır.

2.2. KAA Olumlu ve Olumsuz Özellikleri

KAA Olumlu Özellikleri:

- Hareketlilik: Kablosuz haberleşen düğümler, kapsama alanı içerisinde herhangi bir kısıtlama olmaksızın hareket edebilmektedirler. Buna bağlı olarak dinamik ağ topolojileri kolayca kullanılabilmektedir.
- Taşınabilirlik: Herhangi bir kablolama ve enerji tesisi/altyapısı gerektirmediğinden, mevcut herhangi bir KAA, bir yerden başka bir yere kolaylıkla taşınabilir.
- Yeniden kullanılabilirlik: Fiziksel ortamlardan çeşitli verileri algılamayı amaçlayan algılayıcı düğümler, değişik şekillerde ve farklı uygulamalarda yeniden kullanılabilmektedir.
- Kolay kullanım: Algılayıcı düğümler, yeni bir düzenleme gerektirmeden, aralarında dinamik bir biçimde organize olarak değişen koşullara uyum sağlayabilmektedir.
- Ölçeklenebilirlik: Mevcut bir KAA'ya yeni düğümlerin ya da başka bir KAA'nın eklenmesi kolayca ve dinamik bir şekilde mümkün olabilmektedir.

- Düşük maliyet: Kablosuz iletişim ve mikro-elektromekanik sistem teknolojilerindeki gelişmelere paralel olarak, algılayıcı düğüm maliyetleri oldukça düşüktür ve gün geçtikçe azalmaktadır.

KAA Olumsuz Özellikleri:

- Kısıtlı kaynaklar: Küçük algılayıcı düğümler, hesaplama yetenekleri, bellek kapasiteleri ve enerji kaynakları sınırlı olacak şekilde üretilmektedir.
- Yönetim ve izlenebilirlik zorluğu: Algılayıcı düğümler, algılama ve basit hesaplama gibi işlevler öngörülerek tasarlandığından, uzaktan yönetim ve trafik mühendisliğinin gerektirdiği algoritmalar, nispî olarak KAA'lar için yüksek kaynak kullanımına sebep olmaktadır.
- Yüksek hata olasılığı: Algılama ortamının coğrafik ve fiziksel yapısına bağlı olumsuz etkenler ve kablosuz iletişim karakteristiklerine bağlı olarak, bit hata oranları klasik kablolu iletişim sistemlerine göre çok yüksek olabilmektedir.
- Sınırlı servis kalitesi desteği: Yüksek bit hata oranı ve dinamik topolojisine bağlı olarak, KAA servis kalitesi desteği oldukça sınırlıdır (Raghavendra ve diğ., 2004).

2.3. KAA Uygulama Alanları

KAA'ların literatürde sunulan önemli uygulama alanları altı temel grupta incelenmektedir:

- Askeri uygulamalar: Askeri birliklerin cephane ve araç/gereçlerinin izlenmesi, hedef tespiti, savaş alanını gözaltında tutma, düşmanları ve düşman hattını keşfetme, hasar tespiti ve değerlendirmesi, nükleer, biyolojik ve kimyasal saldırıların tespiti vb.
- Çevre uygulamaları: Orman yangınlarının tespiti, sel tespiti, tarım arazilerinin değerlendirilmesi vb.
- Endüstriyel uygulamalar: Süreç izleme ve kontrol, enerji hatlarının izlenmesi, taşımacılık, titreşim izleme vb.
- Tıbbi uygulamalar: İnsan fizyolojisinin ve psikolojisinin uzaktan gözlemlenmesi, hastane içerisindeki hasta ve doktorların izlenmesi vb.

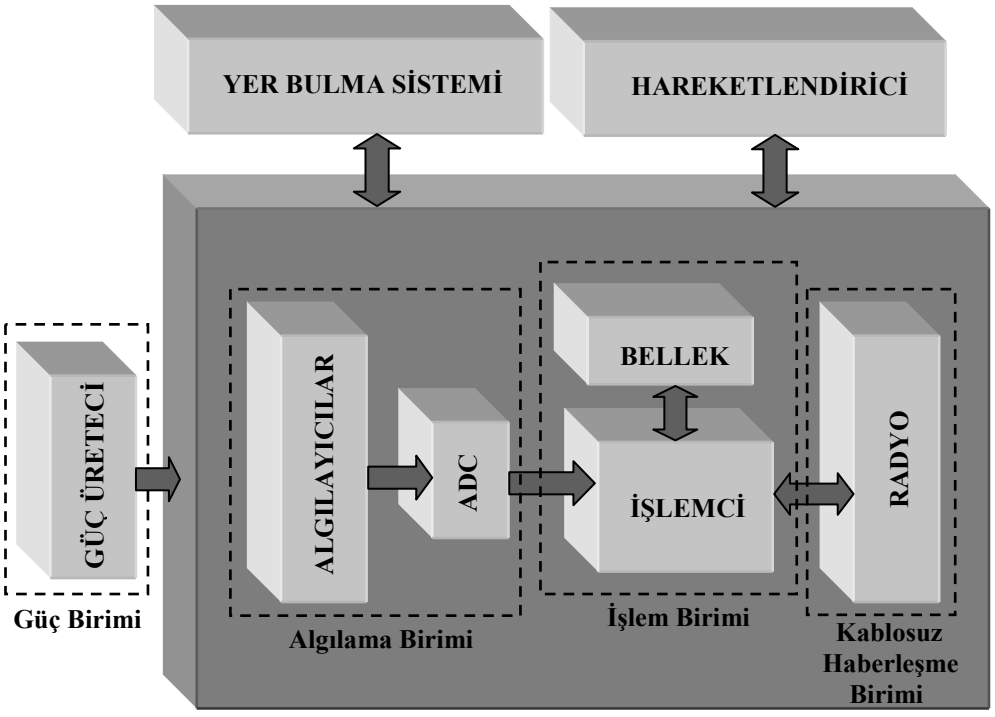
- Bina otomasyonu: Güvenlik, konum/yer tayini (çalışan, malzeme, araç gibi), aydınlatma tesisatı kontrolü, yangın alarmı, deprem tahmini vb.
- Diğer ticari uygulamalar: Binaların çevresel kontrolü, araç hırsızlığının tespiti ve gözlenmesi, araç takibi ve belirleme, stok kontrol yönetimi vb (Raghavendra ve diğ., 2004).

2.4. KAA Düğüm Mimarisi

Bir KAA düğümü, Şekil 2.1'de görüldüğü gibi dört temel birimden oluşmaktadır. Bunlar;

- İşlem birimi (mikroişlemci ve bellek),
- Kablosuz haberleşme birimi (kısa mesafeli radyo alıcı/verici),
- Algılama birimi (algılayıcı ve analog/sayısal çevirici) ve
- Güç birimidir.

KAA düğümleri, uygulama gereksinimlerine bağlı olarak konum/yer bulma sistemi ve hareketlendirici gibi ek birimler de içerebilmektedir (Akyildiz ve diğ., 2002, Raghavendra ve diğ., 2004).

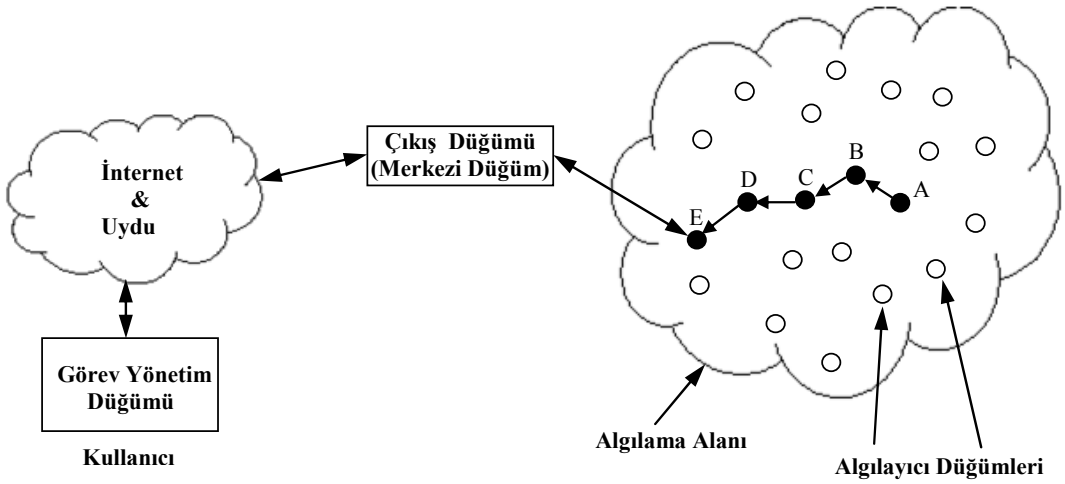


Şekil 2.1: KAA düğüm mimarisi.

2.5. KAA Topolojisi

Şekil 2.2'de örnek bir KAA topolojisi görülmektedir. Bu ağ topolojisini oluşturan birimler ve işlevleri şunlardır:

- KAA düğümleri: Fiziksel ortamdaki büyüklükleri algılama, işleme ve yönlendirme işlemlerini gerçekleştirir. Veriler, algılayıcı düğümler üzerinden çıkış düğümüne gönderilir.
- Çıkış düğümü (sink)/Merkezi düğüm: Algılama alanı içerisindeki KAA düğümleri tarafından gönderilen verileri toplayan ve son kullanıcıya internet ya da uydu üzerinden ulaştıran düğümdür.
- Kullanıcı: Çıkış düğümünden gelen bilgileri işler ve değerlendirir.



Şekil 2.2: KAA topolojisi.

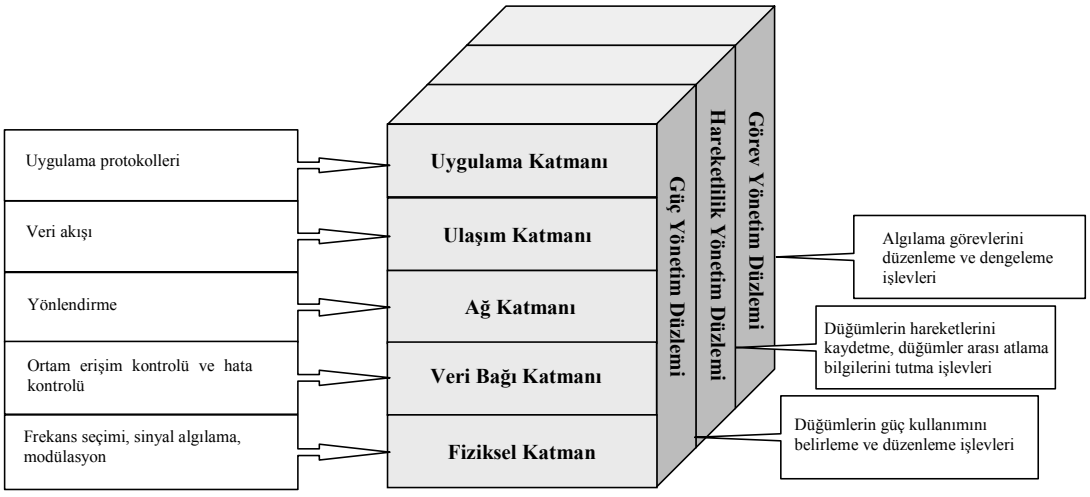
2.6. KAA Protokol Mimarisi

KAA protokol mimarisi, fiziksel, veri bağı, ağ, ulaşım ve uygulama katmanlarını içeren güç, hareketlilik ve görev yönetim düzlemlerinden oluşmaktadır (Şekil 2.3). Algılayıcı düğümler ve merkezi düğüm bu temel protokol yapısını kullanmaktadır.

- Fiziksel Katman: Frekans seçimi, taşıyıcı frekans üretimi, sinyal belirleme ve modülasyon fonksiyonlarından oluşmaktadır. Kablosuz iletişim için ISM (Industries, Scientific, Medical, Endüstriyel, Bilimsel ve Tıbbi) frekans bandı kullanılmaktadır.

- Veri Bağı Katmanı: Veri akışını çoklama (multiplexing), veri çerçevesi belirleme, ortam erişim kontrol (OEK) ve hata kontrol işlevlerini içermektedir. Ağ içerisinde noktadan-noktaya ve noktadan-çok noktaya güvenli bağlantılar sağlar. Ortam erişim kontrolü, KAA düğümleri arasında veri transferi için gerekli ağ bağlantılarını düzenler. Hata kontrolü mekanizmaları ise gönderilen verinin güvenli olarak iletimden sorumludur.
- Ağ Katmanı: Yönlendirme protokolleri yardımıyla, mevcut minimum enerji yolu, minimum atlama yolu gibi değişik ölçütleri esas alarak, çıkış düğümüne giden en uygun veri yolunu tespit eder (Raghavendra ve diğ., 2004).
- Ulaşım Katmanı: Düzenli iletişim, akış/tıkanıklık kontrolü, paket kayıp oranının iyileştirilmesi ve muhtemel servis kalitesi taleplerinin sağlanması gibi işlevleri yerine getirmektedir (Wang ve Sohraby, 2006).
- Uygulama Katmanı: Algılama görevlerine bağlı olarak farklı tür ve amaçta geliştirilen uygulama yazılımları bu katmanda tanımlanmaktadır.

Güç yönetim düzlemi, algılayıcı düğümlerin enerji kullanımını yönetir ve düzenler. Hareketlilik yönetim düzlemi, algılayıcı düğümlerin hareketlerini algılayarak ve kaydederek, kullanıcıya giden yolu (bağlantı) korur. Görev yönetim düzlemi ise KAA içerisindeki düğümlerin tüm algılama görevlerini planlar ve ayarlar (Akyildiz ve diğ., 2002).



Şekil 2.3: KAA protokol mimarisi.

2.7. Sonuç

Bir KAA düğümü; algılama, veri işleme ve kablosuz ortamdan verinin gönderimi gibi işlevleri yerine getirmektedir. Gözlem yapılacak ortama rastgele dağıtılabilen KAA düğümleri, birbirlerini tanıyabilmekte ve işbirliği içerisinde geniş bir alanda ölçüm vazifesini gerçekleştirebilmektedir. Sağlık, askeri, bina güvenliği, orman yangınlarının önceden tespiti gibi çok çeşitli alanlarda kullanılmaktadır.

Tezin bu bölümünde KAA düğüm mimarisi, topolojisi ve protokol mimarisi incelenerek tez çalışması olarak gerçekleştirilen yeni bir KAA veri bağı katmanı güvenlik protokolünün çalışacağı ortam ve sahip oldukları kaynaklar (hesaplama/bellek kapasitesi) hakkında kısa bir bilgi verilmektedir.

3. KABLOSUZ ALGILAYICI AĞ GÜVENLİĞİ

3.1. Giriş

Geleneksel ağlar için tasarlanan ve günümüzde birçok uygulamada yaygın olarak kullanılan güvenlik yöntemleri; algılayıcı düğümlerin, kısıtlı enerji kaynaklarına, yetersiz bellek kapasitesine ve sınırlı işlem kabiliyetlerine sahip olmaları, kolay erişilebilir ve fiziksel saldırılara açık alanlara yerleştirilmeleri ve insanlarla/ölçüm yapılacak fiziksel ortamla doğrudan etkileşimde bulunmalarından dolayı, KAA'larda doğrudan uygulanamamaktadır (Perrig ve diğ., 2004).

Genel olarak literatürde araştırmalara ve çalışmalara yön verilirken KAA'larda güvenlik konuları üç temel kategoride değerlendirilmektedir;

- KAA güvenliğindeki engeller,
- Güvenli KAA gereksinimleri,
- Saldırılar ve savunma önlemleri.

Bu bölümde, tez çalışmasında önerilen KAA güvenlik protokolünün güvenliğindeki engeller, gereksinimler ve karşılaşabileceği saldırı türleri ve savunma önlemleri hakkında bilgi verilmektedir.

3.2. KAA Güvenliğindeki Engeller

KAA'ların geleneksel ağlarla karşılaştırıldığında sınırlı kaynaklar, güvenli olmayan haberleşme ve gözetimsiz işlem gibi birçok kısıtlaması bulunmaktadır. Bunlar için güvenlik mekanizmaları geliştirirken, aşağıdaki alt bölümlerde de sunulan kısıtlamaların neler olduğunu değerlendirerek güncel güvenlik tekniklerinden faydalanılmalıdır (Dağlar ve Çayırcı, 2005).

3.2.1. Sınırlı kaynaklar

Yeni bir güvenlik yöntemi tasarlanırken KAA kaynaklarının (veri belleği, mikroişlemci, güç kaynağı) kapasitelerini göz önüne almak gerekmektedir. Ayrıca, bu kaynakların, küçük/sınırlı bir algılayıcı düğümde yer aldığı da unutulmamalıdır.

3.2.2. Güvenli olmayan haberleşme

Güvenli olmayan (açık) haberleşme, KAA veri güvenilirliği için başka bir tehdit unsurudur. KAA güvenliğinin, haberleşmede kullanılan tanımlanmış klasik protokoller yardımıyla temin edilmesi oldukça önemli zorlukları ve dezavantajları içermektedir. Bu nedenle günümüzde özellikle etkin yeni KAA güvenlik protokollerinin geliştirilmesi amacıyla gerçekleştirilen araştırmalar artan bir önem arz etmektedir.

3.2.3. Gözetimsiz işlem

Algılayıcı düğümler uzun zaman gözetimsiz bırakıldığından fiziksel saldırılara maruz kalabilmektedir. Uzaktan yönetildikleri için enerji durumları bilinemeyebilir ve ağ içerisinde diğer düğümlerle bağlantısının devam edip etmediğinin kontrolü mümkün olmayabilir. Tüm bu dezavantajlarına rağmen, merkezi bir yönetim noktasının bulunmaması ise KAA yaşam süresini arttırmaktadır (Xiao, 2007).

3.3. Güvenli KAA Gereksinimleri

KAA'lar geleneksel bilgisayar ağlarıyla bazı ortak özelliklere sahip olsalar da kendilerine özgü çeşitli gereksinimleri bulunmaktadır. Güvenlikle ilgili gereksinimler ve özellikler kısaca şöyle özetlenebilir:

- Veri gizliliği: Ağ güvenliğinin en önemli parçasıdır. Algılayıcı ağ, komşularına veri/bilgi sızdırmamalıdır. Algılayıcı bilgisinin (veri, kimlik, herkese açık anahtar), saldırılara karşı korunması için uygun bir yöntem ile şifrelenerek iletilmesi gerekmektedir.

- Veri bütünlüğü: İletişim esnasında çevresel şartlar ya da kötü amaçlı üçüncü düğümler nedeniyle, iletilen veri değişebilmekte dolayısıyla veri kayıpları meydana gelebilmektedir. Veri bütünlüğü özelliği ile iletişim esnasında kaynak düğümde elde edilen ilk verinin değiştirilmemesi garanti edilir. Bu işlem veri asıllama (kimlik denetimi) ile sağlanabilmektedir.
- Veri tazeliği: Algılayıcı ağlar anlık değişen verileri/ölçümleri algıladığı ve işlediği için sadece gizlilik ve güvenliğin sağlanması yeterli değildir. Aynı zamanda her mesajın tazeliğinin de garanti edilmesi gerekir. Verinin tazeliği, verinin yeni olduğunu belirtir; böylece kötü amaçlı üçüncü düğümlerin eski mesajları tekrar göndermediği de garanti edilir.
- Kullanılabilirlik: Algılayıcı ağın yaşam süresince işlevselliğini yitirmeden çalışması esastır. Muhtemel saldırılar, ek hesaplama ve ek haberleşme ihtiyacından kaynaklanan daha çok enerji tüketimi, algılayıcı düğümün kullanılırlığını azaltan etkenlerdir ve bunlara karşı alınacak önlemler KAA ömrünü beklenilen düzeyde tutmak için oldukça önemlidir.
- Kendini örgütleme (Self–organization): KAA farklı durumlara göre yeteri kadar kendini iyileştiren ve kendini örgütleyen esnek/bağımsız algılayıcı düğümlerden oluşan tipik bir eşe eş (Ad–Hoc) ağdır. Bu özelliğin, KAA ile ilgili yapılan tüm çalışmalarda ve uygulamalarda dikkate alınması zorunludur.
- Senkronizasyon: Algılayıcı ağ uygulamaları eş zamanlı çalışma (zaman senkronizasyonu) gerektirmektedir. Ayrıca düğümler, işbirliği içerisinde uygulamaları gerçekleştirebilmek için de senkronize çalışırlar. Güç kaybını en aza indirgemek amacıyla, algılayıcı düğümler bazı zaman periyotlarında ve birbirleriyle senkronizasyon içerisinde "uyku" konumunda çalışırlar (radyo alıcı-vericilerini kapatırlar).
- Güvenli yer belirleme: Algılayıcı düğüm, saldırı bulunan bir ortamda coğrafik konumunu doğru olarak belirleyebilir. Güvenli yer belirleme, güvenli coğrafik yönlendirme için ön koşuldur.
- Kimlik doğrulama: Alıcı düğümlerin gelen mesaj kaynağını/göndereni doğrulaması gerekmektedir. Kimlik doğrulama, alıcının mesajın gerçekten doğru kaynaktan geldiğini garanti etmesine olanak sağlar (Xiao, 2007, Perrig ve diğ., 2002).

3.4. Saldırılar ve Savunma Önlemleri

KAA düğümleri birçok durumda uzaktan izlenilen bir ortama rastgele yerleştirilmektedir. Bu nedenle uygulama alanında korunmasız olarak bulunan düğümler, çeşitli saldırılarla kolaylıkla ele geçirilebilir ve fiziksel hasara maruz bırakılabilir. Ayrıca, bunlar yeniden programlanarak saldırı üreten zararlı düğümler haline çevrilebilme riskini barındırırlar. Kullanım/kurulum kolaylıkları ve ucuz maliyetlerinin beraberinde getirdiği bu önemli dezavantajlar sebebiyle, kablosuz algılayıcı ağlar diğer ağlara nazaran oldukça az güvenilirdir (Karaboğa ve Ökdem, 2006).

KAA'ların güvenliğini tehdit eden hizmet engelleme (Denial of Service, DoS), Sybil, trafik analiz, düğüm kopyalama (Node Replication, NR), gizliliğin ortadan kaldırılması (Attacks Against Privacy, AAP) ve fiziksel olmak üzere birçok saldırı tipi bulunmaktadır. Alt bölümlerde bu saldırı tipleri ve savunma önlemlerine değinilmektedir (Pahtan ve diğ., 2006, Bandırmalı ve diğ., 2008a).

3.4.1. Hizmet engelleme (DoS) saldırıları ve savunma önlemleri

DoS tipi saldırılar, algılayıcı ağın beklenilen işlevinin azalmasına ya da tamamen kaybolmasına yol açan herhangi bir olay olarak adlandırılabilir. Donanım hataları, yazılım sorunları, kaynak tüketimi ve çevresel şartlar DoS tipi saldırıların temel kaynaklarıdır. DoS tipi saldırılarda genelde yazılım açıkları kullanılsa da algılayıcı ağ protokol ve tasarım açıkları önemli bir rol oynamaktadır.

KAA her biri farklı görevleri yerine getiren katmanlı bir protokol yapısına sahiptir. Bu katmanlara yapılabilecek saldırı türleri farklılık gösterir. Her katmanın güvenlik açıkları tanımlanarak bu açıklara dayalı olası DoS tipi saldırılara karşı çözüm önerileri Tablo 3.1'de verilmektedir (Wood ve Stankovic, 2002, Cakiroglu ve diğ., 2006).

Tablo 3.1: Hizmet engelleme (DoS) saldırıları ve savunma önlemleri.

KAA Katmanı	**Saldırı Türü**	**Savunma Önlemleri**
Ulaşım Katmanı	Sel (flooding)	İstemci bulmaca
	Senkronizasyon bozma (desynchronization)	Yetki
Ağ Katmanı	İhmal etme (neglect)	Fazlalık, araştırma
	Konumlama (homing)	Şifreleme
	Yanlış yönlendirme (misdirection)	Çıkış filtreleme, yetki, izleme
	Kara delik (black holes)	Yetki, izleme, fazlalık
Veri Bağı Katmanı	Çarpışma (collision)	Hata düzeltme kodu
	Tüketme (exhaustion)	Hız sınırlaması
	Haksızlık (unfairness)	Küçük çerçeveler
Fiziksel Katman	Bozma (jamming)	Yayılma spektrumu, öncelik mesajları, daha düşük görev çevrimi, bölge haritalama, mod değiştirme
	Kurcalama (tampering)	Kurcalama-kanıtlama, saklama

3.4.2. Sybil saldırısı ve savunma önlemleri

Sybil saldırısı, kötü niyetli bir düğümün istenmeyen bir şekilde birden fazla kimliğe sahip olmasıdır. Kötü niyetli bir düğümün ek kimlikleri, sybil düğüm olarak ifade edilmektedir. Sybil saldırıları, doğrudan ve dolaylı haberleşme, uydurulmuş ve çalınmış kimlikler, eş zamanlılık (simultaneity) olmak üzere üç şekilde sınıflandırılmaktadır.

Sybil saldırılarına karşı savunma yapmak için ağdaki kimliklerin gerçek düğümlere ait olup olmadığını saptayan bazı onaylama mekanizmalarına ihtiyaç vardır. Bu amaçla doğrudan onaylama ve dolaylı onaylama olmak üzere iki yöntem tanımlanmaktır. Doğrudan onaylamada düğüm kimliğinin güvenilir ve geçerli olup olmadığı doğrudan test edilir. Dolaylı onaylamada ise herhangi bir güvenilir düğüm tarafından, ağa dahil olan yeni düğümün kimlik geçerliliği test edilir.

Sybil saldırısına karşı savunma yöntemlerinden bir diğeri de "rastgele anahtar ön dağıtım" tekniği kullanmaktır. Bu tekniğin temelindeki fikir, sınırlı sayıda anahtarla bir anahtar döngüsü kurmaktır. Kimlikleri rastgele üreten bir düğüm birden çok kimliği almak için yeterli anahtara sahip olamayacaktır ve böylece şifreleme ve şifre çözme mesajları bulunmadığından ağdaki mesajları değiştirmesi olanaksız hale gelecektir (Newsome ve diğ., 2004).

3.4.3. Trafik analiz saldırıları ve savunma önlemleri

KAA'lar bir "çıkış düğümü/merkezi düğüm"le ("sink") haberleşebilen birçok düşük güçlü algılayıcılardan oluşmaktadır. Düğümler tarafından toplanan veri son olarak çıkış düğümüne/merkezi düğüme yönlendirilir. Saldırılar, ağı kullanılamaz duruma getirebilmek için çıkış düğümünü/merkezi düğümü etkisizleştirir (disable).

"Saldırı izleme oranı" (rate monitoring attack), çıkış düğümüne/merkezi düğüme yakın olan düğümlerin, uzak olan düğümlere nazaran daha çok paket iletme eğilimini gösterir. Saldırılarda, paket gönderen düğümler izlenilerek, en çok paket gönderen düğüm takip edilir. "Rastgele yürüme gönderme" tekniği kullanılarak saldırı izleme oranı düşürülebilir. Bu yöntemle, algılayıcının ebeveyn düğümden başka bir düğüme zaman zaman paket gönderilerek, saldırganın algılayıcıdan çıkış düğümüne/merkezi düğüme açık bir yol belirlemesi zorlaştırılır.

3.4.4. Düğüm kopyalama saldırıları ve savunma önlemleri

Bu saldırı türünde saldırgan, ağ içerisinde bulunan algılayıcı bir düğümün kimlik bilgisini (ID) kopyalayarak mevcut algılayıcı ağa düğüm eklemeye çalışır. Kopyalanmış bir düğüm, paketlerin bozulması veya yanlış yönlendirilmesiyle algılayıcı ağ performansını ciddi bir şekilde bozabilir. Eğer saldırgan tüm ağa fiziksel erişim kazanabilirse kopyalanmış algılayıcı düğüm için şifreleme anahtarlarını kopyalayabilir ve ağın stratejik noktalarına kopyalanmış düğümler ekleyebilir. Saldırgan, kopyalanmış düğüm ekleyerek ağın belirli bir bölümünü kolayca değiştirebilir hatta tüm ağ bağlantısını kesebilir.

Düğüm kopyalama saldırılarını belirlemek için “rastgele dağıtım çoklu gönderim” (randomized multicast) ve “seçilmiş hat çoklu gönderim” (line–selected multicast) olmak üzere iki algoritma kullanılmaktadır. Rastgele dağıtım çoklu gönderimde “düğüm yayım stratejisi”nden (node broadcasting strategy) faydalanılır. Düğüm yayım stratejisinde her bir algılayıcı tüm ağa doğrulanmış yayım mesajı dağıtır. Çelişkili veya kopyalanmış talepleri alan herhangi bir düğüm çelişkili düğümleri iptal eder.

Seçilmiş hat çoklu gönderim, rastgele dağıtım çoklu gönderim algoritmasının haberleşme maliyetini düşürmek için geliştirilmiştir. Bu yöntem, “söylenti yönlendirme” (rumor routing) esasına dayanır ve kopyalamayı tespit etmek için ağın topolojisini kullanır.

3.4.5. Gizliliğe karşı saldırılar ve savunma önlemleri

Algılayıcı ağ teknolojisi, küçük algılayıcı araçları belirli bir alana etkin şekilde yayarak, otomatik veri toplama yeteneklerinde büyük bir artış olacağını öngörür. Bununla birlikte, algılayıcı düğümler bu durumu kötüye de kullanabilir. Algılayıcı ağlar artan veri toplama kapasiteleri sağladığından, ilgi özellikle gizlilik problemleri üzerine yoğunlaşmaktadır. İstenmeyen (üçüncü) kullanıcılar birden çok algılayıcı girişleri arasındaki bağlantıyı bilirlerse, hassas/gizli bilgiyi elde etmek için görünüşte zararsız veriyi bile kullanabilirler. Bunun en tanınmış örneklerinden biri “panda–avcı problemi”dir. “Avcı” trafiği izleyerek “panda”ların pozisyonunu işaret edebilir. Algılayıcı ağlar uzaktan erişim ile büyük miktarda bilgiyi kolayca elde ettikleri için gizliliğin temini zorlaşır. İstenmeyen (üçüncü) düğümlerin izlemeyi sürdürmesi için fiziksel olarak ağ içerisinde bulunmaması gerekir. Uzaktan erişimle tek bir istenmeyen düğüm eş zamanlı olarak birçok bölgeyi izleyebilir. Gizliliği ortadan kaldırmak üzere, izleme ve gizli dinleme (eavesdropping), trafik analiz ve gizleme gibi saldırılar bulunmaktadır.

Gizliliğin ihlaline karşı, savunma önlemi olarak birkaç önemli yöntem bulunmaktadır. Bunlardan birincisi kaynağı bilinmeyen (anonymity) mekanizmalardır. Kesin konum bilgisi, kullanıcının tanımlanmasını ya da hareketinin

izlenmesini sağlayabilir. Bu durum gizlilik için bir tehlike oluşturur. Kaynağı belli olmayan mekanizmalar veri gönderilmeden önce veriyi kişiselleştiremez. İkincisi, plan (policy) temelli yaklaşımlardır. Erişim kontrol kararları ve kimlik doğrulama gizlilik politikalarının karakteristiklerine dayanır. Bir diğer savunma önlemi ise "bilgi seli"dir (flooding). Algılanan nesnelerin konum bilgileri yayıldığından, veri kaynağının konumunu izlemekten kaynaklanan bir dış saldırganı engellemek için anti–trafik analiz mekanizmaları önerilmektedir.

3.4.6. Fiziksel saldırılar ve savunma önlemleri

Algılayıcı ağlar genellikle saldırı hatlarında çalışır. Fiziksel saldırılar diğerlerinden farklı olarak algılayıcıları kalıcı şekilde yok eder. Örneğin, saldırılar şifrelenmiş sırları açığa çıkarabilir, donanımı bozabilir, algılayıcının programını değiştirebilir veya saldırganın kontrolü altındaki kötü amaçlı bir düğümle değiştirilebilir.

Fiziksel saldırılar, KAA'lar için gözetimsiz özellikleri ve sınırlı kaynaklarından dolayı büyük bir tehdit oluşturmaktadır. Algılayıcı düğümlerin çeşitli saldırılara karşı koruma geliştirmeleri için fiziksel donanımları güçlendirilebilir. Algılayıcıyı fiziksel saldırılara karşı koruyabilmek için rastgeleleştirilmiş saat sinyalinin eklenmesi, program sayıcının sınırlandırılması, güçlü düşük frekanslı algılayıcı kullanımı gibi önlemler alınabilir. Örneğin, rastgeleleştirilmiş saat sinyali ile gözlemlenen bir olay veya kritik işlemler arasına rastgele bir gecikme eklenerek fiziksel saldırı önlenebilir (Xiao, 2006, Bandırmalı ve diğ., 2007).

3.5. Sonuç

KAA'lar sınırlı bellek ve hesaplama kapasiteleri, kısıtlı güç kaynaklarına sahip olmaları, kolay erişilebilir alanlara yerleştirilmeleri ve insanlar ya da ölçüm yapılacak fiziksel ortam ile doğrudan etkileşimde bulunmaları sebebiyle birçok saldırıya açıktırlar.

Özellikle saldırı hatlarında çalışmaları öngörüldüğünden dolayı KAA'lar, başta fiziksel saldırılar olmak üzere, kötü niyetli düğümler tarafından düğümlerin ele

geçirilip kopyalanması, hizmet engellemelerine maruz kalmaları, ağ içerisindeki veri akışının analizi ve sahte kimliklere sahip sanal düğümlerin oluşturulması gibi birçok saldırı türüyle karşı karşıyadır. Sırasıyla bu saldırı türlerine karşı özel olarak geliştirilmiş rastgeleleştirilmiş saat sinyali, rastgele dağıtım çoklu gönderim, küçük çerçeveler kullanma, rastgele yürüme gönderme ve rastgele anahtar ön dağıtım yöntemlerinin kullanımı ile KAA'lar güvenli ve güvenilir olarak çalıştırılabilmektedir.

Bu bölümde KAA güvenliğindeki engeller, güvenli KAA gereksinimleri, saldırılar ve savunma önlemleri ayrıntılı olarak açıklanarak, geliştirilecek yeni KAA güvenlik protokolünün temelleri oluşturulmaktadır.

4. ŞİFRELEME YÖNTEMLERİ VE KAA VERİ BAĞI KATMANI GÜVENLİK PROTOKOLLERİNDE KULLANIMI

4.1. Giriş

Şifreleme, gizli ve yüksek öneme sahip verilerin çeşitli mantıksal ve matematiksel ifadeler kullanılarak okunamamasını sağlayan bir bilim dalıdır. Simetrik anahtar (gizli anahtar, secret key) ve asimetrik anahtar (açık anahtar, public key) olmak üzere iki temel şifreleme yöntemi mevcuttur. Simetrik anahtar şifreleme tekniğiyle geliştirilen algoritmaların güvenliği anahtar uzunluğuna bağlı olarak değişir. Bu algoritmalar oldukça hızlı çalışırlar. Asimetrik anahtar şifreleme tekniğiyle geliştirilen algoritmalar ise şifreleme ve şifre çözme için birbiriyle matematiksel olarak ilişkili iki farklı anahtar kullandıklarından dolayı güvenli fakat özellikle büyük veri yığınları için çok yavaştırlar (Menezes ve diğ., 1996).

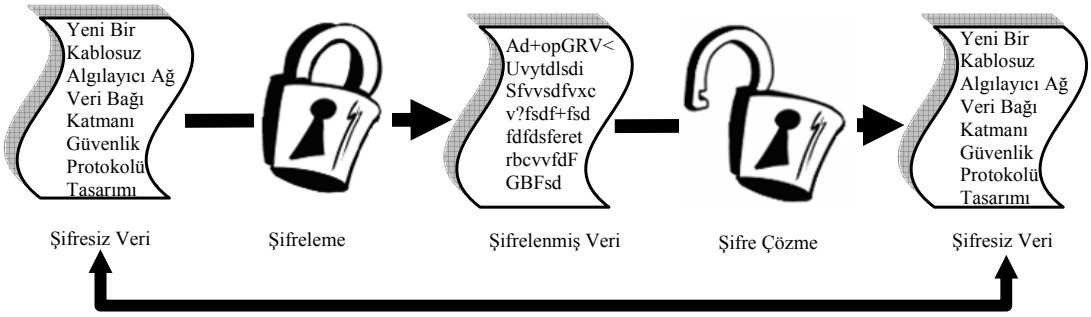
Şifreleme yöntemleri kullanılarak, kablolu ve kablosuz ağlar için literatürde geliştirilmiş birçok güvenlik protokolü bulunmaktadır. Fakat bu güvenlik protokollerini KAA'larda aynen kullanmak KAA'ların sınırlı kaynakları açısından uygun olmamaktadır. Bu sebeple, KAA'lar için yeni güvenlik protokolleri geliştirme çalışmaları sürdürülmektedir.

Bu bölümde, şifreleme yöntemleri ve mevcut KAA güvenlik protokolleri hakkında ayrıntılı bilgi verilerek, tez çalışmasında kullanılan şifreleme algoritması (SEA) ve veri güvenilirlik düzeyini arttırıcı yaklaşımlar (CTR ve CBC–MAC) detaylı olarak açıklanmaktadır.

4.2. Şifreleme Yöntemleri

Özel bir yöntem kullanmadan okunabilen ve anlaşılabilen veriye şifresiz veri (plaintext, cleartext), şifresiz verinin herhangi bir şekilde gizlenmesini sağlayan

metoda şifreleme (encryption), şifresiz verinin şifrelenmesiyle oluşan okunamayan anlamsız ifadelere ise şifrelenmiş veri (ciphertext) adı verilmektedir. Şifreli verinin orijinal şifresiz veri haline dönme işlemi ise şifre çözme (decryption) olarak adlandırılmaktadır (Şekil 4.1).



Şekil 4.1: Şifreleme ve Şifre Çözme.

Şifreleme ile hassas verinin güvenli bir şekilde saklanması ya da güvenli olmayan bir ağda iletilen verinin planlanan alıcılar dışında herhangi bir alıcı tarafından anlaşılamaması sağlanır.

Şifrelenmiş verinin güvenliği, büyük oranda şifreleme algoritmasının dayanıklılığı ve anahtarın gizliliğine bağlıdır (PGP, 2004).

Şifreleme yöntemlerinin başarımı şu ana kriterlere göre belirlenir:

- Kırılma süresinin uzunluğu,
- Şifreleme/şifre çözme işlemlerinde harcanan zaman (zaman karmaşıklığı),
- Şifreleme/şifre çözme işlemlerinde ihtiyaç duyulan bellek miktarı (bellek karmaşıklığı),
- Bu algoritmaya dayalı şifreleme uygulamalarının esnekliği,
- Bu uygulamaların dağıtımındaki kolaylık ya da algoritmaların standart hale getirilebilmesi ve
- Algoritmanın kurulacak sisteme uygunluğu (Tektaş ve diğ., 2003).

4.2.1. Simetrik anahtar şifreleme yöntemleri

Simetrik anahtar şifreleme yöntemlerinde, verinin şifrelenmesinde kullanılan anahtar ile şifrelenmiş verinin şifresinin çözülmesinde kullanılan anahtar aynıdır.

Bunlara örnek olarak DES (Data Encryption Standard, Veri Şifreleme Standardı), TDES (Triple DES, Üçlü DES), AES (Advanced Encryption Standard, İleri Şifreleme Standardı), SKIPJACK ve RC4 verilebilir. Simetrik anahtar şifreleme yöntemleri blok ve akış (stream) şifreleme yöntemleri olmak üzere ikiye ayrılmaktadır. Blok şifreleme yöntemleri, şifresiz (orijinal) veriyi veya şifreli veriyi bloklara bölerek şifreleme/şifre çözme işlemini yaparken, akış şifreleme yöntemleri, bir bit veya bayt üzerinde şifreleme/şifre çözme işlemi yapmaktadır.

Simetrik anahtar kullanan şifreleme algoritmaları teorik olarak, olası bütün anahtarları sırasıyla denemek yoluyla kırılabilir. Olası anahtarların denenmesi işlemi de, anahtarın uzunluğu arttıkça güçleşecektir. Buna örnek olarak Tablo 4.1'de belirli anahtar uzunluklarına göre şifreyi deneyerek bulma süreleri verilmiştir (Tektaş ve diğ., 2003).

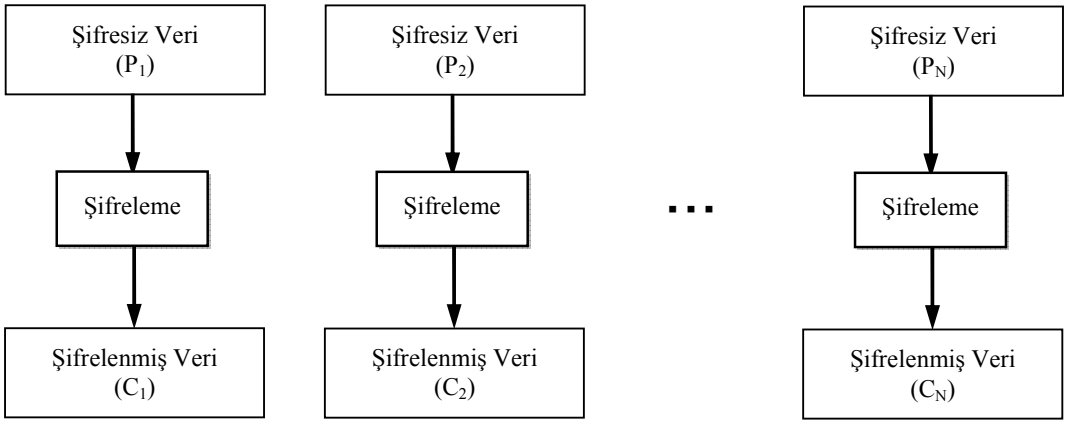
Tablo 4.1: Anahtar uzunluklarına göre şifre çözme süreleri.

Anahtar Uzunluğu	**Sayı Değeri**	**10^6 şifre/s**	**10^9 şifre/s**	**10^{12} şifre/s**
32 bit	$4x10^9$	36 dakika	2,16 saniye	2,16 milisaniye
40 bit	10^{12}	6 gün	9 dakika	1 saniye
56 bit	$7,2x10^{16}$	1142 yıl	1 yıl 2 ay	10 saat
64 bit	$1,8x10^{19}$	292000 yıl	292 yıl	3,5 ay
128 bit	$1,7x10^{38}$	$5,4x10^{24}$ yıl	$5,4x10^{21}$ yıl	$5,4x10^{18}$ yıl

4.2.1.1. Blok şifreleme yöntemleri

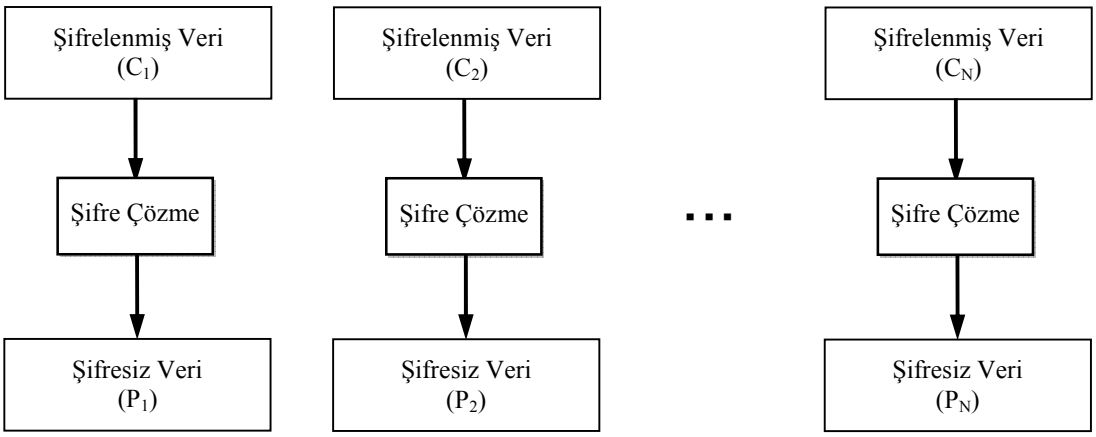
Blok şifreleme, şifresiz veriyi bloklara bölme, her bloğu şifreleyerek şifreli veri bloklarına dönüştürme ve bu şifreli blokları şifreli veri çıkışı olarak gruplandırma

şeklinde tanımlanabilmektedir. Şekil 4.2.a'da blok şifreleme sisteminin şifreleme, Şekil 4.2.b'de ise şifre çözme temel yapısı görülmektedir.



P: Şifresiz Veri (Plaintext)
C: Şifrelenmiş Veri (Ciphertext)
1,2, …, N: Veri Blok Sayısı

a)



b)

Şekil 4.2: Blok şifreleme sistemi a) Şifreleme b) Şifre çözme.

Bir blok şifreleme sistemi matematiksel olarak şöyle ifade edilmektedir:
$Z_2 = \{0,1\}$, $Z_2^n = Z_2 \times \ldots \times Z_2 = \{(x_{n-1}, \ldots, x_0) : x_i \in Z_2\}$ ve K ise anahtar uzayı olsun. $E: K \times Z_2^n \rightarrow Z_2$ ve her $k \in K$ için E(k,p) tersi alınabilir bir fonksiyondur. Bu fonksiyona şifreleme fonksiyonu denir. Blok şifreleme sistemi ile şifrelenmiş olan bir verinin şifresini çözmek için aynı sistemi şifreli veriye aynı anahtar ile uygulamak

gerekir. Şifreleme fonksiyonunun tersi uygulanarak şifre çözme işlemi yapılır. Şifre çözme fonksiyonu D(k,c) şeklinde gösterilir.

Blok şifreleme yöntemlerinin güvenilirliğini arttırmak için literatürde tanımlanmış olan çeşitli blok şifreleme yaklaşımları alt bölümde ayrıntılı bir şekilde açıklanmaktadır.

4.2.1.1.1. Blok şifreleme yaklaşımları

Blok şifreleme yöntemini farklı uygulamalarda kullanabilmek için, NIST (National Institute of Standards and Technology, Ulusal Standart ve Teknoloji Enstitüsü) tarafından güvenliği arttıran çeşitli şifreleme yaklaşımları tanımlanmaktadır. Bunlar üç temel kategoride sınıflandırılmaktadır:

- Veri gizliliğini arttıran yaklaşımlar,

i. ECB (Electronic Codebook, Elektronik Kod Kitabı)
ii. CBC (Cipher Block Chaining, Kapalı Metin Zincirleme)
iii. CFB (Cipher Feedback, Şifreyi Geri Besleme)
iv. OFB (Output Feedback, Çıktıyı Geri Besleme)
v. CTR (Counter, Sayaç)

- Veri asıllaması sağlayan yaklaşımlar,

i. CBC–MAC (Cipher Block Chaining Message Authentication Code, Kapalı Metin Zincirleme Mesaj Asıllama Kodu)
ii. OMAC/CMAC (One Key MAC/Cipher-based MAC, Tek Anahtarlı MAC)

- Veri gizliliğini arttıran ve aynı zamanda veri asıllaması sağlayan yaklaşımlar,

i. CCM (Counter with CBC–MAC, CBC–MAC ile Sayaç)
ii. GCM (Galois/Counter Mode, Galois/Sayaç Yaklaşımı)

4.2.1.1.1.1. ECB yaklaşımı

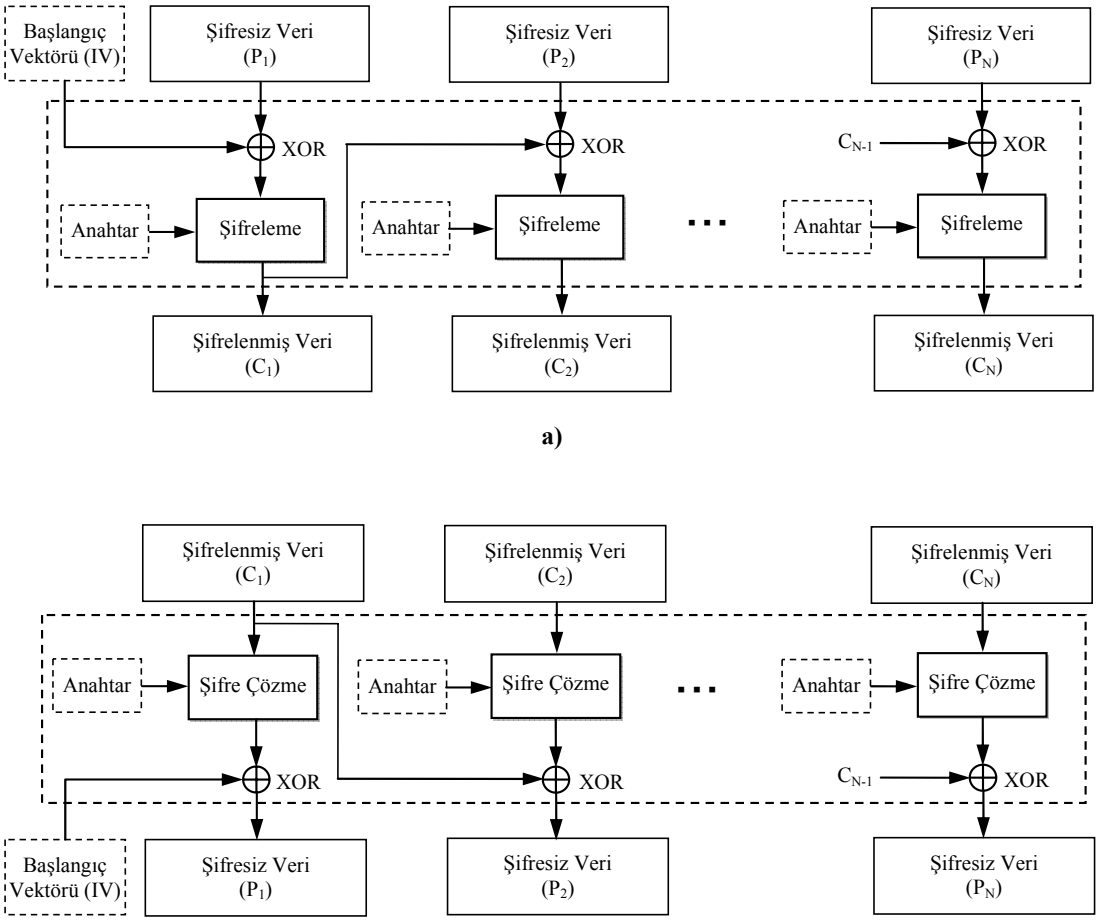
En basit yaklaşımdır. Bu yaklaşımda, şifresiz veri bloklara bölünür ve her blok aynı anahtarla şifrelenerek şifreli veri blokları elde edilir. Şifre çözme işleminde de şifreleme işleminde olduğu gibi her blok aynı anahtarla işleme girerek şifresiz veri blokları elde edilir. “Kod Kitabı” ifadesinin kullanılmasının nedeni, verilen bir

anahtara karşılık her bir şifresiz veri bloğu için tek bir şifreli veri bloğunun var olmasıdır. Yani her olası şifresiz veri bloğu kalıbına karşılık bir kayıt barındıran devasa bir kod kitabı hayal edilebilir.

ECB yaklaşımı, şifreleme anahtarı gibi küçük boyuttaki veriler için idealdir. Uzun boyuttaki veriler için ECB yaklaşımının kullanılması güvenli olmayabilir. Eğer veri biçimli bir yapıya sahipse, şifre analizcisi için bu düzeni açığa çıkarmak mümkündür. Örneğin, verinin hep aynı önceden belirlenmiş kalıplarla başladığı biliniyorsa o zaman şifre analizcisi üzerinde çalışabileceği birçok şifreli/şifresiz veri çiftine sahip olabilir, bu durum özellikle uzun boyuttaki verilerin şifre çözümünü kolaylaştırır.

4.2.1.1.1.2. CBC yaklaşımı

ECB yaklaşımındaki güvenlik eksiklerini giderebilmek için geliştirilen bir yaklaşımdır. Şekil 4.3.a'da CBC yaklaşımıyla veri şifreleme işlemi görülmektedir. Şifreleme algoritmasının girdisini, o an ki şifresiz veri bloğu ile önceki şifreli veri bloğunun XOR'lanması oluşturmaktadır. Her blok için aynı anahtar kullanılmaktadır. Ardışık şifresiz veri blokları sanki birbirine zincirlenmiş gibidir. Bu modelde, her bir şifresiz veri bloğu için, şifreleme fonksiyonunun girişinin, şifresiz veri bloğu ile doğrudan bir bağlantısı bulunmamaktadır. Böylece şifresiz veri bloklarının gözlenmesi mümkün değildir. Şekil 4.3.b'de ise CBC yaklaşımıyla şifre çözme işlemi görülmektedir. Her bir şifreli veri bloğu şifre çözme algoritmasından geçirilir ve elde edilen sonuç, önceki şifreli veri bloğu ile XOR'lanır ve şifresiz veri bloğu elde edilir.



Şekil 4.3: CBC yaklaşımı a) Şifreleme b)Şifre çözme.

Şifreli verinin ilk bloğunu oluşturmak için bir başlangıç vektörü (Initial Vector, IV) kullanılmaktadır. IV, şifresiz verinin ilk bloğu ile XOR'lanır. Şifre çözme işleminde ise, şifresiz verinin ilk bloğunu elde etmek için IV, şifre çözme algoritmasının çıktısı ile XOR'lanır.

IV, şifre bloğu ile aynı uzunluktaki bir veri bloğudur. Şifrelemenin farklılığını sağlamak için kullanılır. Yani aynı şifresiz veri iki kez şifrelendiğinde farklı iki şifreli veri üretmeyi sağlamaktadır. Sadece gönderici ve alıcı tarafından bilinmesi gereken bir bilgidir. En az şifreleme anahtarı kadar iyi korunmalıdır.

Bu yaklaşım, tez çalışmamı karşılaştırdığım TinySEC ve LLSP veri bağı katmanı güvenlik protokollerindeki veri gizliliğini arttırmak için kullanılmaktadır.

4.2.1.1.1.3. CFB yaklaşımı

Bu yaklaşımda, CBC yaklaşımında olduğu gibi şifresiz verinin parçaları birbirine zincirlenir öyle ki herhangi bir şifresiz veri biriminin şifreli verisi o ana kadar olan tüm şifresiz verinin bir fonksiyonudur.

CFB ya da OFB yaklaşımı kullanılarak blok şifreleme akış şifrelemeye dönüştürülebilir. Akış şifreleme, veriyi blok uzunluğunun tam katı olacak şekilde doldurma işlemine ihtiyaç duymaz ve gerçek zamanlı çalışabilir.

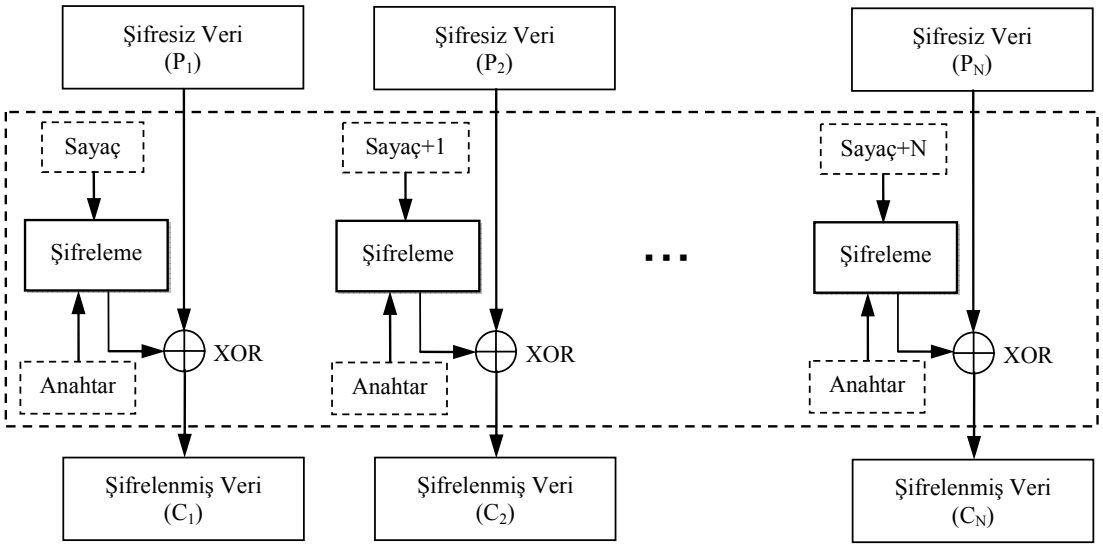
4.2.1.1.1.4. OFB yaklaşımı

OFB yaklaşımı, yapısal olarak CFB yaklaşımına benzemektedir. CFB yaklaşımında, kaydırmalı yazmaca (shift register) geri beslenen, şifreli veri birimi iken; OFB yaklaşımında, şifreleme fonksiyonunun çıkışıdır.

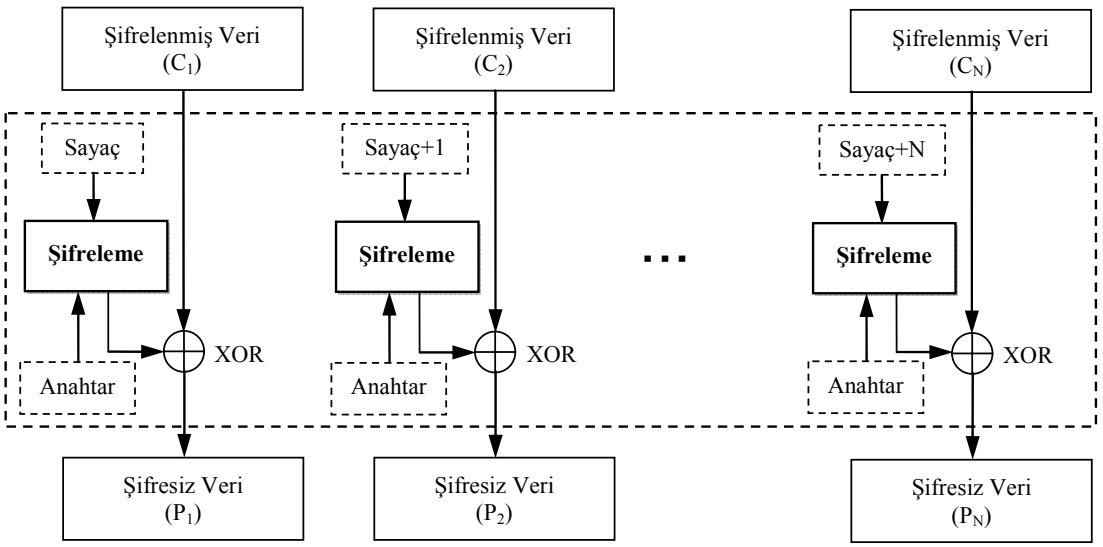
OFB yaklaşımının bir üstünlüğü, iletimde bit hatalarının yayılmamasıdır. Örneğin, şifrelenmiş birinci veri bloğunda bit hatası meydana gelirse, sadece elde edilen şifresiz birinci veri bloğu bundan etkilenir, sonraki şifresiz veri blokları kesintiye uğramaz. Fakat CFB yaklaşımında, şifrelenmiş birinci veri bloğu aynı zamanda kaydırmalı yazmacın girişi olduğundan, kesintiye sebep olur (Sakallı, 2006, Başkök, 2007, Yerlikaya, 2006).

4.2.1.1.1.5. CTR yaklaşımı

Şekil 4.4.a'da CTR yaklaşımıyla şifreleme işlemi görülmektedir. Şifreleme algoritmasının girdisini, sayaç değeri oluşturmaktadır. Şifrelenmiş veri bloğu, şifrelenen sayaç değeri ve şifresiz veri bloğunun XOR'lanması ile elde edilmektedir. Şekil 4.4.b'de ise CTR yaklaşımıyla şifre çözme işlemi görülmektedir. CTR yaklaşımı ile şifre çözme işleminde en dikkat edilmesi gereken işlem, sistemde şifre çözme algoritması yerine şifreleme algoritmasının kullanılmasıdır. Yine sayaç şifreleme algoritması ile şifrelenerek şifrelenmiş veri bloğuyla XOR'lanıp şifresiz veri bloğu elde edilmektedir.



a)



b)

Şekil 4.4: CTR yaklaşımı a) Şifreleme b)Şifre çözme.

CTR yaklaşımında da OFB ve CFB yaklaşımlarındaki gibi blok şifreleme akış şifrelemeye dönüştürülebilir. Sayaç değeri en basit ve en çok kullanılan şekli ile 1'den N'e kadar bir değer olabildiği gibi uzun bir süre tekrarlanmayacağı garanti edilen sıralı dizi üreten herhangi bir fonksiyon da olabilir.

Diğer yaklaşımlarla karşılaştırıldığında CTR yaklaşımının üstünlüklerinden biri yüksek hızlı uygulama sağlamasıdır. CTR, tam olarak paralelize edilebilmekte ve

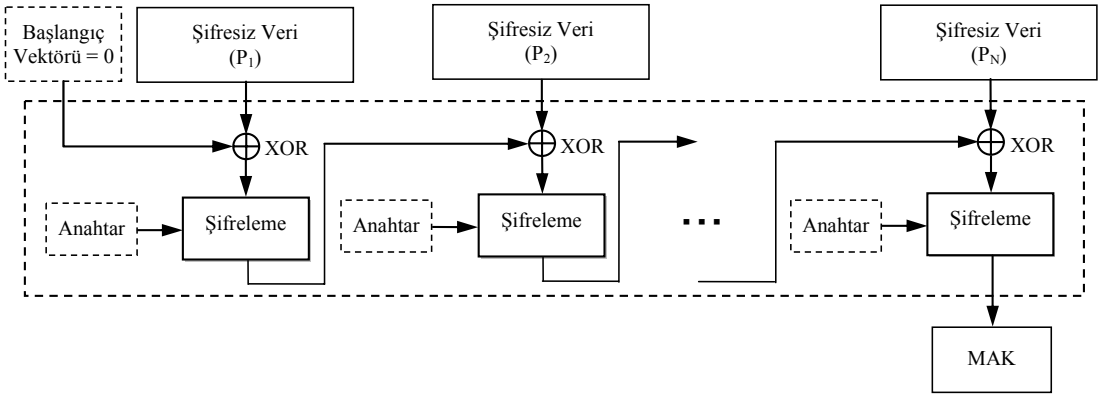
hızı arttırmak için ön işlem (pre-processing) yapılabilmektedir. Diğer üstünlükleri ise, düşük hata yayılma oranına sahip olması ve isteğe bağlı değişik (arbitrary) veri uzunluklarında kullanılmasıdır. Tüm bunları güvenlikten ödün vermeden yapmaktadır (Tirtea ve Deconinck, 2004). CTR yaklaşımının CBC, CFB ve OFB yaklaşımlarından farklı olarak bir önceki blokla hiçbir bağlantısı yoktur. Böylece bir blokta meydana gelebilecek hataların yayılması önlenmeye çalışılmaktadır.

CTR yaklaşımı, sunulan tezde önerilen YP içerisinde veri gizliliğini arttırmak için kullanılmaktadır.

4.2.1.1.1.6. CBC–MAC yaklaşımı

CBC–MAC yaklaşımı, bir blok şifreleme algoritması ve bir anahtar ile şifresiz veriden Mesaj Asıllama Kodu (MAK) elde edilmesini sağlayan bir yöntemdir. Bu yöntemde, Şekil 4.5'de görüldüğü gibi şifresiz veri; CBC yaklaşımı, anahtar ve blok şifreleme algoritması kullanılarak ardışık şekilde şifrelenir, elde edilen en son bloğun istenen boyuttaki (16, 32, 64…-bit) en önemli bitleri MAK olarak seçilip alıcısına gönderilecek olan pakete eklenir. Burada, başlangıç vektörünün değeri "0" olarak belirlenmektedir.

CBC–MAC oldukça etkili, hızlı ve sabit uzunluktaki veri büyüklüklerinde güvenlidir (Bellare ve diğ., 2000).



Şekil 4.5: CBC–MAC yaklaşımı.

CBC–MAC yaklaşımı, sunulan tezde önerilen YP içerisinde veri asıllaması/bütünlüğü sağlamak için kullanılmaktadır.

4.2.1.1.1.7. CMAC yaklaşımı

CMAC yaklaşımı, CBC–MAC gibi blok şifreleyici temelli bir mesaj asıllama kodu yöntemidir. CBC–MAC, sadece sabit uzunluktaki veri büyüklüklerinde güvenli olduğu ve değişik uzunluktaki veri büyüklüklerinde güvenlik sağlayamadığından dolayı yeni bir asıllama kodu yaklaşımı olarak CMAC geliştirilmiştir. CMAC, OMAC1 (One–Key CBC–MAC1) ile eş değerdedir.

Bu yaklaşımda, anahtar değerinden k_1 ve k_2 olmak üzere iki tane alt anahtar türetilir. Veri sabit blok uzunluklarına bölündükten sonra elde edilen son blok tam bir blok ise k_1 anahtarı ile değilse k_2 anahtarı ile şifreleme işlemi yapılır. Ayrıca son blok tam blok değilse son bloğa "0"'lar ile doldurma işlemi uygulanmaktadır (Iwata ve Kurosawa, 2003).

4.2.1.1.1.8. CCM yaklaşımı

CCM, hem veri asıllaması hem de gizlilik sağlamak için tasarlanan bir asıllanmış şifreleme yaklaşımıdır. CCM yaklaşımı, sadece 128-bit blok büyüklüğündeki blok şifreleyiciler için tanımlanmıştır. Özellikle AES algoritması ile birlikte kullanılmaktadır. Bu yaklaşıma örnek olarak AES–CCMP (Advanced Encryption Standard–Counter Mode with Cipher Block Chaining Mode Protocol) verilebilir.

CCM yaklaşımında, şifreleme CTR yaklaşımıyla, asıllama işlemi ise CBC–MAC yaklaşımıyla yapılmaktadır. Her iki durumda da aynı şifreleme anahtarı kullanılmaktadır. Sistemin güvenliği, CTR ve CBC–MAC yaklaşımlarının birleşimiyle birlikte kullanılan blok şifreleme algoritmasına da dayanmaktadır (Dworkin, 2004).

4.2.1.1.1.9. GCM yaklaşımı

GCM, CCM'de olduğu gibi hem asıllama hem de gizlilik sağlamak için tasarlanan bir asıllanmış şifreleme yaklaşımıdır. GCM yaklaşımı da 128-bit blok büyüklüğüne sahip blok şifreleyiciler için tanımlanmaktadır.

GCM yaklaşımı, şifreleme için CTR, asıllama için yeni Galois yaklaşımlarının bir arada kullanılmasından oluşmaktadır. Bu yaklaşımdaki temel özellik, asıllama için kullanılan Galois alanı hesabıdır. Kullanılan $GF(2^{128})$ alanı, $x^{128}+x^7+x^2+x^1+1$ polinomuyla tanımlanmaktadır. GCM yaklaşımı, IEEE 802.1AE (MACsec) Ethernet güvenliği, ANSI (INCITS) Fibre Channel Güvenlik Protokolleri (FC-SP), IEEE P1619.1 manyetik bant depolama ve IETF IPsec standardında kullanılmaktadır (Dworkin, 2004).

4.2.1.1.2. Blok şifreleme algoritmaları

Blok şifreleme algoritmalarına, DES, TDES, AES, SKIPJACK ve SEA örnek olarak verilebilir. Blok şifreleme algoritmalarının gücünün tanımlanmasında, kullanılan S kutuları, döngü sayısı, anahtarların XOR işlemine sokulması, blok uzunluğu, anahtarın uzunluğu ve özelliği büyük önem taşımaktadır. Ayrıca algoritmaya yapılan saldırılara karşı dayanıklılıkta, günümüz algoritmalarının gücünün ölçülmesinde önemli bir kıstastır.

Blok şifreleme algoritmalarının, karıştırma (confusion) ve yayılma (diffusion) teknikleriyle dayanıklılığı arttırılmaktadır. Karıştırma, şifresiz ve şifreli veri arasındaki ilişkiyi gizlemeyi amaçlarken, yayılma, şifresiz verideki izlerin şifreli veride sezilmemesini sağlamak için kullanılmaktadır. Karıştırma ve yayılma, sırasıyla yer değiştirme ve lineer dönüşüm (transformasyon) işlemleri ile gerçeklenmektedir. Feistel ağları ve yer değiştirme permütasyon ağları olmak üzere iki ana blok şifreleme mimarisi bulunmaktadır. Her ikisi de yer değiştirme ve lineer dönüşüm kullanmaktadır. Ayrıca her iki mimari birden fazla şifreleme işleminin birleşmesi ile oluşturulur. Her şifreleme adımına döngü denir. Genellikle her döngüde anahtar materyali kullanılmaktadır.

Blok şifreleme algoritmalarının önemli özellikleri aşağıda yer almaktadır:

- Anahtar: Blok şifreleme algoritmalarında anahtarın uzunluğu ya da bit sayısı en temel saldırı olan geniş anahtar arama saldırısına karşı güçlü olmalıdır. Örneğin DES algoritması 56-bit anahtar kullanırken AES algoritması DES'in bu zaafını örter niteliktedir ve 128, 192, 256-bit anahtar seçenekleri sunmaktadır. Ayrıca anahtarın rastlantısal olması gerekmektedir.
- Döngü Sayısı: Blok şifreleme algoritmalarında döngü sayısı iyi seçilmek zorundadır. Çünkü lineer dönüşüm ve yer değiştirmelerin bu seçilen değerle algoritmaya yeterli gücü vermesi gerekmektedir. Ayrıca yapılan saldırıların başarısız olması için en önemli şartlardan birisidir.
- S kutuları: S kutuları bir blok şifreleme algoritmasının en önemli ana elemanıdır. Çünkü algoritmadaki tek doğrusal olmayan yapıdır ve dolayısıyla algoritmaya güç vermektedir. S kutuları için üç önemli nokta vardır. Bunlar;

i. SAC (Strict Avalanche Criteria): 1 bit giriş değişimi sonucunda her çıkış bitinin değişme olasılığı ½ olur.
ii. S kutularının genişliği: Şifre analiz saldırıları düşünüldüğünde büyük bir kutu küçüğüne oranla daha iyi olacaktır. Ayrıca diferansiyel saldırılardan korunmak için büyük sayıda çıkış bitleri ve lineer saldırılardan korunmak için büyük sayıda giriş bitleri gereklidir.
iii. S kutusu gereksinimleri: Çıkışların dağılımları saldırılara karşı kontrol edilmeli, çıkışlar girişe göre lineer olmamalı, S kutusunun her sırasındaki değerler tek olmalıdır (Şahin ve diğ., 2005).

Blok şifreleme algoritmalarına yönelik saldırı tipleri aşağıda tanımlanmaktadır:

- Temel Saldırılar: Blok uzunluğu n bit olan ve k bit anahtar uzunluğuna sahip bir blok şifreleme algoritması için en temel saldırılardan biri "sözlük" saldırısıdır. Bu saldırıda k bitlik anahtarı kullanan saldırgan bir şifresiz veriyi mümkün olan 2k anahtarla şifreler ve şifreli verileri sıralı bir sözlükte tutar. Daha sonra gizli anahtarla şifrelenmiş seçilmiş bir şifresiz veri elde eder ve uygun eşleşmeyi sözlükten kontrol eder. Bir diğer saldırı türü olan "kod kitabı" saldırısında, saldırgan mümkün olan şifresiz veri bloklarından gizli bir anahtar ile şifrelenmiş veri blokları elde eder ve bunları bir tabloya (kod kitabı) depolar. Ele geçirdiği bir şifreli veriyi, önceden elde ettiği tablo ile karşılaştırarak şifresiz veriyi

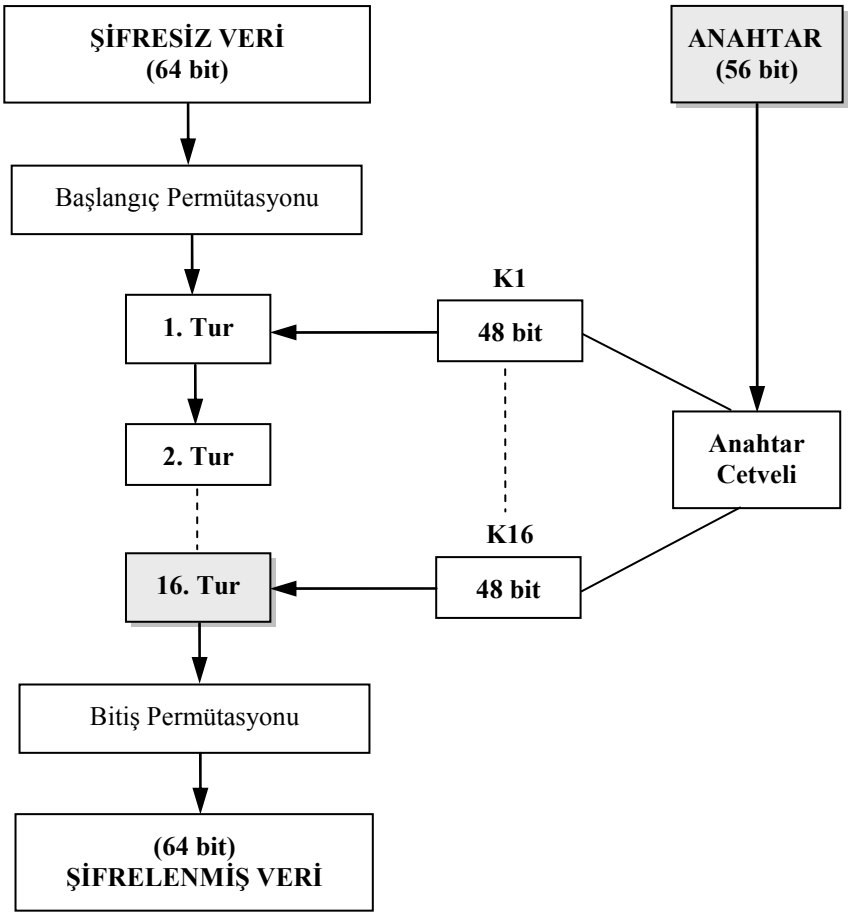
bulmaya çalışır. “Geniş anahtar arama” saldırısında ise 2k olası anahtar denenerek şifreli veriden anlamlı bir şifresiz veri elde edilince saldırı tamamlanır.

- Gelişmiş Saldırılar: Bu tür saldırılar matematiksel bir fikrin kullanılması ile geliştirilmişlerdir. Bu saldırılara örnek olarak lineer şifre analiz, diferansiyel şifre analiz, kesik diferansiyel şifre analiz, imkansız diferansiyel şifre analiz verilebilir (Şahin ve diğ., 2005).

4.2.1.1.2.1. DES şifreleme algoritması

Şekil 4.6’da görülen DES şifreleme algoritması, birçok çalışmada referans olarak gösterilen ilk simetrik anahtar blok şifreleme algoritmasıdır. Bu algoritma, 64 bitlik veri bloklarını, 64 bitlik bir anahtarın 8. ve katları (8, 16, 24, 32, 40, 48, 56 ve 64) bitlerini göz ardı ederek, 56 bitlik anahtar yardımıyla şifrelemektedir.

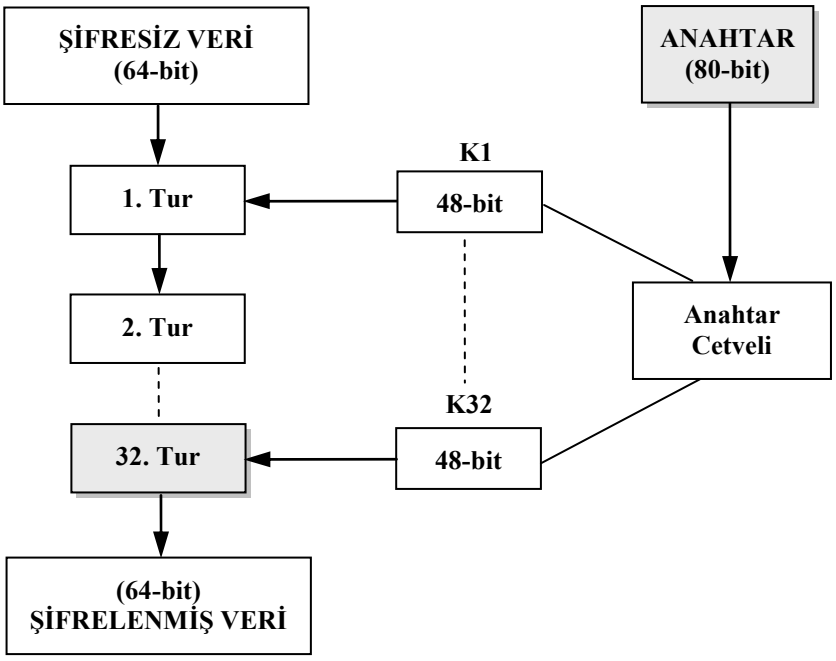
KAA’larda kullanılması uygun olarak görülmekle birlikte aşırı kod uzunluğu, enerji tüketimini ve bellek kullanımını, uygulamalar tarafından kabul edilemez miktarlarda arttırmaktadır. 16 döngü sonunda şifrelenmiş verinin elde edilmesi, algoritmayı güçlü yapmakla birlikte anahtar uzunluğunun küçük olması güvenliğini azaltmaktadır (Sakallı ve Buluş, 2002).



Şekil 4.6: DES şifreleme algoritması.

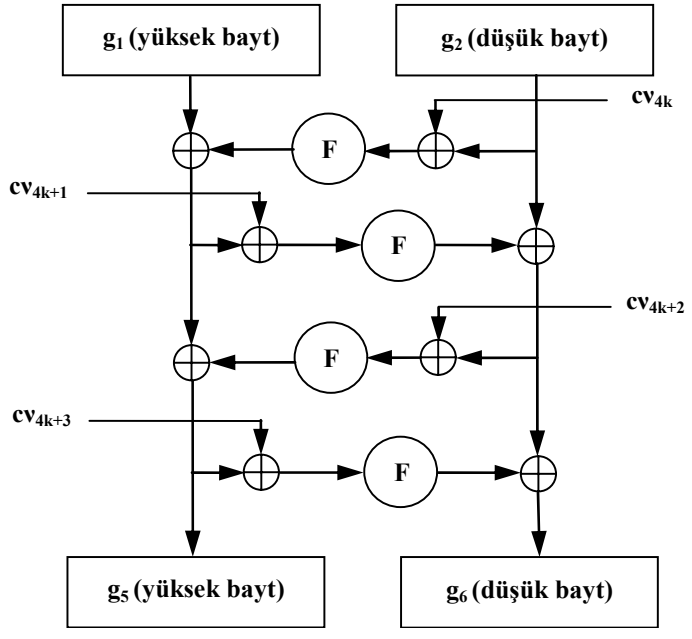
4.2.1.1.2.2. SKIPJACK şifreleme algoritması

1987 yılında geliştirilen SKIPJACK şifreleme algoritması, 1993'te kullanılmaya başlanmış bir simetrik anahtar blok şifreleme algoritmasıdır. Şekil 4.7'de görülen algoritmanın akış şemasından da anlaşılacağı üzere, 64 bitlik şifresiz veri blokları 80 bitlik anahtar kullanarak 32 döngü sonunda şifrelenerek, şifrelenmiş veri blokları elde edilmektedir. Döngü sayısı arttıkça algoritmanın güvenliği de üssel olarak artmaktadır. SKIPJACK, gizli olarak tutulması gereken (hassas) her türlü veriyi şifrelemek için kullanılan güvenli bir algoritmadır (Brickell ve diğ., 1993).



Şekil 4.7: SKIPJACK şifreleme algoritması.

Kural A ve Kural B olarak adlandırılan iki yöntemin arka arkaya yinelenerek (8 defa Kural A, 8 defa Kural B ve arkasından tekrar 8 defa Kural A, 8 defa Kural B) çalışmasıyla 32 döngü sonunda şifrelenmiş veri elde edilir. Her döngü, Şekil 4.8'de gösterilen lineer olmayan anahtarlanmış G permütasyonun eklenmesiyle bir lineer geri beslemeli kaymalı kaydedici şeklinde tanımlanır.



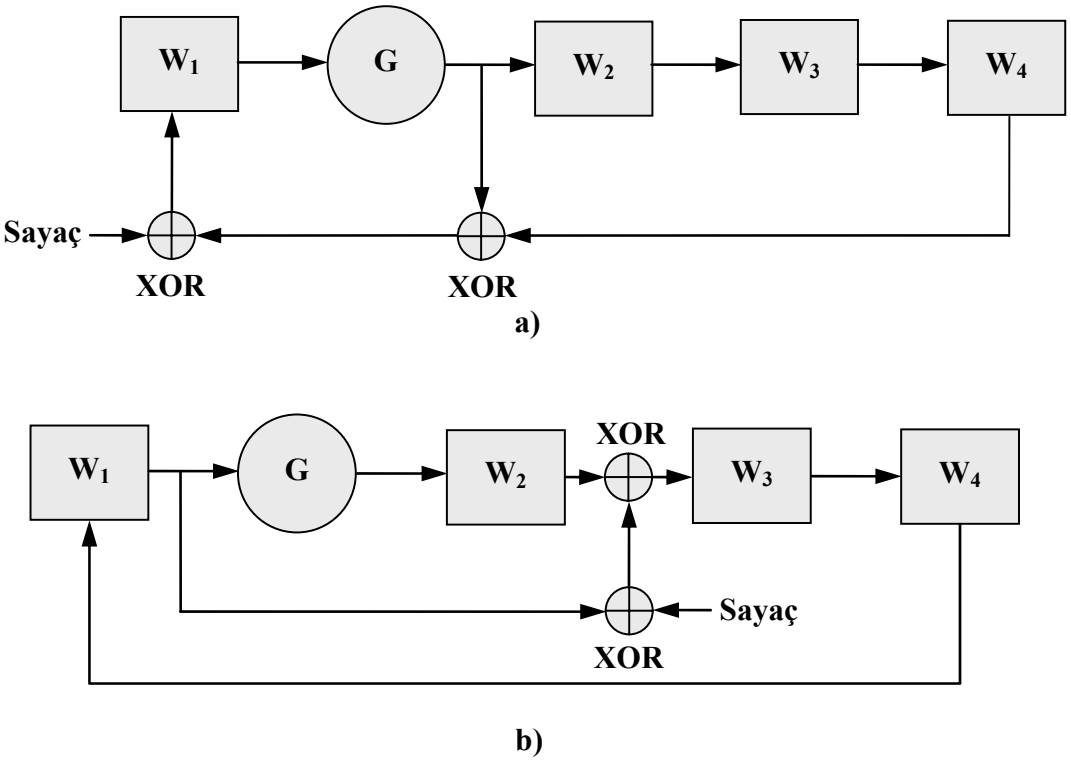
Şekil 4.8: G permütasyonu.

Şekil 4.9.a'da, Kural A ve Şekil 4.9.b'de Kural B'nin tanımlamaları görülmektedir. Kural B, temelde Kural A'nın tersidir. Sayaç, döngü sayısını göstermektedir. G, F fonksiyonu (8x8 bit S kutusu, F tablosu olarak adlandırılır) olarak tanımlanan 4 döngüden oluşan Feistel permütasyonudur ve G'nin her döngüsünde anahtarın 8 biti kullanılır (Brickell ve diğ., 1993, NIST, 1998).

Kural A aşağıdaki şekilde çalışır:

- G, w1 (32-bit)'i değiştirir (permutes).
- Yeni w1, G çıkışı, sayaç ve w4 (32-bit)'ün XOR'lanmasından oluşur.
- w2 (32-bit) ve w3 (32-bit) değerleri 32-bit sağa kaydırılır. Yani w2'i w3 olur, w3'de w4 olur.
- Yeni w2, G çıkışıdır.
- Sayaç bir arttırılır.

Kural B'de Kural A'ya benzer olarak çalışır (NIST, 1998).



Şekil 4.9: SKIPJACK şifreleme algoritması a) Kural A b) Kural B.

4.2.1.1.2.3. AES şifreleme algoritması

AES şifreleme algoritması, mevcut standart olan DES şifreleme algoritmasının gelişen teknoloji ve artan işlemci hızları karşısında güvenilirliğini yitirmeye başlaması sonucu ortaya çıkmış bir algoritmadır. Eylül 1997'de NIST tarafından geliştirilecek yeni algoritma için çağrıda bulunulmuş ve uzun değerlendirme çalışmaları sonucunda Kasım 2000'de beş finalistten Rijndael algoritmasının AES olarak adlandırılmasına karar verilmiştir.

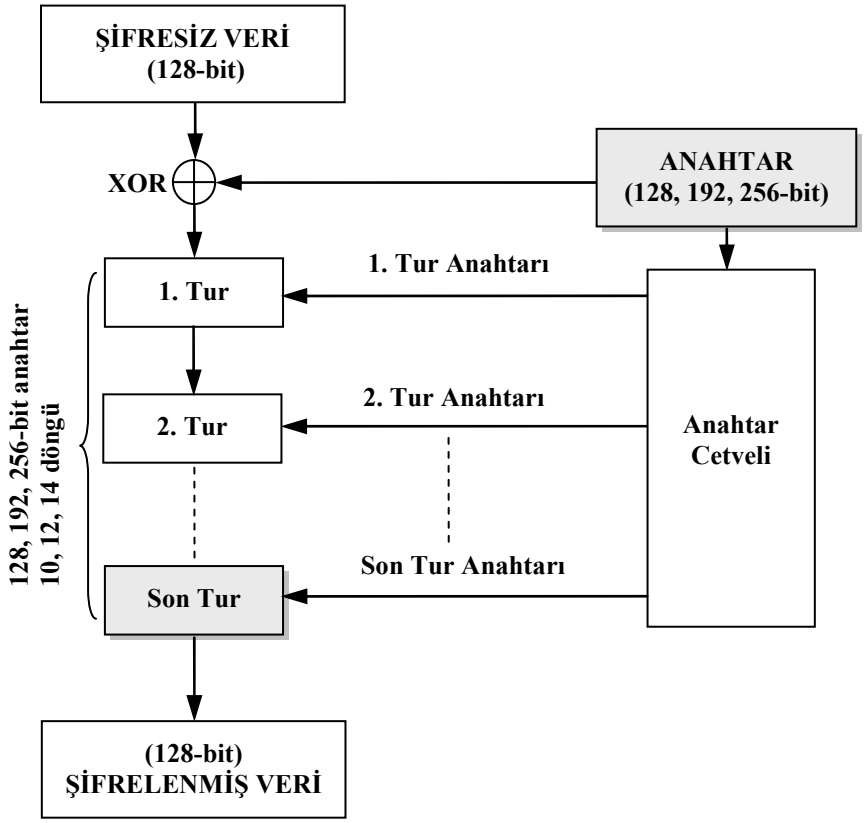
AES algoritması, 128 bitlik veri bloklarını 128, 192, 256-bit anahtar seçenekleri ile şifreleyen bir simetrik anahtar blok şifreleme algoritmasıdır. Algoritmadaki döngü sayısı anahtar uzunluğuna bağlıdır. 128, 192, 256 bitlik anahtarlar ile sırasıyla 10, 12, 14 döngü sonunda şifrelenmiş veri elde edilmektedir.

AES algoritmasında, son döngü hariç, her döngü dört katmandan oluşmaktadır ve her döngüde sırasıyla;

- Baytların yer değiştirmesi,
- Satırların ötelenmesi,
- Sütunların karıştırılması,
- Döngü anahtarı ile XOR'lama

işlemleri yapılmaktadır.

Son döngüde; sütunları karıştırma katmanı atlanır, ancak diğer katmanların uygulanmasında bir değişiklik söz konusu değildir. Şekil 4.10'da AES şifreleme algoritmasının yapısı görülmektedir (Başkök, 2007).



Şekil 4.10: AES şifreleme algoritması.

4.2.1.1.2.4. SEA şifreleme algoritması

SEA şifreleme algoritması, algılayıcılar gibi sınırlı kaynaklara sahip ortamlarda kullanılmak için geliştirilen simetrik anahtar blok şifreleme algoritmasıdır. 2006 yılında Belçikalı bir grup tarafından geliştirilen ve literatüre giren bu uygulamanın üzerindeki çalışmalar halen devam etmektedir. Sınırlı kaynaklarda kullanılmak için geliştirilen simetrik anahtar şifreleme algoritmaları üzerine hâlihazırda yapılmış çok fazla çalışma bulunmamaktadır. Genellikle literatürde olan ve her alanda kullanılan şifreleme algoritmaları sınırlı kaynaklarda da kullanılmaya çalışılmaktadır. Bu durumda da sınırlı kaynaklar için hayati öneme sahip olan enerji yükü artmakta ve başarım düşmektedir. Sınırlı kaynaklara sahip olan araçları en iyi ve en güvenli şekilde kullanmak için şifreleme algoritmaları üzerine son yıllarda birçok çalışma sürdürülmektedir. Şekil 4.11'de algoritması görülen SEA, kaynakları en etkin şekilde ve güvenliği de göz önüne alarak geliştirilen en yeni simetrik anahtar blok şifreleme algoritmalarından birisidir.

Sınırlı kaynaklara sahip cihazlarda kullanmak için geliştirilecek algoritmaların örneğin düşük bellek gereksinimi, küçük kod büyüklüğü ve sınırlı komut seti gibi belirli tasarım ölçütleri olması gerekmektedir. SEA algoritmasının bütün bu tasarım ölçütlerini sağlamasının yanında ek olarak esnek bir yapıya sahip olması da amaçlanmıştır. SEA algoritması, sınırlı kaynaklara sahip ağlarda düşük maliyetli şifreleme ve kimlik denetimi için iyi bir çözüm sunmaktadır (Standaert ve diğ., 2006).

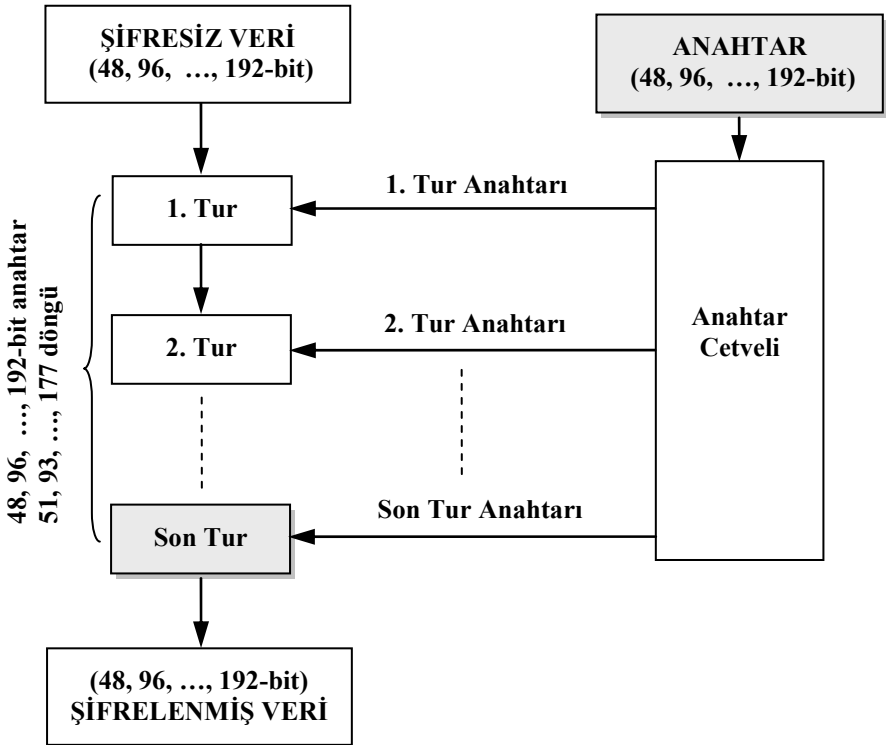
SEA şifreleme algoritmasında;

- Bit-bit XOR,
- Yerine koyma işlemi (S kutuları),
- Kelime (word) sağa ve sola döndürme,
- Bit döndürme,
- Mod 2b toplama

işlemleri yapılmaktadır.

SEA algoritmasının en önemli özelliklerinden biri, şifresiz veri ve anahtar büyüklüğünün esnek olmasıdır. Örneğin 8 bit işlemci kullanarak 48, 96,144,…-bit blok şifreleyiciler türetilebilir. Şifresiz veri ve anahtar büyüklüğü 6'nın katı olmalıdır.

$SEA_{n,b}$ şeklinde ifade edilmektedir.
n: şifresiz veri ve anahtar büyüklüğü
b: kelime büyüklüğü
$n_b = n/2b$: Feistel dallanması başına kelime sayısı
n_r: Şifreleme tur sayısı
Uygun bir güvenlik düzeyinin sağlanabilmesi için kelime uzunluğunun $b \geq 8$ ve tur sayısının en az $n_r = 3n/4+2(n_b+[b/2])$ olması gerekmektedir.



Şekil 4.11: SEA şifreleme algoritması.

SEA algoritmasının güvenlik analizleri yapılmış, lineer ve diferansiyel, çoklu lineer, bumerang, kare ataklar, kesilmiş ve imkânsız diferansiyeller, ara değer kestirimi atakları, ilgili anahtar atakları, cebirsel ataklar gibi şifre analiz yöntemlerine karşı kanıtlanabilir derece de dirençli olduğu tespit edilmiştir (Standaert ve diğ., 2006, Macé ve diğ., 2008).

4.2.1.1.2.5. SKIPJACK, AES ve SEA şifreleme algoritmalarının güvenlik ve başarım kıyaslaması

SKIPJACK, AES ve SEA şifreleme algoritmalarının, genel bir güvenlik değerlendirmesi ve bir benzetim modelinde elde edilen genel başarım karşılaştırması Tablo 4.2'de görülmektedir (Wong, NIST, 1998, Başkök, 2007, Standaert ve diğ., 2006, Macé ve diğ., 2008). En büyük anahtar boyutuna (192-bit) sahip SEA, diğerlerine kıyasla ihmal edilebilir seviyede artan gecikme ve güç tüketim sonuçlarına mukabil, oldukça yüksek güvenlik sunmaktadır.

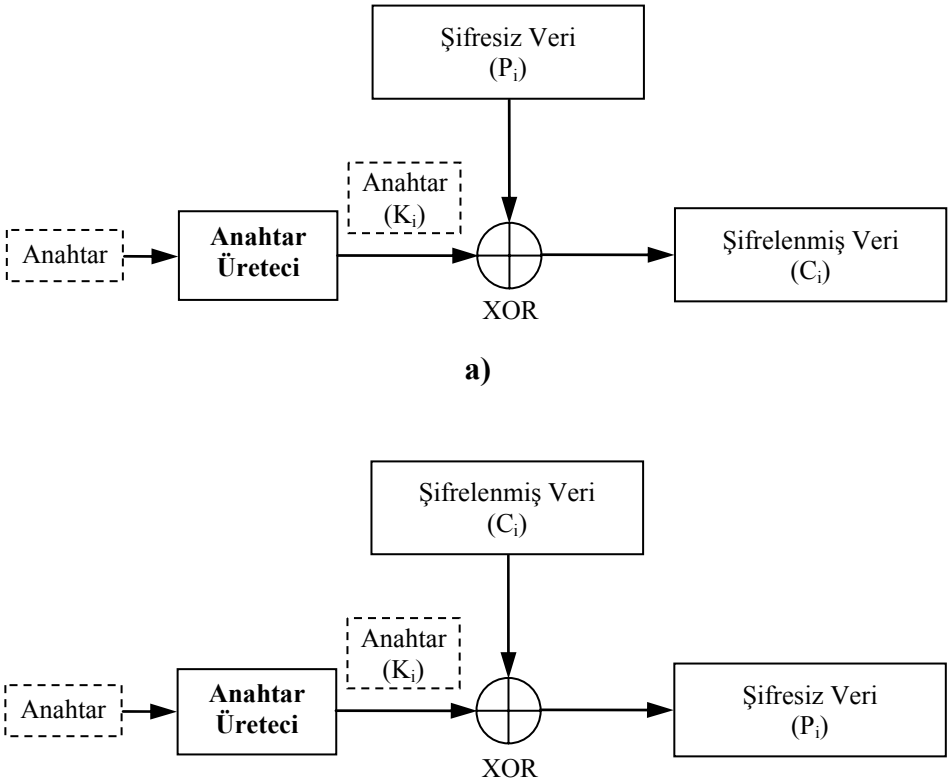
Tablo 4.2: Genel güvenlik ve başarım karşılaştırması (Bandirmali ve diğ., 2009).

	SEA	SKIPJACK	AES
Anahtar (bit)	192	80	128
Şifresiz veri (bit)	192	64	128
Güvenlik	Yüksek	Düşük	Yüksek
Hesaplama Karmaşıklığı	Düşük	Orta	Çok yüksek
Bellek (bayt)	Az	Az	Çok fazla
Gecikme (ms)	n	0,97 x n	1,003 x n
Güç tüketimi (mW)	m	0,94 x m	1,14 x m

4.2.1.2. Akış şifreleme yöntemleri

Akış (stream) şifreleme sistemleri, blok uzunlukları bir olan blok dönüştürücüler olarak düşünülebilir. Akış şifreleyicilerde, her birim zamanda şifrelenecek olan verinin yalnızca bir biti şifrelenir ve bu şifreleme işlemi bir XOR kapısı ile gerçekleştirilir.

Akış şifreleme sistemleri genel olarak Şekil 4.12.a'da görülen şifreleme, Şekil 4.12.b'de görülen şifre çözme temel yapısına sahiptirler. Şekil 4.12.a'daki şifreleme işleminde, şifrelenecek olan şifresiz verinin bit dizisi şeklindeki ifadesi olan P dizisinin bir biti ile üretilen anahtar dizisinin bir biti XOR işlemine tabii tutularak şifrelenmiş veriye ait olan C biti elde edilir. Bu işlem şifresiz veriye ait bütün bitler için uygulanarak istenilen şifrelenmiş veri elde edilir. Şekil 4.12.b'deki gibi şifrelenmiş veriden şifresiz veri elde edilmek istendiği takdirde ise XOR işleminin özelliğinden faydalanarak; aynı şekilde şifrelenmiş verinin bir biti ile veriyi şifrelemek için kullanılan aynı anahtar biti XOR işlemine tabii tutulur. Bu işlem şifrelenmiş veriye ait bütün bitler için tekrarlanarak şifresiz veri elde edilir.



Şekil 4.12: Akış şifreleme sistemi a) Şifreleme b) Şifre çözme.

Kullanılan akış şifreleme algoritması yeterince güvenli ise şifrelenmiş veriden şifresiz veriyi elde edebilmek için anahtarın bilinmesi şarttır. Anahtarın bulunabilmesi ancak bütün olasılıkların denenmesi ile mümkün olabilir, fakat bu işlem çok uzun zaman alır ve bu süre zarfında anahtar değiştirilmiş ve yeni bir anahtar kullanılmaya başlanmış olabilir.

Akış şifreleme sistemleri, genel olarak makul bit uzunluklarına sahip anahtar tohumu dizilerini ve/veya şifresiz veriyi kullanarak olabildiğince özgün, rastgele, kendini tekrarlamayan ve tahmin edilmesi zor olan anahtar dizisi üretilmesinde kullanılır. Üretilen anahtarın kalitesi akış şifreleme sistemlerinin temel problemidir. Üretilen anahtarın çok uzun zaman sonra kendini tekrarlaması, kullanılan anahtarın bulunmasını zorlaştırır ve bu da algoritmanın güvenliği açısından oldukça büyük öneme sahiptir. Akış şifreleme sistemleri rastgele bit üreten sonlu durum makineleri olarak algılanabilir. Burada rastgeleden kasıt, önceki bitlere bakarak bir sonraki bitin tahmin edilemez olmasıdır. Rastgele bir dizi elde etmenin mümkün olmadığı göz

önünde bulundurulursa, burada amaç eldeki sistemlerle olabildiğince karmaşık bir bit üretme algoritması kullanarak yeterince rastgele diziler elde etmektir.

Akış şifreleme sistemleri, eşzamanlı ve eşzamansız olmak üzere iki gruba ayrılmaktadır. Eşzamanlı akış şifreleme sistemleri, anahtar dizisini şifresiz veriden bağımsız olarak üreten sistemlerdir. Eşzamansız akış şifreleme sistemleri ise, anahtar dizisini sisteme girilen anahtar tohumlarıyla birlikte şifrelenmiş verinin sabit sayıdaki şifreli bitlerini kullanarak oluşturmaktadırlar (Doğan, 2006).

Şimdiye kadar standart hale gelmiş herhangi bir akış şifreleyici bulunmamaktadır. En yaygın olarak kullanılan akış şifreleyici RC4'tür. Blok şifreleyiciler de, blok şifreleme yaklaşımları kullanılarak akış şifreleyici ve anahtar dizi üreteci olarak kullanılabilmektedir.

Akış şifreleme sistemleri, blok şifreleme sistemlerinden çok daha yüksek hızlarda çalışır ve daha az donanım karmaşıklığına sahiptir. Ancak, akış şifreleme sistemleri doğru kullanılmazsa ciddi güvenlik problemlerine neden olabilmektedir. Akış şifreleme sistemlerinde şifrelenen her bitin önceki bitlerle bağlantısı olduğundan oluşacak bir bit hatası, bütün şifrelenmiş/şifresiz verinin hatalı olmasına yol açmaktadır (Menezes ve diğ., 1996).

4.2.2. Asimetrik anahtar şifreleme yöntemleri

Asimetrik anahtar şifreleme algoritmalarında, verinin şifrelenmesi için açık anahtar (public key), şifre çözme için ise matematiksel/mantıksal olarak açık anahtara bağlı özel bir anahtar (private key) kullanılmaktadır.

Simetrik anahtar algoritmalarının en büyük problemlerinden biri anahtar paylaşımının ve anahtarı korumanın zor olmasıdır. Veriyi şifrelemek ve şifresini çözmek için aynı anahtar kullanılmaktadır. Saldırıda bulunan kişilerin anahtarı ele geçirme ve kopyalama olasılığı yüksektir.

Asimetrik anahtar algoritmaları, matematiksel olarak birbirine bağlı iki ayrı anahtar kullanarak anahtar dağılım probleminin üstesinden gelmeye çalışmaktadır.

Açık anahtar şifreleme sistemlerinin bulunuşu belki de şifreleme alanındaki en önemli yenilik olarak görülebilir. Daha önceki sistemlerin aksine matematik üzerine kurulan bu sistemin en önemli dayanağı farklı iki anahtar ile çalıştırılmasıdır.

Buna rağmen, açık anahtar sistemi diğer sistemlerden kesinlikle daha güvenlidir şeklinde bir sonuç çıkarılamaz. Çünkü bu sistemlerin güvenliği, kullanılan anahtarların uzunluğu üstüne kurulu olduğundan, bu yargıda belirleyici olan anahtardır. Yeni sistemin eski geleneksel sistemleri tamamıyla devre dışı bıraktığı, açık anahtar sistemindeki işlem ya da hesaplama yükünün yoğunluğu sebebiyle günümüzde geçerli olan bir durum değildir. Ayrıca sistemin, anahtar dağılımında diğer sistemlere göre daha kolay bir yöntem getirmiş olması belli açılardan doğruluk taşımasına rağmen anahtar üretimindeki yoğun emek ve kaynak kullanımı, diğer sistemlerle karşılaşıldığında kullanıcı açısından sıfır problem anlamına gelmemektedir. Bir başka değişle sistemin güvenilirliği için anahtar seçiminde diğer sistemlere oranla daha fazla çaba gerekmektedir (Cobb, 2004).

Asimetrik anahtarlar başlangıç noktası olarak asal sayıları kullanırlar. İşlemin ilk kısmı, çok uzun iki asal sayı çarpılarak çok büyük bir sayı oluşturmaktır. Bu sayıdan matematiksel hesaplamalarla özel anahtar türetilir. Özel anahtardan da açık anahtar oluşturulur (Cobb, 2004).

Asimetrik anahtar şifreleme algoritmalarına, RSA (Ron Rivest, Adi Shamir, and Leonard Adleman) ve ECC (Elliptical Curve Cryptography, Eliptik Eğri Şifreleme) örnek olarak verilebilir.

RSA, açık ve özel anahtarların her birini oluşturmak için asal sayıları kullanır. İki büyük asal sayıyı çarpma esasına dayanır, fakat sonucu çarpanlarına ayırmak oldukça zordur (Hellman, 1978).

ECC algoritması, ECDLP (Elliptic Curve Discrete Logarithm Problem, Eliptik Eğri Ayrık Logaritma Problemi) problemi üzerine kurulmuş bir şifreleme sistemidir. Eliptik eğri yaklaşımı standart RSA sisteminden daha zengin matematiksel prosedürler içermektedir (Yerlikaya ve diğ., 2005).

Özel /açık anahtar kavramları güçlü olarak düşünülmesine rağmen 128 bitlik simetrik bir algoritmayla aynı seviyede güvenliği sağlamak için en az 2304 bit anahtara ihtiyaç duyulmaktadır. Asimetrik algoritmaların işlem gücü yavaştır ve büyük miktarda veri şifrelemek için onları kullanmak elverişsizdir. (Cobb, 2004). Simetrik anahtar şifreleme algoritmaları oldukça hızlıdır, donanımla gerçekleştirilmeleri kolaydır. Bunlar "gizlilik" güvenlik hizmetini yerine getirirler, fakat ölçeklenebilir değildirler. Güvenli anahtar dağıtımı ve bütünlük, kimlik denetimi gibi güvenlik hizmetlerini gerçekleştirmek oldukça zordur. Asimetrik anahtar şifreleme algoritmalarında ise anahtar yönetimi ölçeklenebilir, kırılması zor ve bütünlük, kimlik denetimi gibi güvenlik hizmetleri kolaylıkla sağlanabilir olmakla birlikte simetrik anahtar şifreleme algoritmalarıyla karşılaştırıldıklarında yaklaşık 1500 kat kadar yavaştır (Cobb, 2004). Ayrıca anahtar uzunlukları bazı uygulamalar için kullanışlı değildir.

4.3. Mevcut KAA Güvenlik Protokolleri

Güvenli olmayan iletişim kanallarını, etkin bir şekilde kullanılabilir hale getirmek amacıyla literatürde tanımlanmış birçok genel güvenlik protokolü bulunmaktadır.

Kablolu ağlar için RC4 şifreleme algoritmasını kullanan WEP (Wired Equivalent Privacy, Kablolu Eşdeğer Gizlilik) ve daha sonra yine aynı şifreleme algoritmasını kullanan TKIP (Temporal Key Integrity Protocol, Geçici Anahtar Bütünlük Protokolü) protokolleri ile kablosuz ağlar için AES şifreleme algoritmasını kullanan AES–CCMP (Advanced Encryption Standard–Counter Mode with Cipher Block Chaining Mode Protocol) protokolü geliştirilmiştir. Ayrıca, Internet güvenliğini sağlamak için IPSec (Internet Protocol Security, İnternet Protokol Güvenliği), SSL/TLS (Secure Sockets Layer/Transport Layer Security, Güvenli Soketler Katmanı/Ulaşım Katmanı Güvenliği) ve SSH (Secure Shell, Güvenli Kabuk) gibi

protokoller literatürde sunulmuştur. Ancak, bu protokollerin KAA'larda kullanımı, algılayıcı düğümlere nispî olarak aşırı işlem yükü getirmektedir. Zira bu düğümler, hesaplama kabiliyetleri oldukça sınırlı küçük donanımlardır (Samiah ve diğ., 2007).

Kablosuz hücresel telefon ağları ve tasarsız ağlar, KAA'lara oldukça benzer bir şekilde geliştirilmiştir; ancak, bunların da mevcut güvenlik tasarımlarının KAA'lara uyarlanmasında ciddi sınırlamalar bulunmaktadır.

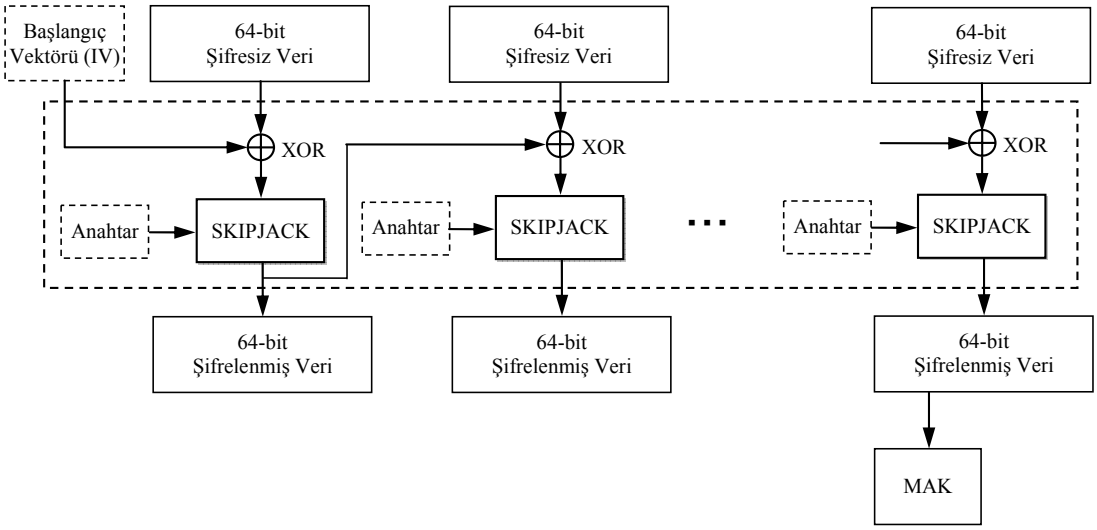
Alt bölümlerde, yaygın olarak kullanılan ve ilerleyen bölümlerde karşılaştırma amacıyla da gerekli olan KAA veri bağı katmanı güvenlik protokollerinden TinySEC ve LLSP üzerinde ayrıntılı olarak durulmaktadır.

4.3.1. TinySEC

TinySEC, mevcut olan güvenlik protokollerinin algılayıcı ağ güvenliği için yeterli olmamasından yola çıkılarak 2004 yılında Berkeley Üniversitesinde geliştirilen bir veri bağı katmanı güvenlik mimarisidir. TinySEC yine Berkeley Üniversitesi tarafından geliştirilmiş olan TinyOS işletim sistemi üzerinde çalışmaktadır ve TinyOS'un yeni sürümlerinde ekli durumdadır.

TinySEC, güvenlik için sadece asıllama TinySEC–Auth (Authentication Only TinySEC) ve asıllama ve şifreleme (TinySEC–AE, Authenticated Encryption TinySEC) olmak üzere iki farklı kullanım seçeneği sunmaktadır. Sadece asıllama seçeneği, mesaj asıllama kodu ile tüm paketi doğrularken, veriyi şifrelememektedir. Asıllama ve şifreleme seçeneğinde ise, hem veri şifrelenirken hem de paket (başlık+veri) bir Mesaj Asıllama Kodu (MAK) ile doğrulanır.

Şekil 4.13'de görüldüğü gibi TinySEC'te, mesaj gizliliğini sağlamak için SKIPJACK simetrik blok şifreleme algoritması, blok şifreleme yöntemlerinden birisi olan CBC yaklaşımı ile birlikte kullanılmaktadır. Şifrelemenin güvenliğini arttırmak için ayrıca 8 baytlık bir başlangıç vektörü (IV) bulunmaktadır. Mesaj bütünlüğü ve doğruluğunu sağlamak için CBC–MAC yaklaşımı ile 4 baytlık bir MAK hesaplanmakta ve pakete eklenmektedir (Karlof ve diğ., 2004, Bandirmali ve diğ., 2008b).



Şekil 4.13: TinySEC güvenlik protokolü.

TinySEC'in paket yapısı, standart TinyOS paket yapısına TinySEC tarafından kullanılan kısımların eklenmesiyle oluşturulmuştur. Şekil 4.14'de TinySEC'in ve TinyOS'un paket yapıları görülmektedir.

Hedef (2 bayt)	**AM (1 bayt)**	**Uzunluk (1 bayt)**	**Kaynak (2 bayt)**	**Sayaç (2 bayt)**	**Veri (0…29 bayt)**	**MAK (4 bayt)**

Hedef: Varış adresi.
AM: Aktif mesaj dinleyicisi tipi (TCP port numaraları benzeri).
Uzunluk: Gönderilecek verinin boyu.
Kaynak: Kaynak adresi.
Sayaç: 16-bit boyunda bir sayaç.

a)

Hedef (2 bayt)	**AM (1 bayt)**	**Uzunluk (1 bayt)**	**Veri (0…29 bayt)**	**MAK (4 bayt)**

b)

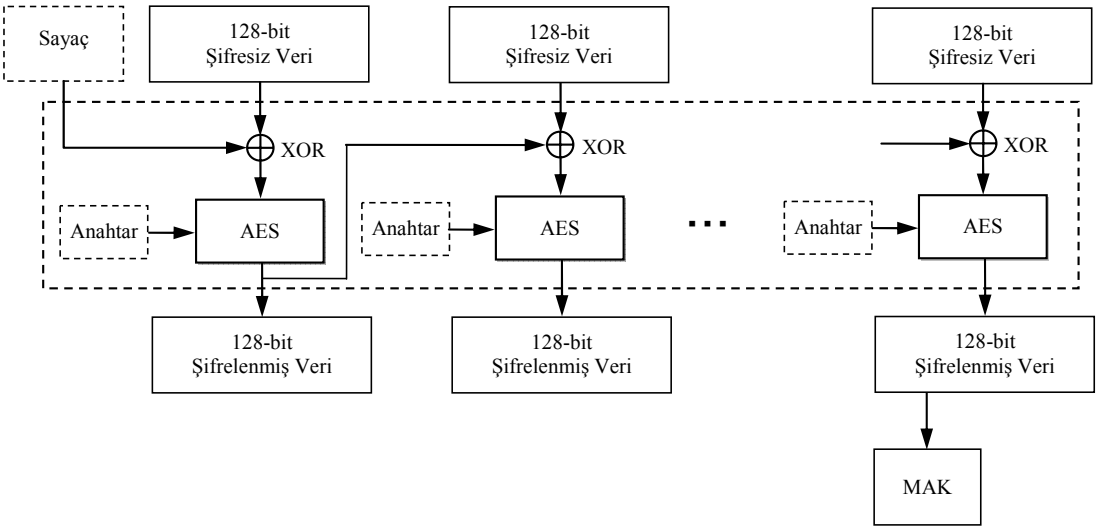
Hedef (2bayt)	**AM (1 bayt)**	**Uzunluk (1 bayt)**	**Grup (1 bayt)**	**Veri (0…29 bayt)**	**CRC (2 bayt)**

c)

Şekil 4.14: TinySEC paket formatları ve TinyOS paket formatı a) TinySEC asıllama ve şifreleme paket formatı b) TinySEC asıllama paket formatı c) TinyOS paket formatı.

4.3.2. LLSP

KAA'lar için geliştirilen bir başka protokol de, enerji etkin veri bağı katmanı güvenlik protokolü LLSP'dir (Link Layer Security Protocol). Şekil 4.15'de görüldüğü gibi LLSP'de mesaj gizliliği, simetrik anahtar blok şifreleme yöntemlerinden AES ve CBC yaklaşımının bir arada kullanılması ile sağlanmaktadır. AES, güvenliği yüksek ve mevcut kablosuz ağlarda kullanılan bir şifreleme algoritmasıdır. TinySEC'e benzer şekilde, bu protokolde de şifrelemenin güvenliğini arttırmak için bir başlangıç vektörü (IV) kullanılmakta ve mesaj bütünlüğü/doğruluğu için CBC–MAC yaklaşımı ile 4 baytlık bir MAK elde edilmektedir.



Şekil 4.15: LLSP güvenlik protokolü.

LLSP'nin TinySEC protokolünden en önemli olumlu farkı, Şekil 4.16'da da görüldüğü gibi aynı paketlerin tekrar gönderilip gönderilmediğinin denetimini yapmak için kullanılan sayaç değerinin, paket başlığına eklenmesi yerine, gönderici ve alıcı arasında bir kaymalı kaydedici (shift register) kullanılarak sağlanmasıdır. Böylece paket ek yükü (overhead) azalmaktadır. Sonuç olarak LLSP, güvenlik düzeyinden ödün vermeden, her paket için güvenlik ek yükünü en aza indirecek bir şekilde tasarlanmıştır (Ren ve diğ., 2005, Lighfoot ve diğ., 2007).

Hedef (2 bayt)	AM (1 bayt)	Uzunluk (1 bayt)	Kaynak (2 bayt)	Veri (0…29 bayt)	MAK (4 bayt)

Şekil 4.16: LLSP paket formatı.

Bu bölümde detaylı olarak açıklanan TinySEC ve LLSP protokolleri, tez çalışmamda önerdiğim YP ile enerji tüketimi ve bellek kullanımı açısından karşılaştırılarak YP'nin performansı ve kullanılabilirliği tespit edilmeye çalışılmaktadır.

4.4. Sonuç

Simetrik ve asimetrik anahtar şifreleme olarak ikiye ayrılan şifreleme yöntemlerinden KAA'ların sınırlı kaynakları göz önüne alındığında veri şifreleme için bir simetrik anahtar şifreleme yönteminin kullanılmasının daha uygun olduğu görülmektedir. Simetrik anahtar şifreleme yöntemlerinden de akış şifreleme algoritması yerine bir blok şifreleme algoritması kullanmak veri güvenilirliği açısından bir tercih sebebi olarak görülmektedir. Şifreleme algoritmasının gizliliğini arttırmak ve veri asıllaması/bütünlüğü sağlamak için ise blok şifreleme yaklaşımlarının kullanılması gerekmektedir.

Kablolu ve kablosuz ağlar için literatürde geliştirilmiş olan güvenlik protokollerini KAA'ların sınırlı kaynakları düşünüldüğünde doğrudan KAA'larda kullanmak uygun bulunmamaktadır. Bu sebeple, KAA'lar için yeni güvenlik protokolleri geliştirilme çalışmaları sürdürülmektedir.

KAA uygulamalarında, veri güvenilirliğini arttırmak için eklenen güvenlik yöntemlerinin çalışma süresi, bellek kullanımı ve enerji tüketim miktarlarını arttırması olağandır. Bu bakımdan uygulamaların ihtiyaçlarının iyi tespit edilmesi oldukça önemlidir. Zira, basit bir geniş ölçekli çevre ya da endüstriyel KAA uygulamasında güvenlik çok fazla önem taşımazken enerji tüketiminin büyük önemi bulunmaktadır. Diğer yandan, askeri ve sağlık uygulamalarında ise güvenlik büyük önem taşırken, algılayıcı düğüm enerji tüketimi, nispeten göz ardı edilebilir bir gösterge olarak değerlendirilmektedir.

Bütün bu parametreler ışığında, değişik ve giderek artan uygulama alanları da dikkate alındığında, KAA düğümleri tarafından transfer edilen veri doğruluğu, bütünlüğü ve gizliliği, ancak yüksek güvenlik sağlayan yeni bir veri bağı katmanı güvenlik protokolü geliştirilerek sağlanabilir.

5. YENİ BİR KABLOSUZ ALGILAYICI AĞ VERİ BAĞI KATMANI GÜVENLİK PROTOKOLÜ TASARIMI

5.1. Giriş

Yeni bir KAA veri bağı katmanı güvenlik protokolü geliştirirken, güvenilirlik, başarım (gecikme, bellek kullanımı, enerji tüketimi) ve kullanılırlık ölçütleri göz önüne alınır:

- Güvenilirlik: Bir veri bağı katmanı güvenlik protokolü üç temel güvenilirlik özelliğini sağlamalıdır:

i. Erişim denetimi ve mesaj asıllaması/bütünlüğü: İzinsiz olarak ağa katılmak isteyen düğümler önlenmelidir. Ağda bulunan gerçek düğümler, izinsiz olarak ağa katılmış düğümlerden gelen mesajları saptayabilmeli ve reddetmelidir. Mesaj asıllaması/bütünlüğü, her pakette bir mesaj asıllama kodu kullanılarak sağlanabilir.

ii. Gizlilik: Ağda yetkisiz olarak bulunan düğümlerden veriler/mesajlar gizli tutulmalıdır. Bu, şifreleme kullanılarak başarılabilir. Şifreleme, saldırıda bulunan düğümlerin mesajları değiştirmesini ve şifrelenmiş mesajlar hakkında bilgi edinilmesini önlemelidir. Bu özellik anlamsal (semantik) güvenlik olarak da bilinir.

iii. Tekrarlama saldırılarını önleme: İki geçerli düğüm arasındaki iletişime kulak misafiri olan bir saldırgan düğümün, mesajları tekrar aynı kaynak düğüm tarafından gönderterek saldırıda bulunması önlenmelidir. Mesaj, geçerli düğümde oluşturulduğundan, aynı alıcı tarafından tekrar kabul edilecektir. Bu tür saldırıyı önlemek için tekdüze olarak artan bir sayaç kullanılabilir. Her mesaj gönderildiğinde sayaç değeri bir arttırılarak, eski sayaç değerindeki mesajların tekrar tekrar alınması engellenebilir.

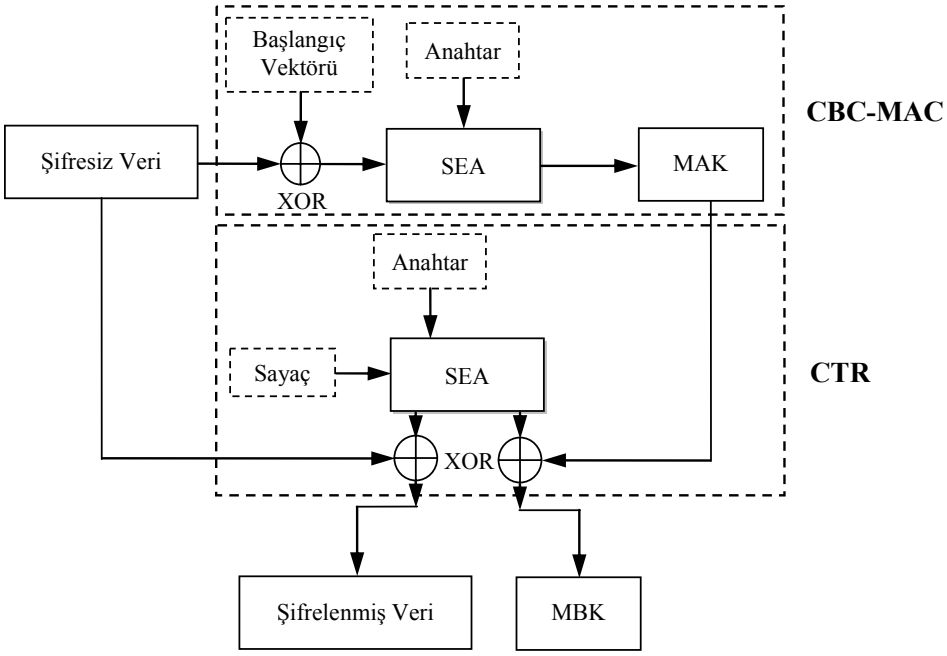
- Başarım: Şifrelemeden faydalanılan bir sistemde, RAM ve işlemci kullanımının artışına paralel olarak gönderilen mesajın uzunluğu da artabilmektedir. Buna bağlı olarak, iletişim verimliliği azalırken uçtan–uca gecikme olumsuz

etkilenmekte ve KAA'lar için oldukça önemli bir kısıtlayıcı olan güç tüketimi artmaktadır. KAA'larda paket ek yükünü en aza indirmek için makul ölçülerde koruma sağlayan bir güvenlik mekanizması oluşturmak önemlidir. Geleneksel kablolu/kablosuz ağ güvenliğinde, 8–16 bayt arasında ek yük artışı önemsiz kabul edilmektedir. Ancak KAA'larda 8 baytlık bir ek yük, toplam paket büyüklüğünün %25'ini oluşturmaktadır.

- Kullanılırlık: Güvenlik mekanizması, mevcut sistemlere kolayca uygulanabilmeli ve kullanılabilmelidir.

Farklı KAA uygulamalarının gereksinimleri esas alındığında, ölçütlerin öncelikleri de değişebilmektedir. Örneğin, çevresel izlemede gizlilik çok büyük önem taşımazken, sağlık ve askeri uygulamalarda güvenlik ilkelerini oluşturan mesaj gizliliği, asıllaması, bütünlüğü ve kaynak/hedef adres doğrulaması oldukça önemlidir. Bu nedenlerle, örneğin sağlık uygulamaları için planlanan bir çalışmada, enerji etkin bir güvenlik protokolü yerine, iletişim güvenliği ve mesaj (veri) güvenilirliği arttırılmış bir güvenlik protokolü tasarımı amaçlanmalıdır. Geleneksel olarak, yüksek güvenlik sağlayan yöntemler aşırı hesaplama gerektirmektedir ve bunların algılayıcı düğümlerde kullanımı enerji tüketiminin oldukça artmasına yol açmaktadır.

Bu bölümde, özellikle sınırlı kaynakları bulunan düğümlerde kullanılan SEA blok şifreleme algoritması ile veri gizliliği ve güvenilirliğini arttırmak için kullanılan CTR ve CBC–MAC mesaj asıllama/bütünlük denetim yaklaşımlarının birlikte (bütünleşik) kullanımıyla geliştirilen Yeni Bir KAA Veri Bağı Katmanı Güvenlik Protokolü (YP) detaylı bir şekilde açıklanmaktadır (Şekil 5.1). YP, veri asıllaması/bütünlüğü, tekrarlama saldırılarını önleme ve veri gizliliği ölçütleri temel alınarak gerçekleştirilen bir güvenlik protokolüdür. C program kodları EK–A ve EK–B'de yer alan YP'de sunulan CTR yaklaşımı ve SEA şifreleme algoritması kullanılarak mesaj gizliliği ve güvenilirliği arttırılırken, CBC–MAC asıllama yaklaşımı ile mesaj bütünlüğü ve doğruluğu sağlanmaktadır. Ayrıca YP geçerlilik değerlendirmesinin yapıldığı bu bölümde, klasik TinySEC ve LLSP yöntemleriyle karşılaştırmalı başarım sonuçlarının yanı sıra YP'nin kullanıldığı basit bir KAA uygulamasının OPNET yazılımı ile modellenmesi ve benzetimi de sunulmaktadır.



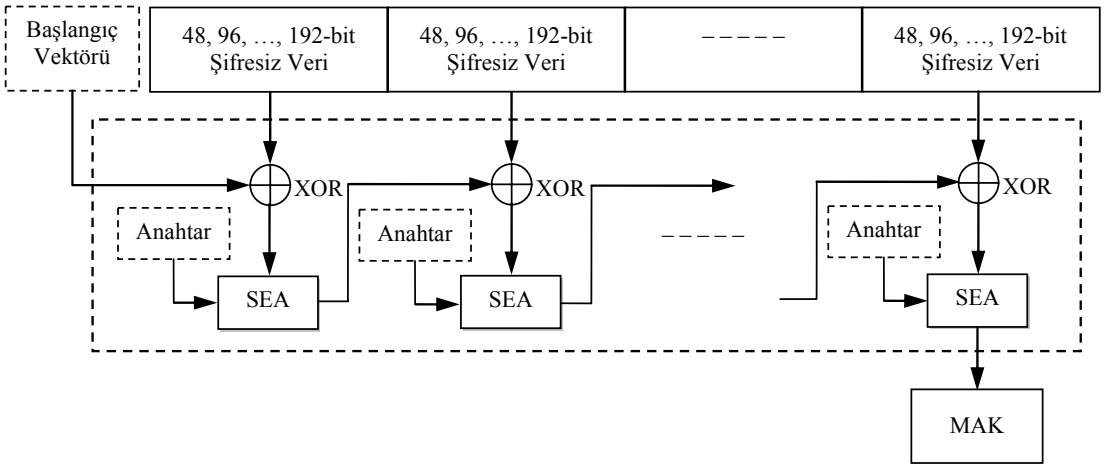
Şekil 5.1: Önerilen YP'nin blok şeması.

5.2. YP Mesaj Asıllaması, Bütünlüğü ve Güvenilirlik

Asıllama olmaksızın sadece şifrelemenin kullanılması güvenilir değildir. Asıllama yapılmadan şifrelenmiş mesajlar, şifresiz mesajda tahmin edilebilir değişikliklere sebep olabilmektedir. Asıllama mekanizması bulunmayan alıcılar, üçüncü şahıslardan kaynaklanan saldırıların oluşturduğu değişiklikleri saptayamayabilir. Ayrıca asıllanmamış mesajlar "kes ve yapıştır" saldırılarına da açıktır. Bu olumsuzlukları da göz önünde bulundurarak güvenlik protokolleri geliştirilirken, mesaj asıllamasını ve bütünlüğünü sağlamak amacıyla mesaj asıllama kodu (MAK) kullanılmaktadır. MAK, iletilen mesajları şifrelemeyip, mesajın hedefe doğru bir şekilde ulaştığını tespit/teyit etmek amacıyla geliştirilmiş bir yöntemdir.

Önerilen YP'de, MAK hesabı CBC–MAC yaklaşımı kullanılarak yapılmaktadır. CBC–MAC oldukça etkili, hızlı ve güvenlidir. Geleneksel güvenlik protokolleri 8 ya da 16 baytlık MAK kullanmakla birlikte, KAA'larda mesaj asıllaması için çoğunlukla 4 baytlık MAK değeri yeterli olmaktadır. Bu durumda dahi saldırıda bulunan bir düğüm, belirli bir mesaj için oluşturulan MAK değerini ancak 2^{31} denemeden sonra elde edebilir (Jonsson, 2003).

Şekil 5.2'de görüldüğü gibi YP'de şifrelenmiş veri blokları birbirine zincirlenmiş bir yapıdadır. Burada SEA algoritmasının girdisini, XOR'lanmış mevcut şifresiz veri bloğu ve önceki şifreli veri bloğu oluşturmaktadır. Şifrelenmiş en son veri bloğunun ilk 4 baytı MAK değeri olarak seçilmekte ve pakete eklenmektedir. Saldırıda bulunan bir düğüm şifrelenmiş mesajda herhangi bir değişiklik (veri ekleme, bozma, çıkartma vb.) yaptığında, bu durum, paketteki mevcut MAK değeri ve alıcı (hedef) düğümün hesapladığı MAK değeri ile karşılaştırılarak kolaylıkla tespit edilebilmektedir.



Şekil 5.2: YP'de CBC–MAC yaklaşımıyla MAK hesabı.

5.3. YP Mesaj Gizliliği ve Güvenilirlik

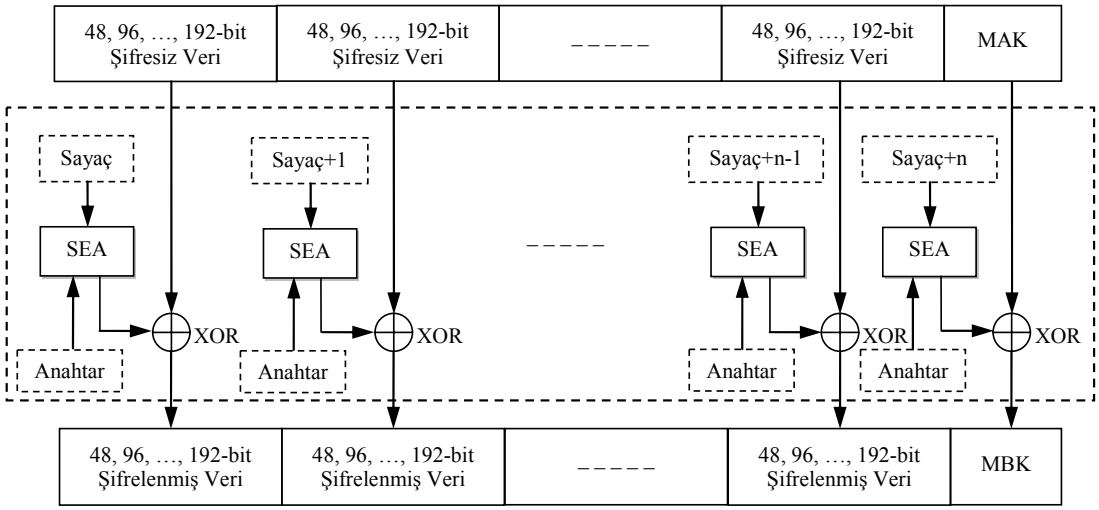
Literatürde, özellikle sınırlı kaynaklar için geliştirilmiş çok az sayıda şifreleme algoritması (örneğin TEA (Wheeler ve Needham (1995)) ve Yuval's proposal (Yuval (1997))) bulunmaktadır. Hâlihazırda, bu algoritmalar basit lineer ve diferansiyel şifre analiz yöntemleriyle dahi incelendiğinde kanıtlanabilir derecede güvenlik sağlayamadığı görülmektedir. Kablolu ya da kablosuz ağlarda kullanılabilen simetrik blok şifreleme algoritmalarından AES ve benzerleri ise KAA uygulamaları açısından genellikle yüksek maliyet, düşük güvenlik düzeyi ve yetersiz başarım gibi önemli sınırlamalara sahiptir (Standaert ve diğ., 2006). Bu nedenlerle, önerilen KAA veri bağı katmanı güvenlik protokolünde, kanıtlanabilir derecede güvenliği yüksek, başta lineer ve diferansiyel şifre analiz yöntemlerine karşı olmak üzere dayanıklı ve

özellikle sınırlı kaynaklar için geliştirilmiş SEA simetrik blok şifreleme algoritması kullanılmaktadır.

Uygulama çalışmaları sonucunda, SEA şifreleme algoritmasının geleneksel AES şifreleme algoritmasına kıyasla, kaynakları (bellek, enerji ve mikroişlemci) oldukça verimli kullanarak daha yüksek başarım sağladığı ortaya konulmuştur. Bu nedenle SEA, düşük şifreleme maliyetiyle sınırlı kaynaklara sahip sistemlerde etkin kullanım için oldukça uygundur. Ayrıca, SEA veri blok büyüklüğü, AES (128-bit) şifreleme algoritmasındaki gibi sabit değildir. İstenen güvenlik düzeyine uygun ve dinamik bir şekilde değişik veri blok büyüklüklerinde (48, 96,…, 192-bit) şifreleme yapılabilmektedir (Standaert ve diğ., 2006, Macé ve diğ., 2008, Bandirmali ve diğ., 2009).

Bu tez çalışmasında sunulan YP'de, TinySEC ve LLSP veri bağı katmanı güvenlik protokollerinden farklı olarak, veri güvenilirliğini (veri gizliliği) arttırmak için, blok şifreleme yaklaşımlarından birisi olan CTR kullanılmaktadır. CTR yaklaşımıyla birlikte, geleneksel TinySEC protokolünde başlık paketine eklenen ve ek yük getiren bir sayaç değerine gerek kalmamaktadır.

Şekil 5.3'de görüldüğü gibi rastgele oluşturulan bir sayaç değeri SEA şifreleme algoritması ile şifrelenerek şifresiz veri bloğu ile XOR'lanmakta ve şifrelenmiş veri bloğu elde edilmektedir. MAK değerinin de şifrelenerek alıcıya (hedefe) iletilmesi sağlanmaktadır. Diğer bir ifadeyle hem veri hem de MAK değeri şifrelenmektedir. Böylece veri gizliliği artarken, aynı zamanda eklenen sayaç ile veri tazeliği de garanti edilmektedir (Jonsson, 2003).



Şekil 5.3: YP'de CTR yaklaşımıyla veri şifreleme.

5.4. YP Tasarım Aşamaları

Şekil 5.4'de görüldüğü gibi şifrelenecek veri (mesaj), seçilen güvenlik düzeyine uygun ve dinamik olarak belirlenebilen boyuttaki veri bloklarına (48, 96, …, 192-bit) bölündükten sonra üzerinde işlem yapılmaktadır (EK−A, EK−B).

MAK değeri, veri ve anahtar üzerinden CBC–MAC asıllama yaklaşımı kullanılarak aşağıdaki işlemler sonunda hesaplanmaktadır:

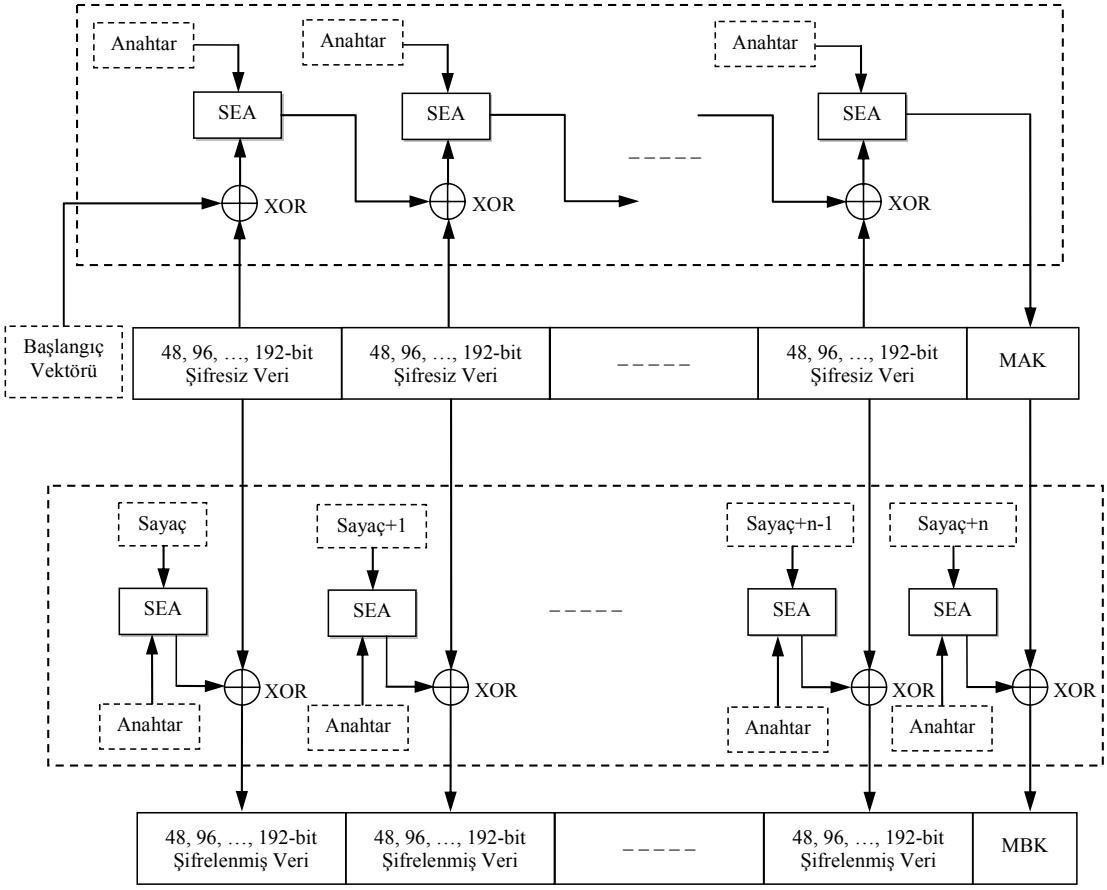
- İlk veri bloğu ve başlangıç vektörü XOR işlemine tabi tutularak elde edilen değer SEA ile şifrelenmektedir.
- İkinci veri bloğu ve şifrelenmiş ilk blok XOR'lanarak elde edilen değer SEA ile şifrelenmektedir.
- Bu süreç, bütün veri blokları şifreleninceye kadar tekrar edilmektedir.
- En son şifrelenmiş veri blok değerinin ilk en önemli (MSB) 32 biti MAK değeri olarak kullanılmaktadır.

SEA ile şifrelenmiş veri blokları elde edilirken, CTR yaklaşımının bütünleşik kullanımı aşağıdaki işlemler ile gerçekleştirilmektedir:

- SEA ile şifrelenen bir sayaç değeri ve ilk şifresiz veri bloğu XOR'lanarak ilk şifrelenmiş veri bloğu elde edilmektedir.

- Sayaç değeri arttırılarak tekrar şifrelenir. Şifrelenmiş yeni sayaç değeri ve ikinci şifresiz veri bloğu XOR'lanarak ikinci şifrelenmiş veri bloğu elde edilmektedir.
- Bu süreç, tüm veri blokları bitinceye kadar tekrar edilmektedir.
- Son olarak, CBC–MAC yaklaşımı kullanılarak elde edilen MAK değeri ve şifrelenmiş en son sayaç değerinin (Sayaç+n) XOR'lanmasıyla Mesaj Bütünlük Kodu (MBK) elde edilmektedir.

Böylece alıcısına (hedefe) iletilmeden önce hem veri hem de pakete eklenen MAK değeri şifrelenmiş olmaktadır. Bu iki yöntemin SEA ile bütünleşik kullanımı, yeni geliştirilen YP'nin, TinySEC ve LLSP protokollerinden kanıtlanabilir ve oldukça üst derecede güvenli olduğunu analitik bir yaklaşımla doğrulamaktadır. Ayrıca bu yöntemlerle birlikte mesaj gizliliği, asıllaması ve bütünlüğüne ek olarak, şifreleme için kullanılan sayaç değeri tekrar gönderme saldırılarını da önlemektedir (Bandirmali ve Erturk, 2009).



Şekil 5.4: YP bütünleşik yapısı.

YP'nin paket formatı Şekil 5.5'de görülmektedir. YP'de şifreleme işlemi yapılırken zaten bir sayaç değeri kullanılmasından dolayı paket başlığına tekrar bir sayaç değeri eklemeye gerek görülmemektedir. Böylece LLSP'de olduğu gibi paket ek yükü de azalmaktadır.

Şekil 5.5: YP paket formatı.

5.5. YP'de Anahtar Yönetimi

Anahtar yönetimi (key management), ağ boyunca şifrelenmiş anahtarların paylaşım ve dağıtım şeklini belirlemektedir. YP, herhangi bir anahtar yönetimi kullanımı açısından sınırlı değildir. Tablo 5.1'de KAA'lar için değişik anahtar yönetim mekanizmaları verilmektedir.

Tablo 5.1: Veri bağı katmanı güvenliği için farklı anahtar yönetim mekanizmaları.

Anahtar Yönetim Mekanizması	Güçlü Yönleri	Zayıf Yönleri
Ağ–Paylaşım Anahtarı	Basit ve uygulaması kolaydır. Pasif katılım ve yerel yayını destekler.	Düğüm uzlaşmasına duyarlıdır.
Bağlantı Anahtarı	Düğümlerin uzlaşmasından etkilenmez.	Anahtar dağıtım protokolüne ihtiyaç duyar. Pasif katılım ve yerel yayına imkân vermez.
Grup Anahtarı	Düğümlerin uzlaşmasına hassas değildir. Pasif katılım ve yerel yayını destekler.	Anahtar dağıtım gerektirir. Ek işlevler kazandırmakla birlikle düğüm güç ihtiyacını arttırır.

YP'de gizli anahtar kullanım esası benimsenmiştir. Ancak bu durumda hem alıcının, hem de göndericinin gizli anahtarı bilmesi gerekir. Bu amaçla değişik anahtar paylaşım yaklaşımları kullanılabilmektedir;

- Ağ–Paylaşım anahtar: Tüm ağdaki düğümler tarafından kullanılan ortak anahtar.
- Bağlantı anahtarı: İki düğümün haberleşeceği zaman üzerinde anlaştıkları anahtar (oturum anahtarı gibi).

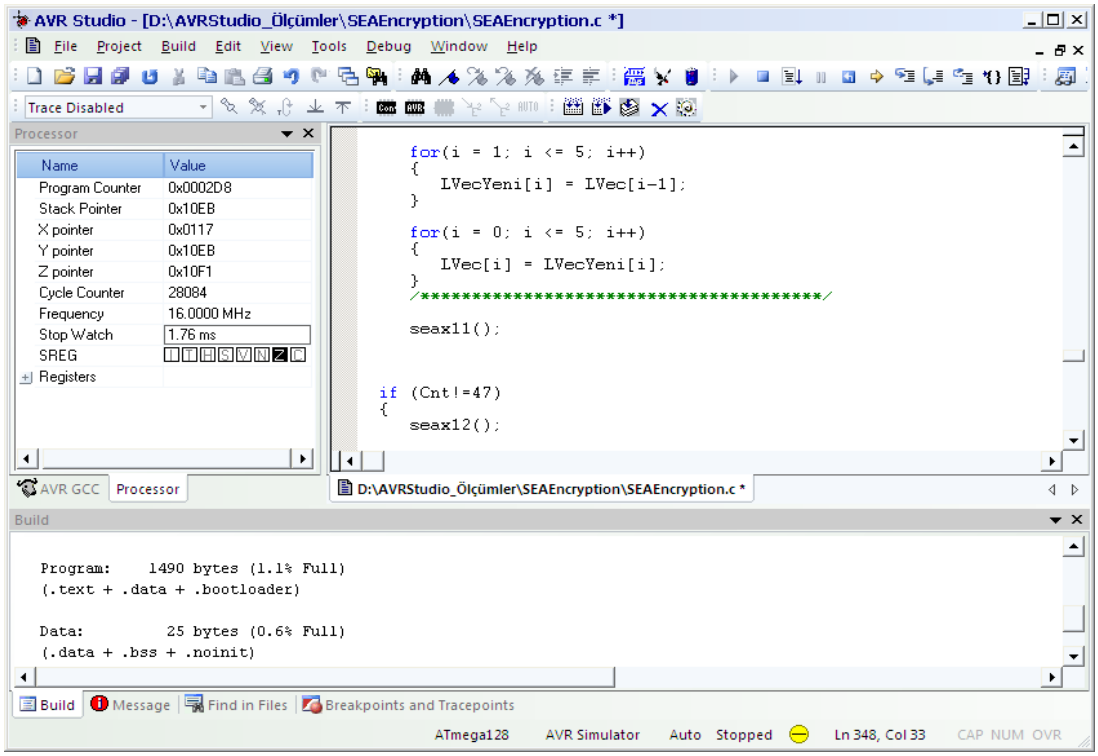
- Grup anahtarı: Yakın düğümlerin kullandığı, daha çok yerel yayınlara ihtiyaç duyulan durumlarda kullanılan anahtar (Karlof ve diğ., 2004).

5.6. YP'nin Geçerlilik Değerlendirmesi

KAA güvenlik protokollerinin değerlendirmesinde oldukça önemli olan bellek kullanım ve enerji tüketim ölçütleri, önerilen YP ile LLSP ve TinySEC nicel karşılaştırmasında da esas alınmaktadır. Güvenlik düzeyinin niteliği açısından ise özellikle içermekte olduğu SEA ile CTR ve CBC–MAC yaklaşımlarının bütünleşik kullanımı, önceki bölümlerde detaylandırıldığı gibi, YP'yi karşılaştırmada kullanılan bu iki geleneksel güvenlik protokolüne kıyasla hâlihazırda oldukça üstün kılmaktadır. Zira CTR yaklaşımında sözde rastgele değişim (pseudorandom permutation, PRP) kullanımı, bu yaklaşımın güvenilirlik düzeyinin yeterli seviyede olduğunun temel göstergesidir. (Bellare ve diğ., 1997)'de detaylıca gösterildiği üzere CTR yaklaşımı kullanılarak temin edilebilecek aşılması mümkün olmayan güvenilirlik düzeyi, klasik ECB, CBC, CFB ve OFB şifreleme yaklaşımlarına nazaran daha yüksektir (Lipmaa ve diğ., 2000). Ayrıca CTR yaklaşımı, 48, …, 192-bitlik tahmin edilmesi mümkün olmayan sayaç başlangıç değeri ile, ön hesaplama (precomputation) türü saldırılara önemli bir seviyede direnç sağlamaktadır. Aynı zamanda CTR yaklaşımının "seçilmiş-şifresiz veri" saldırılarına karşı yüksek güvenlik sağladığı (Bellare ve diğ., 1997)'de ispatlanmış bulunmaktadır. YP, CTR yaklaşımını CBC–MAC ile birlikte kullanmak suretiyle "seçilmiş-şifreli veri" saldırılarına karşı da oldukça yüksek bir düzeyde dayanıklılık göstermektedir (Shi ve diğ., 2005). CBC–MAC yaklaşımı kullanıldığında, saldırıda bulunan bir düğümün "m" blok uzunluğundaki "q" mesajı ele geçirdikten sonra sahte mesaj üretme olasılığı "$5m^2q^2/2^{128}$"den daha çok değildir. Örneğin, saldırıda bulunan bir düğüm, bir ay içinde mesaj boyutu 10000 bayttan daha uzun olmayan saniyede 1000 mesaj ele geçirmiş olsun. Saldırıda bulunan bu düğümün yeni sahte mesaj üretme olasılığı 10 trilyonda 1'den daha azdır (Black ve Rogaway, 2000). Bu açıklamalardan da anlaşılabildiği gibi bu iki yaklaşımının birlikte kullanılması YP güvenlik düzeyini son derece arttırmaktadır. Literatürde sunulan bu tür protokoller için, "güvenlik" ve "veri güvenilirliği" düzeylerini belirleyen somut ve sayısal (ölçülebilir) bir kıstas

bulunmamasına bağlı olarak, bu bölümde önerilen çalışmanın nitelik kıyaslamasının üzerinde detaylıca durulamamaktadır.

Örnek KAA uygulamalarında da kullanılabilecek önerilen YP, TinySEC ve LLSP'nin bellek kullanım ve enerji tüketim değerlerini elde edebilmek için AVR Studio ve WinAVR yazılımları (Şekil 5.6) kullanılmış bulunmaktadır. AVR Studio, ATMEL firması tarafından geliştirilmiş ve ATMEL AVR mikrodenetleyiciler için Assembly ve C dillerinde programların yazılarak derlenebildiği bir geliştirme ve benzetim aracıdır. AVR Studio ile yazılan programların herhangi bir mikroişlemcinin üzerinde çalışması izlenebilmekte ve değerlendirilebilmektedir.



Şekil 5.6: AVR Studio benzetim aracı.

KAA uygulamalarında yaygın olarak tercih edilen MicaZ kablosuz algılayıcı düğümleri, 128 Kbayt kod belleği, 4 Kbayt veri belleği ve 16 MHz hızında çalışan ATMEGA 128L mikrodenetleyicisi içermektedir. Sunulan bu çalışmada gerçeğe yakın sonuçlar elde edebilmek için YP, TinySEC ve LLSP'nin benzetimleri, AVR Studio ortamında Tablo 5.2'de özellikleri yer alan ATMEGA 128 mikrodenetleyicisi kullanılarak gerçekleştirilmiş bulunmaktadır. Sonuçların anlamlı ve karşılaştırmalı

değerlendirilebilmesi amaçlarıyla üç protokolde de işlem gören veri boyutu sabit ve eşit (768 bit) kabul edilmektedir.

Tablo 5.2: ATMEGA 128 mikrodenetleyici özellikleri.

ATMEGA 128	Büyüklükleri
Flash	131072 bayt
EEPROM	4096 bayt
Dâhili SRAM	4096 bayt
Harici SRAM	65536 bayt
I/O	64 bayt
Genişletilmiş I/O	160 bayt
Maksimum hız	16 MHz

SEA işlem sürelerinin başarım karşılaştırmasının anlamlı ve kolay olması için $SEA_{(192,192)}$, $[SEA_{(96,96)}+(CTR+CBC–MAC)]^{YP}$ ve $[SEA_{(192,192)}+(CTR+CBC–MAC)]^{YP}$ uygulamalarına ait şifreleme ve şifre çözme toplam süreleri $SEA_{(96,96)}$'ya göre normalize edilerek sunulmaktadır. Normalize işlemi yapılırken, $SEA_{(96,96)}$ uygulamasına ait şifreleme ve şifre çözme toplam süresi "1 saniye" kabul edilerek, $SEA_{(192,192)}$, $[SEA_{(96,96)}+(CTR+CBC–MAC)]^{YP}$ ve $[SEA_{(192,192)}+(CTR+CBC–MAC)]^{YP}$ şifreleme ve şifre çözme toplam süreleri nispî bir değer olarak yeniden hesaplanmaktadır. Örneğin; şifreleme ve şifre çözme toplam süreleri $SEA_{(96,96)}$ için 24,62 ms ve $[SEA_{(96,96)}+(CTR+CBC–MAC)]^{YP}$ için 47,39 ms olarak elde edilmiştir. Bu durumda, $SEA_{(96,96)}$'ya ait şifreleme ve şifre çözme toplam süresine göre normalize $[SEA_{(96,96)}+(CTR+CBC–MAC)]^{YP}$ şifreleme ve şifre çözme değeri 1,92 (1,92 = 47,39 ms / 24,62 ms) olarak hesaplanır. Diğer bir ifadeyle, $[SEA_{(96,96)}+(CTR+CBC–MAC)]^{YP}$ uygulaması için $SEA_{(96,96)}$'ya kıyasla yaklaşık olarak 1,92 kat daha fazla gecikme görülmektedir.

YP için uyarlanan veri güvenilirliği arttırılmış SEA şifreleme algoritması karşılaştırmalı başarım sonuçları Tablo 5.3'de sunulmaktadır. $[SEA_{(96,96)}+(CTR+CBC–MAC)]^{YP}$ uygulamasının şifreleme ve şifre çözme toplam süresi $SEA_{(96,96)}$'ya kıyasla 1,92 kat daha fazlayken bu değer $SEA_{(192,192)}$ uygulaması

için 2,50 ve $[SEA_{(192,192)}+(CTR+CBC–MAC)]^{YP}$ uygulaması için ise 4,68 katına çıkmaktadır. Önerilen protokolde güvenlik düzeyini arttırıcı ilave işlevler şifreleme ve şifre çözme sürelerini de nispi olarak arttırmaktadır.

Tablo 5.3: YP için uyarlanan veri güvenilirliği arttırılmış SEA işlem süreleri.

	Normalize[nz] Şifreleme ve Şifre Çözme Toplam Süresi
$SEA_{(96^{vb},96^{ab})}$	1^{nz}
$SEA_{(192,192)}$	2,50
$SEA_{(96,96)}+(CTR+CBC–MAC)^{YP}$	1,92
$SEA_{(192,192)}+(CTR+CBC–MAC)^{YP}$	4,68

[nz]: Tablodaki diğer değerler, standart $SEA_{(96,96)}$ şifreleme ve şifre çözme toplam süresi "1 saniye" esas alınarak normalize edilmiş sonuçlardır.
[vb]: Veri blok boyutu (bit)
[ab]: Anahtar boyutu (bit)

Tablo 5.4'de YP ve LLSP'nin, TinySEC ile normalize edilen bellek kullanım değerleri karşılaştırmalı olarak görülmektedir. 96-bit veri blok/anahtar boyutlu YP'nin bellek kullanım değeri, standart TinySEC protokolüne kıyasla 1,57 kat daha fazlayken, bu değer 192-bit veri blok/anahtar boyutlu (oldukça yüksek güvenlik düzeyinde) YP için 1,78'e çıkmaktadır. Benzer bir şekilde standart LLSP için değerlendirme gerçekleştirildiğinde ise bellek kullanım miktarının 2,26 kat arttığı sonucuna ulaşılmaktadır. Geliştirilen ve içerdiği yeni yaklaşımlara bağlı olarak güvenlik düzeyi oldukça üst düzeyde bulunan YP ile TinySEC protokolü karşılaştırıldığında bellek kullanımda ortaya çıkan bu artışın, genel olarak KAA düğümlerine nispî olarak aşırı bir yük getirmediği değerlendirilmektedir (Vitaletti, 2006).

Tablo 5.4: YP ile diğer protokollerin bellek kullanım karşılaştırması.

	Normalize[nz] Bellek Kullanım Değeri
$TinySEC_{(64,80)}$	1^{nz}
$LLSP_{(128,128)}$	2,26
$YP_{(96,96)}$	1,57
$YP_{(192,192)}$	1,78

[nz]: Tablodaki diğer değerler, geleneksel $TinySEC_{(64,80)}$ bellek kullanım değeri "1 bayt" esas alınarak normalize edilmiş sonuçlardır.

Tablo 5.5'de YP ve LLSP'nin, TinySEC ile normalize edilen enerji tüketim değerleri karşılaştırmalı olarak sunulmaktadır. 96-bit veri blok/anahtar boyutlu YP'nin normalize enerji tüketim değerinin, TinySEC protokolüne kıyasla 0,88'e düştüğü görülmektedir. Bu değer 192-bit veri blok/anahtar boyutlu YP için 2,14'e çıkmaktadır. Benzer bir yaklaşımla LLSP için yapılan değerlendirmede ise enerji tüketiminin 12,01 kat arttığı sonucuna ulaşılmaktadır. YP'nin iletişim güvenlik düzeyini ve veri güvenilirliğini dinamik olarak oldukça üst düzeye çıkarmakla birlikte, TinySEC ile kıyaslandığında enerji tüketiminin çok fazla artmadığı görülmektedir. Ayrıca diğer geleneksel yöntem olan LLSP'nin basit ölçekte bir güvenlik iyileştirmesi getirmesine rağmen, TinySEC'e kıyasla enerji tüketiminin çok yüksek düzeyde ortaya çıkması da dolaylı olarak YP'nin üstünlüğünü kanıtlamaktadır. Protokollerin çalışması ile düğümlerde tüketilen enerji değerleri ve toplam çalışma süreleri (şifreleme ve şifre çözme) doğru orantılı olarak değişmektedir. Dolayısıyla, bu bölümde ayrıca çalışma süreleri karşılaştırması yapılmamıştır ve benzetim modelleri yardımıyla gerçekleştirilen gecikme analizi Bölüm 5.7.3'de ele alınmaktadır.

Tablo 5.5: YP ile diğer protokollerin enerji tüketim karşılaştırması.

	Normalize[nz] Enerji Tüketim Değeri
$TinySEC_{(64,80)}$	1^{nz}
$LLSP_{(128,128)}$	12,01
$YP_{(96,96)}$	0,88
$YP_{(192,192)}$	2,14

[nz]: Tablodaki diğer değerler, geleneksel $TinySEC_{(64,80)}$ enerji tüketim değeri "1 joule" esas alınarak normalize edilmiş sonuçlardır.

Önerilen YP'nin başarımı hakkında çok daha anlaşılabilir ve bütüncül bir fikir edinmek için Tablo 5.6'da sunulan bir diğer önemli parametre olan tümleşik bellek kullanım ve enerji tüketim göstergesi (BE) sonuçlarını incelemek gerekmektedir. Bir KAA veri bağı katmanı güvenlik protokolü, az bellek kullanımı gerektirirken yüksek enerji tüketebilir ya da tam tersi bir durum söz konusu olabilir. Bu açıdan bakıldığında BE göstergesi, YP'nin, TinySEC ve LLSP ile anlamlı bir şekilde karşılaştırıldığı gerçek bir değerlendirme için yüksek öneme sahiptir. TinySEC ile normalize edilen 96-bit veri blok/anahtar boyutlu YP ve LLSP BE değerleri, sırasıyla 1,38 ve 27,20 olarak elde edilmektedir. Bu sonuçlar genel bir bakışla göstermektedir ki, YP kullanımı geleneksel eşleniği LLSP'ye nazaran bellek kullanım ve enerji tüketim ölçütleri açısından oldukça önemli bir iyileştirmeyi (19,71 kat) beraberinde getirmektedir. Sonuç olarak, önerilen YP'nin, sınırlı kaynaklara sahip KAA'larda üstün bir başarım sağlamasına ek olarak, yüksek güvenliği ön planda tutan maliyet etkin bir çözüm sunduğu da açıkça görülebilmektedir.

Tablo 5.6: YP ile diğer protokollerin "bellek kullanım x enerji tüketim" karşılaştırması.

	Normalize[nz] Bellek Kullanım x Enerji Tüketim (BE) Değeri
$TinySEC_{(64,80)}$	1^{nz}
$LLSP_{(128,128)}$	27,20
$YP_{(96,96)}$	1,38
$YP_{(192,192)}$	3,82

[nz]: Tablodaki diğer değerler, geleneksel $TinySEC_{(64,80)}$ "Bellek kullanım x Enerji tüketim" değeri "1 bayt x 1 joule" esas alınarak normalize edilmiş sonuçlardır.

Elde edilen sonuçlardan genel olarak anlaşılmaktadır ki iletişim güvenlik ve veri güvenilirlik düzeylerini arttırıcı tüm ek işlevler, uygulamaların çalışma sürelerini (şifreleme/şifre çözme) olumsuz etkilemektedir. Bu durum paralelinde özellikle enerji tüketim göstergesinde önemli bir artışa yol açmaktadır. Kablosuz iletişim (RF) bileşenlerini de içeren tam bir KAA uygulaması düşünüldüğünde, bu artışın özellikle RF için gerekli enerji miktarından oldukça küçük kalması gerçeği, önerilen YP'nin özellikle sağlık ve askeri alanlarda kullanılabilirliğini desteklemektedir.

5.7. YP'nin Kullanıldığı Basit Bir KAA Uygulamasının Modellenmesi ve Benzetimi

Bu bölümde, önerilen YP ve TinySEC güvenlik protokollerini içeren bir KAA uygulama modelinin ve benzetiminin OPNET geliştirme ve benzetim yazılımı yardımıyla gerçekleştirilmesi sunularak bu protokollerin gecikme sonuçları üzerindeki etkisi karşılaştırmalı olarak incelenmektedir. KAA uygulama modelinde ortama erişim protokolü olarak, KAA'lar için geliştirilen ve literatürde sunulan Enerji–etkin ve Gecikme–duyarlı Merkezileştirilmiş OEK (EGMOEK) protokolü kullanılmaktadır (Ceken, 2008).

5.7.1. Bütünleşik YP ve ortama erişim katmanı protokolü (EGMOEK)

Bu bölümde, önerilen YP ve TinySEC güvenlik protokollerinin EGMOEK protokolü ile kullanılarak bütünleşik modellenmesi sunulmaktadır. Bu model, kablosuz algılayıcı düğüm (KAD) ve merkezi düğüm (MD) olmak üzere iki süreçten oluşmaktadır.

5.7.1.1. Bütünleşik YP ve EGMOEK protokolünün KAD modeli

Şekil 5.7'de görülen kablosuz algılayıcı düğüm EGMOEK modeli, aşağıdaki temel işlevleri içermektedir:

- Bağlantı kurma,
- Gecikme duyarlı trafikler için ek zaman dilimi isteği,
- Önceden tahsis edilen zaman dilimlerini kullanma,
- Önceden tahsis edilen zaman dilimlerinden tahsis bilgisini kaldırma,
- Kendi zaman dilimlerinde verilerini gönderme ve
- Gecikme hassasiyeti olmayan algılayıcı düğümler için enerji tüketimini azaltan uyku (radyo alıcı/vericilerini kapatma) işlemi.

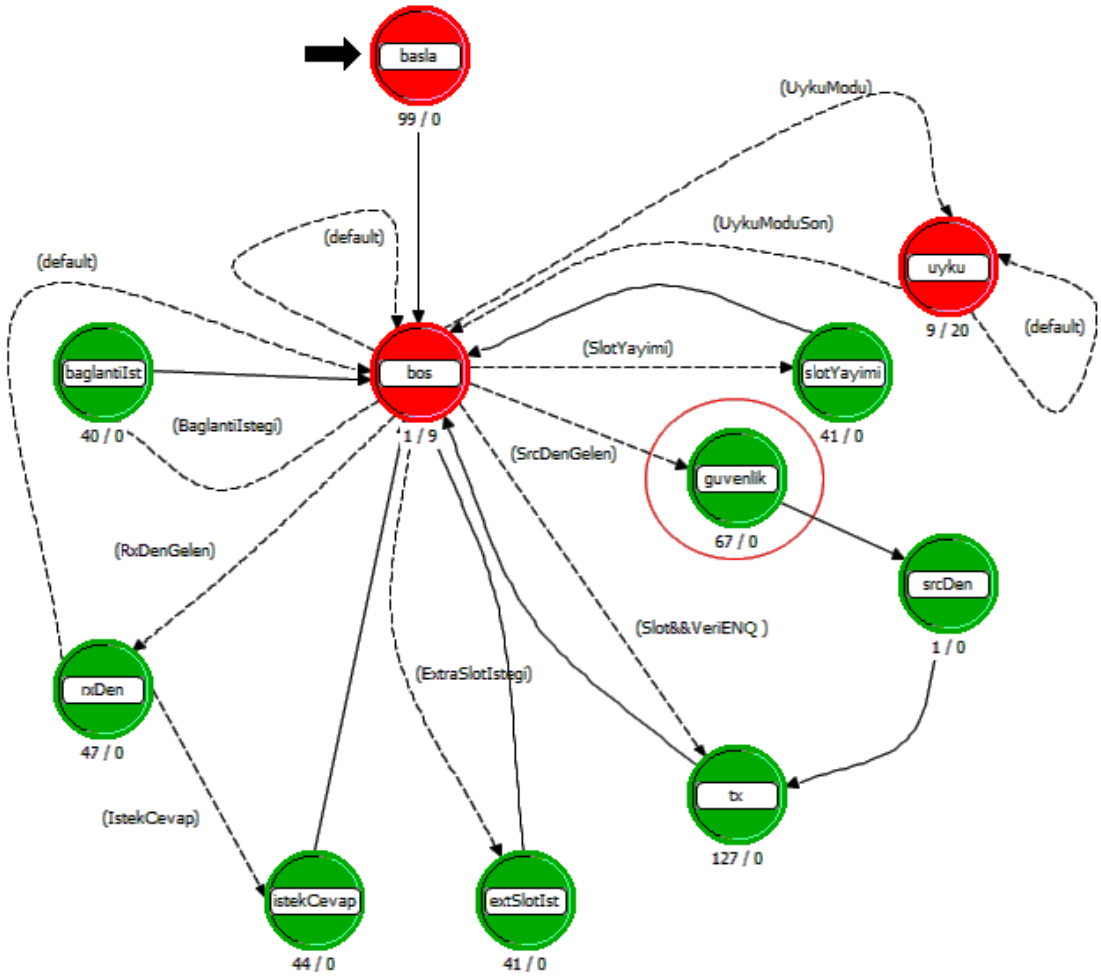
Sunulan çalışmada EGMOEK protokolüne, ortamdan algıladığı veriyi YP ya da TinySEC protokolü ile güvenilir/güvenli hale getirerek MD'ye gönderilmesini sağlayan yeni bir işlev ("güvenlik") daha eklenmiştir. Böylece EGMOEK modelinin, dış ortamdan gelecek olan veri gizliliği saldırılarına/ataklara karşı güvenli olması amaçlanmaktadır. Bu ek işlem, sistemdeki hesaplama karmaşıklığını arttırmakla birlikte iletişimin güvenliğini arttırmaktadır.

KAD süreç modelindeki her bir durum makinesi aşağıdaki şekilde çalışmaktadır:

- "basla" durum makinesi, benzetimdeki diğer süreçler başlatılıncaya kadar bir gecikme gerçekleştirir ve kontrol değişkenleri yükler.
- "bos" durum makinesi, bir kesme isteği gelinceye kadar bekler.
- "baglantiIst" durum makinesi, bağlantı kurma bilgisini içeren bağlantı istek paketini oluşturur ve MD'ye bu paketi gönderir.

- “istekCevap” durum makinesi, MD tarafından tahsis edilen zaman dilimlerinin (slot) sayısını belirler.
- “srcDen” durum makinesi, üst katmandan gelen algılanmış veriyi alır, paketlere böler ve paketleri kuyruğa ekler.
- “guvenlik” durum makinesi, ortamdan algılanmış olan veriyi YP ya da TinySEC protokollerinden birini kullanarak güvenilir/güvenli hale getirir.
- “tx” durum makinesi, üst katmandan alınan veri paketleri KAD için tahsis edilen zaman dilimlerinde hedefine gönderilir.
- “uyku” durum makinesi, gecikme hassasiyeti olmayan veri trafikleri, sahip oldukları enerjiyi korumak için tanımlanan bir zaman aralığında alıcı/vericilerini kapatır.
- “extSlotIst” durum makinesi, gecikmeye duyarlı veri trafiklerinin ek bantgenişliği ihtiyacını bildirmek için ek zaman dilimi istek paketi oluşturur.
- “slotYayimi” durum makinesinde, ek zaman dilimi dağılım paketi oluşturulur ve MD’ye gönderilir.
- “rxDen” durum makinesi, KAD’a gelen tüm paketleri işler (Ceken, 2008).

KAD süreç modelinde gerçekleştirilen tüm işlemlerin algoritması genel hatları ile EK−C’de görülmektedir. Ayrıca bu süreç modele ait C kodları EK−A’da verilmektedir.



Şekil 5.7: KAD süreç modeli.

5.7.1.2. Bütünleşik YP ve EGMOEK protokolünün MD modeli

Merkezi düğüm, aynı küme (cluster) içerisinde bulunan algılayıcı düğümlerin çevreden algıladığı tüm verileri toplamakta ve algılayıcı düğümlerin kablosuz ortama erişimini düzenlemektedir. Şekil 5.8'de görülen EGMOEK protokolünde amaçlanan merkezi düğüm işlevleri üç temel süreç içermektedir:

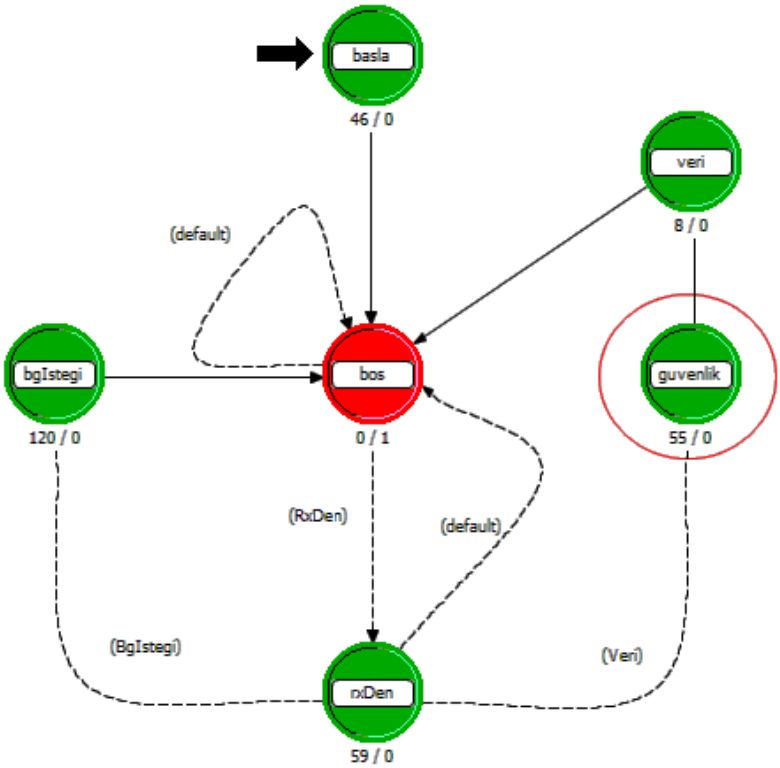
- Herhangi bir KAD için zaman dilimi tahsisi,
- MD'ye gelen veri paketlerini üst katmana gönderme ve
- Zaman dilimi tablosunu (ST) kullanarak gecikmeye duyarlı veri trafikleri için ek zaman dilimleri tahsis etme/dağıtmadır.

Sunulan çalışmada, orijinal EGMOEK modelinden farklı olarak, MD tarafından alınan ve güvenilir/güvenli halde bulunan verilerin güvenlik denetimlerini

gerçekleştirmek amacıyla yeni bir işlev (“güvenlik”) daha bulunmaktadır. Bu süreçteki her bir durum makinesi aşağıdaki şekilde çalışmaktadır:

- “basla” durum makinesinde, süreç başlar ve kontrol değişkenlerine ilk değer ataması yapılır.
- “bos” durum makinesi, bir kesme gelinceye kadar bekler.
- “rxDen” durum makinesi, gelen veri paketinin formatına uygun olan bir sonraki durum makinesine paketi gönderir.
- “guvenlik” durum makinesi, veri paketinin içindeki güvenilir/güvenli verinin güvenlik denetimlerini gerçekleştirir.
- “bgIstegi” durum makinesi, bağlantı isteklerini işler ve ek zaman dilimi isteklerini kabul eder ya da tahsisi kaldırır. Ayrıca, zaman dilimi tablosunu yöneten “adaletli dağılım algoritması”nı da çalıştırır.
- “veri” durum makinesinde, “algılanmış bilgi”, özel bir görevi yerine getirmek için üst katmana gönderilir (Ceken, 2008).

MD süreç modelinde gerçekleştirilen tüm işlemlerin algoritması genel hatları ile EK−D’de görülmektedir. Ayrıca bu süreç modele ait C kodları EK−B’de verilmektedir.

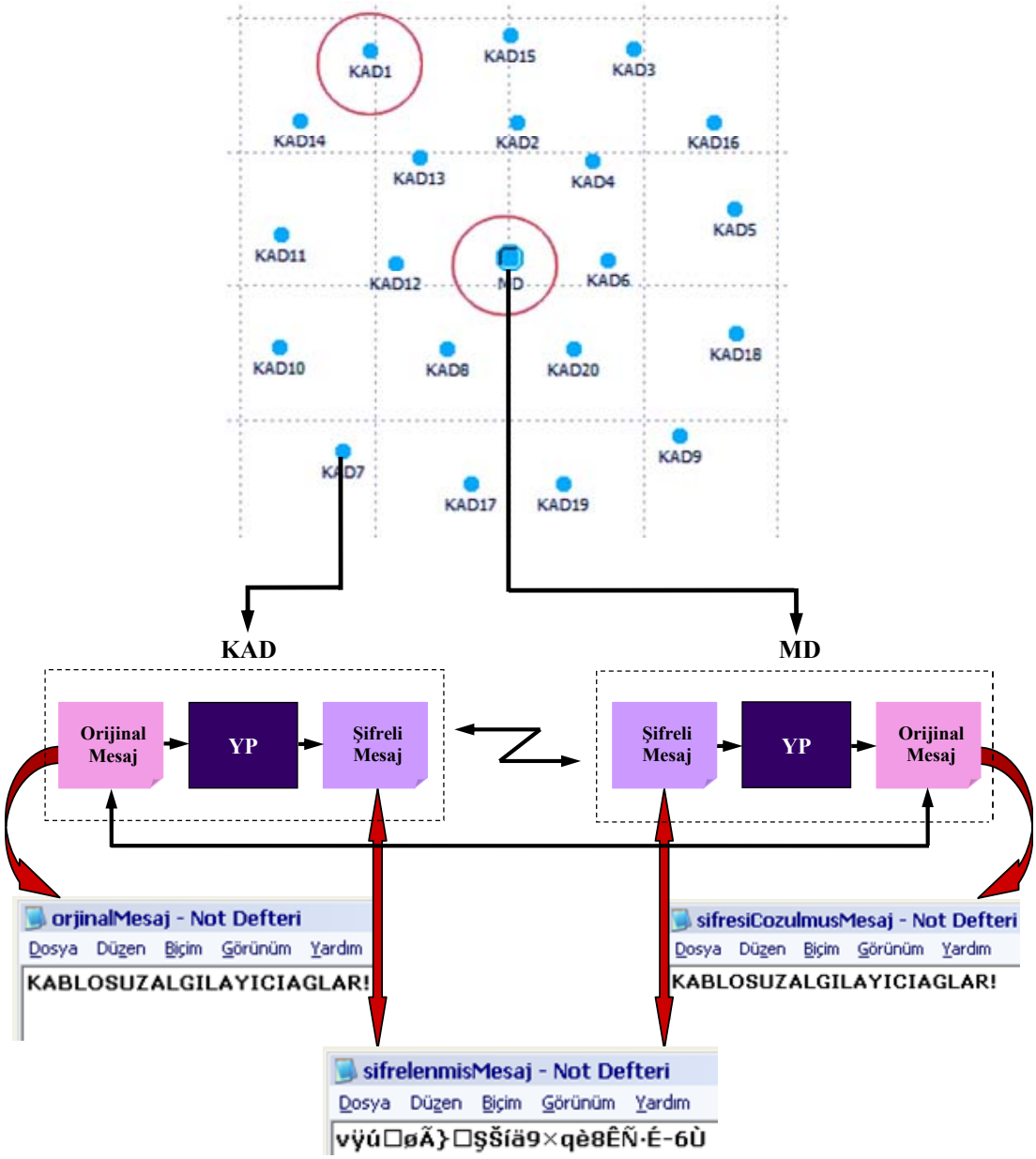


Şekil 5.8: MD süreç modeli.

5.7.2. YP'nin kullanıldığı benzetim modeli ve parametreleri

Şekil 5.9'da sunulan örnek ağ modelinde, 20 adet ortama rastgele dağıtılmış KAA düğümleri (KAD'lar) ve bir adet diğer algılayıcı düğümlere göre kaynakları daha iyi olan merkezi düğüm/baz istasyonu (MD) bulunmaktadır. Uygulamada, algılayıcı düğümler, ortamdan aldıkları bilgileri (algı), modelde kullanılan iki farklı güvenlik protokollerinden biriyle güvenilir/güvenli hale getirerek MD'ye göndermektedir. Bu işlem benzetim çalışma süresince tekrarlanmaktadır. Şekilde ayrıca örnek olarak bir orijinal algılayıcı düğüm mesajı ("KABLOSUZALGILAYICIAGLAR!") YP tarafından şifrelenerek ("vÿú□øÃ}□ŞŠíä9×qè8ÊÑ·É-6Ù") iletilmekte ve alıcıda YP şifre çözme mekanizması tarafından tekrar orijinal mesaj ("KABLOSUZALGILAYICIAGLAR!") elde edilmektedir.

Benzetim ortamında kullanılan TDMA çerçevesi, yukarı bağlantıda, 1 milisaniyede 220 zaman dilimi ve 1 veri paketi içermektedir. Benzetim süresi 3600 saniyedir. Diğer benzetim parametreleri Tablo 5.7'de görülmektedir.



Şekil 5.9: Güvenli KAA uygulama benzetim modeli.

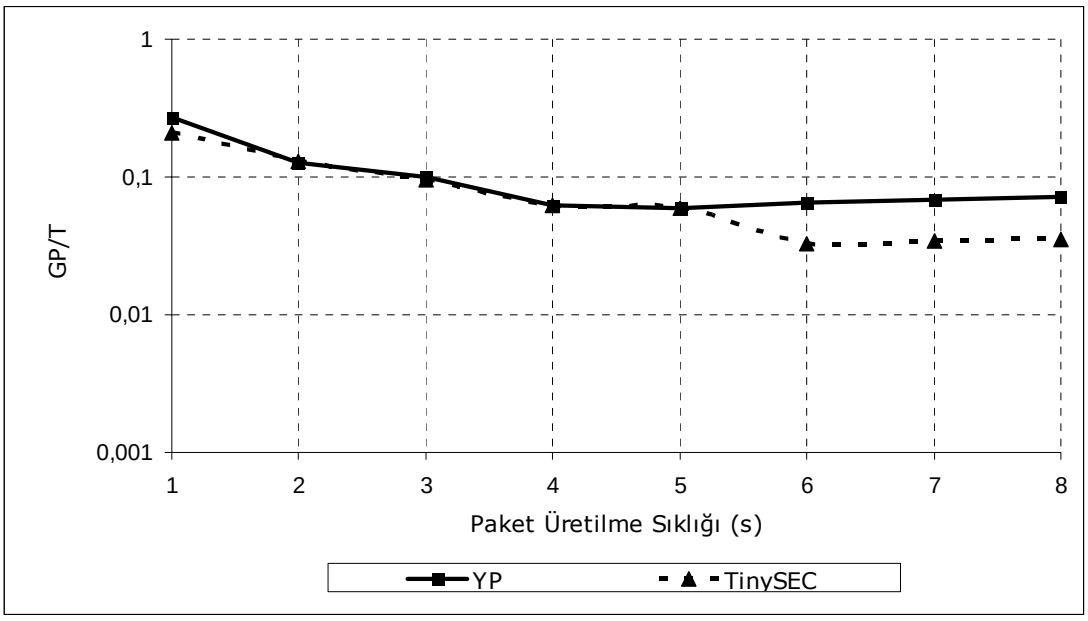
Tablo 5.7: Benzetim parametreleri.

Parametreler	Değerler
KAD mesaj büyüklüğü	1000* Bayt
Veri alış/veriş bit hızı	1 Mbit/s
Verici gücü (MD & KAD'lar)	10 mW
KAA düğüm sayısı	20
Alan büyüklüğü	100 m x 100 m
*Üstel dağılım fonksiyonu kullanılarak üretilmiştir.	

5.7.3. Benzetim Sonuçları

Bu bölümde, KAA uygulama modelinin değişik ağ yük şartları altında elde edilen benzetim sonuçları sunulmaktadır.

Şekil 5.10'da, YP ve TinySEC protokollerinin örnek uygulama modelinde oluşturduğu ortalama gecikme değerlerinin KAD–MD arasındaki toplam ortalama uçtan uca gecikme değerlerine oranı (GP/T), KAD paket üretme sıklığı değiştirilerek incelenmektedir. Oldukça düşük ağ yük durumlarına karşılık gelen paket üretilme sıklığının 8−5 s aralığında, GP/T oranı YP ve TinySEC için sırasıyla %7,08 ve %3,54 gibi ihmal edilebilir değerlerdedir. Ağ yük miktarının arttırılmasına bağlı olarak bu oran, özellikle yüksek güvenlik sağlayan YP için diğer geleneksel TinySEC çözümün sonucuna yaklaşmaktadır (%5,94). Diğer bir ifadeyle YP kullanımı ile oluşan gecikme değerinin doğal olarak toplam uçtan uca gecikme içerisindeki payı oransal olarak az da olsa artmakla birlikte YP'nin hem sunmuş olduğu yüksek güvenlik düzeyi hem de gecikme başarım göstergesi açısından olumlu ve tercih edilebilirliği anlaşılmaktadır.



Şekil 5.10: YP ve TinySEC ortalama gecikme değerlerinin KAD–MD arasındaki toplam ortalama uçtan uca gecikme değerlerine oranı.

5.8. Sonuç

Değişik ve giderek artan uygulama alanları da dikkate alındığında, KAA düğümleri tarafından iletilen verilerin doğruluğu, bütünlüğü ve gizliliği, ancak iletişim güvenliği ve veri güvenilirliği arttırılmış yeni bir güvenlik protokolü geliştirilerek sağlanabilir. Önerilen YP'de özellikle sınırlı kaynaklar için tasarlanmış bir simetrik anahtar blok şifreleme algoritması olan SEA ile protokolün veri güvenilirliğini arttırması için CTR ve CBC−MAC yaklaşımları bütünleşik olarak kullanılmaktadır. SEA algoritması ve CTR yaklaşımı ile mesaj gizliliği ve mesaj tazeliği sağlanırken SEA algoritması ve CBC−MAC yaklaşımı ile mesaj asıllaması/bütünlüğü temin edilmektedir. Böylece KAA uygulamaları için dinamik olarak belirlenebilen oldukça yüksek güvenlikli iletişim imkânı sunulurken, bellek kullanımda ve özellikle enerji tüketimde geleneksel çözümlere kıyasla önemsiz bir oranda artış gerçekleşmektedir.

KAA uygulamalarında, veri güvenilirliğini arttırmak için eklenen güvenlik yaklaşımlarının çalışma süresi, bellek kullanımı ve enerji tüketim miktarlarını arttırması beklenen bir sonuçtur. Bu bakımdan uygulamaların ihtiyaçlarının iyi tespit edilmesi oldukça önemlidir. Zira basit bir geniş ölçekli çevre ya da endüstriyel KAA uygulamasında güvenlik çok fazla önem taşımazken enerji tüketiminin büyük önemi bulunmaktadır. Diğer yandan, askeri ve sağlık uygulamalarında ise güvenlik büyük önem taşırken, algılayıcı düğüm enerji tüketimi, nispî olarak göz ardı edilebilir bir gösterge olarak değerlendirilmektedir.

Önerilen YP, kullanımına bağlı olarak gerçekleşen gecikme sonuçlarının toplam ortalama uçtan uca gecikme sonuçlarına oranı değerlendirildiğinde, özellikle artan ağ yük değerlerinde geleneksel TinySEC yöntemine kıyasla ölçülebilir başarım göstergelerinden olan gecikme (ve dolaylı olarak enerji tüketimi) açısından olumlu olduğu anlaşılmaktadır. Buna ek olarak YP'nin sunmuş olduğu veri güvenilirliği içermiş olduğu ek işlevlere bağlı olarak hâlihazırda oldukça üst düzeyde bulunmaktadır. Bu sonuçlar genel olarak ele alındığında, tez çalışmasının özünü teşkil eden "iletişim güvenliği ve veri güvenilirliği arttırılırken" diğer somut başarım ölçütleri açısından muhtemel olumsuzlukların CTR ve CBC–MAC yaklaşımlarının

YP içerisinde oldukça etkin bir şekilde uyarlanılması ile ortadan kaldırıldığı görülmektedir.

6. SONUÇLAR VE ÖNERİLER

KAA'lar, sağlık ve askeri uygulamalar başta olmak üzere çevre uygulamalarından endüstriyel ortamlara kadar çok değişik alanlarda kullanılmaktadır. KAA'ların geleneksel ağlardan farklı olarak sınırlı kaynaklara sahip düğümlerden oluşması, geleneksel iletişim güvenlik yöntemlerinin KAA düğümlerine doğrudan uygulanması açısından önemli olumsuzluklar doğurmaktadır. Diğer yandan KAA'lar özellikle askeri uygulamalar başta olmak üzere birçok alanda veri gizliliği, bütünlüğü, tazeliği ve kimlik doğrulaması gibi temel güvenlik gereksinimlerini sağlamak zorundadır. Bu temel güvenlik gereksinimlerinin üst düzeyde sağlanmasının en iyi yolu ise bir veri bağı katmanı güvenlik protokolü kullanılmasıdır. Sağlık ya da askeri olmak üzere, kritik KAA uygulamalarında kullanılacak bir güvenlik protokolünün, enerji tüketimi nispî olarak göz ardı edilirken, sunulan tez çalışmasında da esas alındığı gibi yüksek güvenlik sağlayan bir yapıda olması tercih edilir.

Yüksek güvenliğe sahip bir protokolde temel bileşen, kullanılan şifreleme algoritmasıdır. Şifreleme algoritması verinin bir uçtan diğer uca güvenilir bir şekilde iletilmesini temin eder. Bununla birlikte, şifreleme algoritmasının gizliliğini arttırmak ve veri asıllaması/bütünlüğü sağlamak için ise çeşitli blok şifreleme yaklaşımları kullanılmaktadır. Literatürde kablolu ve kablosuz ağlar için geliştirilen birçok şifreleme algoritmaları ve şifreleme yaklaşımları bulunmaktadır. Fakat bu geleneksel şifreleme algoritmalarının KAA'larda doğrudan uygulanması, sistem başarımını olumsuz etkileyen bir aşırı yük oluşturmaktadır. KAA uygulama alanlarının artmasıyla birlikte, günümüzde özellikle sınırlı kaynaklara sahip cihazlar için geliştirilen güncel şifreleme algoritmaları dikkati çekmektedir. Bu algoritmaların, çeşitli şifreleme yaklaşımlarıyla güvenilirlik düzeyleri arttırılarak KAA protokolleri gerçekleştirilmektedir.

Tez çalışmasında, özellikle sınırlı kaynakları bulunan düğümlerde kullanılan ve güvenliği değişik şifre analiz yöntemleri ile kanıtlanmış ve ölçeklenebilir bir yapıya

sahip SEA blok şifreleme algoritması ile gizliliği ve güvenilirliği arttırmak için kullanılan CTR ve CBC–MAC asıllama/bütünlük denetim yöntemlerinin yeni ve bütünleşik bir yaklaşımıyla geliştirilen YP (Yeni Bir KAA Veri Bağı Katmanı Güvenlik Protokolü) gerçekleştirilmiş bulunmaktadır. YP, veri asıllaması, tekrarlama saldırılarını önleme ve veri gizliliği ölçütleri temel alınarak tasarlanan bir güvenlik protokolüdür. Gerçeklenen bu yeni protokolde CTR yaklaşımı ve SEA şifreleme algoritması kullanılarak mesaj (veri) gizliliği ve güvenilirliği arttırılırken, CBC−MAC asıllama yaklaşımı ile mesaj (veri) bütünlüğü ve doğruluğu sağlanmaktadır.

Önerilen YP protokolü ile iki önemli katkı sağlanmıştır;

- Sınırlı kaynaklara sahip düğümlerde kullanılan, güvenliği kanıtlanmış, ölçeklenebilir bir şifreleme algoritması ve şifreleme yaklaşımlarıyla literatürde bulunan veri bağı katmanı güvenlik protokollerine kıyasla yüksek güvenlik sağlayan bir KAA veri bağı katmanı güvenlik protokolü gerçekleştirilmiştir.
- Enerji kaynaklarının etkin kullanımından ziyade özellikle iletilen verinin güvenilirliğinin (doğruluğunun) garanti edilmesini hedefleyen askeri ve sağlık uygulamalarında yüksek güvenlik sağlanmıştır.

Tez çalışmasının yukarıda ifade edilen özelliklerinin yanı sıra bazı ek katkıları da bulunmaktadır. Bunlar aşağıda maddeler halinde sıralanmıştır;

- Geliştirilen protokolde, hâlihazırda bulunan ve kablosuz ortamlarda kullanılan bir şifreleme algoritması (SKIPJACK) yerine literatürde sınırlı kaynaklar için sunulmuş ve çeşitli ortamlarda denenerek etkinliği ve güvenliği kanıtlanmış ölçeklenebilir yeni bir şifreleme algoritması (SEA) kullanılarak kaynak kullanımı azaltılmıştır.
- Literatürde tanımlanmış şifreleme yaklaşımları kullanılarak protokolün güvenliği arttırılmıştır.
- Önerilen YP'nin kullanıldığı örnek bir uygulamaya ait benzetim modeli gerçekleştirilerek araştırmacıların kullanımına sunulmuştur.

Tez çalışması ile YP, TinySEC ve LLSP'nin karşılaştırmalı değerlendirilebilmesi için bu protokollerin AVRStudio ve WinAVR yazılımlarıyla derlenerek benzetimleri gerçekleştirilmiştir.

YP için uyarlanan, $[SEA_{(96,96)}+(CTR+CBC–MAC)]^{YP}$ uygulamasının şifreleme ve şifre çözme toplam süresi yalın $SEA_{(96,96)}$'ya kıyasla 1,92 kat daha fazlayken bu değer $SEA_{(192,192)}$ uygulaması için 2,50 ve $[SEA_{(192,192)}+(CTR+CBC–MAC)]^{YP}$ uygulaması için ise 4,68 katına çıkmıştır. YP'de güvenlik düzeyini arttırmak için kullanılan CTR ve CBC–MAC şifreleme yaklaşımlarının, toplam işlem sürelerini de kabul edilebilir sınırlar içerisinde kalacak şekilde arttırdığı (maliyet-etkin olduğu) görülmüştür.

YP ve LLSP'nin, TinySEC ile normalize edilen bellek kullanım değerleri karşılaştırmalı olarak incelendiğinde, 96-bit veri blok/anahtar boyutlu YP'nin bellek kullanım değeri, standart TinySEC protokolüne kıyasla 1,57 kat daha fazlayken, bu değer 192-bit veri blok/anahtar boyutlu (oldukça yüksek güvenlik düzeyinde) YP için 1,78'e çıkmıştır. Benzer bir şekilde standart LLSP için değerlendirme gerçekleştirildiğinde ise bellek kullanım miktarının 2,26 kat arttığı sonucuna ulaşılmıştır. Geliştirilen ve içerdiği yeni yaklaşımlara bağlı olarak güvenlik düzeyi oldukça üst düzeyde bulunan YP ile TinySEC protokolü karşılaştırıldığında bellek kullanımda ortaya çıkan bu artışın, genel olarak KAA düğümlerine nispî olarak aşırı bir yük getirmediği değerlendirilmektedir.

YP ve LLSP'nin, TinySEC ile normalize edilen enerji tüketim değerleri karşılaştırmalı olarak değerlendirildiğinde ise 96-bit veri blok/anahtar boyutlu YP'nin enerji tüketim değeri, TinySEC protokolüne kıyasla 0,88'e düştüğü görülmüştür. Bu değer 192-bit veri blok/anahtar YP için 2,14'e çıkmıştır. Benzer bir yaklaşımla LLSP için yapılan değerlendirmede ise enerji tüketim değerinin 12,01 kat arttığı sonucuna ulaşılmıştır. Veri güvenilirlik düzeyini dinamik olarak oldukça üst düzeye çıkarmakla birlikte, YP'nin TinySEC ile kıyaslandığında enerji tüketiminin çok fazla artmadığı görülmektedir. Ayrıca diğer geleneksel yöntem olan LLSP'nin basit ölçekte bir güvenlik iyileştirmesi getirmesine rağmen, TinySEC'e kıyasla enerji

tüketiminin çok yüksek düzeyde ortaya çıkması da YP'nin üstünlüğünü kanıtlamaktadır.

YP ve LLSP'nin, TinySEC ile normalize edilen bütünleşik bellek kullanım ve enerji göstergesi (BE) değerlerine bakıldığında, 96-bit veri blok/anahtar boyutlu YP'nin BE değerinin 1,38 ve 192-bit veri blok/anahtar boyutlu YP'nin BE değerinin ise 3,82'e çıktığı görülmüştür. Bu sonuçlardan, üst düzey güvenlik sağlayan 96-bit ve 192-bit veri blok/anahtar boyutlu YP BE değerinin daha basit düzeyde güvenlik sağlayan TinySEC'e oranla fazla artmadığından, sınırlı kaynaklara sahip donanımlar için kullanılabilirliği ortaya çıkmaktadır. LLSP'nin BE değeri ise TinySEC'e kıyasla 27,20 katına ulaşmaktadır. Bu sonuç, LLSP kullanımının sınırlı kaynaklara sahip donanımlara aşırı yük getirmesine sebep olduğunu da açıkça göstermektedir.

YP ve TinySEC protokollerinin örnek uygulama benzetim modelinde oluşturduğu ortalama gecikme değerlerinin kablosuz algılayıcı düğüm (KAD)–merkezi düğüm (MD) toplam ortalama uçtan uca gecikme değerlerine oranı (GP/T), oldukça düşük ağ yük durumlarına karşılık gelmekte olan KAD paket üretilme sıklığının 8−5 s aralığında, YP ve TinySEC için sırasıyla yaklaşık olarak %7 ve %3 seviyesindedir. Ağ yük miktarının arttırılmasına bağlı olarak bu oran, özellikle yüksek güvenlik sağlayan YP için diğer geleneksel TinySEC çözümün sonucuna yaklaşmaktadır (%6). Özetle, YP kullanımı ile oluşan gecikme değerinin toplam uçtan uca gecikme içerisindeki payının oransal olarak azalan bir ivme ile artması, YP'nin sunduğu yüksek güvenlik özelliğinde olduğu gibi, gecikme başarım ölçütü açısından da olumlu ve tercih edilirliğini göstermektedir.

Sonuçlar genel olarak ele alındığında, tez çalışmasının özünü teşkil eden "veri güvenilirliği arttırılırken" diğer somut başarım ölçütleri açısından muhtemel olumsuzlukların CTR ve CBC–MAC yaklaşımlarının YP içerisinde oldukça etkin bir şekilde uyarlanılması ile ortadan kaldırıldığı görülmektedir.

6.1. Tartışma ve Öneriler

Bu tez çalışmasında, önerilen YP'nin, AVRStudio ve WinAVR yazılımlarıyla derlenerek benzetimleri gerçekleştirilmiş ve üzerinde çalışmalar yapılmıştır. Ayrıca YP ve TinySEC protokollerinin kullanıldığı örnek bir KAA uygulamasının OPNET geliştirme ve benzetim yazılımı ile modellenmesi ve benzetimi gerçekleştirilmiş ve elde edilen gecikme sonuçları karşılaştırmalı olarak değerlendirilmiştir. Geliştirilen YP'nin fiziksel olarak KAA düğümlerinde gerçekleştirilmesi ve sağlık alanında kullanımı bir sonraki çalışma olarak düşünülmektedir.

Tez çalışmasından esinlenerek, güvenli yönlendirme algoritmaları konusunda araştırma ve geliştirme çalışmaları gerçekleştirilebilir.

Tez çalışmasında gerçekleştirilen YP, literatürde sunulan değişik anahtar yönetim yöntemleriyle birlikte kullanılarak, veri güvenilirliği, asıllanması/bütünlüğü ve tazeliğinin yanı sıra şifreleme ve şifre çözme anahtarlarının güvenli dağıtımını ve ağda kullanımını sağlayabilir.

KAYNAKLAR

Akyildiz I. F., Su W., Sankarasubramaniam Y., Cayirci E., "Wireless Sensor Networks: Survey", ***Computer Networks***, 38 (2), 393–422, (2002).

Bandırmalı N., Bayılmış C., Ertürk İ., "Kablosuz Algılayıcı Ağlarda Güvenlik", ***Ulusal Teknik Eğitim, Mühendislik ve Eğitim Bilimleri Genç Araştırmacılar Sempozyumu UMES'07***, Kocaeli, 37–40, 20–22 Haziran (2007).

Bandırmalı N., Ertürk İ., Çeken C., Bayılmış C., "Yüksek Riskli Kablosuz Algılayıcı Ağlarda Güvenlik ve Şifreleme Uygulaması", ***Ağ ve Bilgi Güvenliği Ulusal Sempozyumu ABG'08***, Girne, KKTC, 216–220, 16–18 Mayıs (2008a).

Bandırmalı N., Çeken C., Ertürk İ., Bayılmış C., "Skipjack Şifreleme Algoritması Kullanarak Gecikme Duyarlı ve Enerji Etkin Kablosuz Algılayıcı Ağ Güvenlik Hizmeti", ***Elektrik, Elektronik ve Bilgisayar Mühendisliği Sempozyumu ELECO'08***, Bursa, 152–157, 26–30 Kasım (2008b).

Bandirmali N., Erturk I., Ceken C., "Securing Data Transfer in Delay-sensitive and Energy-aware WSNs Using the Scalable Encryption Algorithm*",* ***International Symposium on Wireless and Pervasive Computing ISWPC'09***, Melbourne, Australia, 288–293, 11–13 February (2009).

Bandirmali N., Erturk I., "Increasing the Reliability of Security Protocols for WSNs", ***IEEE International Conference on Application of Information and Communication Technologies AICT'09***, Baku, Azerbaijan, 1–5, 14–16 October (2009).

Başkök M. D., "AES Şifreleme Algoritmasının Modellenmesi", Yüksek Lisans Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, (2007).

Bellare M., Desai A., Jokipii E., Rogaway P., "A Concrete Security Treatment of Symmetric Encryption", ***IEEE 38th Annual Symposium on Foundations of Computer Science,*** Miami Beach, FL, USA, 394–403, 20–22 October (1997).

Bellare M., Kilian J., Rogaway P., "Security of the Cipher Block Chaining Message Authentication Code", ***Journal of Computer and System Sciences***, 61 (3), 362–399, (2000).

Black J., Rogaway P., "A Suggestion for Handling Arbitrary-Length Messages with the CBC–MAC", ***Block Cipher Modes Workshop 1***, Baltimore, Maryland, USA, 20 October (2000).

Brickell E. F., Denning D. E., Kent S. T., Maher D. P., Tuchman W., 1993, *Skipjack Review, Interim Report: The Skipjack Algorithm* [online], http://www.austinlinks.com/Crypto/skipjack-review.html (**Ziyaret tarihi: 30 Mayıs 2009**).

Cakiroglu M., Ozcerit A. T., Ekiz H., Cetin O., "MAC Layer DoS Attacks in Wireless Sensor Networks: A Survey", ***International Conference on Wireless Networks ICWN'06***, Las Vegas, Nevada, USA, 26–29 June (2006).

Ceken C., "An Energy Efficient and Delay Sensitive Centralized MAC Protocol for Wireless Sensor Networks", ***Computer Standards and Interfaces***, 30 (1–2), 20–31, (2008).

Chan H., Perrig A., Przydatek B, Song D., "SIA: Secure Information Aggregation in Sensor Networks", ***Journal of Computer Security***, 15 (1), 69–102, (2007).

Cobb C., "Cryptography for Dummies", ***Wiley Publishing***, Inc., (2004).

Dağlar M., Çayırcı E., "Yardımlaşan Nesne Ağlarında Güvenlik Sorunları ve Çözümler", ***Ağ ve Bilgi Güvenliği Ulusal Sempozyumu ABG'05***, İTÜ Süleyman Demirel Kültür Merkezi, İstanbul, 9–11 Haziran (2005).

Doğan A.Y., "SFINKS Dizi Şifreleme Algoritmasının VHDL ile Yazılımı ve FPGA Üzerinde Gerçeklenmesi", ***İstanbul Teknik Üniversitesi Elektronik Mühendisliği Bitirme Projesi***, İstanbul, (2006).

Dworkin M., "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality", ***NIST Special Publication 800–38C***, (2004).

Dworkin M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication", ***NIST Special Publication 800–38D***, (2007).

Hellman M. E., "An Overview of Public Key Cryptography", ***IEEE Communications Magazine***, 16 (6), 24–32, (1978).

Iwata T., Kurosawa K., "OMAC: One-key CBC MAC", ***10th Fast Software Encryption International Workshop FSE'03***, Springer LNCS, 2887, 129–153, (2003).

Jonsson J., "On the Security of CTR+CBC–MAC", ***Selected Areas in Cryptography,*** Springer LNCS, 2595, 76–93, (2003).

Karaboğa D., Ökdem S., "Kablosuz Algılayıcı Ağlarında Güvenli İletişim Teknikleri", ***Ulusal Elektronik İmza Sempozyumu***, Gazi Üniversitesi, Ankara, 07–08 Aralık (2006).

Karlof C., Sastry N., Wagner D., "TinySEC: A Link Layer Security Architecture for Wireless Sensor Networks", ***The 2nd ACM Conference on Embedded Networked Sensor Systems SENSYS'04***, Baltimore, Maryland, USA, 162–175, 03–05 November (2004).

Lighfoot L.E., Ren J., Li T., "An Energy Efficient Link Layer Security Protocol for Wireless Sensor Networks ", ***IEEE Electro/Information Technology Conference***, EIT 2007, IL, USA, 233–238, 17–20 May (2007).

Lipmaa H., Rogaway P., Wagner D., "CTR Mode Encryption", ***Block Cipher Modes Workshop 1***, Baltimore, Maryland, USA, 20 October (2000).

Liu A., Ning P., "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks", ***Information Processing in Sensor Networks IPSN'08***, St. Louis, Missouri, USA, 22–24 April (2008).

Luk M., Mezzour G., Perrig A., Gligor V., "MiniSEC: A Secure Sensor Network Communication", ***Information Processing in Sensor Networks IPSN'07***, Cambridge, Massachusetts, USA, 25–27 April (2007).

Macé F., Standaert F.-X., Quisquater J.-J., "FPGA Implementation(s) of a Scalable Encryption Algorithm", ***IEEE Transactions on Very Large Scale Integration (VLSI) Systems***, 16 (2), 212–216, (2008).

Menezes A. J., Oorschot P. C., Vanstone S. A., 1996. "Handbook of Applied Cryptography", ***CRC Press***, (1997).

NIST, "SKIPJACK and KEA Algorithm Specifications", ***National Institute of Standards and Technology,*** Version 2.0, 29 May (1998).

Newsome J., Shi E., Song D., Perrig A., "The Sybil Attack in Sensor Networks: Analysis & Defenses", ***Information Processing in Sensor Networks IPSN'04***, Berkeley, California, USA, 259–268, 26–27 April (2004).

OPNET, 2009, *OPNET Modeler 15.0 Documentation [online]*, OPNET Technologies, Release 15.0, http://www.opnet.com (**Ziyaret tarihi: 25 Aralık 2009**).

Pahtan A. S. K., Lee H. W., Hong C. S., "Security in Wireless Sensor Networks: Issues and Challenges", ***Advanced Communication Technology ICACT'06***, 2, 1043–1048, 20–22 February (2006).

Park T., Shin K.G., "LISP: A Lightweight Security Protocol for Wireless Sensor Networks", ***ACM Transactions on Embedded Computing Systems***, 3 (3), 1–27, (2004).

Perrig A., Szewczyk R., Wen V., Culler D., Tygar J. D., "SPINS: Security Protocols for Sensor Networks", ***Wireless Networks,*** 8 (5), 521–534, (2002).

Perrig A., Stankovic J., Wagner D., "Security in Wireless Sensor Networks", ***Communication of the ACM***, 47 (6), 53–57, (2004).

Callas J., 2006, *An Introduction to Cryptography* [online], PGP Corporation, https://supportimg.pgp.com/guides/Introduction_To_Cryptography_eng.pdf (**Ziyaret tarihi: 20 Kasım 2009**).

Raghavendra C. S., Sivalingam K. M., Znati T., "Wireless Sensor Networks**"**, ***Springer***, (2004).

RSA Laboratories, http://www.rsa.com/rsalabs/node.asp?id=2174 (**Ziyaret tarihi: 23 Aralık 2009).**

Ren J., Li T., Aslam D., "A Power Efficient Link Layer Security Protocol (LLSP) for Wireless Sensor Networks", ***Military Communications Conference MILCOM'05***, Atlantic City, New Jersey, USA, 2, 1002–1007, 17–20 October (2005).

Sakallı M. T., Buluş E., "DES'in TMS320C6711 DSP Cihazı Üzerindeki Uygulaması, Performansı Ve Karşılaştırılması", ***Elektrik, Elektronik ve Bilgisayar Mühendisliği Sempozyumu***, Bursa, 18–22 Aralık (2002).

Sakallı M.T., "Modern Şifreleme Yöntemlerinin Gücünün İncelenmesi", Doktora Tezi, ***Trakya Üniversitesi Fen Bilimleri Enstitüsü***, Edirne, (2006).

Samiah A., Aziz A., Ikram N., "An Efficient Software Implementation of AES-CCM for IEEE 802.11i Wireless Standard", ***31st Annual International Computer Software and Applications Conference COMPSAC'07***, Beijing, China, 689–694, 23–27 July (2007).

Shi W., Lee H.-H.S., Ghosh M., Lu C., Boldyreva A., "High Efficiency Counter Mode Security Architecture via Prediction and Precomputation", ***32nd International Symposium on Computer Architecture***, 14–24, 04–08 June (2005).

Standaert F.-X., Piret G., Gershenfeld N., Quisquater J.-J., "SEA: a Scalable Encryption Algorithm for Small Embedded Applications", ***Smart Card Research and Advanced Applications***, Springer LNCS, 3928, 222–236, (2006).

Şahin A., Buluş E., Sakallı M.T., "Modern Blok Şifreleme Algoritmalarının Gücünün İncelenmesi", ***Mühendislik Bilimleri Genç Araştırmacılar Kongresi***, MBGAK'05, İstanbul, 86–92, 17–19 Kasım (2005).

Tektaş M., Baba F., Çalışkan E.M., "Şifreleme Algoritmalarının Sınıflandırılması ve Bir Kredi Kartı Uygulaması", ***3rd International Advanced Technologies Symposium***, Ankara, 18–20 Ağustos (2003).

Tirtea R., Deconinck G., "Specifications Overview for Counter Mode of Operation. Security Aspects in Case of Faults", ***12th IEEE Mediterranean In Electrotechnical Conference MELECON'04***, Dubrovnik, Croita, 2, 769–773, 12–15 May (2004).

Vitaletti A., Palombizio G., "Rijndael for Sensor Networks: is Speed the Main Issue?", ***2nd Workshop on Cryptography for Ad Hoc Networks WCAN'06***, S. Servolo Island, Venice, Italy, 16 July (2006).

Wang C., Sohraby K., "A Survey of Transport Protocols for Wireless Sensor Networks", ***IEEE Network***, 34–40, (2006).

Wheeler D. J., Needham R., "TEA, a Tiny Encryption Algorithm", ***2nd Fast Software Encryption International Workshop FSE'95***, Springer LNCS, 1008, 363–366, (1995).

Wood A.D., Stankovic J.A., "Denial of Service in Sensor Networks", ***IEEE Computer***, 35 (10), 54–62, (2002).

Xiao Y., "Security in Distributed, Grid, Mobile, and Pervasive Computing", ***CRC Press***, (2007).

Yerlikaya T., Buluş E., Arda D., "Asimetrik Kripto Sistemler ve Uygulamaları", ***Mühendislik Bilimleri Genç Araştırmacılar Kongresi MBGAK'05***, İstanbul, Türkiye, 24–31, 17–19 Kasım (2005).

Yerlikaya T., "Yeni Şifreleme Algoritmalarının Analizi", Doktora Tezi, ***Trakya Üniversitesi Fen Bilimleri Enstitüsü***, Edirne, (2006).

Yuval G., "Reinventing the Travois: Encryption/MAC in 30 ROM Bytes", ***4th Fast Software Encryption International Workshop FSE'97***, Springer LNCS, 1267, 205–209, (1997).

Zhu S., Setia S., Jajodia S., "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks", ***10th ACM Conference on Computer and Communications Security***, Washington, DC, USA, 62–72, 27–30 October (2003).

EK−A: KAD−OEK MODÜLÜ SÜREÇ MODELİNE AİT PROGRAM KODLARI

Bu ekte, YP'nin Kablosuz Algılayıcı Düğüm (KAD) süreç modelinin C programlama dilinde yazılan kodları verilmektedir. Bu ekte sunulan kodlarda Ceken (2008) makalesindeki çalışmada geliştirilen kodlardan faydalanılmıştır ve bu kısımlar italik olarak gösterilmektedir.

"basla" Durum Makinesi:

```
slot_genisligi=0.001;
my_id = op_id_self();
my_node_id = op_topo_parent (my_id);

op_ima_obj_attr_get (my_node_id, "TerminalID", &TermID);
op_ima_obj_attr_get (my_node_id, "HedefID", &HedefID);
op_ima_obj_attr_get (my_node_id, "UykuModu", &uykuModu);
op_ima_obj_attr_get (my_node_id, "Uretec.Start Time", &baslama);
op_ima_obj_attr_get (my_node_id, "Uretec.Stop Time", &bitis);

/* Transmitter veri hizini al */
tx_id = op_topo_assoc (my_id, OPC_TOPO_ASSOC_OUT, OPC_OBJTYPE_RATX, 0);
comp_id = op_topo_child (tx_id, OPC_OBJTYPE_COMP, 0);
tx_ch_id = op_topo_child (comp_id, OPC_OBJTYPE_RATXCH, 0);
op_ima_obj_attr_get (tx_ch_id, "data rate", &tx_data_rate);


for (i=0;i<SLOT_SAYISI;i++)
benim_slot[i] =-1;

/* durum degiskeni */
istek_ack=0;
baglanti_kuruldu=OPC_FALSE;
replay=OPC_FALSE;
connection=OPC_FALSE;
extraSlot=OPC_FALSE;
l=0;

op_prg_mem_free(&satablo);

paketGucu=0;
gondermeSure=0;
almaSure=0;
bosSure=0;
uykuSure=0;

esikAsildi=OPC_FALSE;
sleepModeFlag=OPC_FALSE;
paketGeldi=OPC_FALSE;
```

```
Cnt =0x01;

RKey[0] = 0x01;
RKey[1] = 0x00;
RKey[2] = 0x00;
RKey[3] = 0x00;
RKey[4] = 0x00;
RKey[5] = 0x00;
LKey[0] = 0x00;
LKey[1] = 0x00;
LKey[2] = 0x00;
LKey[3] = 0x00;
LKey[4] = 0x00;
LKey[5] = 0x00;

/* Struct yapisi icin bellekten yer ayir */

op_prg_mem_free(&sptr);
sptr = (Paket_Data *)op_prg_mem_alloc (sizeof (Paket_Data));

ptextlength = 24;
blocksize = 12;

        paketGucuEvh = op_stat_reg("Gonderilen Paketin Gucu", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
        gondermeSureEvh = op_stat_reg("Gonderme Suresi", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
        almaSureEvh = op_stat_reg("Alma Suresi", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
        bosSureEvh = op_stat_reg("Bos Suresi", OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
        uykuSureEvh = op_stat_reg("Uyku Suresi", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);


/* baslama ve bitis zamanlarina göre self interrupt ayarla */

op_intrpt_schedule_self (baslama, 10);
op_intrpt_schedule_self (op_sim_time ()+baslama+3,40);

if (uykuModu)
        op_intrpt_schedule_self (op_sim_time ()+50,50);     /* sleep mode icin */

op_intrpt_schedule_self (op_sim_time (), 0);
```

“bos” Durum Makinesi:

```
Giriş İşlemleri:
currentTime=op_sim_time();

Çıkış İşlemleri:
current_intrpt_type = op_intrpt_type ();
bosSure+=op_sim_time()-currentTime;
if(UykuModu)
        sleepModeFlag=OPC_TRUE;
```

“baglantiIst” Durum Makinesi:

```
pkptr = op_pk_create_fmt ("cc_WSN_conreq_pk");
op_pk_nfd_set (pkptr, "SourceID", TermID);
su_an = op_sim_time();

/* hangi zaman dilimi? */
kullanilan_slot = (int) floor ((su_an / slot_genisligi) + EPSILON);
simdiki_slot = kullanilan_slot % SLOT_SAYISI;

/* Paketi gondermek icin yeterli zaman var mi? */
kalan_zaman = ((kullanilan_slot + 1)*slot_genisligi) - su_an;
pk_genisligi = (double) op_pk_total_size_get (pkptr);
pk_zamani = (double) pk_genisligi / tx_data_rate;

if((pk_zamani<kalan_zaman)&&!Busy)
{
        if (op_prg_odb_ltrace_active ("tdma"))
        {
                printf ("Terminal %d band genisligi istek paketini %f de
gönderiyor\n",TermID,op_sim_time ());
                printf ("\n");
        }

        op_pk_send (pkptr, TX_OUT_STRM);

}
else    /* Bir sonraki slot bos mu ona bak */
{
        op_intrpt_schedule_self ((op_sim_time()+(simdiki_slot+1)*slot_genisligi),10);
}
```

“istekCevap” Durum Makinesi:

```
replay=OPC_FALSE;
if(connection)
{
        benim_slot[l++] =dataSlot;
        baglanti_kuruldu=OPC_TRUE;
}
else if(extraSlot)
{
        benim_slot[l++] =dataSlot;
        extraSlot=OPC_FALSE;
}
```

“guvenlik” Durum Makinesi:

```
sprintf(DosyaAdi, "c:/orjinalMesaj.txt");
fptr = fopen(DosyaAdi, "r");

for (i=0; i < ptextlength; i++)
{
    fscanf(fptr,"%c", &metin[i]);
}
```

```
 fclose(fptr);

 for (i=0; i < ptextlength; i++)
{
      plaintext[i] = (byte)metin[i];
}
mac = (byte *) malloc(4*sizeof(byte));
dataAuth.messageIntegrityCode = (byte *) malloc(4*sizeof(byte));
dataAuth.cipdecipText = (byte *) malloc(ptextlength*sizeof(byte));

mac = sea_cbc_encryption(plaintext);
dataAuth = sea_ctr_encryption(plaintext,mac);

for(m=0; m<ptextlength; m++)
{
      sptr->Veri[m]= dataAuth.cipdecipText[m];
}

for(m=0; m<4; m++)
{
      sptr->Mic[m]= dataAuth.messageIntegrityCode[m];
}

sprintf(DosyaAdi1, "c:/sifrelenmisMesaj.txt");
fptr1 = fopen(DosyaAdi1, "w");

for(m = 0; m<ptextlength; m++)
{
      fprintf(fptr1, "%c",sptr->Veri[m]);
}

fprintf(fptr1, "\n");
fclose(fptr1);
```

"srcDen" Durum Makinesi:

```
ustKatmandanPaketAl();
```

"tx" Durum Makinesi:

```
if(baglanti_kuruldu)
{
          su_an = op_sim_time();

          /* hangi zaman dilimi? */
          kullanilan_slot = (int) floor ((su_an / slot_genisligi) + EPSILON);
          simdiki_slot = (kullanilan_slot % SLOT_SAYISI);

          /* Paketi gondermek icin yeterli zaman var mi?*/
          kalan_zaman = ((kullanilan_slot + 1)*slot_genisligi) - su_an;
          pk_genisligi = (double) op_pk_total_size_get (op_subq_pk_access (0, OPC_QPOS_HEAD));
          pk_zamani = (double) pk_genisligi / tx_data_rate;

          /* Su anki zaman dilimi benimki ise ve paket gondermeye
          yetecek kadar zaman varsa paketi gonder */
```

```
for(i=0;i<SLOT_SAYISI;i++)
{
        if(benim_slot[i]==-1)
            break;

        if((simdiki_slot==benim_slot[i])&&(pk_zamani<kalan_zaman))
        {
            /* paketi kuyruktan al ve gonder */
            pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);
            op_pk_send(pkptr, TX_OUT_STRM);

                break;
        }

}


if((!uykuModu)&&(!Mesgul)&&VeriENQ&&(((int)su_an%100>55)&&((int)su_an%100<95)))
    {
        pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);
        op_pk_send(pkptr, TX_OUT_STRM);
    }

    if(VeriENQ)
    {
        for(i=0;i<5;i++)
            sonraki_slotlar[i]=-1;

        k=0;
        j=0;

        for(;;)
        {
            if(benim_slot[k]!=-1)
            {
                for(i=1;i<SLOT_SAYISI;i++)
                    if(benim_slot[k]!=((simdiki_slot+i)%SLOT_SAYISI))
                    {
                        sonraki_slotlar[j++]=i;
                        break;
                    }

                k++;
                }
                else
                    break;
        }

        k=0;
        for(;;k++)
        {
            if(sonraki_slotlar[k]==-1)
                break;
            if(sonraki_slotlar[k]<ilk)
                ilk=sonraki_slotlar[k];

        }

        op_intrpt_schedule_self (((kullanilan_slot+ilk)*slot_genisligi+EPSILON),0);
```

```
        }
}
```

"uyku" Durum Makinesi:

```
/* Giriş İşlemleri: */
if(!paketGeldi)
{
        currentTime=op_sim_time();
        op_intrpt_schedule_self (op_sim_time()+50,60);      /* sleep mode end icin */
}
paketGeldi=OPC_FALSE;

/* Çıkış İşlemleri: */
if(UykuModuSon)//sleep mode end intrpt ise
{
        sleepModeFlag=OPC_FALSE;
        op_intrpt_schedule_self (op_sim_time ()+50,50);     /* yeniden sleep mode a girmesi icin */
        uykuSure+=op_sim_time()-currentTime;

}

else
{       paketGeldi=OPC_TRUE;
        ustKatmandanPaketAl();
}
```

"extSlotIst" Durum Makinesi:

```
pkptr = op_pk_create_fmt ("cc_WSN_burstreq_pk");
op_pk_nfd_set (pkptr, "SourceID", TermID);
su_an = op_sim_time();

/* hangi zaman dilimi? */
kullanilan_slot = (int) floor ((su_an / slot_genisligi) + EPSILON);
simdiki_slot = kullanilan_slot % SLOT_SAYISI;

/* Paketi gondermek icin yeterli zaman var mi? */
kalan_zaman = ((kullanilan_slot + 1)*slot_genisligi) - su_an;
pk_genisligi = (double) op_pk_total_size_get (pkptr);
pk_zamani = (double) pk_genisligi / tx_data_rate;

if((pk_zamani<kalan_zaman)&&!Mesgul)
{
        if (op_prg_odb_ltrace_active ("tdma"))
        {
                printf ("Terminal %d band genisligi istek paketini %f de
gönderiyor\n",TermID,op_sim_time ());
                printf ("\n");
        }
        op_pk_send (pkptr, TX_OUT_STRM);

}
else    /* Bir sonraki slot bos mu ona bak */
{
        op_intrpt_schedule_self ((op_sim_time()+(simdiki_slot+1)*slot_genisligi),40);
}
```

“slotYayimi” Durum Makinesi:

```
pkptr = op_pk_create_fmt ("cc_WSN_burstreq_pk");
op_pk_nfd_set (pkptr, "SourceID", TermID);
su_an = op_sim_time();

/* hangi zaman dilimi? */

kullanilan_slot = (int) floor ((su_an / slot_genisligi) + EPSILON);
simdiki_slot = kullanilan_slot % SLOT_SAYISI;

/* Paketi gondermek icin yeterli zaman var mi? */
kalan_zaman = ((kullanilan_slot + 1)*slot_genisligi) - su_an;
pk_genisligi = (double) op_pk_total_size_get (pkptr);
pk_zamani = (double) pk_genisligi / tx_data_rate;

if((pk_zamani<kalan_zaman)&&!Mesgul)
{
        if (op_prg_odb_ltrace_active ("tdma"))
        {
                printf ("Terminal %d band genisligi istek paketini %f de
gönderiyor\n",TermID,op_sim_time ());
                printf ("\n");
        }
        op_pk_send (pkptr, TX_OUT_STRM);

}
else    /* Bir sonraki slot bos mu ona bak */
{
        op_intrpt_schedule_self ((op_sim_time()+(simdiki_slot+1)*slot_genisligi),40);
}
```

“rxDen” Durum Makinesi:

```
currentTime=op_sim_time();
if((pkptr = op_pk_get (RX_IN_STRM))!=OPC_NIL)
{
        paketGucu = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_POWER);
        op_stat_write(paketGucuEvh, paketGucu);
        op_pk_nfd_get (pkptr, "DestID", &destID);

         if(destID==TermID)
         {
                op_pk_format(pkptr,pk_format);

                if (strcmp("cc_WSN_conrep_pk",pk_format)==0)
                {
                        op_pk_nfd_get (pkptr, "DataSlot", &dataSlot);
                        connection=OPC_TRUE;
                        replay=OPC_TRUE;
                }

                else if (strcmp("cc_WSN_burstrep_pk",pk_format)==0)
                {
                        op_pk_nfd_get (pkptr, "DataSlot", &dataSlot);
                        extraSlot=OPC_TRUE;
                        replay=OPC_TRUE;
```

```
                }

        }

        op_pk_destroy(pkptr);
        }

        almaSure+=op_sim_time()-currentTime;
        op_stat_write(almaSureEvh, almaSure);
```

Fonksiyonlar:

```
static void ustKatmandanPaketAl()
{
   int hucre_sayisi;
   int i;
   Packet* pkptr;

   FIN (UstKatmandanPaketAl());

   if(((pkptr =  op_pk_get (SRC_IN_STRM))!=OPC_NIL))
   {
        hucre_sayisi=(int) floor (op_pk_total_size_get(pkptr)/(34*8));
        if(hucre_sayisi<1)
        hucre_sayisi=1;

        for(i=1;i<hucre_sayisi+1;i++)
        {
             pkptr=op_pk_create_fmt("nb_WSN_SEA_CCM_dataPK");
             op_pk_nfd_set (pkptr, "SourceID",TermID);
             op_pk_nfd_set(pkptr, "Data", sptr, op_prg_mem_copy_create, op_prg_mem_free, sizeof
(Paket_Data));
              op_subq_pk_insert (0, pkptr, OPC_QPOS_TAIL);
        }

        if((!uykuModu)&&((!esikAsildi)&&(op_subq_stat (0, OPC_QSTAT_BITSIZE)>12000)))
        {
                esikAsildi=OPC_TRUE;
                op_intrpt_schedule_self (op_sim_time(),40);
        }

        }
        FOUT;
}


/**********************************/
/*** 1 BIT SAGA DÖNDÜRME ***/
/**********************************/

byte sagadondurme(byte sag)
{
   int bitg;
        FIN (sagadondurme(byte sag));

   /* 1 bit saga döndürme */
  if(( sag & 0x01) == 0x01)
```

```
      bitg = 1;
   else
      bitg = 0;

   sag = sag >> 1;

   if( bitg == 1)

   sag = sag | 0x80;

   FRET (sag);

}

/**********************************/
/*** 1 BIT SOLA DÖNDÜRME ***/
/**********************************/

byte soladondurme(byte sol)
{
   int bitl;

         FIN (soladondurme(byte sol));

   /* 1 bit sola döndürme */

   if(( sol & 0x80) == 0x80)

      bitl = 1;
   else
      bitl = 0;

   sol = sol << 1;

   if( bitl == 1)

   sol = sol | 0x01;
   sol = sol & 0xff;

   FRET (sol);

}

/**********************************/
/*** SEAX1 ***/
/**********************************/

void seax1()
{
 byte a, b, c, d, e, f, g, h;

 FIN (seax1());

 /* 1. Grup ********************************/

 a = RVec[0] + RKey[0];
 b = RVec[1] + RKey[1];
 c = RVec[2] + RKey[2];
 d = c;
```

```
 a =(d & b) ^ a;
  /***************************************/
 e = sagadondurme(a);
 /***************************************/

 LVec[0] = LVec[0] ^ e;
 f = (c & a) ^ b;
 LVec[1] = LVec[1] ^ f;
 g = (f | a) ^ c;

 /***************************************/
 h = soladondurme(g);
 /***************************************/

 LVec[2] = LVec[2] ^ h;

 a = RVec[3] + RKey[3];
 b = RVec[4] + RKey[4];
 c = RVec[5] + RKey[5];
 d = c;

 a =(d & b) ^ a;

 /***************************************/
 e = sagadondurme(a);
 /***************************************/

 LVec[3] = LVec[3] ^ e;
 f = (c & a) ^ b;
 LVec[4] = LVec[4] ^ f;
 g = (f | a) ^ c;

 /***************************************/
 h = soladondurme(g);
 /***************************************/

  LVec[5] = LVec[5] ^ h;

 FOUT;
}

/*********************************/
/*** SEAX2 ***/
/*********************************/

void seax2()
{
 byte a, b, c, d, e, f, g, h;

 FIN (seax2());

 /*2. Grup   ***************************/
 a = RKey[0] + Cnt;
 b = RKey[2] & RKey[1];
 a = b ^ a;
 /***************************************/
 c = sagadondurme(a);
 /***************************************/
 LKey[1] = LKey[1] ^ c;
```

```
  d = (RKey[2] & a) ^ RKey[1];
  LKey[2] = LKey[2] ^ d;
  e = (d | a) ^ RKey[2];

  /*****************************************/
  f = soladondurme(e);
  /*****************************************/

  LKey[3] = LKey[3] ^ f;

  a = RKey[5] & RKey[4];
  b = a ^ RKey[3];

  /*****************************************/
  c = sagadondurme(b);
  /*****************************************/

  LKey[4] = LKey[4] ^ c;
  d = RKey[5] & b;
  e = d ^ RKey[4];
  LKey[5] = LKey[5] ^ e;
  f = e | b;
  g = f ^ RKey[5];

  /*****************************************/
  h = soladondurme(g);
  /*****************************************/

  LKey[0] = LKey[0] ^ h;

  FOUT;
}

/*********************************/
/*** SEAX3 ***/
/*********************************/

void seax3()
{
  byte a, b, c, d, e, f, g;

  FIN (seax3());

  /* 3. Grup  *************************/

  a = LVec[0] + LKey[0];
  b = LVec[1] + LKey[1];
  c = LVec[2] + LKey[2];
  a = (c & b) ^ a;

  /*****************************************/
  d = sagadondurme(a);
  /*****************************************/

  RVec[0] = RVec[0] ^ d;
  e = (c & a) ^ b;
  RVec[1] = RVec[1] ^ e;
  f = (e | a) ^ c;
```

```
  /****************************************/
  g = soladondurme(f);
  /****************************************/
  RVec[2] = RVec[2] ^ g;

  a = LVec[3] + LKey[3];
  b = LVec[4] + LKey[4];
  c = LVec[5] + LKey[5];
  a = (c & b) ^ a;

  /****************************************/
  d = sagadondurme(a);
  /****************************************/

  RVec[3] = RVec[3] ^ d;
  e = (c & a) ^ b;
  RVec[4] = RVec[4] ^ e;
  f = (e | a) ^ c;

  /****************************************/
  g = soladondurme(f);
  /****************************************/

  RVec[5] = RVec[5] ^ g;

  FOUT;
}

/**********************************/
/*** SEAX4 ***/
/**********************************/

void seax4()
{
  byte a, b, c, d, e, f, g, h;

  FIN (seax4());

  /* 4. Grup    ***************************/

  a = LKey[0] + Cnt;
  b = LKey[2] & LKey[1];
  a = b ^ a;

  /****************************************/
  c = sagadondurme(a);
  /****************************************/

  RKey[1] = RKey[1] ^ c;
  d = (LKey[2] & a) ^ LKey[1];
  RKey[2] = RKey[2] ^ d;
  e = (d | a) ^ LKey[2];

  /****************************************/
  f = soladondurme(e);
  /****************************************/
  RKey[3] = RKey[3] ^ f;

  a = LKey[5] & LKey[4];
```

```
  b = a ^ LKey[3];

  /****************************************/
  c = sagadondurme(b);
  /****************************************/

  RKey[4] = RKey[4] ^ c;
  d = LKey[5] & b;
  e = d ^ LKey[4];
  RKey[5] = RKey[5] ^ e;
  f = e | b;
  g = f ^ LKey[5];

  /****************************************/
  h = soladondurme(g);
  /****************************************/

  RKey[0] = RKey[0] ^ h;

  FOUT;
}

byte* encryption(byte temp[])
{
   byte LVecYeni[6],RVecYeni[6],*cipher;
   int i,j,z;

   FIN (encryption(byte temp[]));

   k=0;
   Cnt =0x01;

   cipher = (byte *) malloc(blocksize*sizeof(byte));

    for(z=0; z<6; z++)
    {
      RVec[z]=temp[z];
      LVec[z]=temp[z+6];
    }
 /***LVec*** Bir karakter SAGA döndürüldü */

while(k <= 23)
{
    LVecYeni[0] = LVec[5];

    for(i = 1; i <= 5; i++)
    {
      LVecYeni[i] = LVec[i-1];
    }

    for(i = 0; i <= 5; i++)
    {
      LVec[i] = LVecYeni[i];
    }
    /****************************************/

    seax1();

  if (Cnt!=47)
```

```
  {
    seax2();

    /* Cnt degerini bir arttiriyor */

    Cnt = Cnt+1;
      /***RVec*** Bir karakter SAGA döndürüldü*/

    RVecYeni[0] = RVec[5];

    for(j = 1; j <= 5; j++)
    {
      RVecYeni[j] = RVec[j-1];
    }

    for(j = 0; j <= 5; j++)
    {
      RVec[j] = RVecYeni[j];
    }

    /*************************************/

    seax3();

    seax4();

    /* Cnt degerini bir arttiriyor */

    Cnt = Cnt+1;
   }
   k++;
}

/*************************************/
/*************************************/
/*************************************/

/* Cnt degerini bir azaltiyor */

while(Cnt > 0 )
{    Cnt = Cnt - 1;

    seax4();

    /***RVec*** Bir karakter SAGA döndürüldü*/

    RVecYeni[0] = RVec[5];

    for(j = 1; j <= 5; j++)
    {
      RVecYeni[j] = RVec[j-1];
    }

    for(j = 0; j <= 5; j++)
    {
      RVec[j] = RVecYeni[j];
    }
    /*************************************/
```

```
    seax3();

    /* Cnt degerini bir azaltiyor */

    Cnt = Cnt - 1;

    seax2();
     /***LVec*** Bir karakter SAGA döndürüldü */

    LVecYeni[0] = LVec[5];

    for(i = 1; i <= 5; i++)
    {
      LVecYeni[i] = LVec[i-1];
    }

    for(i = 0; i <= 5; i++)
    {
      LVec[i] = LVecYeni[i];
    }

    /*************************************/

    seax1();

    if (Cnt == 1)
    break;
}

  for(z=0;z<6;z++)
  {
     cipher[z] = RVec[z];
     cipher[z+6] = LVec[z];
  }

  FRET (cipher);
 }

byte* sea_cbc_encryption(byte plaintext[])
{
    int k,z,n,m;
    byte *mac, temp[12], *cipher;
            byte ciphertext[24];

    byte iv[12]= {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

    FIN (sea_cbc_encryption(byte plaintext[]));

    mac = (byte *) malloc(4*sizeof(byte));
    cipher = (byte *) malloc(blocksize*sizeof(byte));

    n = (int)ceil(ptextlength/blocksize);

    k = 0;

    for(m=0; m<n; m++)
    {
       for(z=0; z<blocksize; z++)
       {
```

```
        temp[z] = plaintext[z+k];
      }

      for(z=0; z<blocksize; z++)
      {
        temp[z]=(temp[z] ^ iv[z]);
      }
      /***********************************/
      /*   SIFRELEME   ISLEMI  */
      /***********************************/

      cipher = encryption(temp);

      /***********************************/
      /*   SIFRELEME   ISLEMI  */
      /***********************************/

      for(z=0;z<blocksize;z++)
      {
        ciphertext[z+k] = cipher[z];
      }
      for(z=0;z<blocksize;z++)
      {
        iv[z] = ciphertext[z+k];
      }

      k = k + blocksize;
    }
 for (z=0; z<4; z++)
 {
   mac[z] = ciphertext[(ptextlength-blocksize)+z];
 }

 FRET (mac);
 }

ctrMode sea_ctr_encryption(byte plaintext[], byte *mac)
{
   int k,m,n,z,i;
   byte *cipher;
   ctrMode dataAuth;

   byte counter[12]= {0x11, 0xee, 0xdd, 0xcc, 0xbb, 0xaa, 0x99, 0x88, 0x77, 0x66, 0x55, 0x01};

         FIN (sea_ctr_encryption(byte plaintext[], byte *mac));

   cipher = (byte *) malloc(blocksize*sizeof(byte));

   dataAuth.messageIntegrityCode = (byte *) malloc(4*sizeof(byte));
   dataAuth.cipdecipText = (byte *) malloc(ptextlength*sizeof(byte));

    n = (int)ceil(ptextlength/blocksize);

    k = 0;
    for(m=0; m<n; m++)
    {
        for (i = blocksize - 1; i >= 0; i--)
        {
            counter[i]++;
```

```
            if (counter[i])
            break;
      }
      cipher = encryption(counter);
      for(z=0; z<blocksize; z++)
      {
        dataAuth.cipdecipText[z+k]=(plaintext[z+k] ^ cipher[z]);
      }
      k = k + blocksize;
   }

   for (i = blocksize - 1; i >= 0; i--)
   {
       counter[i]++;

       if (counter[i])
       break;
   }

   cipher = encryption(counter);

   for(z=0; z<4; z++)
   {
     dataAuth.messageIntegrityCode[z]=(mac[z] ^ cipher[z]);
   }

FRET (dataAuth);
}
```

Durum Makineleri Arası Geçiş Şartları:

```
#define SRC_IN_STRM             (0)
#define TX_OUT_STRM             (1)
#define SINK_OUT_STRM           (0)
#define RX_IN_STRM              (1)

#define VeriENQ                 (!(op_subq_empty(0)))

#define SrcDenGelen             (current_intrpt_type == OPC_INTRPT_STRM) &&
(op_intrpt_strm () == SRC_IN_STRM)

#define RxDenGelen              (current_intrpt_type == OPC_INTRPT_STRM) &&
(op_intrpt_strm () == RX_IN_STRM)

#define Mesgul                  (op_stat_local_read (0) == 1.0)

#define Slot                    (current_intrpt_type == OPC_INTRPT_SELF) &&
((op_intrpt_code () == 0))

#define Son                     (current_intrpt_type == OPC_INTRPT_STAT)

#define BaglantiIstegi          (op_intrpt_type() == OPC_INTRPT_SELF) && (op_intrpt_code
() == 10)

#define  BaglantiIstegiSon      (op_intrpt_type() == OPC_INTRPT_SELF) && (op_intrpt_code
() == 20)
#define ExtraSlotIstegi         (op_intrpt_type() == OPC_INTRPT_SELF) && (op_intrpt_code
() == 40)
```

```
#define SlotYayimi                  (op_intrpt_type() == OPC_INTRPT_SELF) && (op_intrpt_code
() == 70)

#define UykuModu                    (op_intrpt_type() == OPC_INTRPT_SELF) && (op_intrpt_code
() == 50)

#define UykuModuSon                 (op_intrpt_type() == OPC_INTRPT_SELF) && (op_intrpt_code
() == 60)
#define  IstekCevap             (replay == OPC_TRUE)

#define ReqRep                      (current_intrpt_type == OPC_INTRPT_REMOTE) &&
(op_intrpt_code () == ISTEK_ACK)

slot_ayirma_tablosu *satablo;
```

EK−B: MD−OEK MODÜLÜ SÜREÇ MODELİNE AİT PROGRAM KODLARI

Bu ekte, YP'nin MD süreç modelinin C programlama dilinde yazılan kodları verilmektedir. Bu ekte sunulan kodlarda Ceken (2008) makalesindeki çalışmada geliştirilen kodlardan faydalanılmıştır ve bu kısımlar italik olarak gösterilmektedir.

"basla" Durum Makinesi:

```
slot_genisligi=0.001;
my_id = op_id_self();
my_node_id = op_topo_parent (my_id);

/* Durum degiskenlerini baslat */
data=0;
brequest=0;
connection=OPC_FALSE;

op_prg_mem_free(&satablo);
satablo = (slot_ayirma_tablosu*) op_prg_mem_alloc (sizeof(slot_ayirma_tablosu));

for (i=0;i<SLOT_SAYISI;i++)
{
        satablo->slot_term[i]=0;
        satablo->slot_ayrildi[i]=0;
        satablo->slot_oncelik[i]=0;
}

Cnt =0x01;

   RKey[0] = 0x01;
   RKey[1] = 0x00;
   RKey[2] = 0x00;
   RKey[3] = 0x00;
   RKey[4] = 0x00;
   RKey[5] = 0x00;
   LKey[0] = 0x00;
   LKey[1] = 0x00;
   LKey[2] = 0x00;
   LKey[3] = 0x00;
   LKey[4] = 0x00;
   LKey[5] = 0x00;

/* Struct yapisi icin bellekten yer ayir */

op_prg_mem_free(&sptr);
sptr = (Paket_Data *)op_prg_mem_alloc (sizeof (Paket_Data));

ptextlength = 24;
blocksize = 12;
```

“bos” Durum Makinesi:

Çıkış İşlemleri:
current_intrpt_type = op_intrpt_type ();

“rxDen” Durum Makinesi:

```
if((pkptr = op_pk_get (RX_IN_STRM))!=OPC_NIL)
{
        op_pk_format(pkptr,pk_format);

                if (strcmp("nb_WSN_SEA_CCM_dataPK",pk_format)==0)
                {
                        op_pk_nfd_get (pkptr, "SourceID", &TermID);
                        data=1;
                        brequest=0;
                        if (op_prg_odb_ltrace_active ("tdma"))
                        {
                                printf ("Baz istasyonu veri paketini gönderiyor");
                                printf ("\n");
                        }
                }
                else if (strcmp("cc_WSN_conreq_pk",pk_format)==0)
                {
                        if (op_prg_odb_ltrace_active ("tdma"))
                        {
                                printf ("Baz istasyonu istek paketini aldi");
                                printf ("\n");
                        }
                                brequest=1;
                                connection=OPC_TRUE;
                                data=0;
                                op_pk_nfd_get (pkptr, "SourceID", &TermID);

                        }
                else if (strcmp("cc_WSN_burstreq_pk",pk_format)==0)
                {
                        if (op_prg_odb_ltrace_active ("tdma"))
                        {
                                printf ("Baz istasyonu istek paketini aldi");
                                printf ("\n");
                        }
                                brequest=1;
                                data=0;
                                op_pk_nfd_get (pkptr, "SourceID", &TermID);
                        }

                else
                        {
                          op_pk_destroy(pkptr);
                        }
}
```

“guvenlik” Durum Makinesi:

```
mac = (byte *) malloc(4*sizeof(byte));
```

```
mic = (byte *) malloc(4*sizeof(byte));
dataAuth.messageIntegrityCode = (byte *) malloc(4*sizeof(byte));
dataAuth.cipdecipText = (byte *) malloc(ptextlength*sizeof(byte));

op_pk_nfd_get(pkptr,"Data",&sptr);

for(m=0; m<ptextlength; m++)
{
       plaintext[m] = sptr->Veri[m];
}

for(m=0; m<4; m++)

{
      mic[m] = sptr->Mic[m];
}
dataAuth = sea_ctr_decryption(plaintext,mic);
mac = sea_cbc_decryption(dataAuth.cipdecipText);

sprintf(DosyaAdi, "c:/sifresiCozulmusMesaj.txt");
fptr = fopen(DosyaAdi, "w");

if (strcmp(mac,dataAuth.messageIntegrityCode))
{
     for(m = 0; m<ptextlength; m++)
     {
          fprintf(fptr, "%c ",dataAuth.cipdecipText[m]);
     }
}
else
          op_pk_destroy(pkptr);

   fclose(fptr);
```

"bgIstegi" Durum Makinesi:

```
if(connection)
{         for(i=0;i<SLOT_SAYISI;i++)
                  if(satablo->slot_ayrildi[i]==0)
                  {
                           bos_slot=OPC_TRUE;
                           break;
                  }

          if(bos_slot)
          {
                  for(i=1;i<SLOT_SAYISI;i++)
                  {

                           if(satablo->slot_ayrildi[i]==0)
                           {
                                    satablo->slot_term[i]=TermID;
                                    satablo->slot_ayrildi[i]=1;
                                    satablo->slot_oncelik[i]=1;
                                    ayrildi=OPC_TRUE;
                                    break;

                           }
```

```
		}

		if(ayrildi)
		{
			pkptr=op_pk_create_fmt("cc_WSN_conrep_pk");
			op_pk_nfd_set (pkptr, "DestID",TermID);
			op_pk_nfd_set (pkptr, "DataSlot",i);
			op_pk_send(pkptr,TX_OUT_STRM);
		}
	/* Yeterli bos slot yoksa */
	}else
	{
		if (op_prg_odb_ltrace_active ("tdma"))
		{
			printf ("yeterli bos slot yok...\n");
		}

	}
}

else
/* Extra Slot istegi */
{
for(i=0;i<SLOT_SAYISI;i++)
	if(satablo->slot_ayrildi[i]==0)
	{
		bos_slot=OPC_TRUE;
		break;
	}

if(bos_slot)
{
	for(i=1;i<SLOT_SAYISI;i++)
	{
		if(satablo->slot_ayrildi[i]==0)
		{
			satablo->slot_term[i]=TermID;
			satablo->slot_ayrildi[i]=1;
			satablo->slot_oncelik[i]=0;
			ayrildi=OPC_TRUE;
			break;
		}

	}

		if(ayrildi)
		{
			pkptr=op_pk_create_fmt("cc_WSN_conrep_pk");
			op_pk_nfd_set (pkptr, "DestID",TermID);
			op_pk_nfd_set (pkptr, "DataSlot",i);
			op_pk_send(pkptr,TX_OUT_STRM);

		}


	/* Yeterli bos slot yoksa */
	}else
	{
		if (op_prg_odb_ltrace_active ("tdma"))
```

```
                {
                        printf ("yeterli bos slot yok...\n");
                }

        }

}
```

“veri” Durum Makinesi:

```
if(TermID==1)
        op_pk_send (pkptr,3);
else if(TermID==2)
        op_pk_send (pkptr,0);
else
        op_pk_destroy(pkptr);
```

Fonksiyonlar:

```
/**********************************/
/*** 1 BIT SAGA DÖNDÜRME ***/
/**********************************/

byte Invsagadondurme(byte sag)
{
   int bitg;

        FIN (Invsagadondurme(byte sag));

   /* 1 bit saga döndürme */

   if(( sag & 0x01) == 0x01)

      bitg = 1;
   else
      bitg = 0;

   sag = sag >> 1;

   if( bitg == 1)

   sag = sag | 0x80;

   FRET (sag);

}

/**********************************/
/*** 1 BIT SOLA DÖNDÜRME ***/
/**********************************/

byte Invsoladondurme(byte sol)
{
   int bitl;

        FIN (Invsoladondurme(byte sol));
```

```
   /* 1 bit sola döndürme */
   if(( sol & 0x80) == 0x80)

      bitl = 1;
   else
      bitl = 0;

  sol = sol << 1;
  if( bitl == 1)

   sol = sol | 0x01;
   sol = sol & 0xff;
   FRET (sol);

}

/*********************************/
/*** SEAX1 ***/
/*********************************/
void Invseax1()
{
 byte a, b, c, d, e, f, g, h;

 FIN (Invseax1());

 /* 1. Grup ********************************/

 a = RVec[0] + RKey[0];
 b = RVec[1] + RKey[1];
 c = RVec[2] + RKey[2];
 d = c;
 a =(d & b) ^ a;

 /*****************************************/
 e = Invsagadondurme(a);
 /*****************************************/

 LVec[0] = LVec[0] ^ e;
 f = (c & a) ^ b;
 LVec[1] = LVec[1] ^ f;
 g = (f | a) ^ c;

 /*****************************************/
 h = Invsoladondurme(g);
 /*****************************************/

 LVec[2] = LVec[2] ^ h;

 a = RVec[3] + RKey[3];
 b = RVec[4] + RKey[4];
 c = RVec[5] + RKey[5];
 d = c;

 a =(d & b) ^ a;

 /*****************************************/
 e = Invsagadondurme(a);
 /*****************************************/
```

```
  LVec[3] = LVec[3] ^ e;
  f = (c & a) ^ b;
  LVec[4] = LVec[4] ^ f;
  g = (f | a) ^ c;

  /*****************************************/
  h = Invsoladondurme(g);
  /*****************************************/
   LVec[5] = LVec[5] ^ h;
  FOUT;
}
/*********************************/
/*** SEAX2 ***/
/*********************************/

void Invseax2()
{
  byte a, b, c, d, e, f, g, h;
  FIN (Invseax2());

  /*2. Grup   ****************************/
  a = RKey[0] + Cnt;
  b = RKey[2] & RKey[1];
  a = b ^ a;

  /*****************************************/
  c = Invsagadondurme(a);
  /*****************************************/

  LKey[1] = LKey[1] ^ c;
  d = (RKey[2] & a) ^ RKey[1];
  LKey[2] = LKey[2] ^ d;
  e = (d | a) ^ RKey[2];

  /*****************************************/
  f = Invsoladondurme(e);
  /*****************************************/

  LKey[3] = LKey[3] ^ f;

  a = RKey[5] & RKey[4];
  b = a ^ RKey[3];

  /*****************************************/
  c = Invsagadondurme(b);
  /*****************************************/

  LKey[4] = LKey[4] ^ c;
  d = RKey[5] & b;
  e = d ^ RKey[4];
  LKey[5] = LKey[5] ^ e;
  f = e | b;
  g = f ^ RKey[5];

  /*****************************************/
  h = Invsoladondurme(g);
  /*****************************************/

  LKey[0] = LKey[0] ^ h;
```

```
  FOUT;
}

/*********************************/
/*** SEAX3 ***/
/*********************************/
void Invseax3()
{
  byte a, b, c, d, e, f, g;

  FIN (Invseax3());

 /* 3. Grup  *************************/

  a = LVec[0] + LKey[0];
  b = LVec[1] + LKey[1];
  c = LVec[2] + LKey[2];
  a = (c & b) ^ a;

  /***************************************/
  d = Invsagadondurme(a);
  /***************************************/

  RVec[0] = RVec[0] ^ d;
  e = (c & a) ^ b;
  RVec[1] = RVec[1] ^ e;
  f = (e | a) ^ c;

  /***************************************/
  g = Invsoladondurme(f);
  /***************************************/

  RVec[2] = RVec[2] ^ g;
  a = LVec[3] + LKey[3];
  b = LVec[4] + LKey[4];
  c = LVec[5] + LKey[5];
  a = (c & b) ^ a;

  /***************************************/
  d = Invsagadondurme(a);
  /***************************************/

  RVec[3] = RVec[3] ^ d;
  e = (c & a) ^ b;
  RVec[4] = RVec[4] ^ e;
  f = (e | a) ^ c;

  /***************************************/
  g = Invsoladondurme(f);
  /***************************************/

  RVec[5] = RVec[5] ^ g;

  FOUT;
}

/*********************************/
/*** SEAX4 ***/
/*********************************/
```

```
void Invseax4()
{
 byte a, b, c, d, e, f, g, h;

 FIN (Invseax4());

 /* 4. Grup    ***************************/

 a = LKey[0] + Cnt;
 b = LKey[2] & LKey[1];
 a = b ^ a;

 /*****************************************/
 c = Invsagadondurme(a);
 /*****************************************/

 RKey[1] = RKey[1] ^ c;
 d = (LKey[2] & a) ^ LKey[1];
 RKey[2] = RKey[2] ^ d;
 e = (d | a) ^ LKey[2];

 /*****************************************/
 f = Invsoladondurme(e);
 /*****************************************/

 RKey[3] = RKey[3] ^ f;

 a = LKey[5] & LKey[4];
 b = a ^ LKey[3];

 /*****************************************/
 c = Invsagadondurme(b);
 /*****************************************/

 RKey[4] = RKey[4] ^ c;
 d = LKey[5] & b;
 e = d ^ LKey[4];
 RKey[5] = RKey[5] ^ e;
 f = e | b;
 g = f ^ LKey[5];

 /*****************************************/
 h = Invsoladondurme(g);
 /*****************************************/

 RKey[0] = RKey[0] ^ h;

 FOUT;
}

byte* decryption(byte temp[])
{
   byte LVecYeni[6],RVecYeni[6], *decipher;
   int i,j,z,k;

   FIN (decryption(byte temp[]));

   k=0;
   Cnt=0x01;
```

```
    decipher = (byte *) malloc(ptextlength*sizeof(byte));

    for(z=0; z<6; z++)
    {
      RVec[z]=temp[z];
      LVec[z]=temp[z+6];
    }


while(k <= 23)
{
    Invseax1();

    /***LVec*** Bir karakter SOLA döndürüldü */

    for(i = 0; i <= 4; i++)
    {
      LVecYeni[i] = LVec[i+1];
    }

    LVecYeni[5] = LVec[0];

    for(i = 0; i <= 5; i++)
    {
      LVec[i] = LVecYeni[i];
    }
    /*************************************/

    if (Cnt !=47)
    {

      Invseax2();

      /* Cnt degerini bir arttiriyor */

      Cnt = Cnt+1;

      Invseax3();

      /***RVec*** Bir karakter SOLA döndürüldü */

      for(j = 0; j <= 4; j++)
      {
        RVecYeni[j] = RVec[j+1];
      }

      RVecYeni[5] = RVec[0];

      for(j = 0; j <= 5; j++)
      {
        RVec[j] = RVecYeni[j];
      }
      /*************************************/
      Invseax4();

      /* Cnt degerini bir arttiriyor */

      Cnt = Cnt+1;
```

```
    }
    k++;
}

    /* Cnt degerini bir azaltiyor */
while(Cnt > 0 )
{
    Cnt = Cnt - 1;
    Invseax4();
    Invseax3();

    /***RVec*** Bir karakter SOLA döndürüldü */

    for(j = 0; j <= 4; j++)
    {
      RVecYeni[j] = RVec[j+1];
    }

    RVecYeni[5] = RVec[0];

    for(j = 0; j <= 5; j++)
    {
      RVec[j] = RVecYeni[j];
    }
    /*************************************/

    /* Cnt degerini bir azaltiyor */

    Cnt = Cnt - 1;

    Invseax2();

    Invseax1();

    /***LVec*** Bir karakter SOLA döndürüldü */

    for(i = 0; i <= 4; i++)
    {
      LVecYeni[i] = LVec[i+1];
    }

    LVecYeni[5] = LVec[0];

    for(i = 0; i <= 5; i++)
    {
      LVec[i] = LVecYeni[i];
    }
    /*************************************/
    if (Cnt == 1)
    break;
}
    for(z=0;z<6;z++)
    {
        decipher[z] = RVec[z];
        decipher[z+6] = LVec[z];
    }

  FRET (decipher);
}
```

```
byte* encryptionDecrypt(byte temp[])
{
   byte LVecYeni[6],RVecYeni[6],*cipher;
   int i,j,z,k;

   FIN (encryptionDecrypt(byte temp[]));

   k=0;
   Cnt =0x01;
   cipher = (byte *) malloc(12*sizeof(byte));

    for(z=0; z<6; z++)
    {
      RVec[z]=temp[z];
      LVec[z]=temp[z+6];
    }

 /***LVec*** Bir karakter SAGA döndürüldü */

while(k <= 23)
{
    LVecYeni[0] = LVec[5];

    for(i = 1; i <= 5; i++)
    {
      LVecYeni[i] = LVec[i-1];
    }

    for(i = 0; i <= 5; i++)
    {
      LVec[i] = LVecYeni[i];
    }
    /*************************************/

    Invseax1();

  if (Cnt!=47)
  {
     Invseax2();

     /* Cnt degerini bir arttiriyor */
     Cnt = Cnt+1;

     /***RVec*** Bir karakter SAGA döndürüldü*/

     RVecYeni[0] = RVec[5];

     for(j = 1; j <= 5; j++)
     {
        RVecYeni[j] = RVec[j-1];
     }

     for(j = 0; j <= 5; j++)
     {
        RVec[j] = RVecYeni[j];
     }

     /*************************************/
```

```
      Invseax3();

      Invseax4();
       /* Cnt degerini bir arttiriyor */

      Cnt = Cnt+1;

    }
    k++;
}

/*************************************/
/*************************************/
/*************************************/

   /* Cnt degerini bir azaltiyor */

while(Cnt > 0 )
{
     Cnt = Cnt - 1;

     Invseax4();

     /***RVec*** Bir karakter SAGA döndürüldü*/

     RVecYeni[0] = RVec[5];

     for(j = 1; j <= 5; j++)
     {
       RVecYeni[j] = RVec[j-1];
     }

     for(j = 0; j <= 5; j++)
     {
       RVec[j] = RVecYeni[j];
     }
     /*************************************/

     Invseax3();
     /* Cnt degerini bir azaltiyor */

     Cnt = Cnt - 1;

     Invseax2();
     /***LVec*** Bir karakter SAGA döndürüldü */

     LVecYeni[0] = LVec[5];

     for(i = 1; i <= 5; i++)
     {
       LVecYeni[i] = LVec[i-1];
     }

     for(i = 0; i <= 5; i++)
     {
       LVec[i] = LVecYeni[i];
     }

     /*************************************/
```

```
    Invseax1();
    if (Cnt == 1)
    break;
}
 for(z=0;z<6;z++)
 {
    cipher[z] = RVec[z];
    cipher[z+6] = LVec[z];
 }

 FRET (cipher);
}

ctrMode sea_ctr_decryption(byte plaintext[], byte mic[])
{
   int k,m,n,z,i;
   byte *decipher;
   ctrMode dataAuth;

   byte counter[12]= {0x11, 0xee, 0xdd, 0xcc, 0xbb, 0xaa, 0x99, 0x88, 0x77, 0x66, 0x55, 0x01};

   FIN (sea_ctr_decryption(byte plaintext[], byte mic[]));

   decipher = (byte *) malloc(blocksize*sizeof(byte));
   dataAuth.messageIntegrityCode = (byte *) malloc(4*sizeof(byte));
   dataAuth.cipdecipText = (byte *) malloc(ptextlength*sizeof(byte));

   n = (int)ceil(ptextlength/blocksize);

   k = 0;

   for(m=0; m<n; m++)
   {
      for (i = blocksize - 1; i >= 0; i--)
      {
          counter[i]++;

          if (counter[i])
          break;
      }
      decipher = encryptionDecrypt(counter);
      for(z=0; z<blocksize; z++)
      {
        dataAuth.cipdecipText[z+k]=(plaintext[z+k] ^ decipher[z]);
      }

      k = k + blocksize;
   }

   for (i = blocksize - 1; i >= 0; i--)
   {
       counter[i]++;

       if (counter[i])
       break;
   }

   decipher = encryptionDecrypt(counter);
   for(z=0; z<4; z++)
```

```
    {
      dataAuth.messageIntegrityCode[z]=(mic[z] ^ decipher[z]);
    }

FRET (dataAuth);
}

byte* sea_cbc_decryption(byte plaintext[])
{
    int k,z,n,m;
    byte *mac, temp[12], *decipher, *deciphertext;

    byte iv[12]= {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

    FIN (sea_cbc_decryption(byte plaintext[]));

    mac = (byte *) malloc(4*sizeof(byte));
    decipher = (byte *) malloc(blocksize*sizeof(byte));
    deciphertext = (byte *) malloc(ptextlength*sizeof(byte));

    n = (int)ceil(ptextlength/blocksize);

    k = 0;

    for(m=0; m<n; m++)
    {
        for(z=0; z<blocksize; z++)
        {
           temp[z] = plaintext[z+k];
        }
        for(z=0; z<blocksize; z++)
        {
           temp[z]=(temp[z] ^ iv[z]);
        }

        /************************************/
        /*   SIFRE ÇÖZME   ISLEMI  */
        /************************************/

        decipher = encryptionDecrypt(temp);
        /************************************/
        /*   SIFRE ÇÖZME   ISLEMI  */
        /************************************/

        for(z=0; z<blocksize; z++)
        {
           deciphertext[z+k] = decipher[z];
        }
        for(z=0; z<blocksize; z++)
        {
           iv[z] = deciphertext[z+k];
        }
        k = k+ blocksize;
     }
     for (z=0; z<4; z++)
     {
         mac[z] = deciphertext[(ptextlength-blocksize)+z];
     }
```

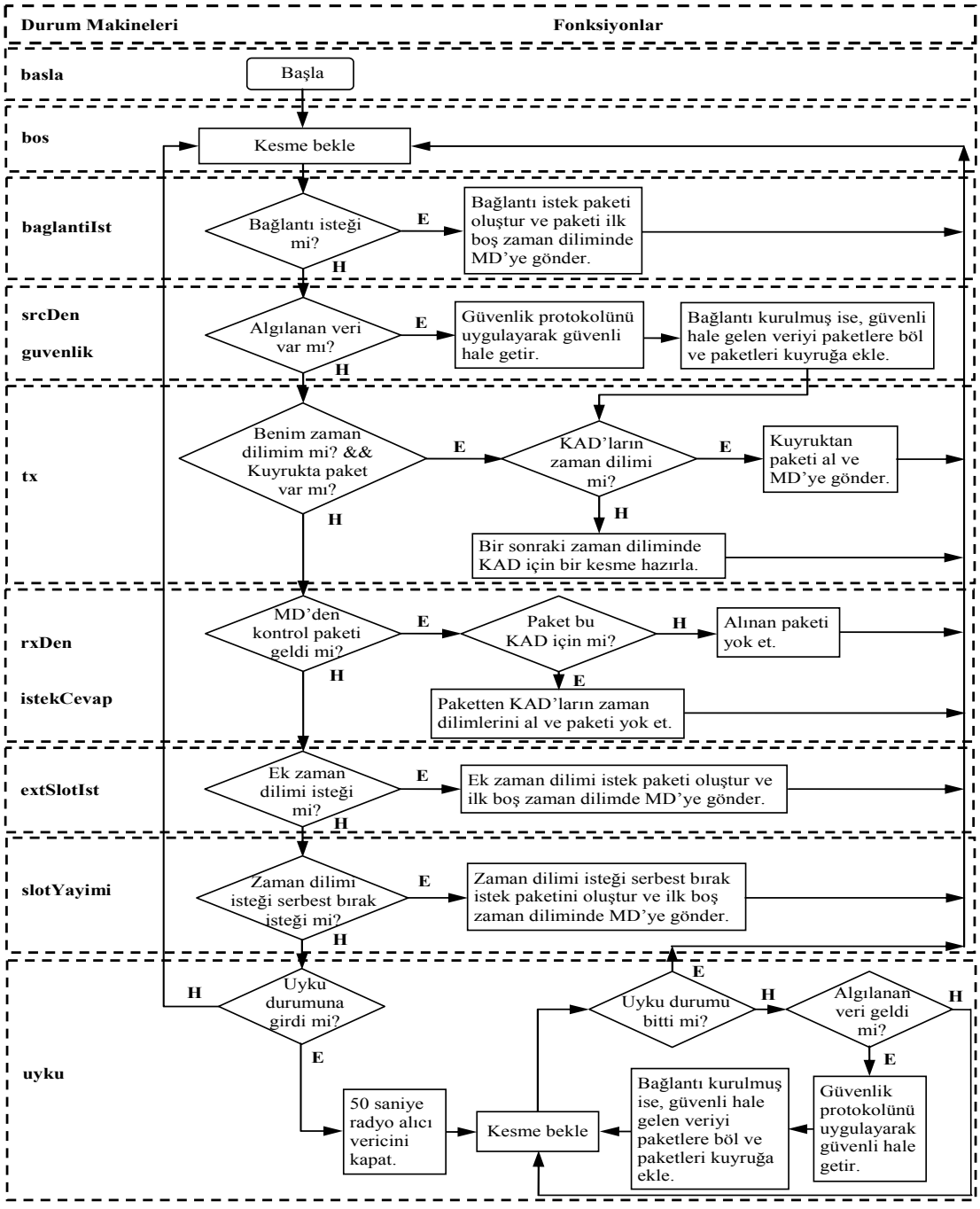
```
FRET (mac);
}
```

Durum Makineleri Arası Geçiş Şartları:

```
#define SRC_IN_STRM          (0)
#define TX_OUT_STRM          (1)
#define SINK_OUT_STRM        (0)
#define RX_IN_STRM           (1)
#define VeriENQ              (!(op_subq_empty(0)))

#define RxDen                (current_intrpt_type == OPC_INTRPT_STRM) &&
(op_intrpt_strm () == RX_IN_STRM)

#define Veri                 (data == 1)

#define BgIstegi             (brequest == 1)

slot_ayirma_tablosu *satablo;
```

EK−C: KAD SÜREÇ MODELİNİN ALGORİTMASI

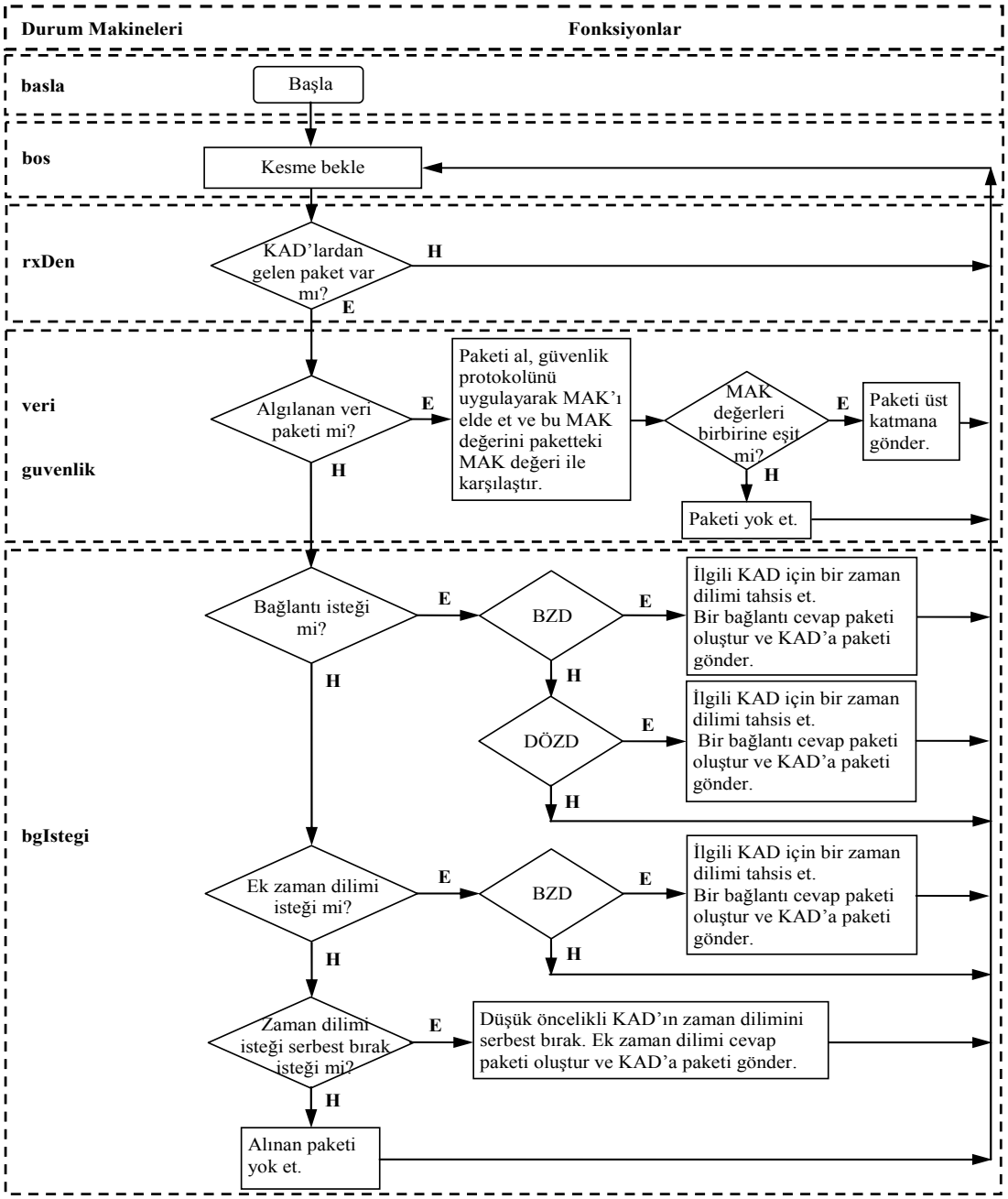
Bu ekte, YP'nin Kablosuz Algılayıcı Düğüm (KAD) süreç modelinin algoritması verilmektedir (Şekil C1). Algoritmada, Ceken (2008) makalesinde sunulan OEK esas alınmış bulunmaktadır.



Şekil C1: KAD süreç modelinin algoritması.

EK−D: MD SÜREÇ MODELİNİN ALGORİTMASI

Bu ekte, YP'nin Merkezi Düğüm (MD) süreç modelinin algoritması verilmektedir (Şekil D1). Algoritmada, Ceken (2008) makalesinde sunulan OEK esas alınmış bulunmaktadır.



BZD: Boş zaman dilimi var mı (örneğin ST'de tahsis edilmiş zaman dilimi=0)
DÖZD: Düşük öncelikli zaman dilimi var mı (örneğin ST'de zaman dilimi önceliği=0)

Şekil D1: MD süreç modelinin algoritması.

KİŞİSEL YAYINLAR VE PROJELER

A. Uluslararası Hakemli Dergilerde Yayımlanan Makaleler (SCI tarafından taranan)

1. Bandirmali N., Erturk I., "HighSEC: A New Highly Reliable Link Layer Security Protocol For Wireless Sensor Networks", 2009 (Gönderildi).

B. Uluslararası Bilimsel Toplantılarda Sunulan ve Bildiri Kitabında Basılan Bildiriler

1. Bandirmali N.**,** Erturk I., "Increasing the Reliability of Security Protocols for WSNs", ***IEEE International Conference on Application of Information and Communication Technologies, AICT'09***, Baku, Azerbaijan, 1–5, 14–16 October (2009).

2. Bandirmali N., Erturk I., Ceken C., "Securing Data Transfer in Delay-sensitive and Energy-aware WSNs Using the Scalable Encryption Algorithm", ***IEEE International Symposium on Wireless and Pervasive Computing, ISWPC'09***, Melbourne, Australia, 288–293, 11–13 February (2009).

3. Bandirmali N., Ceken C., Bayilmis C., Erturk I., "Effects of Transferring Data Traffic on Video Traffic in Wireless ATM Channels", ***International Conference on Electrical and Electronics Engineering, ELECO'05***, Bursa, Türkiye, 402–406, 07–11 December (2005).

4. Bandırmalı N., Çeken C., Bayılmış C., Ertürk İ., "Kablosuz ATM MAC Protokolerinden PRMA/DA ve MAC-GB'nin Karşılaştırmalı İncelemesi", ***4. Uluslararası İleri Teknolojiler Sempozyumu***, Konya, Türkiye, 372–375, 28–30 Eylül (2005).

C. Ulusal Hakemli Dergilerde Yayımlanan Makaleler

1. Bandırmalı N., Ertürk İ., "Yeni Bir Kablosuz Algılayıcı Ağ Veri Bağı Katmanı Güvenlik Protokolü", ***Politeknik Dergisi, Gazi Üniversitesi***, 2009 (Kabul edildi).

2. Bandırmalı N., Çeken C., Bayılmış C., Ertürk İ., "Video Trafiğinin Kablosuz ATM Ortam Erişim Kontrol Mekanizmalarına Etkisinin Karşılaştırmalı İncelemesi", ***Politeknik Dergisi, Gazi Üniversitesi***, 9 (3), 175–180, (2006).

D. Ulusal Bilimsel Toplantılarda Sunulan ve Bildiri Kitaplarında Basılan Bildiriler

1. Bandırmalı N., Çeken C., Ertürk İ., Bayılmış C., "Skipjack Şifreleme Algoritması Kullanarak Gecikme Duyarlı ve Enerji Etkin Kablosuz Algılayıcı Ağ Güvenlik Hizmeti", ***Elektrik ve Elektronik ve Bilgisayar Mühendisliği Sempozyumu, ELECO'08***, Bursa, Türkiye, 152–157, 26–30 Kasım (2008).

2. Bandırmalı N., Ertürk İ., Çeken C., Bayılmış C., "Yüksek Riskli Kablosuz Algılayıcı Ağlarda Güvenlik ve Şifreleme Uygulaması", ***Ağ ve Bilgi Güvenliği Ulusal Sempozyumu, ABG'08***, Girne, KKTC, 216–220, 16–18 Mayıs (2008).

3. Harmankaya A. O., Demiray H. E., Ertürk İ., Bayılmış C., Bandırmalı N., "Kablosuz Ağlarda Güvenlik Protokollerinin Karşılaştırmalı İncelenmesi", ***Ağ ve Bilgi Güvenliği Ulusal Sempozyumu, ABG'08***, Girne, KKTC, 51–57, 16–18 Mayıs (2008).

4. Bandırmalı N., Bayılmış C., Ertürk İ., "Kablosuz Algılayıcı Ağlarda Güvenlik", ***Ulusal Teknik Eğitim, Mühendislik ve Eğitim Bilimleri Genç Araştırmacılar Sempozyumu, UMES'07***, Kocaeli, Türkiye, 37–40, 20–22 Haziran (2007).

5. Bandırmalı N., Ertürk İ., "Algılayıcı Ağ İşletim Sistemi ve Uygulama Alanları", ***Ulusal Teknik Eğitim, Mühendislik ve Eğitim Bilimleri Genç Araştırmacılar Sempozyumu, UMES'07***, Kocaeli, Türkiye, 136–139, 20–22 Haziran (2007).

6. Bandırmalı N., Çeken C., Bayılmış C., Ertürk İ., "Kablosuz Erişim Yöntemlerinin Karşılaştırılmalı İncelemesi", ***Elektrik, Elektronik, Bilgisayar Mühendisliği 11. Ulusal Kongresi, EMO***, İstanbul, Türkiye, 95–98, 22–25 Eylül (2005).

7. Bayılmış C., Ertürk İ., Çeken C., Bandırmalı N., "DSR ve AODV MANET Yönlendirme Protokollerinin Başarım Değerlendirmesi", ***Elektrik, Elektronik, Bilgisayar Mühendisliği 11. Ulusal Kongresi, EMO***, İstanbul, Türkiye, 280–283, 22–25 Eylül (2005).

E. Görev Aldığı Projeler

1. Yardımcı Araştırmacı***, Kablosuz Algılayıcı Ağlar İle Bir Acil Tespit ve Kurtarma Sistemi Tasarımı ve Uygulaması,*** Sanayi Bakanlığı, San–Tez, *00246.STZ.2008-1,* 2008–2010.

2. Yardımcı Araştırmacı***, Akıllı Anten Kullanan Genişbant Kablosuz Sistemler için Servis Kalitesi Destekli Ortam Erişim Protokolü Tasarımı ve Benzetimi*****,*** KOÜ, Bilimsel Araştırmalar Birimi, 2008/022, 2008–2010.

3. Yardımcı Araştırmacı, ***Kablosuz Algılayıcı Ağlarda Akıllı Anten Kullanan Sektörel Tarama Tabanlı Konum Tahmini: OPNET ile Simülasyon Çalışması***, TÜBİTAK, 109E138, 2009–…(Devam ediyor).

ÖZGEÇMİŞ

Necla BANDIRMALI, 1980 yılında Bandırma'da doğdu. İlk, orta ve lise eğitimini Bandırma'da tamamladı. 1997 yılında girdiği Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümü'nden, 2002 yılında Bilgisayar Teknik Öğretmeni olarak mezun oldu. 2002–2005 yılları arasında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Bilgisayar Eğitimi Anabilim Dalı'nda Yüksek Lisans öğrenimini tamamladı. 2006 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Bilgisayar Eğitimi Anabilim Dalı'ndaki doktora eğitimine halen devam etmektedir. 2003 yılından itibaren Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümünde Araştırma Görevlisi olarak çalışmaktadır.